HP-UX Reference

Release 11.0

System Administration Commands

Section 1M

Volume 2 of 5

Edition 1

HEWLETT® PACKARD

B2355-90166

E1097

Printed in: United States

© Copyright 1997 Hewlett-Packard Company

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Copyright Notices. ©Copyright 1983-1997 Hewlett-Packard Company, all rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

©Copyright 1979, 1980, 1983, 1985-93 Regents of the University of California

This software is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

©Copyright 1980, 1984, 1986 Novell, Inc.
©Copyright 1986-1992 Sun Microsystems, Inc.
©Copyright 1985, 1986, 1988 Massachusetts Institute of Technology.
©Copyright 1989-1993 The Open Software Foundation, Inc.
©Copyright 1986 Digital Equipment Corporation.
©Copyright 1990 Motorola, Inc.
©Copyright 1990-1995 Cornell University
©Copyright 1989-1991 The University of Maryland
©Copyright 1988 Carnegie Mellon University
©Copyright 1991-1997 Mentat, Inc.
©Copyright 1996 Morning Star Technologies, Inc.
©Copyright 1996 Progressive Systems, Inc.
©Copyright 1997 Isogon Corporation

Trademark Notices. UNIX is a registered trademark in the United States and other countries, licensed exclusively through The Open Group.

X Window System is a trademark of the Massachusetts Institute of Technology.

MS-DOS and Microsoft are U.S. registered trademarks of Microsoft Corporation.

OSF/Motif is a trademark of the Open Software Foundation, Inc. in the U.S. and other countries.

Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. the manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition: October 1997 (HP-UX Release 11.0)

Volume Two
Table of Contents

Section 1M

Volume Two
Table of Contents

Section 1M

Table of Contents
Volume Two

Section 1M: System Administration Commands

Table of Contents
Volume Two

Table of Contents
Volume Two

Table of Contents
Volume Two

Table of Contents
Volume Two

Section 1M

System Administration Commands

Section 1M

System Administration Commands

NAME

intro - introduction to system maintenance commands and application programs

DESCRIPTION

This section describes commands that are used chiefly for system maintenance and administration purposes. The commands in this section should be used in conjunction with other sections of this manual, as well as the HP-UX System Administration manuals for your system.

Command Syntax

Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [*option* (*s*)] [*cmd_arg* (*s*)]

where the elements are defined as follows:

name Name of an executable file.

option One or more *option*s can appear on a command line. Each takes one of the following forms:

- *no_arg_letter*
A single letter representing an option without an argument.

- *no_arg_letters*
Two or more single-letter options combined into a single command-line argument.

- *arg_letter<>opt_arg*
A single-letter option followed by a required argument where:
arg_letter
is the single letter representing an option that requires an argument,
opt_arg
is an argument (character string) satisfying the preceding *arg_letter*,
<> represents optional white space.

cmd_arg Path name (or other command argument) *not* beginning with **-**, or **-** by itself indicating the standard input. If two or more *cmd_arg*s appear, they must be separated by white space.

RETURN STATUS

Upon termination, each command returns two bytes of status, one supplied by the system giving the cause for termination, and (in the case of "normal" termination) one supplied by the program (for descriptions, see *wait*(2) and *exit*(2)). The system-supplied byte is 0 for normal termination. The byte provided by the program is customarily 0 for successful execution and non-zero to indicate errors or failure such as incorrect parameters in the command line, or bad or inaccessible data. Values returned are usually called variously "exit code", "exit status", or "return code", and are described only where special conventions are involved.

WARNINGS

Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings and therefore become confused upon encountering a null character (the string terminator) within a line.

SEE ALSO

getopt(1), exit(2), wait(2), getopt(3C), hier(5), Introduction(9).

NAME

accept, reject - allow/prevent LP printer queuing requests

SYNOPSIS

/usr/sbin/accept *destination* ...

/usr/sbin/reject [**-r**[*reason*]] *destination* ... [**-r**[*reason*] *destination* ...] ...

DESCRIPTION

The **accept** command permits the **lp** command (see *lp*(1)) to accept printing requests for each named LP printer or printer class *destination* queue.

The **reject** command causes the **lp** command to reject subsequent printing requests for each named *destination* queue. Requests already queued will continue to be processed for printing by the **lpsched** scheduler (see *lpsched*(1M)).

Use the **lpstat** command (see *lpstat*(1)) to find the status of destination queues.

For an overview of LP command interactions, see *lp*(1).

Options

The **reject** command can have the following option.

-r[*reason*] Specifies a string that is used to explain why the **lp** command is not accepting requests for a destination. *reason* applies to all queues mentioned up to the next **-r** option. If *reason* or **-r**[*reason*] is omitted, the default is "**reason unknown**". The maximum length of *reason* is 80 bytes.

reason is reported by the **lpstat** command and by the **lp** command when users direct requests to a rejected destination.

EXTERNAL INFLUENCES

Environment Variables

The **LANG** variable determines the language in which messages are displayed. If **LANG** is not specified or is set to the empty string, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

International Code Set Support

Single- and multibyte character code sets are supported.

EXAMPLES

These examples assume you have a system with two printers named **laser1** and **jet2**, and one class named **lj** that includes both printers.

Example 1

To allow all destinations to accept print requests:

```
accept laser1 jet2 lj
```

Example 2

To reject requests to the **lj** class destination, requiring users to choose a printer:

```
reject lj
```

Example 3

To reject requests to the individual printer destinations, requiring all requests to go through the class destination:

```
accept lj
reject -r"use the lj destination" laser1 jet2
```

WARNINGS

accept and **reject** operate on the local system only.

FILES

`/etc/lp`	Directory of spooler configuration data
`/var/adm/lp`	Directory of spooler log files
`/var/spool/lp`	Directory of LP spooling files and directories

SEE ALSO

enable(1), lp(1), lpstat(1), lpadmin(1M), lpsched(1M), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

NAME

acctdisk, acctdusg, accton, acctwtmp, closewtmp, utmp2wtmp - overview of accounting and miscellaneous accounting commands

SYNOPSIS

/usr/sbin/acct/acctdisk

/usr/sbin/acct/acctdusg [**-u** *file*] [**-p** *file*]

/usr/sbin/acct/accton [*file*]

/usr/sbin/acct/acctwtmp *reason*

/usr/sbin/acct/closewtmp

/usr/sbin/acct/utmp2wtmp

DESCRIPTION

Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. The shell procedures, described in *acctsh*(1M), are built on top of the C programs.

Connect time accounting is handled by various programs that write records into **/etc/utmp**, as described in *utmp*(4). The programs described in *acctcon*(1M) convert this file into session and charging records which are then summarized by **acctmerg** (see *acctmerg*(1M)).

Process accounting is performed by the HP-UX system kernel. Upon termination of a process, one record per process is written to a file (normally **/var/adm/pacct**). The programs in *acctprc*(1M) summarize this data for charging purposes; **acctcms** is used to summarize command usage (see *acctcms*(1M)). Current process data can be examined using **acctcom** (see *acctcom*(1M)).

Process accounting and connect time accounting (or any accounting records in the format described in *acct*(4)) can be merged and summarized into total accounting records by **acctmerg** (see **tacct** format in *acct*(4)). **prtacct** is used to format any or all accounting records (see *acctsh*(1M)).

acctdisk reads lines that contain user ID, login name, and number of disk blocks, and converts them to total accounting records that can be merged with other accounting records.

acctdusg reads its standard input (usually from **find -print**) and computes disk resource consumption (including indirect blocks) by login. Only files found under login directories (as determined from the password file) are accounted for. All files under a login directory are assumed to belong to that user regardless of actual owner. If **-u** is given, records consisting of those file names for which **acctdusg** charges no one are placed in *file* (a potential source for finding users trying to avoid disk charges). If **-p** is given, *file* is the name of the password file. This option is not needed if the password file is **/etc/passwd**. (See *diskusg*(1M) for more details.)

accton turns process accounting off if the optional *file* argument is omitted. If *file* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see *acct*(2) and *acct*(4)).

acctwtmp writes a *utmp*(4) record to its standard output. The record contains the current time and a string of characters that describe the *reason* for writing the record. A record type of ACCOUNTING is assigned (see *utmp*(4)). The string argument *reason* must be 11 or fewer characters, numbers, **$**, or spaces. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp ‘uname‘ >> /var/adm/wtmp
acctwtmp "file save" >> /var/adm/wtmp
```

closewtmp writes a DEAD_PROCESS record, for each user currently logged in, to the file **/var/adm/wtmp**. This program is invoked by *runacct* to close the existing **wtmp** file before creating a new one.

utmp2wtmp writes a USER_PROCESS record, for each user currently logged in, to the file **/var/adm/wtmp**. This program is invoked by *runacct* to initialize the newly created **wtmp** file.

FILES

/usr/sbin/acct	Holds all accounting commands listed in section (1M) of this manual.
/var/adm/pacct	Current process accounting file.

`/etc/passwd`	Used for converting login name to user ID
`/var/adm/wtmp`	Login/logoff history file.

SEE ALSO

acctcms(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), diskusg(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

STANDARDS CONFORMANCE

`acctdisk`: SVID2, SVID3

`accton`: SVID2, SVID3

`acctwtmp`: SVID2, SVID3

NAME

acctcms - command summary from per-process accounting records

SYNOPSIS

/usr/sbin/acct/acctcms [*options*] *files*

DESCRIPTION

acctcms reads one or more *files*, normally in the form described in *acct*(4). It adds all records for processes that executed identically-named commands, sorts them, and writes them to the standard output, normally using an internal summary format.

Options

acctcms recognizes the following options:

- **-a** Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, “hog factor”, characters transferred, and blocks read and written, as in *acctcom*(1M). Output is normally sorted by total kcore-minutes.
- **-c** Sort by total CPU time, rather than total kcore-minutes.
- **-j** Combine all commands invoked only once under *****other**.
- **-n** Sort by number of command invocations.
- **-s** Any file names encountered hereafter are already in internal summary format.
- **-t** Process all records as total accounting records. The default internal summary format splits each field into prime- and non-prime-time parts. This option combines the prime and non-prime time parts into a single field that is the total of both, and provides upward compatibility with old (i.e., UNIX System V) style **acctcms** internal summary format records.

The following options can be used only with the **-a** option.

- **-p** Output a prime-time-only command summary.
- **-o** Output a non-prime- (offshift) time only command summary.

When **-p** and **-o** are used together, a combination prime and non-prime time report is produced. All the output summaries are total usage except number of times executed, CPU minutes, and real minutes which are split into prime and non-prime.

EXAMPLES

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
acctcms file ... >today
cp total previoustotal
acctcms -s today previoustotal >total
acctcms -a -s today
```

SEE ALSO

acct(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

WARNINGS

Unpredictable output results if **-t** is used on new-style internal-summary-format files, or if it is not used with old style internal summary format files.

STANDARDS CONFORMANCE

acctcms: SVID2, SVID3

NAME

acctcom - search and print process accounting files

SYNOPSIS

/usr/sbin/acct/acctcom [[*option*]... [*file*]] ...

DESCRIPTION

The **acctcom** command reads *file*, standard input, or **/var/adm/pacct**, in the form described in *acct*(4) and writes selected records to standard output. Each record represents the execution of one process. The output has the following column titles:

COMMAND NAME
USER
TTYNAME
START TIME
END TIME
REAL (SECS)
CPU (SECS)
MEAN SIZE(K)

Optionally, the following can be displayed:

F	**fork()**/**exec()** flag: **1** for **fork()** without **exec()**
STAT	System exit status
HOG FACTOR	
KCORE MIN	
CPU FACTOR	
CHARS TRNSFD	
BLOCKS READ	Total blocks read and written
PRMID	PRM process resource group ID

The command name is preceded by a **#** if a privileged user is *required* to execute the command.

For example, if a user is logged in as **root**, and executes the **date** command to check the time, this does not require a privileged user, and will be shown by **acctcom** without the **#** character on the line. If the user executes the command **date 0731180092** to set the time, this requires a privileged user, and so will be marked with a **#** by **acctcom**.

If a process is not associated with a known terminal, a **?** is printed in the **TTYNAME** field.

The system exit status **STAT** is **0** if the process terminated by calling **exit**. If it is not **0**, it is the signal number that caused the process to terminate. If a core file image was produced as a result of the signal (see *signal*(5)), the value is the signal number plus **0200**.

If no *files* are specified, and if standard input is associated with a terminal or **/dev/null** (as is the case when using **&** in a shell), **acctcom** reads **/var/adm/pacct**. Otherwise, it reads standard input.

If any *file* arguments are given, they are read in their respective order. Each file is normally read forward, that is, in chronological order by process-completion time. The file **/var/adm/pacct** is usually the current file to be examined. A busy system may need several such files of which all but the current file are found in **/var/adm/***pacct[1-9]*.

Options

acctcom recognizes the following values for the *option* argument. Listing options together has the effect of a logical AND.

-a Show some average statistics about the processes selected. Statistics are printed after the output records.

-b Read backwards, showing latest commands first. This option has no effect when standard input is read.

-f Print in octal the **F** flag and system exit status columns in the output.

-h Instead of mean memory size, **MEAN SIZE(K)**, show the fraction of total available CPU time consumed by the process during its execution. This **HOG FACTOR** is computed as:

total-CPU-time`/`*elapsed-time*

`-i`	Print columns containing the I/O counts in the output.
`-k`	Instead of memory size, show total kcore-minutes.
`-m`	Show mean core size (the default).
`-P`	Show the PRM process resource group ID (**`PRMID`**) of each process. See DEPENDENCIES.
`-r`	Show CPU factor: *user-time*`/(`*system-time*`+`*user-time*`)`
`-t`	Show separate system and user CPU times.
`-v`	Exclude column headings from the output.
`-l` *line*	Show only processes belonging to terminal **`/dev/`***line*.
`-u` *user*	Show only processes belonging to *user*, specified as: a user ID, a login name that is then converted to a user ID, a **`#`** which designates only those processes executed by a privileged user, or **`?`** which designates only those processes associated with unknown user IDs. The # and ? characters should be preceded by a backslash (\) and typed as **`\#`** and **`\?`** to prevent the shell from interpreting the **`#`** as the start of a comment, or the **`?`** as a pattern.
`-g` *group*	Show only processes belonging to *group*, specified as either the group ID or group name.
`-s` *time*	Select processes existing at or after *time*, given in the format: *hour*[**`:`***minute*[**`:`***second*]]
`-e` *time*	Select processes existing at or before *time*; see **`-s`**. Using the same *time* for both **`-s`** and **`-e`** shows the processes that existed at *time*; see **`-s`**.
`-S` *time*	Select processes starting at or after *time*; see **`-s`**.
`-E` *time*	Select processes ending at or before *time*; see **`-s`**.
`-n` *pattern*	Show only commands matching *pattern*, where *pattern* is a regular expression as in *ed*(1) except that **`+`** means one or more occurrences.
`-q`	Do not print any output records. Just print the average statistics as with the **`-a`** option.
`-o` *ofile*	Copy selected process records in the input data format to *ofile*. Suppress standard output printing.
`-H` *factor*	Show only processes that exceed *factor*, where *factor* is the "hog factor" as explained in option **`-h`**.
`-O` *time*	Show only those processes with operating system CPU time exceeding *time*; see **`-s`**.
`-C` *sec*	Show only processes with total CPU time, system plus user, exceeding *sec* seconds.
`-I` *chars*	Show only processes transferring more characters than the cut-off number given by *chars*.
`-R` *prmgroup*	Show only processes belonging to process resource group *prmgroup*, specified as either process resource group name or ID number. See DEPENDENCIES.

WARNINGS

`acctcom` only reports on processes that have terminated. For active processes, use the **`ps`** command (see *ps*(1)).

If *time* exceeds the current system clock time, *time* is interpreted as occurring on the previous day.

The accounting flag is not cleared when one processes exec's another, but only when one process forks another. One side-effect of this is that some processes will be marked with **`#`**, when users do not expect them to be.

For example, the **`login`** command requires a privileged user to assume the identity of the user who is logging-in, setting the ASU bit in the accounting flag (which ultimately causes the **`#`** symbol in the **`acctcom`** output). After assuming the user's identity, **`login`** exec's the user's shell. Since the exec does not clear the ASU flag, the shell will inherit it, and be marked with a **`#`** in the **`acctcom`** output.

DEPENDENCIES

HP Process Resource Manager

The **`-P`** and **`-R`** options require the optional HP Process Resource Manager (PRM) software to be installed and configured. See *prmconfig*(1) for a description of how to configure HP PRM, and *prmconf*(4) for the definition of process resource group.

FILES

`/etc/group`
`/etc/passwd`
`/var/adm/pacct`

SEE ALSO

ps(1), su(1), acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), wait(2), acct(4), utmp(4), signal(5).

HP Process Resource Manager: prmconfig(1), prmconf(4) in *HP Process Resource Manager User's Guide*.

STANDARDS CONFORMANCE

`acctcom`: SVID2, SVID3

NAME

acctcon, acctcon1, acctcon2 - connect-time accounting

SYNOPSIS

/usr/sbin/acct/acctcon [*options*]

/usr/sbin/acct/acctcon1 [*options*]

/usr/sbin/acct/acctcon2

DESCRIPTION

The **acctcon1** command converts a sequence of login/logoff records read from its standard input to a sequence of records, one per login session. Its input should normally be redirected from **/var/adm/wtmp**. Its output is ASCII, giving device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time (numeric), and starting date and time. Prime connect time is defined as the connect time within a specific prime period on a non-holiday weekday (Monday through Friday). The starting and ending time of the prime period and the year's holidays are defined in file **/etc/acct/holidays**.

acctcon2 expects as input a sequence of login session records, produced by **acctcon1**, and converts them into total accounting records (see **tacct** format in *acct*(4)).

acctcon combines the functionality of **acctcon1** and **acctcon2** into one program. It takes the same input format as **acctcon1** and writes the same output as **acctcon2**.

acctcon1 recognizes the following *options*:

-p Print input only, showing line name, login name, and time (in both numeric and date/time formats).

-t **acctcon1** maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The **-t** flag causes it to use, instead, the last time found in its input, thus ensuring reasonable and repeatable numbers for non-current files.

acctcon1 and **acctcon** recognize the following *options*:

-l *file* *file* is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hang-up, termination of **login** (see *login*(1)), and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See *init*(1M) and *utmp*(4).

-o *file* *file* is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

EXAMPLES

These commands are typically used as shown below. The file **ctmp** is created only for the use of commands described by the *acctprc*(1M) manual entry:

```
acctcon1 -t -l lineuse -o reboots < wtmp | sort +1n +2 > ctmp
acctcon2 < ctmp | acctmerg > ctacct
```

or

```
acctcon -t -l lineuse -o reboots < wtmp | acctmerg > ctacct
```

FILES

/var/adm/wtmp
/etc/acct/holidays

WARNINGS

The line usage report is confused by date changes. Use **wtmpfix** (see *fwtmp*(1M)) to correct this situation.

SEE ALSO

acct(1M), acctcms(1M), acctcom(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), init(1M), login(1), runacct(1M), acct(2), acct(4), utmp(4).

STANDARDS CONFORMANCE

acctcon1: SVID2, SVID3

acctcon2: SVID2, SVID3

NAME

acctmerg - merge or add total accounting files

SYNOPSIS

/usr/sbin/acct/acctmerg [*options*] [*file*] ...

DESCRIPTION

acctmerg reads its standard input and up to nine additional files, all in the **tacct** format (see *acct*(4)) or an ASCII version thereof. It merges these inputs by adding records whose keys (normally user ID and name) are identical, and expects the inputs to be sorted on those keys.

Options

acctmerg recognizes the following options:

-a	Produce output in ASCII version of **tacct**.
-i	Input files are in ASCII version of **tacct**.
-p	Print input with no processing.
-t	Produce a single record that totals all input.
-u	Summarize by user ID, rather than user ID and name.
-v	Produce output in verbose ASCII format, with more precise notation for floating point numbers.

EXAMPLES

The following sequence is useful for making "repairs" to any file kept in this format:

```
acctmerg -v < file1 > file2
    edit file2 as desired ...
acctmerg -i < file2 > file1
```

SEE ALSO

acct(1M), acctcms(1M), acctcom(1M), acctcon(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

STANDARDS CONFORMANCE

acctmerg: SVID2, SVID3

NAME

acctprc, acctprc1, acctprc2 - process accounting

SYNOPSIS

/usr/sbin/acct/acctprc

/usr/sbin/acct/acctprc1 [**ctmp**]

/usr/sbin/acct/acctprc2

DESCRIPTION

acctprc1 reads input in the form described by *acct*(4), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If **ctmp** is given, it is expected to contain a list of login sessions in the form described in *acctcon*(1M), sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file. The information in **ctmp** helps it distinguish among different login names that share the same user ID.

acctprc2 reads records in the form written by **acctprc1**, summarizes them by user ID and name, then writes the sorted summaries to the standard output as total accounting records.

acctprc combines the functionality of **acctprc1** and **acctprc2** into one program. It takes the same input format as **acctprc1** (but does not accept the ctmp argument) and writes the same output as **acctprc2**.

These commands are typically used as shown below:

```
acctprc1 ctmp < /var/adm/pacct | acctprc2 > ptacct
```

or

```
acctprc < /var/adm/pacct > ptacct
```

EXTERNAL INFLUENCES

Environment Variables

For the output of **acctprc2**, if the user IDs are identical, **LC_COLLATE** determines the order in which the user names are sorted.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **acctprc2** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

FILES

/etc/passwd

SEE ALSO

acct(1M), acctcms(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctsh(1M), cron(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

WARNINGS

Although it is possible to distinguish among login names that share user IDs for commands run normally, it is difficult to do this for those commands run from **cron** for example (see *cron*(1M)). More precise conversion can be done by faking login sessions on the console via the **acctwtmp** program in *acct*(1M).

A memory segment of the mean memory size is a unit of measure for the number of bytes in a logical memory segment on a particular processor.

STANDARDS CONFORMANCE

acctprc1: SVID2, SVID3

acctprc2: SVID2, SVID3

NAME

chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup, turnacct - shell procedures for accounting

SYNOPSIS

/usr/sbin/acct/chargefee *login-name number*

/usr/sbin/acct/ckpacct [*blocks*]

/usr/sbin/acct/dodisk [**-o**] [*files* ...]

/usr/sbin/acct/lastlogin

/usr/sbin/acct/monacct *number*

/usr/sbin/acct/nulladm *file*

/usr/sbin/acct/prctmp

/usr/sbin/acct/prdaily [**-l**] [**-c**] [*mmdd*]

/usr/sbin/acct/prtacct *file* [*heading*]

/usr/sbin/acct/shutacct [*reason*]

/usr/sbin/acct/startup

/usr/sbin/acct/turnacct on | **off** | **switch**

DESCRIPTION

chargefee Can be invoked to charge a *number* of units to *login-name*. A record is written to **/var/adm/fee**, to be merged with other accounting records during the night.

ckpacct Should be initiated via *cron*(1M). It periodically checks the size of **/var/adm/pacct**. If the size exceeds *blocks*, 1000 by default, **turnacct** is invoked with argument *switch*. If the number of free disk blocks in the **/var** file system falls below 500, **ckpacct** automatically turns off the collection of process accounting records via the **off** argument to **turnacct**. When at least this number of blocks is restored, the accounting will be activated again. This feature is sensitive to the frequency at which **ckpacct** is executed, usually by **cron**.

dodisk Should be invoked by **cron** to perform the disk accounting functions. By default, it will do disk accounting on the special files in **/etc/fstab.** If the **-o** flag is used, it does a slower version of disk accounting by login directory. *files* specifies the one or more filesystem names where disk accounting is to be done. If *files* is used, disk accounting will be done on these filesystems only. If the **-o** flag is used, *files* should be mount points of mounted filesystem. If omitted, they should be the special file names of mountable filesystems.

lastlogin Invoked by **runacct** to update **/var/adm/acct/sum/loginlog** which shows the last date on which each user logged in (see *runacct*(1M)).

monacct Should be invoked once each month or each accounting period. *number* indicates which month or period it is. If *number* is not given, it defaults to the current month (01 through 12). This default is useful if **monacct** is to executed via **cron** on the first day of each month. **monacct** creates summary files in **/var/adm/acct/fiscal** and restarts summary files in **/var/adm/acct/sum**.

nulladm Creates *file* with mode 664 and ensures that owner and group are **adm**. It is called by various accounting shell procedures.

prctmp Can be used to print the session record file normally **/var/adm/acct/nite/ctmp** created by **acctcon1** (see *acctcon*(1M)).

prdaily Invoked by **runacct** (see *runacct*(1M)) to format a report of the previous day's accounting data. The report resides in **/var/adm/acct/sum/rprt***mmdd* where *mmdd* is the month and day of the report. The current daily accounting reports may be printed by typing *prdaily*. Previous days' accounting reports can be printed by using the *mmdd* option and specifying the exact report date desired. The **-l** flag prints a report of exceptional usage by login id for the specifed date. Previous daily reports are cleaned up and therefore inaccessible after each invocation of **monacct**. The **-c** flag prints a report of exceptional resource usage by command, and can be used on current day's accounting data only.

prtacct Can be used to format and print any total accounting (**tacct**) file.

shutacct Should be invoked during a system shutdown to turn process accounting off and append a "reason" record to **/var/adm/wtmp**.

startup Should be called by system startup scripts to turn the accounting on whenever the system is brought up.

turnacct An interface to **accton** (see *acct*(1M)) to turn process accounting **on** or **off**. The **switch** argument turns accounting off, moves the current **/var/adm/pacct** to the next free name in **/var/adm/pacct***incr* then turns accounting back on again. (*incr* is a number starting with **1** and incrementing by one for each additional **pacct** file.) **turnacct** is called by **ckpacct**, and thus can be run under **cron** and used to keep **pacct** to a reasonable size.

FILES

/usr/sbin/acct	holds all accounting commands listed in section (1M) of this manual
/var/adm/fee	accumulator for fees
/var/adm/acct/nite	working directory
/var/adm/pacct	current file for per-process accounting
/var/adm/pacct*	used if **pacct** gets large, and during execution of daily accounting procedure
/usr/sbin/acct/ptecms.awk	contains the limits for exceptional usage by command name
/usr/sbin/acct/ptelus.awk	contains the limits for exceptional usage by login id
/var/adm/acct/sum	summary directory, should be saved
/var/adm/wtmp	login/logoff summary

SEE ALSO

acct(1M), acctcms(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctprc(1M), cron(1M), diskusg(1M), fwtmp(1M), runacct(1M), acct(2), acct(4), utmp(4).

STANDARDS CONFORMANCE

chargefee: SVID2, SVID3

ckpacct: SVID2, SVID3

dodisk: SVID2, SVID3

lastlogin: SVID2, SVID3

monacct: SVID2, SVID3

prctmp: SVID2, SVID3

prdaily: SVID2, SVID3

prtacct: SVID2, SVID3

shutacct: SVID2, SVID3

startup: SVID2, SVID3

turnacct: SVID2, SVID3

NAME

arp - address resolution display and control

SYNOPSIS

arp *hostname*

arp -a [*system*] [*core*]

arp [**-d** | **-D**] *hostname*

arp -f *filename*

arp -s *hostname hw_address* [**temp**] [**pub**] [**rif** *rif_address*]

arp -sfc *hostname nport_id*

DESCRIPTION

The **arp** command displays and modifies the Internet-to-Ethernet and Internet-to-Fibre Channel address translation tables used by the Address Resolution Protocol (ARP).

Options

arp has the following keyletter options:

hostname (first form above) Display the current ARP entry for *hostname*, which must appear in the hostname database (see *hosts*(4)), or for the DARPA Internet address expressed in Internet standard "dot" notation.

-a Display all current ARP entries by reading the table from file *core* (default **/dev/kmem**) based on the kernel file *system* (default **/stand/vmunix**).

-d If an ARP entry exists for the host called *hostname*, delete it. This option cannot be used to delete a permanent ARP entry whose IP address is an interface on the local system.

-D (Not recommended). Delete a permanent ARP entry whose IP address is an interface on the local system. The removal of such an ARP entry may result in loss or limitation of network connectivity with remote machines. The local system will no longer respond to ARP requests for this IP address. Consequently, communication with remote systems is possible only when that communication is initiated by the local system. This option should be used with extreme caution.

-f Read file *filename* and set multiple entries in the ARP tables. Fibre Channel entries in the file should be of the form:

> **-sfc***hostname nport_id*

Other entries in the file should be of the form:

> *hostname hw_address*
> [**temp**]
> [**pub**]
> [**rif**
> *rif_address*]

The argument meanings are the same as for the **-s** option.

-s Create an ARP entry for the host called *hostname* with the hardware station address *hw_address*. The hardware station address is given as six hexadecimal bytes separated by colons. If an ARP entry already exists for *hostname*, the existing entry is updated with the new information.

The entry is permanent unless the word **temp** is given in the command.

If the word **pub** is specified, the entry is published, which means that this system will act as an ARP server responding to requests for *hostname* even though the host address is not its own.

The word **rif** specifies source routing information used for token ring networks. This information allows you to specify the particular bridge route which the token ring packet should be delivered. *rif_address* is given as an even number of hexadecimal bytes separated by colons, up to a maximum of 16 bytes.

-sfc Create a permanent ARP entry for the Fibre Channel host called *hostname* with the N_Port address *nport_id*. The N_Port address is given as three hexadecimal bytes separated by colons. If an ARP entry already exists for *hostname*, the existing entry is updated with the new information.

You need superuser privilege to use the **-d**, **-D**, **-f**, **-s** and **-sfc** options.

AUTHOR

arp was developed by HP and the University of California, Berkeley.

SEE ALSO

ifconfig(1M), inet(3N), hosts(4), arp(7P).

a

NAME

arrayinfo - describe general characteristics of a disk array

SYNOPSIS

arrayinfo [**-j** |**-m** |**-s** |**-ar** |**-dr**] *device_file*

DESCRIPTION

arrayinfo displays summarized information for the SCSI disk array associated with the character device file *device_file*.

By default **arrayinfo** returns the following information:

- array vendor ID
- array product ID
- number of attached disk mechanisms
- vendor/product type of attached disk mechanisms. (Assumes all are the same type)

NOTE: The array vendor ID, and product ID information are constant, regardless of the type and quantity of disks attached.

Options:

arrayinfo recognizes the following options:

-j Displays the current setting of certain jumper switches on each disk mechanism, including:

• Automatic Spin Up	(0 Disable / 1 Enable)
• Parity Error Detect	(0 Disable / 1 Enable)
• Unit Attention	(0 Enable / 1 Disable)
• Initiate Synchronous Data Transfer	(0 Disable / 1 Enable)
• SCSI target address of the mechanism	

-m Displays array mapping information, including:

- The disk vendor, and model type of each disk in the array
- The current status of each disk in the array, as determined by the array controller.
- The array sub-channel, and sub-channel addresses for each disk in the array.

-s Displays serial numbers. This option displays serial number information for the disk array controller, and all attached disk mechanisms.

-ar
Displays array revision information. This option displays revision information for the hardware, firmware, and software of the array controller.

-dr
Display disk revisions. This option displays revision information for the hardware, and firmware of each disk in the array.

RETURN VALUE

arrayinfo returns the following values:

0 Successful completion
-1 Command failed (an error occurred).

DEPENDENCIES

This utility is only compatible with HP C2430 disk arrays.

Series 700

arrayinfo must be used with a device file mapped to a unit address that is not in use by the array controller (unconfigured). By convention unit addresses 6 and 7 should not be configured. Array information should be accessible by addressing either of these unit addresses.

Series 800

Any device file (LU) that is mapped to the disk array can be used to access the array information.

AUTHOR

arrayinfo was developed by Hewlett-Packard.

SEE ALSO

dsp(1M).

NAME

arrayscan - search system for disk arrays

SYNOPSIS

arrayscan

DESCRIPTION

arrayscan searches the system I/O buses to locate the address(es) of attached HP disk array devices. The utility can also be used to determine which logical units are configured on a disk array.

arrayscan performs several functions, including:

- Ensuring device special files exist.

  arrayscan verifies that block and character device special files exist for all LUNs configured. On Series 700 systems, device files are created for all possible LUNs.

- Ensuring disk array software was downloaded.

  arrayscan verifies that the disk array software has been downloaded for each disk array it encounters. If **arrayscan** encounters a disk array that does not have disk array software loaded, it automatically downloads the array software.

- Updating **monitor**, and **pscan** device lists.

  Two files, **/etc/hpC2400/hparray.devs**, and **/etc/hpC2400/hparray.luns** are updated by **arrayscan**. **/etc/hpC2400/hparray.devs** is used by the monitor daemon (**/usr/lbin/hpC2400/arraymond**) to determine which devices to monitor. **/etc/hpC2400/hparray.luns** is used by the parity scan utilities (**pscan**, **scn**, and **rpr**) to determine which LUNs to monitor.

RETURN VALUE

arrayscan returns the following values:

0	Successful completion
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **arrayscan**
- SCSI (device level) communications
- system calls

Error messages generated by arrayscan:

arrayscan: Cannot access lock file. Create an empty file <FILE>

Two semaphore files are used by **arrayscan**, **/etc/hpC2400/pscan.lock**, and **/etc/hpC2400/monitor.lock**. If these files do not exist when **arrayscan** begins, it assumes that the monitor daemon is executing. If the period of time required for the monitor daemon to execute expires, and the files still do not exist, it is assumed that they need to be created. You can create these files, if necessary, using the **touch** command (see *touch*(1));

arrayscan: Unable to open Array Parity Scan list <FILE>

arrayscan updates **/etc/hpC2400/hparray.luns**, and **/etc/hpC2400/hparray.devs**. **arrayscan** was unable to write to this file.

arrayscan: Error from process insf.

An error occurred while executing **insf** (see *insf*(1M)). **insf** is used by **arrayscan** on Series 800 systems to create device files for newly configured disk array devices.

arrayscan: Error from process ioscan.

An error occurred while executing **ioscan**. **ioscan** is used by **arrayscan** to scan for all devices. Disk array devices are filtered from the ioscan output.

arrayscan: No SCSI devices identified. Check SCSI connections.

No SCSI devices were identified. Check SCSI cables and power connections and retry the command.

arrayscan: Unable to create char device special file for path <FILE>
: **arrayscan** will create character, and block device files for all disk array devices it encounters. **arrayscan** was unable to create the device file.

arrayscan: Insufficient dynamic memory
: An attempt to allocate dynamic memory failed.

DEPENDENCIES

This utility is supported only on HP C2425D, HP C2430D, HP C3595A and HP C3596A disk array devices.

AUTHOR

arrayscan was developed by HP.

FILES

/etc/hpC2400/hparray.luns
/etc/hpC2400/hparray.devs
/etc/hpC2400/pscan.lock
/etc/hpC2400/monitor.lock

NAME

asecure - control access to Audio on a workstation

SYNOPSIS

/opt/audio/bin/asecure [**-CdelP**] [**+h** *host*] [**-h** *host*] [**+p** *user*] [**-p** *user*] [**+u** *user*] [**-u** *user*] [**+b** *host***,***user*] [**-b** *host***,***user*]

DESCRIPTION

On Series 700 workstations, audio is secured so that only the user on the local workstation can access audio. You use the **asecure** command to modify audio security. This command does not apply to X stations; on an X station, access to audio is unrestricted.

To modify audio security, become root on the local workstation where you want make a change. Then, use **asecure** as follows:

```
/opt/audio/bin/asecure -C
```

When prompted, enter any meaningful password. Issuing **asecure -C** creates the Audio Security File (ASF). The ASF contains information that determines which hosts and users can access the Aserver, and which users (other than the superuser) can modify the ASF.

If needed, you can allow unrestricted access to audio on this workstation. To remove audio security, issue this command:

```
/opt/audio/bin/asecure -d
```

If instead, you wish to modify security, you use **asecure** to make changes to the information in the ASF. (Because the ASF is a binary file, we do not recommend using an editor on this file.) You can use **asecure** to make these types of changes:

- Allow all clients from a remote host to access the server.
- Allow specific users from all other hosts to access the server.
- Allow a specific user from a specific host to access the server.
- Disable access control, allowing complete unrestricted access to the server, but leaving the ASF intact.

Every operation that creates, reinitializes, or changes the contents of the ASF is logged in the **/var/adm/audio/asecure_log** file, so that you can track any changes to the ASF.

OPTIONS

asecure supports the following options:

+b|-b *host,user*
: Add/delete *hostname,username* pair. You must be either superuser or a **privileged user** to do this. You can supply more than one *hostname,username* pair separated by blanks.

  To use either the **+b** or **-b** options, you MUST supply at least one *hostname,username* pair. This option will not work without a pair.

-C
: Create a new ASF file, called the **audio.sec** file. Access control default is enabled with no entries in the access list. Aserver can now be accessed only by local users on the host machine. If an **audio.sec** file already exists, it is re-initialized.

  You must be superuser to execute this option. This option is mutually-exclusive of all other options.

  This option requires a password. This is an extra layer of protection for the contents of the ASF. It is designed to prevent surreptitious manipulation of the ASF. If you are creating a new ASF, you are prompted for a password and an encrypted copy of that password is stored in the new ASF.

  If the ASF already exists, you are prompted for the password. If your password matches the password stored in the ASF, the ASF is then re-initialized.

-d
: Disable access control to the Aserver. This allows unrestricted access by all clients.

-e
: Enable access control to the Aserver. This restricts access to clients listed in the ASF. Enabled is the default state.

+h|-h *host*	Add/delete *hostnames* for ALL users. You must be either superuser or a **privileged user** to do this. You can supply more than one *hostname* separated by blanks.
-l	List the contents of the ASF. This option shows a list of the hostnames and/or usernames that have access to the Aserver.
-P	Change password for **audio.sec** file. You must be superuser to do this. You are prompted once for the old password, then prompted twice for the new password.
+p|-p *user*	Add/delete **privileged users**. You must be superuser to do this and must enter the password given when the ASF was created (see **-C** option). To see a list of privileged users, you must be superuser and use the **-l** option.
+u|-u *user*	Add/delete *usernames* for ALL hosts. You must be either superuser or a **privileged user** to do this. You can supply more than one *username* separated by blanks.

EXAMPLES

List entries in access list.

```
/opt/audio/bin/asecure -l
```

Disable access control. This means anyone can connect to Aserver without restriction.

```
/opt/audio/bin/asecure -d
```

Add **moonbeam** host for all users to access list. Remove **pluto** host for all users from access list.

```
/opt/audio/bin/asecure +h moonbeam -h pluto
```

Add user **comet** for hosts **saturn** and **mercury** to access list.

```
/opt/audio/bin/asecure +b saturn,comet mercury,comet
```

Add user **comet** to access list for all hosts. Remove users **venus** and **neptune** from access list for all hosts.

```
/opt/audio/bin/asecure +u comet -u venus neptune
```

Create new access list.

```
/opt/audio/bin/asecure -C
```

AUTHOR

asecure was developed by HP.

FILES

/var/opt/audio/asecure_log	asecure log pathname
/etc/opt/audio/audio.sec	ASF pathname

SEE ALSO

audio(5), asecure(1M), aserver(1M), attributes(1), convert(1), send_sound(1).

Using the Audio Developer's Kit

NAME

Aserver - start the audio server

SYNOPSIS

/opt/audio/bin/Aserver -f

DESCRIPTION

The **Aserver** command starts the HP-UX Audio server, which can run on a system with audio hardware. See *Audio*(5) for information about which systems have audio hardware. The **-f** option forces the starting of the Audio server; this option is only needed if the Aserver has problems starting.

The Audio Server

Before using any audio tools such as the **Audio Editor**, the system or X station must be running two audio server processes, called **Aserver**. On a Series 700, the Remote Procedure Call daemon (**rpcd**) must also be running.

Normally, the Aserver processes and **rpcd** start automatically when the system is booted. If problems occur on an ENTRIA or ENVIZEX X station, see the X station owner's manual. On a Series 700 Audio hardware, first check if **rpcd** is running. Type the following:

```
ps -e | grep rpcd
```

If it is running, you see a line similar to the following.

```
604 ?  0:36 rpcd
```

If it is not running, see HP 9000/DCE documentation for information on restarting it. If **rpcd** is running, verify that the Aserver is running. Type:

```
ps -e | grep Aserver
```

If the Aserver is running you will see lines similar to the following, which indicate the presence of the two Aserver processes:

```
  1  ?  0:00 Aserver
224  ?  0:00 Aserver
```

If it is not running, become root and restart it as follows:

```
/opt/audio/bin/Aserver
```

If it fails to start, reissue the command with the **-f** option:

```
/opt/audio/bin/Aserver -f
```

Using Audio over the Network

From a workstation, you can also use the Audio Editor and Control Panel over the network. However, the remote system is where the actual playback and recording occur.

The local workstation (or audio client) can be any Series 700 system. The remote system (or audio server) can be a Series 700 or an X station with audio hardware and must have the Aserver processes running. If the server is a workstation, it must also allow access from remote clients (see *asecure*(1M)) and must have **rpcd** running.

To make the system an audio client, set the **AUDIO** variable by modifying the **$HOME/.vueprofile** file as follows:

Korn, Bourne, and POSIX Shells:	**AUDIO=***system_name***; export AUDIO**
C Shell:	**setenv AUDIO** *system_name*

For *system_name*, identify the workstation or X Station running the Aserver.

If the AUDIO variable is not set, the Audio Library attempts to use to the Aserver on the system defined by the DISPLAY variable. If neither DISPLAY nor AUDIO is set, the Aserver on the local machine is used.

DEPENDENCIES

The Audio Server must run on a system that has audio hardware. Note that HP-UX for the 8MB 705 System does not include audio software.

AUTHOR

The Audio Server was developed by HP.

SEE ALSO

audio(5), asecure(1M), attributes(1), convert(1), send_sound(1).

Using the Audio Developer's Kit

NAME

audevent - change or display event or system call audit status

SYNOPSIS

audevent [**-P**|**-p**] [**-F**|**-f**] [**-E**] [[**-e** *event*] ...] [**-S**] [[**-s** *syscall*] ...]

DESCRIPTION

audevent changes the auditing status of the given events or system calls. The *event* is used to specify names associated with certain self-auditing commands; *syscall* is used to select related system calls.

If neither **-P**, **-p**, **-F**, nor **-f** is specified, the current status of the selected events or system calls is displayed. If no events or system calls are specified, all events and system calls are selected.

If the **-E** option is supplied, it is redundant to specify events with the **-e** option; this applies similarly to the **-S** and **-s** options.

audevent takes effect immediately. However, the events and system calls specified are audited only when called by a user currently being audited (see *audusr*(1M)). A list of valid events and associated syscalls is provided in *audit*(5).

Only the super-user can change or display audit status.

Options

audevent recognizes the following options and command-line arguments:

-P	Audit successful events or system calls.
-p	Do not audit successful events or system calls.
-F	Audit failed events or system calls.
-f	Do not audit failed events or system calls.
-E	Select all events for change or display.
-e *event*	Select *event* for change or display.
-S	Select all system calls for change or display.
-s *syscall*	Select *syscall* for change or display.

The following is a list of the valid *events* and the associated *syscall*s (if any):

create	Object creation (**creat()**, **mkdir()**, **mknod()**, **msgget()**, **pipe()**, **semget()**, **shmat()**, **shmget()**)
delete	Object deletion (**ksem_unlink()**, **mq_unlink()**, **msgctl()**, **rmdir()**, **semctl()**, **shm_unlink()**)
readdac	Discretionary access control (DAC) information reading (**access()**, **fstat()**, **fstat64()**, **getaccess()**, **lstat()**, **lstat64()**, **stat()**, **stat64**)
moddac	Discretionary access control (DAC) modification (**chmod()**, **chown()**, **fchmod()**, **fchown()**, **fsetacl()**, **lchmod()**, **lchown()**, **putpmsg()**, **semop()**, **setacl()**, **umask()**)
modaccess	Non-DAC modification (**chdir()**, **chroot()**, **link()**, **lockf()**, **lockf64()**, **rename()**, **setgid()**, **setgroups()**, **setpgid()**, **setpgrp()**, **setre-gid()**, **setresgid()**, **setresuid()**, **setsid()**, **setuid()**, **shmctl()**, **shmdt()**, **symlink()**, **unlink()**)
open	Object opening (**execv()**, **execve()**, **ftruncate()**, **ftruncate64()**, **kload()**, **ksem_open()**, **mmap()**, **mmap64()**, **mq_open()**, **open()**, **ptrace()**, **shm_open()**, **truncate()**, **truncate64()**)
close	Object closing (**close()**, **ksem_close()**, **mq_close()**, **munmap()**)
process	Process operations (**exit()**, **fork()**, **kill()**, **mlock()**, **mlockall()**, **mun-lock()**, **munlockall()**, **nsp_init()**, **plock()**, **rtprio()**, **setcon-text()**, **setrlimit64()**, **sigqueue()**, **ulimit64()**, **vfork()**)
removable	Removable media events (**exportfs()**, **mount()**, **umount()**, **vfsmount()**)

login	Logins and logouts
admin	administrative and superuser events (**acct**(), **adjtime()**, **audctl()**, **audswitch()**, **clock_settime()**, **mpctl()**, **reboot()**, **sched_setparam()**, **sched_setscheduler()**, **serialize()**, **setaudid()**, **setaudproc()**, **setdomainname()**, **setevent()**, **sethostid()**, **setpriority()**, **setprivgrp()**, **settimeofday()**, **stime()**, **swapon()**, **toolbox()**, **utssys()**)
ipccreat	Interprocess Communication (IPC) object creation (**bind()**, **ipccreate()**, **ipcdest()**, **socket()**, **socket2()**, **socketpair()**)
ipcopen	IPC object opening (**accept()**, **connect()**, **fattach()**, **ipcconnect()**, **ipclookup()**, **ipcrecvcn()**)
ipcclose	IPC object deletion (**fdetach()**, **ipcshutdown()**, **shutdown()**)
ipcdgram	IPC datagram (**sendto()** and **recvfrom()**)
uevent1	User-defined event 1
uevent2	User-defined event 2
uevent3	User-defined event 3

AUTHOR

audevent was developed by HP.

SEE ALSO

audisp(1M), audomon(1M), audsys(1M), audusr(1M), getevent(2), setevent(2), audit(4), audit(5).

NAME

audisp - display the audit information as requested by the parameters

SYNOPSIS

audisp [**-u** *username*] [**-e** *eventname*] [**-c** *syscall*] [**-p**] [**-f**] [**-l** *ttyid*] [**-t** *start_time*] [**-s** *stop_time*] *audit_filename* ...

DESCRIPTION

audisp analyzes and displays the audit information contained in the specified *audit_filename* audit files. The audit files are merged into a single audit trail in time order. Although the entire audit trail is analyzed, **audisp** allows you to limit the information displayed, by specifying options. This command is restricted to privileged users.

Any unspecified option is interpreted as an unrestricted specification. For example, a missing **-u** *username* option causes all users' audit information in the audit trail to be displayed as long as it satisfies all other specified options. By the same principle, citing **-t** *start_time* without **-s** *stop_time* displays all audit information beginning from *start_time* to the end of the file.

audisp without any options displays all recorded information from the start of the audit file to the end.

Specifying an option without its required parameter results in error. For example, specifying **-e** without any *eventname* returns with an error message.

Options

-u *username* Specify the login name (*username*) about whom to display information. If no (*username*) is specified, **audisp** displays audit information about all users in the audit file.

-e *eventname* Display audit information of the specified event types. The defined event types are **admin**, **close**, **create**, **delete**, **ipcclose**, **ipccreat**, **ipcdgram**, **ipcopen**, **login**, **modaccess**, **moddac**, **open**, **process**, **readdac**, **removable**, **uevent1**, **uevent2**, and **uevent3** (see *audevent*(1M)).

-c *syscall* Display audit information about the specified system calls.

-p Display only successful operations that were recorded in the audit trail. No user event that results in a failure is displayed, even if *username* and *eventname* are specified.

The **-p** and the **-f** options are mutually exclusive; do not specify both on the same command line. To display both successful and failed operations, omit both **-p** and **-f** options.

-f Display only failed operations that are recorded in the audit trail.

-l *ttyid* Display all operations that occurred on the specified terminal (*ttyid*) and were recorded in the audit trail. By default, operations on all terminals are displayed.

-t *start_time* Display all audited operations occurring since *start_time*, specified as *mmddhhmm*[*yy*] (month, day, hour, minute, year). If no year is given, the current year is used. No operation in the audit trail occurring before the specified time is displayed.

-s *stop_time* Display all audited operations occurring before *stop_time*, specified as *mmddhhmm*[*yy*] (month, day, hour, minute, year). If no year is given, the current year is used. No operation in the audit trail occurring after the specified time is displayed.

AUTHOR

audisp was developed by HP.

SEE ALSO

audevent(1M), audit(4), audit(5).

NAME

audomon - audit overflow monitor daemon

SYNOPSIS

/usr/sbin/audomon [**-p** *fss*] [**-t** *sp_freq*] [**-w** *warning*] [**-v**] [**-o** *output_tty*]

DESCRIPTION

audomon monitors the capacity of the current audit file and the file system on which the audit file is located, and prints out warning messages when either is approaching full. It also checks the audit file and the file system against 2 switch points: *FileSpaceSwitch* (FSS) and *AuditFileSwitch* (AFS) and if either is reached, audit recording automatically switches to the backup audit file if it is available.

The *FileSpaceSwitch* (FSS) is specified as a percentage of the total disk space available. When the file system reaches this percentage, **audomon** looks for a backup audit file. If it is available, recording is switched from the audit file to the backup file.

The *AuditFileSwitch* (AFS) is specified (using *audsys*(1M)) by the size of the audit file. When the audit file reaches the specified size, **audomon** looks for a backup audit file. If it is available, recording is switched from the audit file to the backup file (see *audsys*(1M) for further information on use of this parameter).

If either switch point is reached but no backup file is available, **audomon** issues a warning message.

audomon is typically spawned by **/sbin/init.d/auditing** (as part of the *init*(1M) start-up process) when the system is booted up. Once invoked, **audomon** monitors, periodically sleeping and "waking up" at intervals. Note that **audomon** does not produce any messages when the audit system is disabled.

audomon is restricted to privileged users.

Options

-p *fss* Specify the *FileSpaceSwitch* by a number ranging from 0 to 100. When the audit file's file system has less than *fss* percent free space remaining, **audomon** looks for a backup file. If available, the backup file is designated as the new audit file. If no backup file is available, **audomon** issues a warning message.

The *fss* parameter should be a larger number than the *min_free* parameter of the file system to ensure that the switch takes place before *min_free* is reached. By default, *fss* is 20 percent.

-t *sp_freq* Specify the wake-up switch-point frequency in minutes. The wake-up frequency at any other time is calculated based on *sp_freq* and the current capacity of the audit file and the file system. The calculated wake-up frequency at any time before the switch points is larger than *sp_freq*. As the size of the audit file or the file system's free space approaches the switch points, the wake-up frequency approaches *sp_freq*. *sp_freq* can be any positive real number. Default *sp_freq* is 1 (minute).

-w *warning* Specify that warning messages be sent before the switch points. *warning* is an integer ranging from 0 through 100. The higher the *warning*, the closer to the switch points warning messages are issued. For example, *warning* = 50 causes warning messages to be sent half-way before the switch points are reached. *warning* = 100 causes warning messages to be sent only after the designated switch points are reached and a switch is not possible due to a missing backup file. By default, *warning* is 90.

-v Make audomon more verbose. This option causes **audomon** to also print out the next wake-up time.

-o *output_tty* Specify the tty to which warning messages are directed. By default, warning messages are sent to the console. Note that this applies only to the diagnostic messages **audomon** generates concerning the status of the audit system. Error messages caused by wrong usage of **audomon** are sent to the standard output (where **audomon** is invoked).

AUTHOR

audomon was developed by HP.

SEE ALSO

audsys(1M), audit(5).

NAME

audsys - start or halt the auditing system and set or display audit file information

SYNOPSIS

audsys [**-nf**] [**-c** *file* **-s** *cafs*] [**-x** *file* **-z** *xafs*]

DESCRIPTION

audsys allows the user to start or halt the auditing system, to specify the auditing system "current" and "next" audit files (and their switch sizes), or to display auditing system status information. This command is restricted to super-users.

The "current" audit file is the file to which the auditing system writes audit records. When the "current" file grows to either its Audit File Switch (AFS) size or its File Space Switch (FSS) size (see *audomon*(1M)), the auditing system switches to write to the "next" audit file. The auditing system switches audit files by setting the "current" file designation to the "next" file and setting the new "next" file to NULL. The "current" and "next" files can reside on different file systems.

When invoked without arguments, *audsys* displays the status of the auditing system. This status includes information describing whether auditing is on or off, the names of the "current" and "next" audit files, and a table listing their switch sizes and the sizes of file systems on which they are located, as well as the space available expressed as a percentage of the switch sizes and file system sizes.

Options

audsys recognizes the following options:

-n	Turn on the auditing system. The system uses existing "current" and "next" audit files unless others are specified with the **-c** and **-x** options. If no "current" audit file exists (such as when the auditing system is first installed), specify it by using the **-c** option.
-f	Turn off the auditing system. The **-f** and **-n** options are mutually exclusive. Other options specified with **-f** are ignored.
-c *file*	Specify a "current" file. Any existing "current" file is replaced with the *file* specified; the auditing system immediately switches to write to the new "current" file. The specified *file* must be empty or nonexistent, unless it is the "current" or "next" file already in use by the auditing system.
-s *cafs*	Specify *cafs*, the "current" audit file switch size (in kbytes).
-x *file*	Specify the "next" audit file. Any existing "next" file is replaced with the *file* specified. The specified *file* must be empty or nonexistent, unless it is the "current" or "next" file already in use by the auditing system.
-z *xafs*	Specify *xafs*, the "next" audit file switch size (in kbytes).

If **-c** but not **-x** is specified, only the "current" audit file is changed; the existing "next" audit file remains. If **-x** but not **-c** is specified, only the "next" audit file is changed; the existing "current" audit file remains.

The **-c** option can be used to manually switch from the "current" to the "next" file by specifying the "next" file as the new "current" file. In this instance, the file specified becomes the new "current" file and the "next" file is set to NULL.

In instances where no next file is desired, the **-x** option can be used to set the "next" file to NULL by specifying the existing "current" file as the new "next" file.

The user should take care to select audit files that reside on file systems large enough to accomodate the Audit File Switch (AFS) desired. *audsys* returns a non-zero status and no action is performed, if any of the following situations would occur:

The Audit File Switch size (AFS) specified for either audit file exceeds the space available on the file system where the file resides.

The AFS size specified for either audit file is less than the file's current size.

Either audit file resides on a file system with no remaining user space (exceeds minfree).

AUTHOR

audsys was developed by HP.

FILES

/.secure/etc/audnames File maintained by **audsys** containing the "current" and "next" audit file names and their switch sizes.

SEE ALSO

audit(5), audomon(1M), audctl(2), audwrite(2), audit(4).

NAME

audusr - select users to audit

SYNOPSIS

audusr [[**-a** *user*] ...] [[**-d** *user*] ...] [**-A**|**-D**]

DESCRIPTION

audusr is used to specify *user*s to be audited or excluded from auditing. If no arguments are specified, **audusr** displays the audit setting of every user. **audusr** is restricted to super-users.

Options

audusr recognizes the following options:

-a *user*	Audit the specified *user*. The auditing system records audit records to the "current" audit file when the specified *user* executes audited events or system calls. Use **audevent** to specify events to be audited (see *audevent*(1M)).
-d *user*	Do not audit the specified *user*.
-A	Audit all users.
-D	Do not audit any users.

The **-A** and **-D** options are mutually exclusive: that is, if **-A** is specified, **-d** cannot be specified; if **-D** is specified, **-a** cannot be specified.

Users specified with **audusr** are audited (or excluded from auditing) beginning with their next login session, until excluded from auditing (or specified for auditing) with a subsequent **audusr** invocation. Users already logged into the system when **audusr** is invoked are unaffected during that login session; however, any user who logs in after **audusr** is invoked is audited or excluded from auditing accordingly.

AUTHOR

audusr was developed by HP.

FILES

/tcb/files/auth/*/*	File containing flags to indicate whether users are audited.

SEE ALSO

audevent(1M), setaudproc(2), audswitch(2), audwrite(2). audit(5).

NAME

authck - check internal consistency of Authentication database

SYNOPSIS

authck [**-p**] [**-t**] [**-a**] [**-v**] [**-d** [*domainname*]]

DESCRIPTION

authck checks both the overall structure and internal field consistency of all components of the Authentication database. It reports all problems it finds. Only users who have the *superuser* capability can run this command. When **pwck** is used with the **-s** option, **authck** is run with the **-p** option automatically.

Options

authck recognizes the following options and tests:

-p Check the Protected Password database. The Protected Password database and **/etc/passwd** are checked for completeness such that neither contains entries not in the other. The cross references between the Protected Password database and **/etc/passwd** are checked to make sure that they agree. However, if **Nis+** is configured in your system, the password table is also checked before reporting a discrepancy. This means that a discrepancy would not be reported for a user that does NOT exist in **/etc/passwd** but exists in the Protected Password database as well as the **Nis+ passwd** table. Fields in the Protected Password database are then checked for reasonable values. For example, all time stamps of past events are checked to make sure that they have times less than the times returned by *time*(2).

-t Fields in the Terminal Control database are checked for reasonable values. All time stamps of past events are checked to make sure they have times less than those returned by *time*(2).

-a Shorthand equivalent of using the **-p** and **-t** options together in a single command.

-v Provide running diagnostics as the program proceeds. Produce warnings when unusual conditions are encountered that might not cause program errors in *login*, *password* and *su* programs.

-d Removes Protected Password database entries that are not found in the **Nis+ passwd** table. **Nis+** users may have an entry in the Protected database and not in **/etc/passwd**. Thus, this option removes orphaned Protected database entries: orphaned entries can exist for deleted **Nis+** users. The optional *domainname* specifies the desired **Nis+** domain to use for the **passwd** table. If *domainname* is not specified, the local domain name is used.

FILES

/etc/passwd	System password file
/tcb/files/auth/*/*	Protected Password database
/tcb/files/ttys	Terminal Control database
/tcb/files/auth/system/default	System Defaults database
/usr/sbin/authck	

AUTHOR

SecureWare Inc.

SEE ALSO

getprpwent(3), getprtcent(3), getprdfent(3), authcap(4).

NAME

auto_parms - initial system configuration/DHCP support script

SYNOPSIS

auto_parms

DESCRIPTION

auto_parms is a system initialization script whose primary responsibility lies in handling first time boot configuration and ongoing management of the DHCP lease(s). **auto_parms** is invoked at boot time by the **/sbin/rc** script. Initially, it will load a list of available ethernet interfaces and begin requesting a DHCP lease on each interface, stopping when a valid lease is secured or the list is exhausted.

As a part of checking for the availability of a lease on a particular interface, **auto_parms** will also consult **/etc/rc.config.d/netconf** and examine the variable DHCP_ENABLE[index]. If DHCP_ENABLE[index] is set to ’0’, **auto_parms** will not attempt to request a lease on the the interface designated by ’index’. If DHCP_ENABLE[index] does not exist in **/etc/rc.config.d/netconf**, **auto_parms** will assume that it can attempt the DHCP request over the interface.

Once a lease is secured, the information supplied with the lease will be used to initialize key networking parameters (see *dhcpdb2conf*(1M)).

If **auto_parms** detects that the system is going through a "first time boot" (keyed by the hostname for the system not being set), it will invoke **set_parms** for the purpose of verifying the DHCP supplied parameters as well as collecting any parameters not supplied by DHCP.

For all subsequent boots, the data supplied by a DHCP lease is assumed to be definitive and will be recognized as such by **auto_parms**. Note that in an environment (non-mobile) where DHCP is being used for IP address management, the lease information will not change from boot to boot under normal conditions. This is accomplished by **auto_parms** ensuring that the **dhcpclient** is placed in "lease maintenance mode" prior to exiting.

FILES

/sbin/auto_parms
/sbin/set_parms.util

EXAMPLES

See **/sbin/rc** for invocation context

SEE ALSO

dhcpdb2conf(1M).

NAME

automount - automatically mount NFS file systems

SYNOPSIS

automount [**-nTv**] [**-D** *name* **=** *value*] [**-f** *master-file*] [**-M** *mount-directory*] [**-tl** *duration*] [**-tm** *interval*] [**-tw** *interval*] [*directory map* [*-mount-options*]] ...

DESCRIPTION

automount is a daemon that automatically and transparently mounts NFS file systems as needed. It monitors attempts to access directories that are associated with an **automount** map, along with any directories or files that reside under them. When a file is to be accessed, the daemon mounts the appropriate NFS file system. Maps can be assigned to a directory by using an entry in a direct **automount** map, or by specifying an indirect map on the command line.

automount interacts with the kernel in a manner closely resembling an NFS server:

- **automount** uses the map to locate an appropriate NFS file server, exported file system, and mount options.
- It then mounts the file system in a temporary location, and replaces the file system entry for the directory or subdirectory with a symbolic link to the temporary location.
- If the file system is not accessed within an appropriate interval (five minutes by default), the daemon unmounts the file system and removes the symbolic link.
- If the specified directory has not already been created, the daemon creates it, and then removes it upon exiting.

Since name-to-location binding is dynamic, updates to an **automount** map are transparent to the user. This obviates the need to mount shared file systems prior to running applications that contain internally hard-coded references to files.

If the dummy directory (**/-**) is specified, **automount** treats the *map* argument that follows as the name of a direct map. In a direct map, each entry associates the full path name of a mount point with a remote file system to mount.

If the *directory* argument is a path name, the *map* argument points to an indirect map. An indirect map, contains a list of the subdirectories contained within the indicated *directory*. With an indirect map, it is these subdirectories that are mounted automatically.

A map can be a file or a NIS/NIS+ map; if a file, the *map* argument must be a full path name.

The **-***mount-options* argument, when supplied, is a comma-separated list of options to the **mount** command (see *mount*(1M)) preceded by a **-**. However, any conflicting mount options specified in the indicated map take precedence.

Options

automount recognizes the following options:

-m Option not supported.

-n Disable dynamic mounts. With this option, references through the **automount** daemon succeed only when the target filesystem has been previously mounted. This can be used to prevent NFS servers from cross-mounting each other.

-T Trace. Expand each NFS call and log it in **/var/adm/automount.log** file.

-v Verbose. Log status messages to the system log file (see *syslogd*(1M)).

-D *envar* **=** *value*
Assign *value* to the indicated **automount** (environment) variable *envar*.

-f *master-file* Read the local **master_file** before reading **auto_master** map.

-M *mount-directory*
Mount temporary file systems in the named directory instead of in **/tmp_mnt**.

-tl *duration* Specify a *duration* (in seconds) that a file system is to remain mounted when not in use. The default is 5 minutes.

-tm *interval* Specify an *interval* (in seconds) between attempts to mount a filesystem. The default is 30 seconds.

-tw *interval* Specify an *interval* (in seconds) between attempts to unmount filesystems that have exceeded their cached times. The default is 1 minute.

Environment Variables

Environment variables can be used within an **automount** map. For example, if **$HOME** appears within a map, **automount** expands it to the current value of the **HOME** environment variable.

To protect a reference from affixed characters, surround the variable name with curly braces. Environment variables cannot appear as the key entry in maps.

EXAMPLES

Map Entry Format

A simple map entry (mapping) takes the form:

directory [**-** *mount-options*] *location* ...

where *directory* is the full path name of the directory to mount, when used in a direct map, or the basename of a subdirectory in an indirect map. *mount-options* is a comma-separated list of **mount** options, and *location* specifies a remote filesystem from which the directory may be mounted. In the simple case, *location* takes the form:

host **:** *pathname*

Multiple *location* fields can be specified, in which case **automount** pings all servers in the list and then selects the first host that responds to serve that mount point.

If *location* is specified in the form:

host **:** *path* **:** *subdir*

host is the name of the host from which to mount the file system, *path* is the path name of the directory to mount, and *subdir*, when supplied, is the name of a subdirectory to which the symbolic link is made. This can be used to prevent duplicate mounts when multiple directories in the same remote file system might be accessed. Assume a map for **/home** resembling:

```
mike        hpserver1:/home/hpserver1:mike
dianna      hpserver1:/home/hpserver1:dianna
```

Attempting to access a file in **/home/mike** causes **automount** to mount **hpserver1:/home/hpserver1** and creates a symbolic link called **/home/mike** to the **mike** subdirectory in the temporarily-mounted filesystem. A subsequent file access request in **/home/dianna** results in **automount** simply creating a symbolic link that points to the **dianna** subdirectory because **/home/hpserver1** is already mounted. Given the map:

```
mike        hpserver1:/home/hpserver1/mike
dianna      hpserver1:/home/hpserver1/dianna
```

automount would have to mount the filesystem twice.

A mapping can be continued across input lines by escaping the newline character with a backslash (****). Comments begin with a **#** and end at the subsequent newline character.

Directory Pattern Matching

The **&** character is expanded to the value of the *directory* field for the entry in which it occurs. Given an entry of the form:

```
mike        hpserver1:/home/hpserver1:&
```

the **&** expands to **mike**.

The ***** character, when supplied as the *directory* field, is recognized as the catch-all entry. Such an entry resolves to any entry not previously matched. For example, if the following entry appeared in the indirect map for **/home**:

```
*   &:/home/&
```

this would allow automatic mounts in **/home** of any remote file system whose location could be specified as:

hostname **:/home** *hostname*

Hierarchical Mappings

A hierarchical mapping takes the form:

directory [**/** [*subdirectory*] [**-** *mount-options*] *location* ...] ...

The initial **/** within the **/**[*subdirectory*] is required; the optional *subdirectory* is taken as a file name relative to the *directory*. If *subdirectory* is omitted in the first occurrence, the **/** refers to the directory itself.

Given the direct map entry:

```
/usr/local    \
    /     -ro,intr   shasta:/usr/local       ranier:/usr/local     \
    /bin  -ro,intr   ranier:/usr/local/bin   shasta:/usr/local/bin \
    /man  -ro,intr   shasta:/usr/local/man   ranier:/usr/local/man
```

automount automatically mounts **/usr/local**, **/usr/local/bin**, and **/usr/local/man**, as needed, from either **shasta** or **ranier**, whichever host responded first.

Direct Maps

A direct map contains mappings for any number of directories. Each directory listed in the map is automatically mounted as needed. The direct map as a whole is not associated with any single directory.

Indirect Maps

An indirect map allows specifying mappings for the subdirectories to be mounted under the *directory* indicated on the command line. It also obscures local subdirectories for which no mapping is specified. In an indirect map, each *directory* field consists of the basename of a subdirectory to be mounted as needed.

Included Maps

The contents of another map can be included within a map with an entry of the form:

+*mapname*

mapname can either be a file name, or the name of an NIS/NIS+ map, or one of the special maps described below. If *mapname* begins with a slash then it is assumed to be the pathname of a local file. Otherwise the location of the map is determined by the policy of the name service switch according to the entry for the automounter in **/etc/nsswitch.conf**, such as

automount: *files* **nis**

If the name service is *files* then the name is assumed to be that of a local file in **/etc**. If the key being searched for is not found in the included map, the search continues with the next entry.

Special Maps

Three special maps, **-hosts**, **-passwd**, and **-null**, are currently available: The **-hosts** map uses the **gethostbyname()** map to locate a remote host when the hostname is specified (see *gethostent*(3C)). This map specifies mounts of all exported file systems from any host. For example, if the following **automount** command is already in effect:

```
automount /net -hosts
```

a reference to **/net/hermes/usr** initiates an automatic mount of all file systems from **hermes** that **automount** can mount, and any subsequent references to a directory under **/net/hermes** refer to the corresponding directory on **hermes**. The **-passwd** map uses the *passwd*(4) database to attempt to locate a user's home directory. For example, if the following **automount** command is already in effect:

```
automount /homes -passwd
```

if the home directory for a user has the form **/***dir***/***server***/***username*, and *server* matches the host system on which that directory resides, **automount** mounts the user's home directory as: **/homes** */username*.

For this map, the tilde character (˜) is recognized as a synonym for *username*.

The **-null** map, when indicated on the command line, cancels a previous map for the directory indicated. It can be used to cancel a map given in **auto_master**.

Configuration and the auto_master Map

automount normally consults the **auto_master** configuration map for a list of initial **automount** maps, and sets up automatic mounts for them in addition to those given on the command line. If there are duplications, the command-line arguments take precedence. This configuration database contains arguments to the **automount** command rather than mappings.

Maps given on the command line, or those given in a local master file specified with **-f** override those in the **auto_master** map. For example, given the command:

```
automount /homes /etc/auto.homes /- /etc/auto.direct
```

and the master map file **auto_master** containing:

```
/homes -passwd
```

automount mounts home directories using the **/etc/auto.homes** map instead of the special **-passwd** map in addition to the various directories specified in the **/etc/auto.direct** map.

WARNINGS

Do not send the **SIGKILL** signal (**kill -9**, or **kill -KILL**) to the **automount** daemon. Doing so causes any processes accessing mount directories served by **automount** to hang. A system reboot may be required to recover from this state.

Do not start an **automount** daemon while another is still running. If restarting **automount**, make sure the first daemon and all of its children are not running.

When **automount** receives signal **SIGHUP**, it rereads the **/etc/mnttab** file to update its internal record of currently mounted file systems. If a file system mounted by **automount** is unmounted by a **umount** command, **automount** should be forced to reread the file by sending the **SIGHUP** signal (see *kill*(1)).

Shell file name expansion does not apply to objects not currently mounted.

Since **automount** is single-threaded, any request that is delayed by a slow or nonresponding NFS server delays all subsequent automatic mount requests until it completes.

Programs that read **/etc/mnttab** and then touch files that reside under automatic mount points introduce further entries to the file.

Automatically-mounted file systems are mounted with type **ignore**; they do not appear in the output of either *mount*(1M), or *bdf*(1M).

FILES

/tmp_mnt	directory under which filesystems are dynamically mounted
/etc/mnttab	mount table
/etc/nsswitch.conf	the name service switch configuration file.

SEE ALSO

mount(1M), bdf(1M), passwd(4).

a

NAME

autopush - manage system database of automatically pushed STREAMS modules

SYNOPSIS

autopush -f *file*

autopush -g -M *major* **-m** *minor*

autopush -r -M *major* **-m** *minor*

DESCRIPTION

autopush manages the system database that is used for automatic configuration of STREAMS devices. The command is used in three different ways as dictated by the **-f**, **-g**, and **-r** command-line options described below.

Options

autopush recognizes the following command-line options and arguments:

-f *file* Using the configuration information contained in *file*, load the system database with the names of the STREAMS devices and a list of modules to use for each device. When a device is subsequently opened, the HP-UX STREAMS subsystem pushes the modules onto the stream for the device.

file must contain one or more lines of at least four fields separated by a space as shown below:

major minor lastminor module1 module2 ... moduleN

The first field *major* can be either an integer or a device name. The device name is the name for the device used in the **master** file. The next two fields are integers. If *minor* is set to −1, then all minor devices for the specified *major* are configured and *lastminor* is ignored. If *lastminor* is 0, then only a single minor device is configured. To configure a range of minor devices for a major device, *minor* must be less then *lastminor*. The remaining field(s) list one or more module names. Each module is pushed in the order specified. A maximum of eight modules can be pushed. Any text after a # character in *file* is treated as a comment for that line only.

This option is also used to restore device configuration information previously removed by **autopush -r**. However, when used in such a manner, the entire database is restored, not just the information that was previously removed.

-g -M *major* **-m** *minor*
Display current configuration information from the system database for the STREAMS device specified by the *major* device number (or device name for the device from the **master** file) and *minor* number.

If a range of minors has been previously configured then **autopush -g** returns the configuration information for the first minor in the range, in addition to other information.

-r -M *major* **-m** *minor*
Remove configuration information from the system database for the STREAMS device specified by the *major* device number (or device name for the device from the **master** file and *minor* number. Removal is performed on the database only, not on the original configuration file. Therefore, the original configuration can be restored by using the **-f** *file* option. To permanently exclude a STREAMS device from the database, its information must be removed from the configuration file.

If *minor* matches the first minor of a previously configured range then **autopush -r** removes the configuration information for the entire configured range.

EXAMPLES

If the file **/tmp/autopush.example** contains:

```
75 -1 0 modA modB
test 0 5 modC modA
```

Then **autopush -f /tmp/autopush.example** will cause **modA** and **modB** to be pushed whenever major device **# 75** is opened, and **modC** and **modA** to be pushed for the first six opens of device

test.

This next example lists information about the stream for major device **75** and its minor device **-2**:

```
autopush -g -M 75 -m -2
```

FILES

/usr/lib/nls/msg/C/autopush.cat	NLS catalog for **autopush**.

SEE ALSO

sad(7), streamio(7).

NAME

backup - backup or archive file system

SYNOPSIS

/usr/sbin/backup [**-A**] [*-archive*] [**-fsck**]

DESCRIPTION

The *backup* command uses *find*(1) and *cpio*(1) to save a *cpio* archive of all files that have been modified since the modification time of **/var/adm/archivedate** on the default tape drive (**/dev/update.src**). *backup* should be invoked periodically to ensure adequate file backup.

The **-A** option suppresses warning messages regarding optional access control list entries. *backup*(1M) does not backup optional access control list entries in a file's access control list (see *acl*(5)). Normally, a warning message is printed for each file having optional access control list entries.

The **-archive** option causes *backup* to save all files, regardless of their modification date, and then update **/var/adm/archivedate** using *touch*(1).

backup prompts you to mount a new tape and continue if there is no more room on the current tape. Note that this prompting does not occur if you are running *backup* from *cron*(1M).

The **-fsck** option causes *backup* to start a file system consistency check (without correction) after the backup is complete. For correct results, it is important that the system be effectively single-user while *fsck* is running, especially if **-fsck** is allowed to automatically fix whatever inconsistencies it finds. *backup* does not ensure that the system is single-user.

You can edit **/usr/sbin/backup** to customize it for your system. For example, *backup* uses *tcio*(1) with *cpio* to back up files on an HP CS/80 disc drive's streaming tape. You must modify *backup* to use *cpio*(1) if you want to access a standard HP Tape Drive.

Several local values are used that can be customized:

BACKUPDIRS	specifies which directories to back up recursively (usually **/**, meaning all directories);
BACKUPLOG	file name where start and finish times, block counts, and error messages are logged;
ARCHIVE	file name whose date is the date of the last archive;
REMIND	file name that is checked by **/etc/profile** to remind the next person who logs in to change the backup tape;
FSCKLOG	file name where start and finish times and *fsck* output is logged.

You may want to make other changes, such as whether or not *fsck* does automatic correction (according to its arguments), where *cpio* output is directed, other information logging, etc.

In all cases, the output from *backup* is a normal *cpio* archive file (or volume) which can be read using *tcio* and *cpio* with the **c** option.

File Recovery

backup creates archive tapes with all files and directories specified relative to the root directory. When recovering files from an archive tape created by *backup*, you should be in the root directory and specify the directory path names for recovered files relative to the root directory (**/**). When specifying the directory path name for file recovery by *tcio* and *cpio*, do not precede the leading directory name with a slash. If you prefer, you can also use *cpio* with a **-t** option to determine how files and directories are named on the archive tape before attempting recovery.

WARNINGS

Refer to WARNINGS in *cpio*(1).

When *cpio* runs out of tape, it sends an error to standard error and demands a new special file name from **/dev/tty**.

To continue, rewind the tape, mount the new tape, type the name of the new special file at the system console, and press **Return**.

If *backup* is being run unattended from *cron*(1M) and the tape runs out, *backup* terminates, leaving the *find* process still waiting. Kill this process when you return.

FILES

/var/adm/archivedate parameterized file names

SEE ALSO

cpio(1), find(1), tcio(1), touch(1), cron(1M), fbackup(1M), frecover(1M), fsck(1M), acl(5).

NAME

bdf - report number of free disk blocks (Berkeley version)

SYNOPSIS

/usr/bin/bdf [**-b**] [**-i**] [**-l**] [**-t** *type* | [*filesystem*|*file*] ...]

DESCRIPTION

The **bdf** command displays the amount of free disk space available either on the specified *filesystem* (**/dev/dsk/c0d0s0**, for example) or on the file system in which the specified *file* (such as **$HOME**), is contained. If no file system is specified, the free space on all of the normally mounted file systems is printed. The reported numbers are in kilobytes.

Options

The **bdf** command recognizes the following options:

-b	Display information regarding file system swapping.
-i	Report the number of used and free inodes.
-l	Display information for local file systems only (for example, HFS and CDFS file systems).
-t *type*	Report on the file systems of a given *type* (for example, **nfs** or **hfs**).

RETURN VALUE

The **bdf** command returns 0 on success (able to get status on all file systems), or returns 1 on failure (unable to get status on one or more file systems).

WARNINGS

If file system names are too long, the output for a given entry is displayed on two lines.

The **bdf** command does not account for any disk space reserved for swap space, or used for the HFS boot block (8 KB, 1 per file system), HFS superblocks (8 KB each, 1 per disk cylinder), HFS cylinder group blocks (1 KB - 8 KB each, 1 per cylinder group), and inodes (currently 128 bytes reserved for each inode). Non-HFS file systems may have other items not accounted for by this command.

AUTHOR

bdf was developed by the University of California, Berkeley.

FILES

/etc/fstab	Static information about the file systems.
/etc/mnttab	Mounted file system table.
/dev/dsk/*	File system devices.

SEE ALSO

df(1M), fstab(4), mnttab(4).

NAME

boot - bootstrap process

DESCRIPTION

The Series 700 and 800 bootstrap process involves the execution of three software components:

- **pdc** (see *pdc*(1M),
- **isl** (see *isl*(1M), and
- **hpux**.

After the processor is RESET, **pdc,** the **processor-dependent code** (firmware), performs a self-test and initializes the processor. It then loads and transfers control to **isl**, the operating-system-independent **initial system loader**. **isl**, in turn, loads and transfers control to the **hpux** utility, the HP-UX-specific bootstrap loader. **hpux** then downloads the HP-UX kernel object file from an HP-UX file system and transfers control to the loaded kernel image.

SEE ALSO

hpux(1M), isl(1M), pdc(1M).

NAME

bootpd - Internet Boot Protocol server

SYNOPSIS

/usr/lbin/bootpd [**-d** *debuglevel*] [**-s**] [**-t** *timeout*] [*configfile* [*dumpfile*]]

DESCRIPTION

The **bootpd** daemon implements three functions: a Dynamic Host Configuration Protocol (DHCP) server as defined in RFC1541, an Internet Boot Protocol (BOOTP) server as defined in RFC951 and RFC1395, and a DHCP/BOOTP relay agent as defined in RFC1542.

bootpd can be run through **inetd** (see *inetd*(1M)), or as a stand-alone daemon. It is run by **/etc/inetd** when the following line (or equivalent) is included in the file **/etc/inetd.conf**:

```
bootps dgram udp wait root /usr/lbin/bootpd bootpd
```

bootpd starts when a boot request arrives. If it has not received another boot request after 15 minutes, **bootpd** exits. The **-t** option can be used to specify a different timeout value in minutes (such as **-t20**). With a timeout value of zero (**-t0**), **bootpd** never exits.

To run **bootpd** as a stand-alone daemon, invoke it with the **-s** option. This might be the desired mode of operation for large network installations with many DHCP/BOOTP clients. With the **-s** option, the **-t** option has no effect, since **bootpd** never exits.

The **-d** option sets the verbosity level (1–3) of the logging emitted by the daemon via **syslog** (see *syslog*(3C)).

When **bootpd** receives a DHCP/BOOTP request, it checks whether the client information is in the **/etc/bootptab** database. If the client information is available, **bootpd** sends back the reply. Otherwise, it checks whether there is any matched relay information for the client in the **/etc/bootptab** database. If so, **bootpd** goes through a series of checks to see if it should relay the request. If no matched relay information was found, **bootpd** checks whether the client information is matched by a pool or device group in the **/etc/dhcptab** database. If a match is found, **bootpd** sends back a reply. The request is dropped if no matched group information is found.

To replay to a DCHP or BOOTP request the server puts together a BOOTREPLY message and does a number of checks to ensure the message is sent to the correct destination.

bootpd first checks the **ciaddr** (client IP address) field of the DHCP/BOOTP packet. If this field is nonzero, the BOOTREPLY message is sent to the IP address identified in **ciaddr**.

If the **ciaddr** field is zero, **bootpd** checks the **giaddr** field. If this field is not zero, **bootpd** sends the BOOTREPLY message to the *relay agent* specified in **giaddr** field and the *relay agent* delivers the BOOTREPLY message to the client. If the **giaddr** field is zero, **bootpd** sends the BOOTREPLY message to the client. In both cases, the BOOTREPLY will either be sent to the IP address specified in the **yiaddr** (your IP address) field or as a broadcast message. On HP-UX, there are two ways to specify that the BOOTREPLY should be sent as a broadcast message.

1. The client sets the broadcast flag bit in the **flags** field (bit 0) of the DHCP/BOOTP request packet.
2. Define the **ba** tag in the **bootptab** file (see "Tags for client entries" below)

For the case where the **bootpd** has matched a relay entry in **/etc/bootptab**, it attempts to forward the request to the configured DHCP/BOOTP server.

bootpd first checks whether the relay function is enabled for the requesting client. The relay capability is configurable. If the relay function is disabled, then the request packet is dropped.

Before **bootpd** relays the request, it also examines the **giaddr** (gateway IP address) field. The client sets the **giaddr** field to zero when it sends out the request. If the relay agent finds this field is zero, it fills this field with the primary IP address of the interface on which the request was received; otherwise, the relay agent does not change this field. Then **bootpd** increments the value of the **hops** field, and relays the request to the DHCP/BOOTP servers that have been configured for this client.

If the relay function is enabled for this client, **bootpd** checks the **hops** field of the DHCP/BOOTP request packet. The client sets the **hops** field to 0 when it sends out the DHCP/BOOTP request. The **hops** value is increased every time the request packet is relayed by a relay agent. The maximum hop number can be configured. The maximum possible hop number allowed is 16. The default maximum is set to 4. The request packet is dropped if the hop value exceeds the configured maximum.

Then **bootpd** compares the value of the **secs** (seconds since the client began booting) field of the DHCP/BOOTP packet to the **threshold** value. The client sets the **secs** field to zero when it first sends out the request. The client repeats the request if it does not receive a reply. When the client repeats the request, it sets the **secs** value to the number of seconds since the first request was sent. **bootpd** does not relay the request if the value of the **secs** field is less than the **threshold** value. The **threshold** value can be configured. The default value is 0.

Configuration

Upon startup, **bootpd** reads its configuration files to build its internal database, then listens for boot request packets. The default configuration files are, **/etc/bootptab**, and **/etc/dhcptab**. The **bootptab** file can be specified in the command line. **bootpd** rereads its configuration file when it receives a hangup signal, **SIGHUP**, or when it receives a boot request packet and detects that the configuration file has been updated. If hosts are added, deleted, or modified, their entries in the **bootpd** internal database are updated accordingly when the configuration files are reread.

If **bootpd** receives a **SIGUSR1** signal, it dumps its memory-resident database to the file **/var/tmp/bootpd.dump** or the *dumpfile* specified in the command line.

The configuration file can contain two types of host entries:

1. The client entries, which contains the client information.
2. The relay entries, which contains the configuration to relay DHCP/BOOTP requests for one or more clients.

The configuration uses two-character, case-sensitive tag symbols to represent host parameters. These parameter declarations are separated by colons (**:**). The general format is:

*hostname***:***tg***=***value***:**...**:***tg***=***value***:**...**:***tg***=***value***:**...

where *hostname* is the actual name of a DHCP/BOOTP client in the client entries, and in the case of a relay entry, it can be the actual name of a client if it is an individual relay entry, or it can be a name for a group of clients if it is a group relay entry. *tg* is a two-character tag symbol. Most tags must be followed by an equals-sign, and a value as above. Some can appear in a boolean form with no value (that is, **:***tg***:**).

Blank lines and lines beginning with **#** are ignored in the configuration file. Host entries are separated from one another by newlines; a single host entry can be extended over multiple lines if the lines end with a backslash (****). It is also acceptable for lines to be longer than 80 characters. Tags can appear in any order with the following exceptions: The host name must be the very first field in an entry, and the hardware type tag, **ht**, must precede the hardware address tag, **ha**. and the hardware mask tag, **hm**.

IP addresses are specified in standard Internet dot notation, and can use decimal, octal, or hexadecimal numbers (octal numbers begin with **0**, hexadecimal numbers begin with **0x** or **0X**). Certain tags accept a list of one or more IP addresses (*ip_address_list*). When more than one IP address is listed, the addresses must be separated by whitespace.

The types of tags can be grouped into three categories:

1. The tags that can be used for both the client and the relay entries.
2. The tags that can only be used in the relay entries.
3. The tags that can only be used in the client information entries.

Tag **ip** is used to differentiate a client entry from a relay entry. An entry with tag **ip** defined is treated as a client entry. A relay entry can contain the relay configuration for an individual client, also a hardware address mask mechanism is provided to configure the relay entry for a group of clients. The group client relay entries are kept in a linear sorted table by **bootpd**. When a client does not have an individual relay specification, the linear table is searched to see if there is a match for the client. If there are multiple matched entries in the sorted table, only the first one is used. Tag **hm** is used to differentiate an individual client relay entry from a group relay entry. The linear sorted table is sorted on the value of tag **hm**. The search and match mechanism is explained in the discussion of tag **hm**.

Tags for both kinds of entries

ha=*hardware-address*
This tag specifies the hardware address of the client. The *hardware address* must be specified in hexadecimal; optional periods and/or a leading **0x** can be included for readability. The **ha** tag must be preceded by the **ht** tag (either explicitly or implicitly; see **tc** below).

ht=*hardware-type*
: This tag specifies the hardware type code. *hardware-type* can be an unsigned decimal, octal, or hexadecimal integer corresponding to one of the ARP Hardware Type codes specified in RFC1010. It can also be specified by the symbolic names **ethernet** or **ether** for 10-Mb Ethernet; **ethernet3** or **ether3** for 3-Mb experimental Ethernet; **ieee802**, **tr**, or **token-ring** for IEEE 802 networks; **pronet** for Proteon ProNET Token Ring; **chaos**, and **arcnet**, for Chaos and ARCNET, respectively.

tc=*template-host*
: This tag indicates a table continuation. Often, many host entries share common values for certain tags (such as domain servers, etc.). Rather than repeatedly specifying these tags, a full specification can be listed for one host entry and shared by others via the **tc** mechanism.

The *template-host* is a dummy host that does not actually exist and never sends boot requests. Information explicitly specified for a host always overrides information implied by a **tc** tag symbol. The value of *template-host* can be the host name or IP address of any host entry previously listed in the configuration file.

Sometimes it is necessary to delete a specific tag after it has been inferred via **tc**. This can be done using the construction *tag***@** which removes the effect of *tag*. For example, to completely undo an RFC1034 domain name server specification, use **:ds@:** at an appropriate place in the configuration entry. After removal with **@**, a tag is eligible to be set again through the **tc** mechanism.

Tags for relay entries

bp=*bootp-servers*
: This tag specifies the BOOTP servers that DHCP/BOOTP requests will be relayed to. The value of *bootp-servers* can be one or more individual IP addresses, and/or one or more network broadcast addresses. A relay entry with this tag configured indicates that the relay function is on for the clients specified in this entry. A relay entry missing this symbol means that the relay function is off for the clients specified in this entry.

th=*threshold*
: This tag specifies the *threshold* value in seconds for the entry. The default value is 0.

hp=*hops*
: This tag specifies the maximum *hops* value. If the *hops* value exceeds 16, it is set to 16. The default value is 4.

hm=*hardware-address-mask*
: This tag specifies the mask for the hardware address **ha**. *hardware-address-mask* must be specified in hexadecimal. An optional leading **0x** can be included for readability. The **hm** tag must be preceded by the **ht** tag (either explicitly or implicitly; see **tc** above). Each **0** bit in **hm** specifies that the corresponding bit in **ha** is a "don't-care" bit, each **1** bit in **hm** specifies that the corresponding bit in the **ha** value is ANDed with the **hm** value. If the result is the same and also the hardware type matches, then a match is found. For example,

```
if (((hm & ha)==(client_hw_addr & hm))
    && (ht == client_hw_type))
        then a match is found
        else continue the search
```

Tags for client entries

ba
: This tag specifies that **bootpd** should broadcast the boot reply to the client. As a boolean tag, it causes **bootpd** to send the boot reply on the configured broadcast address of each network interface. You can also assign the tag an IP-address value, which specifies the specific IP or broadcast address for the boot reply.

bf=*filename*
: This tag specifies the *filename* of the bootfile that the client should download. The client's boot request, and the values of the **hd** (see below) and **bf** symbols, determine the contents of the bootfile field in the boot reply packet.

If the client specifies an absolute path name (in its boot request), and that file is accessible on the server machine (see below), **bootpd** returns that path name in the reply packet. If the file is not accessible, the request is discarded; no reply is sent. If the client specifies a relative path name, **bootpd** constructs a full path name by appending the relative path name to the value of the **hd** tag, and tests to determine if the full path name is accessible. If the full path name is accessible, it is returned in the boot reply packet; if not, the request is discarded.

Clients that do not specify boot files in their boot requests always elicit a reply from the server. The exact reply depends on the values of the **hd** and **bf** tags. If the **bf** tag specifies an absolute path name, and the file is accessible, that path name is returned in the reply packet. Otherwise, if the **hd** and **bf** tags together specify an accessible file, that file name is returned in the reply. If a complete file name cannot be determined, or the file is not accessible publicly, the reply contains a zeroed-out bootfile field.

If the **tftp** pseudo-user exists, **bootpd** treats all path names (absolute or relative) as being relative to the home directory of **tftp** and checks there first. If the file is not accessible under the **tftp** home directory or the **tftp** pseudo-user does not exist, **bootpd** checks for the file relative to **/**.

For a file to be available, it must exist, and be publicly readable.

All file names are first tried as *filename***.***hostname* and then simply as *filename*. However, in the case when the **tftp** pseudo-user exists, but *filename***.***hostname* and *filename* are not accessible under the **tftp** home directly, only *filename* is checked relative to **/**.

Note that a file considered to be accessible relative to **/** might not actually be accessible via **tftp** if the command line arguments to **tftpd** disallow that path.

bs=*size*
This tag specifies the size of the bootfile. The parameter *size* can be either a decimal, octal, or hexadecimal integer specifying the size of the bootfile in 512-octet blocks, or the keyword **auto**, which causes the server to automatically calculate the bootfile size at each request. Specifying the **bs** symbol as a boolean has the same effect as specifying **auto** as its value.

ci=*client_ID*
This tag specifies the client identifier of the client. The parameter *client_ID* can be either a hexadecimal integer, or a string contained in double quotes. The *client_ID* is a unique identifier that the DHCP client may use to identify itself to the server. If present, the client identifier supersedes the hardware address, so a client and an entry will only match in one of two situations: one, they both have the same client identifier, or two they both have the same hardware address and neither has a client identifier. If a request has a client identifier, then that is used to match the client up with an entry in the BOOTP server configuration file. One common client ID used is to concatenate the hardware type (e.g. 0x01 for ethernet) with the hardware address.

cs=*ip_address_list*
This tag specifies the IP addresses of RFC865 Quote of the Day (cookie) servers.

dn=*domain_name*
This tag specifies the domain name of the client for Domain Name Server resolution (see RFC1034).

ds=*ip_address_list*
This tag specifies the IP addresses of RFC1034 Domain Name servers.

ef=*filename*
Specifies the name of an extensions file. The file, retrievable via TFTP, contains information which can be interpreted in the same way as the 64-octet vendor-extension field within the BOOTP response. The maximum length of the file is unconstrained. All references to an extensions filename within the file are ignored.

gw=*ip_address_list*
This tag specifies the IP addresses of gateways for the client's subnet. If one of multiple gateways is preferred, it should be listed first.

hd=*home-directory*
This tag specifies a directory name to which the bootfile is appended (see the **bf** tag above). The default value of the **hd** tag is **/**.

hn The presence of this tag indicates that the client's host name should be sent in the boot reply. The **hn** tag is a boolean tag. **bootpd** attempts to send the entire host name as it is specified in the configuration file or hosts database. The configuration file is checked first, if the host name is not found, the hosts(4) database is then checked. If the hostname cannot fit into the reply packet, an attempt is made to shorten the name to just the host field (up to the first period, if present) and then tried. In no case is an arbitrarily truncated host name sent. If nothing reasonable can fit, nothing is sent.

im=*ip_address_list*
This tag specifies the IP addresses of Impress network image servers.

ip=*ip-address*
This tag specifies the IP address of the DHCP/BOOTP client.

lg=*ip_address_list*
This tag specifies the IP addresses of MIT-LCS UDP log servers.

lp=*ip_address_list*
This tag specifies the IP addresses of Berkeley 4BSD printer servers.

md=*merit_dump_file*
This tag specifies the name of a file to dump the core of a client.

na=*ip_address_list*
This tag specifies the IP address(es) of RFC 1001/1002 NetBIOS name server(s) in order of preference.

nb=*ip_address_list*
This tag specifies the IP address(es) of RFC 1001/1002 NetBIOS datagram distribution server(s) in order of preference.

nc=*NetBIOS_node_type*
Specifies the NetBIOS node type code. Allows NetBIOS over TCP/IP clients to be configured as described in RFC1001/1002. The *NetBIOS_node_type* can be an unsigned decimal, octal, or hexadecimal integer corresponding to one of the client types as follows:

0x1 or **B-node** for B-node;
0x2 or **P-node** for P-node;
0x4 or **M-node** for M-node;
0x8 or **H-node** for H-node.

nd=*string*
This tag specifies the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002.

ns=*ip_address_list*
This tag specifies the IP addresses of IEN-116 name servers.

nt=*ip_address_list*
This tag specifies the IP addresses of Network Time Protocol servers. Servers should be listed in order of preference.

rl=*ip_address_list*
This tag specifies the IP addresses of RFC887 Resource Location Protocol servers.

rp=*root_path*
This tag specifies a path name to be mounted as a root disk.

sm=*subnet-mask*
This tag specifies the client's subnet mask. *subnet-mask* is specified as a single IP address.

sr=*destination_ip_address gateway_ip_address ...*
This tag specifies a list of static routes that the client should put in its routing cache. Each route consists of a pair of IP addresses. The first address is the destination address, and the second is the router. Use the **gw=** option to specify the default route (0.0.0.0) as it is not a legal destination address.

ss=*ip_address*
This tag specifies the IP address of a swap server.

T*nnn***=***generic-data*
This is a generic tag where *nnn* is an RFC1533 option field tag number. Use this option to configure RFC1533 options not currently supported with **bootpd** tag names. This option allows one to immediately take advantage of future extensions to RFC1533. The *generic-data* data can be represented as either a stream of hexadecimal numbers or as a quoted string of ASCII

characters. The length of the generic data is automatically determined and inserted into the proper fields of the RFC1541-style boot reply.

to=*offset*
This tag specifies the client's time zone offset in seconds from UTC. The time *offset* can be either a signed decimal integer or the keyword **auto** which uses the server's time zone offset. Specifying the **to** symbol as a boolean has the same effect as specifying **auto** as its value.

ts=*ip_address_list*
This tag specifies the IP addresses of RFC868 Time Protocol servers.

yd=*NIS-domain-name*
Specifies the name of the client's NIS domain.

ys=*ip_address_list*
Specifies the IP address(es) of NIS servers available to the client. Servers should be listed in order of preference.

vm=*magic-cookie*
This tag specifies the RFC1048 vendor information magic cookie. *magic-cookie* can be one of the following keywords: **auto** (indicating that vendor information is determined by the client's request), **rfc1048** (which always forces an RFC1048-style reply), or **cmu** (which always forces a CMU-style reply).

V*nnn***=***generic-data*
This is a generic tag for vendor specific information where *nnn* is a vendor defined option field tag number. The *generic-data* data can be represented as either a stream of hexadecimal numbers or as a quoted string of ASCII characters. The length of the generic data is automatically determined and inserted into the vendor specific field of the RFC1541-style boot reply.

xd=*ip_address_list*
This tag specifies the IP addresses of systems that are running the X Window System Display Manager and are available to the client. Addresses should be listed in order of preference.

xf=*ip_address_list*
This tag specifies the IP addresses of X window System font servers available to the client. Servers should be listed in order of preference.

dhcptab Configuration

The configuration file **/etc/dhcptab** defines groups of IP addresses that to be leased out to clients. It also specifies certain general behaviors of the server, such as whether or not to give addresses from these groups to BOOTP clients or only to DHCP clients.

The configuration file has a format similar to the **/etc/bootptab** configuration file, with a keyword followed by one or more tag symbols. These tag symbols are separated by colons (**:**). The general format is:

*keyword***:***tg***=***value***:** ... :*tg***=***value***:** ... **:***tg***=***value***:** ...

where *keyword* is one of four allowed (non-case-sensitive) symbols and *tg* is a two or more (case-sensitive) character tag symbol. Most tags must be followed by an equals-sign and a value as above. Some can also appear in a boolean form with no value (i.e. **:***tg***:**).

Blank lines and lines beginning with **#** are ignored in the configuration file. Keyword entries are separated from one another by newlines; a single host entry may be extended over multiple lines if each continued line ends with a backslash (****). Lines may be longer than 80 characters. Tags can appear in any order.

IP addresses must be specified in standard Internet "dot" notation, and can use decimal, octal, or hexadecimal numbers (octal numbers begin with **0**, hexadecimal numbers begin with **0x** or **0X**). Certain tags accept a list of one or more IP addresses (*ip_address_list*). When more than one IP address is listed, they must be separated by white space.

The currently recognized keywords are:

dhcp_pool_group
This keyword is followed by tags defining a group of IP addresses to give out to clients on the same subnet, and the characteristics of that group. In addition to the tags defined for DHCP groups, all of the two-letter tags for bootp entries may also be used (except for **ht**, the hardware type tag, **ha**, the hardware address tag, or **ci**, the client ID tag. Required tags are: **subnet-mask**, **addr-pool-start-address**, and **addr-pool-last-address**.

dhcp_device_group
This keyword is used to define a group of IP addresses on a subnet much like **dhcp_pool_group**, but with one exception: all clients in a device group must have the same client class (specified with tag **class-id**). This allows different types of clients to receive different parameters from the server. Required tags are: **class-id**, **subnet-mask**, **addr-pool-start-address**, and **addr-pool-last-address**.

dhcp_default_client_settings
This keyword is followed by tags to be applied to all groups. These tag values can be overridden for a specific group if that tag is defined for that specific group. This keyword simply saves one from entering the same tag for every group. Thus most tags that may be used for **dhcp_pool_group**, and **dhcp_device_group**, may be used here. The tag descriptions specify if a tag may not be used here.

dhcp_server_settings
This keyword is followed by tags that specify a few general behaviors for the dhcp server as a whole.

The currently supported tags for **dhcp_server_settings**:

call-on-unrequited=*filename*
This tag specifies an executable file *filename* that will be called when the server receives a request to which it cannot send a response. Certain arguments will be passed in; the call executed will be:

filename **:** *client-id htype haddr* [*gateway*]

where *client-id* is the client ID in hex if present, or 00 if there is no client ID. *htype* is the hardware type as per the ARP section of the "Assigned Numbers" RFC. *haddr* is the hardware address in hex. *gateway* is the IP address of the bootp relay agent. If the packet was not relayed, then this field is absent.

The currently supported tags for **dhcp_pool_group**, **dhcp_device_group**, and **dhcp_default_client_settings**:

class-name=*classname*
This tag specifies a name to refer to a device group by. It is only applicable to **dhcp_device_group**. The only use that bootpd makes of this field is in logging errors found in the configuration of the group.

pool-name=*poolname*
This tag specifies a name to refer to a pool group by. It is only applicable to **dhcp_pool_group**. The only use that bootpd makes of this field is in logging errors found in the configuration of the group.

class-id=*client-class*
This tag specifies the *client-class* that clients must have to be assigned to this group. This tag is required for **dhcp_device_group** and is inappropriate for any other keyword. Some DHCP clients send out a *client-class* that identifies a class that a client belongs to. For an IP address to be assigned from a device group address pool, not only must the client be on the right subnet, it must send a request with a *client-class* that matches that defined for the **class-id**. This may be specified in either hex or in ASCII (an ASCII string must be enclosed in double quotes).

subnet-mask= *mask*
This tag specifies the subnet mask for the addresses in the group being defined. It is specified as an IP address. This tag is required for both **dhcp_device_group** and **dhcp_pool_group**, and is inappropriate for **dhcp_default_client_settings**.

addr-pool-start-address=*IP-address*
This tag specifies the lowest address in the pool group to be assigned. This tag is required for both **dhcp_device_group** and **dhcp_pool_group**, and is inappropriate for **dhcp_default_client_settings**.

addr-pool-last-address=*ip-address*
This tag specifies the highest address in the pool group to be assigned. This address and the **addr-pool-start-address** define a range of addresses that can be assigned to clients. For the server, no two group address ranges may overlap.

reserved-for-other=*ip-address-list*
This tag is followed by one address that falls in the range of the group. This address is reserved, and will not be assigned to any clients by the DHCP server. Alternatively, a range of addresses may be defined by giving 2 addresses, with the range being the addresses from the first address up to the second address, inclusively. This tag may be repeated to reserve more addresses in the same group. It is not appropriate for **dhcp_default_client_settings**.

lease-time=*seconds*
This tag specifies the time in seconds that a lease should be given to each client. The word "infinite" may be used to specify leases that never expire. The default is "infinite." Note that if a client asks for a shorter lease than is configured for it, it will get that shorter lease time.

lease-grace-period=*percent*
This tag specifies the time after a lease expires during which that lease will not be assigned to a new client. *percent* is the percentage of the configured lease time that this grace period lasts. The default is 5%.

tr=*percent*
This tag specifies the DHCP IP lease renewal time (T1) as a percentage of the total lease-time. This is the time interval from lease assignment until when the client attempts to renew the lease. RFC1541 states that T1 defaults to 50%.

tv=*percent*
This tag specifies the DHCP IP lease rebind time (T2) as a percentage of the total lease-time. This is the time interval from lease assignment until when the client attempts to obtain a new lease from any server. RFC1541 states that T2 defaults to 0.875 times the lease duration (which we round off to 88%).

lease-policy= *policy*
This tag specifies whether or not the assigning of new leases can be done. If *policy* is set to **reject-new-clients** then no new clients can get a lease, and only clients with existing leases will get a response. **accept-new-clients** is the default.

allow-bootp-clients=*boolean*
This tag specifies whether or not bootp clients can be members of the group being defined. The default is **false**. If *boolean* is **TRUE**, then an IP address may be assigned to a client that doesn't have an entry in the **bootptab** file and that is on the same subnet as the group being defined. This address is treated as an infinite lease, and a boot reply is sent to the client. This tag is is not appropriate for **dhcp_device_group**, since bootp clients don't have a client class (and therefore a bootp client would be incapable of matching the client class of the device group). If this tag is used for **dhcp_default_client_settings**, then it is only applicable to pool groups.

call-on-assignment=*filename*
This tag specifies the fully qualified *filename* to be called when an IP address has been assigned to a new client. Some arguments will be passed in, the call will be made as follows:

*filename***:** *client-id htype haddr ipaddr subnet-mask lease-expiration* [*hostname*]

where *client-id* is the client ID in hex if present, or 00 if there is no client ID. *htype* is the hardware type as per the ARP section of the "Assigned Numbers" RFC. *haddr* is the hardware address in hex. *ipaddr* is the IP address that was assigned to the client. *subnet-mask* is the subnet mask of the client represented as an IP address. *lease-expiration* is the bootpd internal representation of when the lease will expire (based on a C call to time()), a value of ffffffff represents an infinite lease. If there is a *hostname* associated with this address, then it is the final argument.

call-on-decline=*filename*
This tag specifies the fully qualified *filename* to be called when an IP address has been declined by a new client. Some arguments will be passed in, the call will be made as follows:

*filename***:** *client-id htype haddr ipaddr subnet-mask*

where *client-id* is the client ID in hex if present, or 00 if there is no client ID. *htype* is the hardware type as per the ARP section of the "Assigned Numbers" RFC. *haddr* is the hardware address in hex. *ipaddr* is the IP address that was declined by the client. *subnet-mask* is the subnet mask of the client represented as an IP address.

call-on-release=*filename*

This tag specifies the fully qualified *filename* to be called when an IP address has been released by a client. Some arguments will be passed in, the call will be made as follows:

filename: *client-id htype haddr ipaddr lease-expiration*

where *client-id* is the client ID in hex if present, or 00 if there is no client ID. *htype* is the hardware type as per the ARP section of the "Assigned Numbers" RFC. *haddr* is the hardware address in hex. *ipaddr* is the IP address that was released by the client. *lease-expiration* is the bootpd internal representation of when the lease would have expired, a value of ffffffff represents an infinite lease.

call-on-lease-extend=*filename*

This tag specifies the fully qualified *filename* to be called when an IP address lease for a client has been extended. Some arguments will be passed in, the call will be made as follows:

filename: *client-id htype haddr ipaddr subnet-mask lease-expiration*

where *client-id* is the client ID in hex if present, or 00 if there is no client ID. *htype* is the hardware type as per the ARP section of the "Assigned Numbers" RFC. *haddr* is the hardware address in hex. *ipaddr* is the IP address that was assigned to the client. *subnet-mask* is the subnet mask of the client represented as an IP address. *lease-expiration* is the bootpd internal representation of when the lease will expire (based on a C call to time()), a value of ffffffff represents an infinite lease.

DHCP/BOOTP Packet

The DHCP/BOOTP packet has the following format:

```
struct dhcp {
    unsigned char   op;              /* packet opcode type */
    unsigned char   htype;           /* hardware addr type */
    unsigned char   hlen;            /* hardware addr length */
    unsigned char   hops;            /* gateway hops */
    unsigned long   xid;             /* 4 bytes transaction ID */
    unsigned short  secs;            /* seconds since boot began */
    unsigned short  flags;           /* if giaddr!=0,client flags*/
    struct in_addr  ciaddr;          /* client IP address */
    struct in_addr  yiaddr;          /* 'your' IP address */
    struct in_addr  siaddr;          /* server IP address */
    struct in_addr  giaddr;          /* gateway IP address */
    unsigned char   chaddr[16];      /* client hardware address */
    unsigned char   sname[64];       /* server host name */
    unsigned char   file[128];       /* boot file name */
    unsigned char   options[312];    /* options area */
};
```

DHCP Option Numbers

The DHCP/BootP options discussed above correspond to the option numbers in RFC1533 as follows:

Number	Tag	Description
1	**sm**	Subnet Mask
2	**to**	Time Offset
3	**gw**	Gateways
4	**ts**	Time Servers
5	**ns**	IEN 116 Name Servers
6	**ds**	Domain Name Servers
7	**lg**	Log Servers
8	**cs**	Cookie Servers
9	**lp**	LPR Servers
10	**im**	Impress Servers
11	**rl**	Resource Location Servers
12	**hn**	Send Host Name in reply
13	**bs**	Boot File Size
14	**md**	Merit Dump File
15	**dn**	Domain Name
16	**ss**	Swap Server

17	**rp**	Root Path
18	**ef**	Extensions Path
28	**ba**	Broadcast Address
33	**sr**	Static Routes
40	**yd**	NIS Domain
41	**ys**	NIS Servers
42	**nt**	NTP Servers
43	**V###**	Vendor Specific Information
44	**na**	NetBIOS Name Servers
45	**nb**	NetBIOS Datagram Distribution Servers
46	**nc**	NetBIOS Node Type
47	**nd**	NetBIOS Scope
48	**xf**	X Font Servers
49	**xd**	X Display Manager
51	**lease-time**	IP Address Lease Time
58	**tr**	Lease Renewal Time (T1)
59	**tv**	Lease Rebinding Time (T2)
60	**class-id**	Class Identifier
61	**ci**	Client Identifier

EXAMPLES

This is an example of a **/etc/bootptab** file:

```
# Common entry

global.defaults:\
    bf=C2300A:\
    hd=/usr/lib/X11/:\
    hn:\
    ht=ether:\
    vm=rfc1048

# Now the actual individual entries

xterm1:\
    tc=global.defaults:\
    ha=08000903212F:\
    ip=190.40.101.22

xterm2:\
    tc=global.defaults:\
    ha=0800090324AC:\
    ip=190.40.101.35

# Common relay entry.

relay-default:\
    ht=ethernet:\
    bp=15.4.3.136 15.13.6.192:\
    th=2:\
    hp=5:

# Relay entry for node2

node2:\
    tc=relay-default:\
    ha=08000902CA00:

# Group relay entry

group-machines:\
    tc=relay-default:\
    ha=080009000000:\
    hm=080009000000:

# Turn the relay off (block the relay) for the following machines.

blocked-machines:\
    ht=ethernet:\
    ha=07000A000000:\
    hm=07000A000000:

# Relay definition for all other machines.

all:\
    tc=relay-default:\
    ha=000000000000:\
    hm=000000000000:
```

This is an example of a **/etc/dhcpptab** file:

```
     # The first entry is for options which define the server's operation.

     DHCP_SERVER_SETTINGS:\
          call-on-unrequited="/tmp/unrequited.script" :\

     # The next entry is for options that will be applied to all groups.
     # Individual options may be overridden for a specific group if the group
     # also configures the option.

     DHCP_DEFAULT_CLIENT_SETTINGS:\
          hn:\
          lease-time=10080:\

     # The next entry defines an address pool for devices with the class
     # id "xterminal" on subnet 15.14.128.  Address leases will be granted
     # for up to 1 week.  The server will use a broadcast message to
     # respond to all client requests.

     DHCP_DEVICE_GROUP:\
          ba:\
          class-name=SUBNET_128_XTERMINAL_GROUP:\
          class-id="xterminal:"\
          subnet-mask=255.255.255.0 :\
          addr-pool-start-address=  15.14.128.1 :\
          addr-pool-last-address=   15.14.128.254 :\
          lease-time=604800 :\
          lease-grace-period=5 :\

     # The next entry grants IP leases to any device on subnet
     # 15.13.128. The script /usr/local/bin/assignment.script will be
     # run whenever a new lease is granted.

     DHCP_POOL_GROUP:\
          pool-name=RED_SUBNET_POOL:\
          call-on-assignment="/usr/local/bin/assignment.script" :\
          subnet-mask=255.255.255.0 :\
          addr-pool-start-address=  15.13.128.100 :\
          addr-pool-last-address=   15.13.128.254 :\
          gw=15.13.128.1
```

WARNINGS

Individual host entries must not exceed 1024 characters.

AUTHOR

bootpd was developed by Carnegie Mellon University, Stanford University, and HP.

FILES

/etc/bootptab
/etc/dhcptab
/etc/services

SEE ALSO

bootpquery(1M), dhcptools(1M), inetd(1M), tftpd(1M), syslog(3C), hosts(4).

DARPA Internet Requests For Comments: RFC865, RFC868, RFC887, RFC951, RFC1010, RFC1034, RFC1048, RFC1084, RFC1395, RFC1533, RFC1534, RFC1541, RFC1542.

NAME

bootpquery - send BOOTREQUEST to BOOTP server

SYNOPSIS

/usr/sbin/bootpquery *haddr* [*htype*] [*options*]

DESCRIPTION

bootpquery is a diagnostic function used to check the configuration of the Internet Bootstrap Protocol (BOOTP) server, *bootpd*(1M). This function can only be run by the superuser, since it uses reserved ports.

bootpquery constructs a boot request with the supplied parameters to send to the BOOTP server, and prints the contents of the BOOTP server reply (as shown in EXAMPLES, below). Note that **bootpquery** formats and prints RFC-1048 or CMU-style vendor information included in the BOOTREPLY.

The BOOTREQUEST packet is broadcast on the BOOTP server port, **bootps**. If a BOOTP server is configured to respond to the request, it returns a BOOTREPLY packet on the BOOTP client port, **bootpc**. **bootpquery** can only display BOOTREPLY packets when the BOOTP server broadcasts the reply on the client port or when the hardware address and IP address supplied in the BOOTREQUEST are those of the host on which **bootpquery** is run.

The following options provide the information for the BOOTREQUEST:

haddr Hardware address of the BOOTP client; used in the BOOTREQUEST. A BOOTP server responds if it has configuration information for a host with this link-level address.

htype Type of address specified as *haddr*; may be **ether** or **ieee802**. The default address type is **ether**.

-i*ipaddr*
Specify the internet address of the BOOTP client to be used in the BOOTREQUEST. If the BOOTP client does not know its IP address, the BOOTP server supplies it in the BOOTREPLY. Otherwise, the server returns the BOOTREPLY directly to ipaddr.

-s*server*
Specify the name of the BOOTP server to receive BOOTREQUEST. When the BOOTP server is known, the BOOTREQUEST is not broadcast.

-v*vendor*
Specify a vendor name to include vendor information in the BOOTREPLAY. *vendor* can be specified as **rfc1048** or **cmu**. For any other *vendor* specification, the first four characters of the parameter are used as the vendor magic cookie.

-f Specify that **bootpd** should broadcast the reply back. This option is only valid for **bootpd** on the HPUX 10.0 (or later) release(s).

-b*bootfile*
Specify a boot file needed by the BOOTP client. If a boot file is specified in the BOOTREQUEST, the BOOTP server responds only if the server host can make the file available.

EXAMPLES

```
/usr/sbin/bootpquery 02608cee018e ether -s hpserver

Received BOOTREPLY from hpserver.hp.com (15.9.18.119)

Hardware Address:    02:60:8c:ee:01:8e
Hardware Type:       ethernet
IP Address:          15.9.18.113
Boot file:      /export/tftpdir/hp-gw2-confg

RFC 1048 Vendor Information:
     Subnet Mask:         255.255.248.0
     Bootfile Size:       6 512 byte blocks
     Domain Name Server:  15.9.18.119
     Host Name:      hp-gw2
```

AUTHOR

bootpquery was developed by HP.

SEE ALSO

bootpd(1M), tftp(1), tftpd(1M).

DARPA Internet Request For Comments RFC951, RFC1048, RFC1084, RFC1395, RFC1542 Assigned Numbers.

NAME

captoinfo - convert a termcap description into a terminfo description

SYNOPSIS

captoinfo [**-1v**] [**-w***n*] [*filenames*]

DESCRIPTION

captoinfo looks in *filenames* for *termcap*(3X) descriptions. For each one found, an equivalent *terminfo*(4) description is written to standard output along with any comments found. The short two letter name at the beginning of the list of names in a **termcap** entry, a holdover from Version 6 UNIX, is removed. Any description that is expressed relative to another description (as specified in the **termcap** *tc=* field) is reduced to the minimum superset before output.

If no *filename* is given, the environment variable **TERMCAP** is used for the filename or entry. If **TERMCAP** is a full pathname to a file, only the terminal whose name is specified in the environment variable **TERM** is extracted from that file. If the environment variable **TERMCAP** is not set, the file **/usr/share/lib/termcap** is read.

Options

captoinfo recognizes the following options:

-1 Print one field per line. If this option is not selected, multiple fields are printed on each line up to a maximum width of 60 characters.

-v Print (verbose) tracing information as the program runs. Additional **-v** options print more information (for example **-v -v -v** or **-vvv**).

-w*n* Change the output width to *n* characters.

DIAGNOSTICS

tgetent failed with return code n (reason).
The termcap entry is not valid. In particular, check for an invalid 'tc=' entry.

unknown type given for the termcap code 'cc'.
The termcap description had an entry for 'cc' whose type was not boolean, numeric or string.

wrong type given for the boolean (numeric, string) termcap code 'cc'.
The boolean termcap entry 'cc' was entered as a numeric or string capability.

the boolean (numeric, string) termcap code 'cc' is not a valid name.
An unknown termcap code was specified.

tgetent failed on TERM=term.
The terminal type specified could not be found in the termcap file.

TERM=term: cap cc (info ii) is NULL: REMOVED
The termcap code was specified as a null string. The correct way to cancel an entry is with an **@**, as in **:bs@:**. Giving a null string could cause incorrect assumptions to be made by any software that uses termcap or terminfo.

a function key for 'cc' was specified, but it already has the value 'vv'.
When parsing the 'ko' capability, the key 'cc' was specified as having the same value as the capability 'cc', but the key 'cc' already had a value assigned to it.

the unknown termcap name 'cc' was specified in the 'ko' termcap capability.
A key that could not be handled was specified in the 'ko' capability.

the vi character 'v' (info 'ii') has the value 'xx', but 'ma' gives 'n'.
The 'ma' capability specified a function key with a value different from that specified in another setting of the same key.

the unknown vi key 'v' was specified in the 'ma' termcap capability.
A vi key unknown to captoinfo was specified in the 'ma' capability.

Warning: termcap sg (nn) and termcap ug (nn) had different values.
terminfo assumes that the sg (now xmc) and ug values were the same.

Warning: the string produced for 'ii' may be inefficient.
The parameterized string being created should be rewritten by hand.

`Null termname given.`
The terminal type was null. This occurs when **`$TERM`** is null or not set.

`cannot open "file" for reading.`
The specified file could not be opened.

`Warning: cannot translate` *capability* **`(unsupported in terminfo).`**
This termcap capability is no longer supported in terminfo, and therefore cannot be translated.

C

WARNINGS

Certain **`termcap`** defaults are assumed to be true. For example, the bell character (**`terminfo`** *bel*) is assumed to be **`^G`**. The linefeed capability (**`termcap`** *nl*) is assumed to be the same for both **`cursor_down`** and **`scroll_forward`** (**`terminfo`** *cud1* and *ind*, respectively). Padding information is assumed to belong at the end of the string.

The algorithm used to expand parameterized information for **`termcap`** fields such as **`cursor_position`** (**`termcap`** *cm*, **`terminfo`** *cup*) sometimes produces a string which, though technically correct, may not be optimal. In particular, the rarely used **`termcap`** operation **`%n`** produces strings that are especially long. Most occurrences of these less than optimal strings are flagged with a warning message, and may need to be recoded by hand.

HP supports only terminals listed on the current list of supported devices. However, the terminfo database contains both supported and nonsupported terminals. If you use nonsupported terminals, they may not work correctly.

AUTHOR

`captoinfo` was developed by AT&T.

SEE ALSO

tic(1M), untic(1M), curses(3X), termcap(3X), terminfo(4), infocmp(1M).

NAME

catman - create the cat files for the manual

SYNOPSIS

/usr/sbin/catman [**-A** *alt-path*] [**-p**] [**-m**] [**-n**] [**-w**] [**-z**] [*sections*]

DESCRIPTION

The **catman** command creates the formatted versions of the online manual from *nroff*(1)-compatible source files. Each manual entry in the **man**∗**.Z** and **man**∗ directories is examined, and those whose formatted versions are missing or out-of-date are recreated. **catman** formats the most recent of the entries, compresses it, and puts it into the appropriate **cat**∗**.Z** directory.

If any changes are made, **catman** recreates the **/usr/share/lib/whatis** database. By default, the **/usr/share/lib/whatis** database is overwritten. If the **MANPATH** environment variable is set to a non-default set of paths, the old database file is saved in **/usr/share/lib/whatis.old** so that, if desired, the system administrator may merge them together.

By default, **catman** searches the **man**∗**.Z** and **man**∗ subdirectories under the following man directories:

- **/usr/share/man**
- **/usr/contrib/man**
- **/usr/local/man**

If **MANPATH** is set in the environment, the directories given in **MANPATH** are checked instead of the default. See *environ*(5) for a description of the **MANPATH** environment variable.

Before running **catman**, remove any existing **cat**∗ directories. If the **-z** option is used, **cat**∗**.Z** directories should be removed instead. If both **cat**∗**.Z** and **cat**∗ directories exist, *man*(1) updates both directories and more space is used.

Any command-line parameters not starting with **-** are interpreted as a list of manual sections (directories) to search. For example:

```
catman 123
```

restricts updating to manual sections 1, 2, and 3 (directories **man1**, **man2**, and **man3**).

Options

catman supports the following options:

-m	Create a merged **/usr/share/lib/whatis** database; i.e., information on new manual entries (added since the last time **catman** was run) is merged into the current database rather than overwriting it. Ignored if selected with the **-n** option.
-n	Prevents creation of **/usr/share/lib/whatis**.
-p	Prints what would be done instead of doing it.
-w	Causes only the **/usr/share/lib/whatis** database to be created. No manual reformatting is done.
-z	Puts the formatted entries in the **cat**∗ directories rather than in the **cat**∗**.Z** directories.
-A *alt-path*	Perform actions based on the given alternate root. With this option, *alt-path* will be prepended to all directory paths, including default paths, the paths defined by **MANPATH**, and the path to **/usr/share/lib/whatis**.

EXTERNAL INFLUENCES

Environment Variables

MANPATH defines parent directories to be used when searching **man**∗ and **man**∗**.Z** directories.

WARNINGS

If unformatted manual entries (those in the **../man**∗ subdirectories) have been removed since the last time **catman** was run, information in the **/usr/share/lib/whatis** database may be lost. The **-m** option may be used to override this, but may result in repeated lines in the database for the same manual entry.

EXAMPLES

Create uncompressed **cat**∗ files for sections 1 and 1m of the manual, but don't create the **/usr/share/lib/whatis** database:

```
catman -z -n 11m
```

Run **catman** from a server to create **cat**∗ entries for a diskless client under the alternate root **/export/shared_roots/OS_700**:

```
catman -A /export/shared_roots/OS_700
```

This will create **cat**∗ manpages under:

```
/export/shared_roots/OS_700/usr/share/man/
/export/shared_roots/OS_700/usr/contrib/man/
/export/shared_roots/OS_700/usr/local/man/
```

and a **whatis** file in:

```
/export/shared_roots/OS_700/usr/share/lib/whatis
```

Create **cat**∗ entries for an application and merge the information with the **/usr/share/lib/whatis** database:

```
MANPATH=/opt/langtools/man
catman -m
```

Note that you may wish to save **MANPATH** before doing this, so as not to lose your current **MANPATH**.

AUTHOR

catman was developed by HP and the University of California, Berkeley.

FILES

/usr/share/man/man∗[.Z]/∗	Unformatted (*nroff*(1)-compatible source) manual entry files [compressed].
/usr/share/man/cat∗[.Z]/∗	Formatted manual pages [compressed].
/usr/local/man/man∗[.Z]/∗	
/usr/local/man/cat∗[.Z]/∗	
/usr/contrib/man/man∗[.Z]/∗	
/usr/contrib/man/cat∗[.Z]/∗	
/usr/share/lib/whatis	Database of manpage entry summaries; utilized by the **man -k** command.
/usr/lbin/mkwhatis	Command to make **whatis** database.

SEE ALSO

compress(1), fixman(1M), man(1), environ(5).

NAME

cfl - configure a logical unit (LUN) on a SCSI disk array

SYNOPSIS

cfl [**-L** *LUN_address*] [**-a -c***list* [**,** *list*] [**-i**]] [**-b** *block_size*] [**-c** *list* [**,** *list*]] [**-d**] [**-f** *flag_word*] [**-k** *num_log_blocks*] [**-l** *sec_tenths*] [**-n** *num_log_blocks*] [**-p** *list*] [**-r** *RAID_level*] [**-s** *num_log_blocks*] [**-t reg**|**sub**] [**-z** *num_log_blocks*] *device_file*

DESCRIPTION

cfl sets configuration parameters, and changes the status of a LUN on the HP SCSI disk array associated with *device_file*.

NOTE: **newarray**, a front-end program for **cfl**, is recommended for doing array configuration (see *newarray*(1M)).

Options

-L *LUN_address*
Specifies which SCSI unit address to affect.

-a -c *list* [**,** *list*] [**-i**]
list is a comma-separated drive list (**c***X***i***Y*,**c***X***i***Y*,...) describing drives on SCSI channel *X*, and SCSI ID *Y* (where *X* and *Y* are decimal numbers). Multiple *lists* are delimited by space characters.

Add a LUN to the set of LUNs known by the controller. If this option is used, the runstring must also contain a value for the **-c** parameter, and can contain values for all other applicable parameters except **-d** (the delete LUN option). If only the **-c** parameter is supplied, a default RAID-level 0 configuration is created with the drives specified in the parameter list. The user may thus specify all the LUN characteristics in one line; create a default configuration and change a few of the parameters to desired values in one line, or create a default configuration and iteratively change its parameters to the desired values. The **-i** option formats the newly added LUN after configuration. If multiple LUNs are to be added and configured, each LUN must be formatted before any other LUNs can be added and configured.

-b *block_size* Set the logical block size of the LUN. *block_size* is specified in bytes.

-c *list* [**,** *list*2] *device_file*
Assign to the LUN a configuration table that describes which drives are associated with the LUN and specifies the order each drive appears in a data stripe. One, or more tables can be assigned to each LUN, depending on the RAID level. Each table can have a maximum of five drives.

-d Delete the LUN from the set of LUNs known by the controller. This option cannot be used simultaneously with the **-a** option.

-f *flag_word* Assign the desired hexadecimal values, given in *flag_word*, to the array's two LUN flag bytes. The default *flag_word* is hex **0072**. User-changeable bits are in Mode Page 0x2b byte 25 (the lsb): bit 4, which enables AEN polling when set; bit 5, which enables parity verification when set, and bit 6, which enables writes with parity verification when set.

-k *num_log_blocks*
Set the reconstruction quantity in blocks. This represents the number of blocks reconstructed in a single reconstruction command. Reconstruction commands are issued at an adjustable interval until the LUN is reconstructed (see the **-l** option).

-l *sec_tenths* Set the reconstruction frequency, the interval between successive reconstruction commands. It is expressed in tenths of a second.

-n *num_log_blocks*
Set the number of logical blocks in the LUN.

-p *list* Create the LUN's disk bit map, which describes the drives associated with the LUN. Either a configuration table or a disk bit map, but not both, is required to configure a LUN; use of the configuration table is recommended.

-r *raid_level* Set the RAID level of the LUN; valid RAID levels are 0, 1, 3 and 5.

-s *num_log_blocks*
: Set the number of blocks in a LUN segment, the part of a data stripe residing on a single disk.

-t reg | **sub**
: Set the LUN type, regular or sub-LUN. A sub-LUN is a LUN that can share its physical drive(s) with another LUN; usually, its data resides on more than one drive. Configurations involving data striping or mirroring should use sub-LUNs.

-z *num_log_blocks*
: Set the number of blocks in the first segment of the LUN.

RETURN VALUE

cfl returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **cfl**
- SCSI (device level) communications
- system calls

All error information is printed to stderr.

Error messages generated by cfl:

```
usage: cfl -L <LUN_addr> -a <-c ...> [-i] <special>  add LUN
 cfl -L <LUN_addr> -b <n> <special>   set logical block size
 cfl -L <LUN_addr> -c <<cXiY,... [cXiY,...]> | none> <special> build
   config table(s)
 cfl -L <LUN_addr> -d <special>           delete LUN
 cfl -L <LUN_addr> -f <n> <special>       set LUN flags
 cfl -L <LUN_addr> -k <n> <special>       set reconstruction amt in
   blocks
 cfl -L <LUN_addr> -l <n> <special>       set reconstruction frequency
 cfl -L <LUN_addr> -n <n> <special>       set number of blocks in LUN
 cfl -L <LUN_addr> -p <cXiY,...> <special> build disk bit map
 cfl -L <LUN_addr> -r <n> <special>       set RAID level
 cfl -L <LUN_addr> -s <n> <special>       set segment size in blocks
 cfl -L <LUN_addr> -t <reg | sub> <special>  set LUN type
 cfl -L <LUN_addr> -z <n> <special>       set segment 0 size in blocks
```

An error in command syntax has occurred. No valid tags were present, or an illegal tag was encountered. Re-enter the command with all required arguments. If a syntax error occurs in a runstring with a legal tag, only the template for that tag will be displayed.

cfl: Arg incompatible with other
: One of the arguments is incompatible with another, for example, when the **-a** (add LUN) and **-d** (delete LUN) are both on the command line.

cfl: Arg out of range
: One of the arguments is larger than its allowed maximum value (or smaller than its allowed minimum value), or is incorrect in form. Check the size, and form of each argument and make appropriate corrections.

cfl: device busy
: To ensure that **cfl** does not modify a disk array that is being used by another process, **cfl** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

cfl: LUN does not exist
: The addressed LUN is not known to the array controller. Only the **-a** option can operate on an

unconfigured LUN. The **-d** option ignores references unconfigured LUNs (and does nothing with them).

cfl: LUN # too big
The LUN number, which is derived from the device special file name, is out of range.

cfl: Multiple args of same type
An argument occurs more than once on the command line.

cfl: Not a disk array
The device being addressed did not identify itself as a SCSI disk array product that is supported by **cfl**.

cfl: Not a raw file
cfl must be able to open the device file for raw access (the character device file).

cfl: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

cfl uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **cfl** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To delete LUN 5 associated with **/dev/rdsk/c2t0d0**:

```
cfl -L 5 -d /dev/rdsk/c2t0d0
```

To add the LUN 0 associated with **/dev/rdsk/c2t0d0**, which will have the following characteristics: logical block size 512 bytes, RAID level of 5, auto reconstruct disabled, reconstruction amount of 64 blocks, reconstruction frequency of .2 seconds, segment size of 64 blocks, type sub-LUN, segment zero size of 1, and drives with SCSI ID 1 on channels 1 through 5, to be striped in the channel order 3, 5, 1, 2 and 4:

```
cfl -L 0 -a  -b 512 -r 5 -f 0072 -k 64 -l 2 -n 123456 -s 64
    -t sub -z 1 -c c3i1,c5i1,c1i1,c2i1,c4i1 /dev/rdsk/c2t0d0
```

WARNING

Changing any configuration parameter except the reconstruction frequency and reconstruction quantity puts the affected LUN in an unusable ("dead") state. You must reformat the LUN before it can be used with the new configuration values. Formatting a LUN destroys all of its user data.

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

cfl was developed by HP.

SEE ALSO

newarray(1M), arraytab(4), vgchange(1M).

C

NAME

ch_rc - change system configuration file

SYNOPSIS

/usr/sbin/ch_rc -a|**-r**|**-l** [**-v**] [**-A**] [**-R** *root*] [**-p** {*parameter*|*parameter*=*value*}...] [*file* ...]

DESCRIPTION

ch_rc manages the addition, modification, removal, and retrieval of information stored in files having the format of those in the **/etc/rc.config.d** directory.

Parameter names are treated as strings. Thus, **X[0]** has no special meaning to **ch_rc** in relation to other parameters named **X[1]**or**X**.

Options

file Specify the file(s) to be used as the configuration database. If no file is specified, the set of files used by **ch_rc** defaults to **/etc/TIMEZONE** and all files in the **/etc/rc.config.d** directory.

Modification and deletion of configuration parameters occurs in the file where the parameter is found.

-a Add or modify a parameter definition. For each parameter specified on the command line, if the parameter is found in the specified (or default) files, it is modified to reflect the specified value. If the parameter is not found, it is added to the specified file(s).

If a new parameter is being defined, one or more files must be specified on the command line; the specified files are those in which the parameter will be defined.

-r Remove a parameter definition. For each parameter name specified on the command line, remove any occurrence of that parameter from the specified file(s).

-l List configuration values. For each parameter specified on the command line, output every definition of the parameter from the specified file(s). Output consists of only the values, one per line.

-p Specify a parameter name or name/value pair. If a name and value is expected, but only a name is specified, the value will be set to the empty string. For example, specifying **FOO** or **FOO=** will result in **FOO** and **FOO=** respectively.

Due to shell quoting rules, if you need a quoted parameter value, you must protect the quotes from the shell. For example,

```
ch_rc -a -p VALUE="a b c" <file>
```

yields:

```
VALUE=a b c
```

which is an error, whereas,

```
ch_rc -a -p VALUE='"a b c"' <file>
```

yields:

```
VALUE="a b c"
```

-v Verbose. When used with the **-l** option, the **-v** option causes a verbose listing to be output. This listing includes a filename followed by the entire line containing the specified parameter for each occurrance of the parameter.

-A The **-A** option is used to list all occurances of array parameters matching the parameters specified on the command line.

For example,

```
ch_rc -l -A -v -p ZZZ file
```

may emit the following output:

```
file: ZZZ[0]=zero
file: ZZZ[5]=five
file: ZZZ[9]=fred
```

-R *root* Normally, the files specified on the command line are used as specified. By specifying a *root* directory with the **-R** option, all files (including the default files if none are specified) will be interpreted relative to *root*.

For example, if *root* is specified as **/foo** and **/etc/TIMEZONE** is specified on the command line, it will be interpreted as **/foo/etc/TIMEZONE**.

RETURN VALUE

ch_rc exits with one of the following values:

0	add/delete/list successful
1	command line syntax/usage error
2	can not access one or more of the listed (or default) files
3	can not open/create/write file
4	memory error
5	no files specified on command line for add option

EXAMPLES

Files in the **/etc/rc.config.d** directory have the following format:

```
# Comments are preceded by pound signs and
# are always on a line of their own.
# Blank lines are allowed.

VARIABLE=value
VARIABLE_2=value2
VARIABLE_3[1]=value3
VARIABLE_3[2]=value4

# All parameters are defined on a single line
# Parameters must not be exported
```

WARNINGS

ch_rc does not interpret configuration files; it only does pattern matching. As a result, if comments appear on lines containing parameter definitions, the comments will also appear in output when using the **-l** option.

ch_rc cannot parse multiple parameter definitions which occur on the same line of a file.

AUTHOR

ch_rc was developed by HP.

FILES

/etc/rc.config	system configuration database driver file
/etc/rc.config.d	directory containing system configuration files

SEE ALSO

rc.config(4).

NAME

chroot - change root directory for a command

SYNOPSIS

/usr/sbin/chroot *newroot command*

DESCRIPTION

The **chroot** command executes *command* relative to the *newroot*. The meaning of any initial slashes (**/**) in path names is changed for *command* and any of its children to *newroot*. Furthermore, the initial working directory is *newroot*.

Note that command suffixes that affect input or output for the **chroot** command use the original root, not the new root. For example, the command:

chroot *newroot command* **> x**

locates file **x** relative to the original root, not the new one.

The *command* variable includes both the command name and any arguments.

The new root path name is always relative to the current root. Even if a **chroot** is currently in effect, the *newroot* argument is relative to the current root of the running process.

This command is restricted to users with appropriate privileges.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

WARNINGS

command cannot be in a shell script.

Exercise extreme caution when referring to special files in the new root file system.

chroot does not search the **PATH** environment variable for the location of *command*, so the absolute path name of *command* must be given.

When using **chroot** to establish a new environment, all absolute path name references to the file system are lost, rendering shared libraries inaccessible. If continued access to shared libraries is needed for correct operation, the shared libraries and the dynamic loader *must* be copied into the new root environment.

SEE ALSO

chdir(2), chroot(2).

STANDARDS CONFORMANCE

chroot: SVID2, SVID3, XPG2, XPG3

NAME

clri - clear inode

SYNOPSIS

`/usr/sbin/clri` *special i-number* ...

DESCRIPTION

The **`clri`** command clears the inode *i-number* by filling it with zeros. *special* must be a special file name referring to a device containing a file system. For proper results, *special* should not be mounted (see WARNINGS below). After **`clri`** is executed, all blocks in the affected file show up as "missing" in an **`fsck`** of *special* (see *fsck*(1M)). This command should only be used in emergencies.

Read and write permission is required on the specified *special* device. The inode becomes allocatable.

WARNINGS

The primary purpose of this command is to remove a file that for some reason does not appear in any directory. If it is used to clear an inode that does appear in a directory, care should be taken to locate the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry in the directory will still point to that file. At that point, removing the old entry destroys the new file, causing the new entry to point to an unallocated inode, so the whole cycle is likely to be repeated again.

If the file system is mounted, **`clri`** is likely to be ineffective.

DEPENDENCIES

`clri` operates only on file systems of type **`hfs`**.

SEE ALSO

fsck(1M), fsdb(1M), ncheck(1M), fs(4).

STANDARDS CONFORMANCE

clri: SVID2, SVID3

NAME

clrsvc - clear x25 switched virtual circuit

SYNOPSIS

clrsvc *line pad-type*

DESCRIPTION

clrsvc clears any virtual circuit that might be established on the specified *line*. *pad-type* indicates to **clrsvc** what **opx25** script to run from **/usr/lbin/uucp/X25**.

DEPENDENCIES

HP 2334A is the only PAD supported at this time, and results in an **opx25** execution of **HP2334A.clr**.

EXAMPLES

A typical invocation is:

```
/usr/lbin/uucp/X25/clrsvc /dev/x25.1 HP2334A
```

AUTHOR

clrsvc was developed by HP.

SEE ALSO

getx25(1M), opx25(1M), getty(1M), login(1), uucp(1).

NAME

config - configure and build an HP-UX system

SYNOPSIS

/usr/sbin/config [**-c** *c_file*] [**-l** *m_file*] [**-m** *master*] [**-r** *path*] [**-s**|**-u**] [**-S**] [**-t**] *system_file*

/usr/sbin/config -M *module_name* [[**-M** *module_name*]...] [**-m** *master*] [**-u**]

DESCRIPTION

config is used to configure the following parts of the operating system:

- device drivers
- swap and dump devices
- tunable system parameters
- kernel modules

config supports the following configurations:

- whole kernel configuration (first form)

  Both the static kernel (**vmunix**) and dynamically loadable modules are generated, and a system reboot is necessary.

- dynamically loadable module configuration (second form)

  Specified loadable modules are dynamically generated and registered with the current system. The newly configured services are available immediately, without requiring a system reboot.

Kernel modules can either be static modules or dynamically loadable modules.

The first form is used to configure the entire kernel; that is, the static kernel and all kernel modules. This type of configuration is called a whole kernel configuration. The second form is used to configure only the dynamically loadable modules.

Static modules are maintained in individual object files which are included or excluded from the static kernel (**vmunix**) based on whether the features they support are required in the system. Such modules are non-loadable and remain linked into the kernel.

Dynamically loadable modules are also maintained in individual object files but they are not statically linked into the kernel. Loadable modules can be configured to be included or excluded from the kernel dynamically, without having to relink the entire kernel or reboot the system. The loadable image generated during the configuration of such modules may be auto-loaded or unloaded by the kernel or demand-loaded or unloaded by the system administrator.

See the *Managing Systems and Workgroups* for information on how to include or remove a subsystem, file system, or kernel module, and how to boot the system.

Whole Kernel Configuration (First Form)

To configure a whole kernel, **config** reads the user-provided description of an HP-UX system (*system_file*), the system description files for kernel modules, and the master kernel configuration table information.

Note that the system file and system description files for kernel modules should only be modified by using the **kmsystem** or **kmtune** system administration commands.

For all kernel modules to be configured, **config** checks the interface functions or symbols used by the modules. If modules rely on symbols not covered by the **$INTERFACE** section of its master file, configuration fails. Otherwise, **config** generates the following output files and directories:

- C program source files (**conf.c** and **space.h**) that define the configuration tables for various parts of the system. Unless kernel modules are configured, these files will not be generated.
- C program header file (**tune.h**) that defines tunable parameters of the system required by kernel and kernel modules.
- C program source files (**mod_conf.c**) that are required by kernel modules. If a **space.h** header file is provided with a module, it is included by the source file.
- a makefile (**config.mk**) to compile the C program produced and relink the newly configured system with statically linked kernel module object file (**vmunix_test**), and to generate kernel

symbol table (**symtab**).

- another makefile (**config.mod**) to generate all dynamically loadable modules to be configured.
- a directory (**dlkm.vmunix_test**) to store the generated dynamically loadable modules, kernel symbol table, and module registry file associated with the kernel being built (**vmunix_test**). This directory here after will be referred to as the kernel function set directory. The files in this directory will be referred to as the kernel function set files.

Many header files are needed to compile **conf.c**. Also, archive library files containing the kernel objects are needed to link the kernel. These files are supplied with the system and are contained in the directories found under **/usr/conf**.

config.mod and the module registry file are not generated if there are no dynamically loadable modules being configured.

config executes the **make** command to compile **conf.c**, to link the kernel with the appropriate kernel libraries and statically linked modules, and to generate the kernel symbol table. It also executes the **make** command with **config.mod** to compile dynamically loadable modules.

The **make** command create several files in a working directory whose location depends on the name of the system file. If *system_file* is **/stand/system**, the working directory is **/stand/build**; otherwise the working directory is the current directory. With successful completion of the **make** command, the following files are generated:

- kernel file

  The kernel file **vmunix_test** is generated in the working directory.
- kernel function set directory

  The kernel function set directory (**dlkm.vmunix_test**) is created in the working directory.
- kernel symbol table

  The kernel symbol table **symtab** is generated in the kernel function set directory.
- dynamically loadable modules

  Dynamically loadable modules are generated under a subdirectory (**mod.d**) of the kernel function set directory.

If the **-u** option is specified, the newly generated kernel file and its kernel function set directory are automatically copied to their default locations, **/stand/vmunix** and **/stand/dlkm**, respectively, on system shutdown or restart. The previous kernel file and its kernel function set directory will be saved as **/stand/vmunix.prev** and **/stand/dlkm.vmunix.prev**, respectively.

Options for Whole Kernel Configuration

When configuring a whole kernel, the **config** command recognizes the following arguments:

-c *c_file*
: Specify the name of the C program source file produced by **config**. The default file name is **conf.c**.

-l *m_file*
: Specify the name of the makefile which is generated by **config**. This is the makefile which will be used by **config** to compile the C program source file and make the new kernel. The default file name is **config.mk**.

-m *master*
: Specify the name of the master kernel configuration information file or directory that **config** should use in creating source files and makefiles. If *master* is a directory, **config** reads all files in that directory to create its data structures. If *master* is a file, only that file is read for creating data structures for **config**. By default, **config** reads the files in the directory **/usr/conf/master.d**. **/usr/conf/master.d** is supplied as part of the HP-UX operating system and should not be modified by anyone who does not fully understand its structure and purpose.

-r *path*
: Search the directory *path* for the libraries and header files needed for making the kernel. By default, **config** uses the directory **/usr/conf**.

-S Statically link all kernel modules into the kernel file. This option only takes effect if kernel modules are configured as loadable.

-s Stop after generating source files and makefiles. **make** is not executed and no kernel (**vmunix_test**) or kernel modules are created. The **-s** option cannot be used with the **-u** option.

-t Give a short table of major device numbers for the character and block devices, the card drivers, the streams drivers and modules that require link routines, the streams devices and the streams modules named in *system_file*. These tables may be useful when creating special device files.

-u Invoke **kmupdate** after successfully configuring the new kernel environment. The **-u** option cannot be used together with the **-s** option.

system_file
The file containing configuration information for the user's system. The default system file is **/stand/system** and when this file is used as input to **config**, the resulting output is placed in the directory **/stand/build**. If a file other than **/stand/system** is used for *system_file*, **config** places its output files in the current directory. The system file is divided into two parts: the first part (mandatory) contains driver specifications; the second part (optional) contains system-dependent information.

Constructing an HP-UX System File

The first part of *system_file* is used to configure:

- device drivers
- pseudo-drivers
- subsystems

Each line has the following format:

devname where *devname* is the driver or subsystem name as it appears in the alias tables, driver install tables or the device tables in the files in the directory, **/usr/conf/master.d**. For example, **scsi** selects the driver for SCSI disk drives, **scsitape** selects the driver for SCSI tape drives, and **nfs** selects the NFS subsystem. Together, the files in **/usr/conf/master.d** contain a complete list of configurable devices, cards, subsystems, and pseudo-drivers.

The optional second part of *system_file* is used to:

- define the swap device
- define the dump device(s)
- provide a mapping of a driver to a hardware path
- define status and values of selected system parameters.

Lines are constructed as indicated below for each category.

(1) *Swap device specification*
No more than one swap specification is allowed. If a swap specification is not given, the system will be configured to swap on the root device at the end of the filesystem.

swap *hw_path offset* [*blocks*]
Configure the swap device location and its size as specified. Arguments are interpreted as follows:

hw_path The hardware path representing the device to configure as the swap device or the string default may be used to indicate using the root device.

offset The swap area location. Boundaries are located at 1K-byte intervals. A negative value specifies that a file system is expected on the device. At boot-up, the super block is read to determine the exact size of the file system, and this value is put in *offset*. If the swap device is auto-configured, this is the mechanism used. If the super block is invalid, the entry will be skipped so that a corrupted super block will not later cause the entire file system to be corrupted by configuring the swap area on top of it. A positive or zero value for *offset* specifies the minimum area that must be reserved. Zero means to reserve no area at the head of the device. A zero value implies that there is no file system on the device.

blocks The number (in decimal) of 1K-byte disk blocks in the swap area. For this swap device specification, only the *blocks* parameter is optional. Zero is the default for auto-configuration. If *blocks* is zero, the entire remainder of the device is automatically configured in as swap area. If *blocks* is non-zero, its absolute value is treated as an upper bound for the size of the swap area. Then, if the swap area size has actually been cut back, the sign of *blocks* determines whether *blocks* remains as is, resulting in the swap area being adjacent to the reserved area, or whether *blocks* is bumped by the size of the unused area, resulting in the swap area being adjacent to the tail of the device.

swap *hw_path options*
Configure the swap device at the location specified using the options specified. The *hw_path* argument is interpreted as it is in the previous example.

The *options* field is used to specify a section. It is only offered for backwards compatibility purposes. For example, **s3** would put the swap area on section 3.

swap lvol
Configure swap on a logical volume.

swap none
Configure the kernel with no swap device.

(2) *Dump device(s) specification*
One or more dump specifications are allowed. If a dump specification is not given, then the primary swap area will be used.

dump *hw_path* [*options*]
Configure the dump device location and its size as specified. Arguments are interpreted as follows:

hw_path The hardware path representing the device to configure as a dump device or the string default may be used to indicate using the primary swap area.

options This field is used to specify a section. It is only offered for backwards compatibility purposes. For example **s3** would put the dump area at section 3.

dump lvol
Configure dump on a logical volume.

dump none
Configure the kernel with no dump device.

(3) *Device driver to hardware path*
One or more driver to hardware path specifications is allowed. If a driver statement is provided, the specified software module is forced into the kernel I/O system at the given hardware path. This can be used to make the system recognize a device that could not be recognized automatically.

driver *hw_path driver_name*
Bind the driver into the kernel I/O system at the given hardware path. Arguments are interpreted as follows:

hw_path The hardware path representing the device to bind the software with.

driver_name
The name of the software module to bind into the kernel at the specified hardware path.

(4) *System parameters*
These parameters should not be modified without a full understanding of the ramifications of doing so (see the *Managing Systems and Workgroups* manual).

Each line contains two fields. The first field can contain up to 20 characters; the second field up to 60 characters. Each line is independent, optional, and written in the following format:

parameter_name number or formula

Interprocess communication consists of messages (**mesg**), semaphores (**sema**) and shared memory (**shmem**) features. If **mesg**, **sema**, and/or **shmem** are specified as 0, the kernel code for these features is not included. If they are specified as 1, the kernel code is included; this is the default. The features can be specified independent of each other. If the code is included, the parameters listed below can be modified:

mesg	**1**
msgmap	*number or formula*
msgmax	*number or formula*
msgmnb	*number or formula*
msgmni	*number or formula*
msgseg	*number or formula*
msgssz	*number or formula*
msgtql	*number or formula*
sema	**1**
semaem	*number or formula*
semmap	*number or formula*
semmni	*number or formula*
semmns	*number or formula*
semmnu	*number or formula*
semume	*number or formula*
semvmx	*number or formula*
shmem	**1**
shmall	*number or formula*
smbrk	*number or formula*
shmmax	*number or formula*
shmmin	*number or formula*
shmmni	*number or formula*
shmseg	*number or formula*

Dynamically Loadable Module Configuration (Second Form)

To configure loadable kernel modules, **config** builds components for the module specified by the **-M** option. If the **-M** option is specified in conjunction with the **-u** option, then **config** builds the loadable module and call upon **kmupdate** to update the loadable image of that module in memory. Updating the loadable image implies replacing the existing loadable image with the newly created loadable image, re-registering the module with the new information, if required, and performing any type-specific initialization; e.g. recreating the special device file, if needed.

When configuring loadable modules, **config** reads the running kernel's system description file, system description files for kernel modules, and the master kernel configuration information table.

Note that system description files for kernel modules should only be modified by using **kmsystem** or **kmtune** system administration commands.

To configure loadable modules, **config** checks the interface functions or symbols used by the modules. If the modules rely on symbols not covered by the **$INTERFACE** section of its master file, configuration fails. **config** then generates the following output files:

- C program header file (**tune.h**) that defines tunable parameters of the system.
- C program source file (**mod_conf.c**) that is required by each kernel module.
- makefile (**config.mod**) to generate specified dynamically loadable modules.
- module registry entry to register the specified modules.

After the above files have been generated, **config** executes the **make** command with **config.mod** to generate dynamically loadable module.

With a successful **make**, the object files of dynamically loadable modules are generated and placed under the kernel function set directory.

If the **-u** option is specified, **kmupdate** is executed by **config**.

All kernel module related files are needed to configure the module. See *kminstall*(1M) for details on kernel module files.

Options for Loadable Module Configuration

When configuring a loadable module, **config** recognizes the following options:

-M *module_name*
Configure the specified loadable module only. A kernel file is not generated in this case. If

successful, the loadable image of the module is generated.

If the specified module is a stub module (see *master*(4)), **config** prints a message and fails. An entire kernel build is required to configure stub modules.

-m *master*
Specify the name of the master kernel configuration information file or directory that **config** should use in creating source files and makefiles. If *master* is a directory, **config** reads all files in that directory to create its data structures. If *master* is a file, only that file is read for creating data structures for **config**. By default, **config** reads the files in the directory **/usr/conf/master.d**. **/usr/conf/master.d** is supplied as part of the HP-UX operating system and should not be modified by anyone who does not fully understand its structure and purpose.

-u Invoking **kmupdate** to update the module.

Kernel Module System Description File

Kernel module description files are placed under **/stand/system.d**. A system file for a module is named after the module name and is unique.

Each file consists of three mandatory and one optional sections.

$VERSION:

The line starting with **$VERSION** indicates the version number for the file format. Version is defined as a decimal number and starts from one.

Format is:

$VERSION *version_number*

Example:

$VERSION 1

$CONFIGURE:

The line starting with **$CONFIGURE** indicates whether the module needs to be configured. If the second field is either **Y** or **y**, the module will be configured on the next build. If the field is either **N** or **n**, the module will not be configured on the build.

Format is:

$CONFIGURE {**Y**|**y**|**N**|**n**}

Example:

$CONFIGURE Y

$LOADABLE:

The line starting with **$LOADABLE** indicates how the module will be configured. If the second field is either Y or y, the module will be configured as a dynamically loadable module.

If the field is either **N** or **n**, the module will be statically linked into the kernel.

If the master file for the module does not have a **$LOADABLE** section, then the system file should not have one either.

Format is:

$LOADABLE {**Y**|**y**|**N**|**n**}

Example:

$LOADABLE Y

$TUNABLE (Optional system parameter section)

The section between the lines starting with **$TUNABLE**, and with **$$$** indicates tunable parameters of the module.

The above mentioned keywords e.g. **$VERSION**, **$CONFIGURE** must start at the beginning of the line without white space or tabs. Field separators can be single white spaces, tabs, or a combination of both.

Lines starting with an asterisk (**`*`**) are comment lines

RETURN VALUE

`config` returns 0 upon successful completion. If an error occurs, a non-zero value is returned.

DIAGNOSTICS

All error messages and warning messages of **`config`** are sent to stderr. Status report messages are sent to stdout. These messages are self explanatory. Some messages are generated by **`make`** or commands called from the makefiles.

FILES

`/usr/conf/master.d/*`	Default input master configuration tables
`/usr/conf/interface.d/*`	Interface files
`/usr/conf/gen/config.sys`	Contains skeleton makefile
`/usr/conf/gen/config.lm`	Contains skeleton makefile for kernel modules
`/stand/system`	Default system file
`/stand/system.d/*`	Default kernel module description files
`/stand/build/conf.c`	Default output configuration table
`/stand/build/tune.h`	Default output system parameter table
`/stand/build/config.mk`	Default output *make*(1) script
`/stand/build/config.mod`	Default kernel module *make*(1) script
`/stand/build/vmunix_test`	Default kernel made by **`config`**
`/stand/build/dlkm.vmunix_test/symtab`	Default kernel symbol table
`/stand/build/dlkm.vmunix_test/mod.d/*`	Default kernel module loadable image
`/stand/build/dlkm.vmunix_test/mod_register`	Default module registry file

SEE ALSO

kminstall(1M), kmmodreg(1M), kmsystem(1M), kmtune(1M), kmupdate(1M), make(1), interface(4), master(4).

NAME

/usr/newconfig/etc/mail/convert_awk - converts old sendmail.cf files to new format.

SYNOPSIS

convert_awk

DESCRIPTION

convert_awk is an **awk** program that will convert pre-HP-UX 10.20 **sendmail.cf** files into the format required by the HP-UX 10.20 **sendmail** (**sendmail** 8.7 and up).

To run it, use:

> **awk -f convert_awk <** *old.cf* **>** *new.cf*

Note that the new sendmail.cf files offer a wealth of new options and features. You should STRONGLY consider making a new **sendmail.cf** file from the distribution version or from the **m4** macros, which are provided in HP-UX 10.20 in **/usr/newconfig/etc/mail/cf**.

SEE ALSO

sendmail(1M).

NAME

convertfs - convert an HFS file system to allow long file names

SYNOPSIS

/usr/sbin/convertfs [**-q**] [*special-file*]

DESCRIPTION

The **convertfs** command converts an existing HFS file system supporting the default maximum file name length of 14 characters into one that supports file names up to 255 characters long. Once an HFS file system is converted to long file names, it cannot be restored to its original state, since the longer file names require a directory representation that is incompatible with the default HFS directory format. Since this is an irreversible operation, **convertfs** prompts for verification before it performs a conversion.

convertfs forces the system to reboot if the root file system is converted. When converting the root file system, the system should be in single-user mode, with all unnecessary processes terminated and all non-root file systems unmounted. Except for the root file system, **convertfs** requires that the file system to be converted be unmounted.

If invoked without arguments, **convertfs** interactively prompts the user with a list of the HFS file systems from **/etc/fstab**. One or more or all of the listed file systems can be selected for conversion. Typically, it is desirable to convert all of the file systems in **/etc/fstab** to avoid inconsistencies between two file systems mounted on the same system.

convertfs can also be invoked with an argument of either a block or character *special-file* of a file system to be converted. Only the block special file should be specified for a mounted root file system.

As part of the conversion process, **convertfs** performs an **fsck** on each file system (see *fsck*(1M)).

Options

-q Do quietly. **convertfs** will perform the conversions without querying the user. Normally **convertfs** prompts the user before converting a file system.

RETURN VALUE

convertfs returns the following values:

0 Success. Either **convertfs** successfully converted the file system, or the file system already allowed long file names.

non-0 Failure. **convertfs** was not able to convert the file system due to some failure in processing.

AUTHOR

convertfs was developed by HP.

FILES

/etc/fstab	Default list of file systems to check.

SEE ALSO

fsck(1M), mkfs(1M), newfs(1M), fs(4), fstab(4).

NAME

cpset - install object files in binary directories

SYNOPSIS

cpset [**-o**] *object directory* [**-** *mode* [**-** *owner* [**-** *group*]]]

DESCRIPTION

The **cpset** command installs the specified *object* file in the given *directory*. The *mode*, *owner*, and *group*, of the destination file can be specified on the command line. If this data is omitted, two results are possible:

- If you have administrative permissions (that is, your numerical ID is less than 100), the following defaults are provided:

mode	**0555**
owner	**bin**
group	**bin**

- If you do not have administrative permissions, the default *mode*, *owner*, and *group* of the destination file are the same as yours.

The **-o** option forces **cpset** to move *object* to **OLD***object* in the destination directory before installing the new object.

cpset reads the **/etc/src/destinations** file to determine the final destination of the file to be installed. The **destinations** file contains pairs of path names separated by spaces or tabs. The first name is the "official" destination (for example: **/usr/bin/echo**). The second name is the new destination. If **echo** is moved from **/usr/bin** to **/usr/local/bin**, the entry in **destinations** would be:

```
/usr/bin/echo   /usr/local/bin/echo
```

When the actual installation happens, **cpset** verifies that the "old" pathname does not exist. If a file exists at that location, **cpset** issues a warning and continues.

This file does not exist on a distribution tape; it is used by sites to track local command movement. The procedures used to build the source are responsible for defining the "official" locations of the source.

Cross Generation

The environment variable **ROOT** is used to locate the destination file (in the form **$ROOT/etc/src/destinations**). This is necessary in the cases where cross generation is being done on a production system.

EXAMPLES

If you are an administrator, all of the following examples have the same effect. They copy file **echo** into **/usr/bin** with *mode*, *owner*, and *group* set to **0555**, **bin**, **bin**, respectively:

```
cpset echo /usr/bin 0555 bin bin
cpset echo /usr/bin
cpset echo /usr/bin/echo
```

If you are not an administrator, the last two examples set *mode*, *owner*, and *group* to your current values.

SEE ALSO

chacl(1), make(1), install(1M), acl(5).

NAME

crashconf - configure system crash dumps

SYNOPSIS

/sbin/crashconf [**-arv**] [**-i**|**-e** *class*] ... [*device*...]

DESCRIPTION

crashconf displays and/or changes the current system crash dump configuration. The crash dump configuration consists of three lists:

- The *crash dump device* list. This list identifies all devices that can be used to store a crash dump.
- The *included class* list. This list identifies all system memory classes that *must* be included in any crash dump.
- The *excluded class* list. This list identifies all system memory classes that *should not* be included in a crash dump.

Most system memory classes are in neither the included class list nor the excluded class list. Instead, the system determines whether or not to dump those classes of memory based on the type of crash that occurs.

Note that certain types of system crash, such as TOC's, require a full crash dump. Also, the system operator may request a full crash dump at the time the dump is taken. In either of these cases, a full dump will be performed regardless of the contents of the excluded class list.

Any changes to the configuration take effect immediately and remain in effect until the next system reboot, or until changed with a subsequent invocation of **crashconf**.

device specifies a block device file name of a device that is a valid destination for crash dumps. All such devices listed on the command line will be added to the end of the current list of crash dump devices, or will replace the current list of crash dump devices, depending on whether **-r** is specified.

class is the name (or number) of a system memory class which should be added to the appropriate class list. The list of system memory classes can be obtained using **crashconf -v**.

class may also be the word **all**, in which case all classes are added to the appropriate list. (The effect of adding all classes to the included class list is to force full crash dumps under all circumstances. The effect of adding all classes to the excluded class list is to disable crash dumps.)

Options

-a The file **/etc/fstab** is read, and all dump devices identified in it will be added to (or will replace) the current list of crash dump devices. This is in addition to any crash dump *device*s specified on the command line. See *fstab*(4) for information on the format of **/etc/fstab**.

-e The *class*es specified with **-e** will be added to (or will replace) the list of excluded (i.e., should not dump) classes. If any of those classes are present in the current included class list, they will be removed from it.

-i The *class*es specified with **-i** will be added to (or will replace) the list of included (i.e., must dump) classes. If any of those classes are present in the current excluded class list, they will be removed from it.

-r Specifies that any changes should replace, rather than add to, the current configuration. Thus, if *device*s or **-a** are specified, the current crash dump device list is replaced with new contents; if *class*es are specified with **-e**, they replace the list of currently excluded classes, and if *class*es are specified with **-i**, they replace the list of currently included classes.

-v Displays the current crash dump configuration. This is the default option if no arguments are specified. If any changes to the current configuration are specified on the same command line as **-v**, the configuration will be displayed *after* the requested changes are made.

RETURN VALUE

Upon exit, **crashconf** returns the following values:

0	Success.
1	The requested configuration changes could not be made.

WARNINGS

The output of **crashconf** is not designed to be parsed by applications or scripts, but only to be read by humans. The output format may change without notice. Applications which require crash dump configuration information should retrieve that information using *pstat*(2).

Dump devices created by *lvcreate*(1M) must be contiguous (**-Cy** option) with bad block relocation turned off (**-rn** option).

AUTHOR

crashconf was developed by HP.

SEE ALSO

lvcreate(1M), crashconf(2), pstat(2), fstab(4).

NAME

crashutil - manipulate crash dump data

SYNOPSIS

/usr/sbin/crashutil [**-q**] [**-v** *version*] *source* [*destination*]

DESCRIPTION

crashutil copies and preserves crash dump data, and performs format conversions on it. Common uses of **crashutil** include:

- Copying portions of a dump that still reside on a raw dump device into a crash dump directory.
- Converting between different formats of crash dumps.
- Copying crash dumps from one directory, or medium, to another.

crashutil will write to its *destination* the crash dump it reads from its *source*. The crash dump format used to write the *destination* is specified with **-v**; if **-v** is not specified, the *destination* will have the same format as the *source*. If no *destination* is specified, *source* is used; the format conversion will be done in place in the *source*, without copying. When **crashutil** completes successfully, the entire contents of the crash dump will exist at *destination*; any portions that had still been on raw dump devices will have been copied to *destination*.

There are three known dump formats:

COREFILE (Version 0) This format, used up through HP-UX 10.01, consists of a single file containing the physical memory image, with a 1-to-1 correspondence between file offset and memory address. Normally there is an associated file containing the kernel image. *source*s or *destination*s of this type must be specified as two pathnames to plain files, separated by whitespace; the first is the core image file and the second is the kernel image file.

COREDIR (Version 1) This format, used in HP-UX 10.10, 10.20, and 10.30, consists of a **core.***n* directory containing an **INDEX** file, the kernel (**vmunix**) file, and numerous **core.***n***.***m* files, which contain portions of the physical memory image. *source*s or *destination*s of this type should be specified as the pathname to a core directory.

CRASHDIR
CURRENT (Version 2 — the current version) This format, used in HP-UX 11.00 and later, consists of a **crash.***n* directory containing an **INDEX** file, the kernel and all dynamically loaded kernel module files, and numerous **image.***m***.***p* files, each of which contain portions of the physical memory image and metadata describing which memory pages were dumped and which were not. *source*s or *destination*s of this type should be specified as the pathname to a crash directory.

Other formats, for example tape archival formats, may be added in the future.

When the *source* and *destination* are different types of files — for example, when *source* is a directory and *destination* is a pair of plain files — both must be specified.

Options

-q (Quiet) Disables the printing of progress messages. Warning and error messages are still printed.

-v *version* Specifies the version of the destination format. Allowed values are **COREFILE**, **COREDIR**, **CRASHDIR**, 0, 1, or 2. Also allowed is the keyword **CURRENT**, which specifies that the destination format should be the same as the current source format. **CURRENT** is the default if **-v** is not specified.

RETURN VALUE

Upon exit, **crashutil** returns the following values:

0 The operation was successful.
1 The operation failed, and an appropriate error message was printed.

EXAMPLES

An HP-UX 11.00 crash dump was saved by *savecrash*(1M) to **/var/adm/crash/crash.2**. The **-p** flag was specified to *savecrash*, specifying that only those portions of the dump which were endangered by swap activity should be saved; the rest are still resident in the raw dump devices. To save the remainder of the dump into the crash dump directory, use:

```
crashutil /var/adm/crash/crash.2
```

If preferred, the completed crash dump directory could be in a different location — perhaps on another machine via NFS:

```
crashutil /var/adm/crash/crash.2 /nfs/remote/otherdir
```

To debug this crash dump using tools which do not understand the most current crash dump format, convert it to the older core directory format:

```
crashutil -v COREDIR /var/adm/crash/crash.2 /tmp/oldcoredir
```

or the even older "core file and kernel" format:

```
crashutil -v COREFILE /var/adm/crash/crash.2 /tmp/corefile
/tmp/kernfile
```

AUTHOR

crashutil was developed by HP.

SEE ALSO

savecrash(1M).

NAME

create_sysfile - create a kernel system file

SYNOPSIS

/usr/lbin/sysadm/create_sysfile [*outfile*]

DESCRIPTION

The **create_sysfile** command creates a kernel generation description file (system file) which can be used as input to the command **config**. The system file is built according to the drivers required by the current system hardware. This command is intended for use during the install process when the system does not have a system file.

The **create_sysfile** command first chooses a template file based on the CPU type of the machine, then it scans the system hardware and includes all drivers it can identify to run the existing hardware. If *outfile* is specified, the resulting system file is sent to *outfile*. If *outfile* is not specified, the output is placed in the file **/stand/system**.

RETURN VALUE

The **create_sysfile** command returns zero upon normal completion or **1** if an error occurred.

DIAGNOSTICS

Errors are sent to stderr. Most of the diagnostic messages from **create_sysfile** are self-explanatory. Errors cause **create_sysfile** to halt immediately.

AUTHOR

create_sysfile was developed by HP.

FILES

/usr/conf/gen/templates/*
/usr/conf/master.d/*

SEE ALSO

config(1M), master(4).

C

NAME

cron - timed-job execution daemon

SYNOPSIS

/usr/sbin/cron

DESCRIPTION

cron executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions placed in crontab files. Users can submit their own crontab files with a **crontab** command (see *crontab*(1)). Users can submit commands that are to be executed only once with an **at** or **batch** command.

Since **cron** never exits, it should be executed only once. This is best done by running **cron** from the initialization process with the startup script **/sbin/init.d/cron** (see *init*(1M)).

cron only establishes a schedule for crontab files and **at**/**batch** command files during process initialization and when it is notified by **at**, **batch**, or **crontab** that a file has been added, deleted, or modified.

When **cron** executes a job, the job's user and group IDs are set to those of the user who submitted the job.

Spring and Autumn Time Transitions

On the days of daylight savings (summer) time transition (in time zones and countries where daylight savings time applies), **cron** schedules commands differently from normal.

In the following description, an **ambiguous time** refers to an hour and minute that occurs twice in the same day because of a daylight savings time transition (usually on a day during the Autumn season). A **nonexistent time** refers to an hour and minute that does not occur because of a daylight savings time transition (usually on a day during the Spring season). **DST-shift** refers to the offset that is applied to standard time to result in daylight savings time. This is normally one hour, but can be any combination of hours and minutes up to 23 hours and 59 minutes (see *tztab*(4)).

When a command is specified to run at an ambiguous time, the command is executed only once at the *first* occurrence of the ambiguous time.

When a command is specified to run at a nonexistent time, the command is executed after the specified time by an amount of time equal to the DST-shift. When such an adjustment would conflict with another time specified to run the command, the command is run only once rather than running the command twice at the same time.

Commands that are scheduled to run during all hours (there is a ***** is in the hour field of the crontab entry) are scheduled without any adjustment.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is set to the empty string, it defaults to "C" (see *lang*(5)). If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

DIAGNOSTICS

A history of all actions taken by **cron** is recorded in **/var/adm/cron/log**.

EXAMPLES

The following examples assume that the time zone is **MST7MDT**. In this time zone, the DST transition occurs one second before 2:00 a.m. and the DST-shift is 1 hour.

Consider the following entries in a crontab file:

```
# Minute    Hour    Month    Day    Month    Weekday Command
# ----------------------------------------------------------
    0       01        *        *        *       Job_1
    0       02        *        *        *       Job_2
    0       03        *        *        *       Job_3
    0       04        *        *        *       Job_4
    0       *         *        *        *       Job_hourly
    0       2,3,4     *        *        *       Multiple_1
    0       2,4       *        *        *       Multiple_2
```

For the period of 1:00 a.m. to 4:00 a.m. on the days of DST transition, the results will be:

Job	Times Run in Fall	Times Run in Spring
`Job_1`	01:00 MDT	01:00 MST
`Job_2`	02:00 MDT	03:00 MDT
`Job_3`	03:00 MST	03:00 MDT
`Job_4`	04:00 MST	04:00 MDT
`Job_hourly`	01:00 MDT	01:00 MST
	02:00 MDT	
	02:00 MST	
	03:00 MST	03:00 MDT
	04:00 MST	04:00 MDT
`Multiple_1`	02:00 MDT	
	03:00 MST	03:00 MDT
	04:00 MST	04:00 MDT
`Multiple_2`	02:00 MDT	03:00 MDT
	04:00 MST	04:00 MDT

WARNINGS

In the Spring, when there is a nonexistent hour because of daylight savings time, a command that is scheduled to run multiple times during the nonexistent hour will only be run once. For example, a command scheduled to run at 2:00 and 2:30 a.m. in the **MST7MDT** time zone will only run at 3:00 a.m. The command that was scheduled at 2:30 a.m. will not be run at all, instead of running at 3:30 a.m.

DEPENDENCIES

HP Process Resource Manager

If the optional HP Process Resource Management (PRM) software is installed and configured, jobs are launched in the initial process resource group of the user that scheduled the job. The user's initial group is determined at the time the job is started, not when the job is scheduled. If the user's initial group is not defined, the job runs in the user default group (**PRMID=1**). See *prmconfig*(1) for a description of how to configure HP PRM, and *prmconf*(4) for a description of how the user's initial process resource group is determined.

AUTHOR

cron was developed by AT&T and HP.

FILES

/var/adm/cron	Main **cron** directory
/var/spool/cron/atjobs	Directory containing **at** and **batch** job files
/var/spool/cron/crontabs	Directory containing crontab files
/var/adm/cron/log	Accounting information

SEE ALSO

at(1), crontab(1), sh(1), init(1M), queuedefs(4), tztab(4).

HP Process Resource Manager: prmconfig(1), prmconf(4) in *HP Process Resource Manager User's Guide*.

STANDARDS CONFORMANCE

cron: SVID2, SVID3

C

NAME

cuegetty - set terminal type, modes, speed, and line discipline for *cue*(1)

SYNOPSIS

/usr/sbin/cuegetty [**-L** *nls_language*] [**-T** *terminal_type*] [**-h**] [**-t** *timeout*] *line* [*speed*]

DESCRIPTION

The **cuegetty**, command, which is very similar to *getty*(1M), is the second process in the series, (*init-cuegetty-cue-work session*) that ultimately connects a user with the HP-UX CUE system. It is invoked by **init** to monitor the terminal lines configured on a system (see *init*(1M)). Each **cuegetty** process resets its process group using **setpgrp**, opens a particular terminal line, and usually sleeps in the **open()** until the machine senses a hardware connection for the terminal. When **open()** returns, **cuegetty** attempts to adapt the system to the terminal speed and type, and displays the contents of the **/etc/issue** file, if it exists. Lastly, **cuegetty** invokes **cue** which displays the Login screen and performs user validation (see *cue*(1)).

To start **cuegetty**, an entry for **cuegetty** should be placed in the **/etc/inittab** file. A typical CUE entry in the **/etc/inittab** file resembles the following:

```
cue:2:respawn:/usr/sbin/cuegetty -L fr_FR.roman8 -h tty0p1
```

See **/usr/newconfig/etc/cue.inittab** for an example **/etc/inittab** file. See *cue*(1) for more details on the CUE system.

Configuration Options and Arguments

cuegetty recognizes the following arguments:

line Name of a tty line in **/dev** to which **cuegetty** is to attach itself. **cuegetty** uses this string as the name of a file in the **/dev** directory to open for reading and writing. By default **cuegetty** forces a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed. However, when **cuegetty** is run on a direct port, **cuegetty** does not force a hangup on the line since the driver ignores changes to zero speed on ports open in direct mode (see *modem*(7)).

-L **nls_language** is used to set the language for the CUE login screens. If the message catalog, **cue.cat**, does not exist for **nls_language**, the default native language, C, is used.

-T **terminal_type** is used to specify the type of terminal that **cuegetty** will be initiated on. Allowed values are **vt320**, **vt100**, **wy60**, and **hp**. The default is **hp**.

-h Tells **cuegetty** not to force a hangup on the line before setting the speed to the default or specified speed.

-t *timeout* **cuegetty** exits if the open on the line succeeds and nothing is typed within *timeout* seconds.

speed A label to a speed and tty definition in the file **/etc/gettydefs**. This definition tells **cuegetty** at what speed to initially run, what the login message should look like, what the initial tty settings are, and what speed to try next should the user indicate that the speed is inappropriate (by typing a *break* character). The default *speed* is 300 baud.

When no optional arguments appear on the command line, **cuegetty** sets the terminal interface as follows:

- Interface *speed*: 300 baud
- Raw mode (awaken on every character)
- Echo suppressed
- Parity: either
- New-line characters: convert to carriage-return, line-feed pair
- Expand tabs on the standard output
- Type login message then read user's name, one character at a time

- If a null character (or framing error) is received, assumed it to be the result of the user pushing the "break" key. This causes **cuegetty** to attempt the next *speed* in the series. The series that **cuegetty** tries is determined by what it finds in **/etc/gettydefs**.

After interface set-up is complete, **cue** is started to accept and validate the user name and password.

WARNINGS

If a supported non-HP terminal (or an HP terminal such as HP 700/60 in VT320, VT100 or WYSE60 mode) is required to run **cuegetty**, make sure that a correct terminal type is specified using the **-T** option. For example, if you want to run **cuegetty** on a vt100 terminal, you should make an entry in the **/etc/inittab** file such as the following entry:

```
tty1:23:respawn:cuegetty -T vt100 -h tty1p1 9600
```

Absence of the **-T** option causes **cuegetty** to assume terminal to be a HP terminal which may then cause the terminal to behave incorrectly and may not even allow user to login.

DEPENDENCIES

cuegetty is available only on Series 800 systems, and is compatible only with the following terminals:

HP 700/92 HP 700/94 HP 2392 HP 2394 VT 100 WYSE 60

See *WARNINGS* if you intend to use a non-HP terminal (or an HP terminal such as HP 700/60 in VT320, VT100, or WYSE60 mode).

FILES

/etc/gettydefs	contains speed and terminal settings used by **cuegetty**
/etc/inittab	**init** reads this file to determine which processes to spawn
/etc/issue	contains issue identification data
/usr/newconfig/etc/cue.inittab	sample **inittab** file with **cuegetty** entry

SEE ALSO

cue(1), env(1), nlsinfo(1), getty(1M), init(1M), ioctl(2), gettydefs(4), inittab(4), environ(5), hpnls(5), lang(5), termio(7).

NAME

dcc - control read and write caching for HP SCSI disk array drives

SYNOPSIS

dcc [*options*] [*drive_list*] *device_file*

DESCRIPTION

dcc displays or changes the read-ahead caching status, and write-immediate reporting status of selected drives on the HP SCSI disk array referenced by *device_file*.

Options

-d	Display only. Displays the read-ahead caching and write immediate reporting status of all selected drives on the HP SCSI disk array. For HP C2430 disk array devices, the number and size of cache segments is displayed. This option cannot be used with any other option.
-r*on*	Read on. Enables read-ahead caching on all selected drives of the HP SCSI disk array. Can be used in combination with one of the write-immediate reporting options.
-r*off*	Read off. Disables read-ahead caching on all selected drives of the HP SCSI disk array. Can be used in combination with one of the write-immediate reporting options.
-w*on*	Write on. Enables write-immediate reporting on all selected drives of the HP SCSI disk array. Can be used in combination with one of the read-ahead caching options.
-w*off*	Write off. Disables write immediate reporting on all selected drives of the HP SCSI disk array. Can be used in combination with one of the read-ahead caching options.
-s*num_segments*	Set the number of cache segments. This option is unique to the HP C2430 disk array. The disk mechanism cache can be segmented into 1, 2, 4, 8 or 16 segments. The default is 2 segments. This option cannot be used with other options.
drive_list	Specify a set of drives. If this optional list is absent, the default set of affected drives is all drives attached to the controller. The list is in the form **c***X***i***Y*,... where *X* (a decimal number) represents SCSI channel number, and *Y* (a decimal number) represents the SCSI ID of the drive. Multiple drives in the list are separated by commas.

RETURN VALUE

dcc returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

ERROR MESSAGES

Errors can originate from problems with:

- **dcc**
- SCSI (device level) communications
- system calls

Error messages generated by dcc:

usage: dcc options [cXiY,...] <special>

An error in command syntax has occurred. Enter command again with the required arguments, in the order shown.

dcc: Arg out of range

One of the arguments is larger than its allowed maximum value (or smaller than its allowed minimum value), or is incorrect in form. Check the size, and form of each argument and make appropriate corrections.

dcc: device busy

To ensure that **dcc** does not modify a disk array that is being used by another process, **dcc** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

dcc: LUN does not exist
The addressed LUN is not known to the array controller.

dcc: LUN # too big
The LUN number, which is derived from the device special file name, is out of range.

dcc: Not a raw file
Utilities must be able to open the device file for raw access.

dcc: Not an HP SCSI disk array
The device is not an HP SCSI disk array.

dcc: Transfer length error
The amount of data actually sent to (or received from) the device was not the expected amount.

SCSI (device level) communication errors:
Sense data associated with the failed operation is printed.

Error messages generated by system calls:
dcc uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **dcc** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To display the status of read and write caching on all the drives of the disk array **/dev/rdsk/c2t2d0** on a Series 700:

```
dcc -d /dev/rdsk/c2t2d0
```

To enable write-immediate reporting on a list of drives on the disk array **/dev/rdsk/c2t2d0** on a Series 800:

```
dcc -won c2i0,c1i0,c5i0,c4i1 /dev/rdsk/c2t2d0
```

To disable read caching and write-immediate reporting on the drives of the disk array **/dev/rdsk/c2t4d0** on a Series 700:

```
dcc -roff -woff /dev/rdsk/c2t4d0
```

To set the number of cache segments on the HP C2430 disk array **/dev/rdsk/c2t2d0** to 4 on a Series 800:

```
dcc -s 4 /dev/rdsk/c2t2d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

dcc was developed by HP.

NAME

dcopy - copy HFS file system with compaction

SYNOPSIS

/usr/sbin/dcopy [**-d**] [**-f***fsize*[**:***isize*]] [**-F hfs**] [**-s***cyl***:***skip*] [**-v**] [**-V**] *source_fs destination_fs*

DESCRIPTION

The **dcopy** command copies an existing HFS file system (*source_fs*) to a new HFS file system (*destination_fs*), appropriately sized to hold the reorganized results. For best results, the source file system should be a raw device, and the destination file system should be a block device. Always run **dcopy** on unmounted file systems. (In the case of the root file system, copy it to a new minidisk.)

If no options are specified, **dcopy** copies files from *source_fs*, compressing directories by removing vacant entries and spacing consecutive blocks in a file by the optimal rotational gap. If options such as **-f** or **-s** are specified, the destination file system structure will be different from that of the source file system.

dcopy makes the destination file system identical to the source file system and preserves the pack and volume labels. Thus, to compress a file system without moving it, use **dcopy** to copy the files to another file system and the **dd** command to copy the file back (see *dd*(1)).

Directory compression is accomplished by running **dcopy** on the primary copy of the file system and allowing the modified directories to propagate to the other copies of the file system in the normal manner.

Options

dcopy recognizes the following options:

-d	Move subdirectories to the beginning of directories.
-f*fsize*[**:***isize*]	Specify the file system size (*fsize*) and inode-list size (*isize*) in blocks. If this option is not specified, the source file-system value is used.
-F hfs	Specify the HFS file system type. The type of a file system can be determined with the **fstyp** command (see *fstyp*(1M)). See DEPENDENCIES.
-s*cyl***:***skip*	Supply device information for creating the best organization of blocks in a file. *cyl* is the number of block per cylinder; *skip* is the number of blocks to skip.
-v	Report size of source and destination file system.
-V	Echo the completed command line, but performs no other actions. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows you to verify the command line.

EXAMPLES

dcopy can be executed with or without options. If no options are specified as in this example, the source and destination file systems are identical. Any differences between the two file systems lie only in the available disk space.

```
dcopy /dev/rdsk/c2d0s4 /dev/dsk/c2d0s5
```

If options are specified, expect a major difference between the source and destination file system structure:

```
dcopy -F hfs -f40960:260 -s45:5 -d /dev/rdsk/c2d0s4 /dev/dsk/c2d0s5
```

WARNINGS

dcopy produces invalid results if run on a mounted file system.

The figures specified in option arguments cannot be smaller than corresponding figures in the source file system.

DEPENDENCIES

dcopy only operates on HFS file systems.

AUTHOR

dcopy was developed by HP.

SEE ALSO

dd(1), fstyp(1M).

STANDARDS CONFORMANCE

dcopy: SVID3

d

NAME

devnm - device name

SYNOPSIS

/usr/sbin/devnm [*name* ...]

DESCRIPTION

For each *name* specified, the **devnm** command identifies the special file associated with the mounted file system where the named file or directory resides.

EXAMPLES

The command:

```
/usr/sbin/devnm /usr
```

produces:

```
/dev/dsk/c1d0s9 /usr
```

if **/usr** is mounted on **/dev/dsk/c1d0s9**.

FILES

/dev/dsk/*

/etc/mnttab Mounted file system table.

SEE ALSO

brc(1M).

STANDARDS COMPLIANCE

devnm: SVID2, SVID3

NAME

df (generic) - report number of free file system disk blocks

SYNOPSIS

/usr/bin/df [**-F** *FStype*] [**-befgiklnv**] [**-t|-P**] [**-o** *specific_options*] [**-V**] [*special*|*directory*]...

DESCRIPTION

The **df** command displays the number of free 512-byte blocks and free inodes available for file systems by examining the counts kept in the superblock or superblocks. If a *special* or a *directory* is not specified, the free space on all mounted file systems is displayed. If the arguments to **df** are path names, **df** reports on the file systems containing the named files. If the argument to **df** is a *special* of an unmounted file system, the free space in the unmounted file system is displayed.

Options

df recognizes the following options:

-b	Report only the number of kilobytes (KB) free.
-e	Report the number of files free.
-f	Report only the actual count of the blocks in the free list (free inodes are not reported).
-F *FStype*	Report only on the *FStype* file system type (see *fstyp*(1M)).
-g	Report the entire structure described in *statvfs*(2).
-i	Report the total number of inodes, the number of free inodes, number of used inodes, and the percentage of inodes in use.
-k	Report the allocation in kilobytes (KB).
-l	Report on local file systems only.
-n	Report the file system name. If used with no other options, display a list of mounted file system types.
-o *specific_options*	Specify options specific to each file system type. *specific_options* is a comma-separated list of suboptions intended for a specific *FStype* module of the command. See the file-system-specific manual entries for further details.
-P	Report the name of the file system, the size of the file system, the number of blocks used, the number of blocks free, the percentage of blocks used and the directory below which the file system hierarchy appears.
-t	Report the total allocated block figures and the number of free blocks.
-v	Report the percentage of blocks used, the number of blocks used, and the number of blocks free. This option cannot be used with other options.
-V	Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **df** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

EXAMPLES

Report the number of free disk blocks for all mounted file systems:

```
df
```

Report the number of free disk blocks for all mounted HFS file systems:

```
df -F hfs
```

Report the number of free files for all mounted NFS file systems:

```
df -F nfs -e
```

Report the total allocated block figures and the number of free blocks, for all mounted file systems:

```
df -t
```

Report the total allocated block figures and the number of free blocks, for the file system mounted as **/usr**:

```
df -t /usr
```

FILES

/dev/dsk/*	File system devices
/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table

SEE ALSO

du(1), df_*FStype*(1M), fsck(1M), fstab(4), fstyp(1M), statvfs(2), mnttab(4).

STANDARDS CONFORMANCE

df: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

df - report number of free CDFS, HFS, or NFS file system disk blocks

SYNOPSIS

/usr/bin/df [**-F** *FStype*] [**-befgiklntv**] [**-B**] [**-o** *specific_options*] [**-V**] [*special* | *directory*]...

DESCRIPTION

The **df** command displays the number of free 512-byte blocks and free inodes available for file systems by examining the counts kept in the superblock or superblocks. If a *special* or a *directory* is not specified, the free space on all mounted file systems is displayed. If the arguments to **df** are path names, **df** reports on the file systems containing the named files. If the argument to **df** is a *special* of an unmounted file system, the free space in the unmounted file system is displayed.

Options

df recognizes the following options:

-b Report only the number of kilobytes (KB) free.

-B Report the total number of blocks allocated for swapping to the file system as well as the number of blocks free for swapping to the file system. This option is supported on HFS file systems only.

-e Report the number of files free.

-f Report only the actual count of the blocks in the free list (free inodes are not reported). When this option is specified, **df** reports on raw devices.

-F *FStype* Report only on the *FStype* file system type (see *fstyp*(1M)). For the purposes of this manual entry, *FStype* can be one of **cdfs**, **hfs**, and **nfs**, for the CDFS, HFS, and NFS file systems, respectively.

-g Report the entire structure described in *statvfs*(2).

-i Report the total number of inodes, the number of free inodes, number of used inodes, and the percentage of inodes in use.

-k Report the allocation in kilobytes (KB).

-l Report on local file systems only.

-n Report the file system name. If used with no other options, display a list of mounted file system types.

-o *specific_options*
Specify options specific to the HFS file system type. *specific_options* is a comma-separated list of suboptions.

The available suboption is:

i Report the number of used and free inodes.

-t Report the total allocated block figures and the number of free blocks.

-v Report the percentage of blocks used, the number of blocks used, and the number of blocks free. This option cannot be used with other options.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

When **df** is used on an HFS file system, the file space reported is the space available to the ordinary user, and does not include the reserved file space specified by **fs_minfree**.

Unreported reserved blocks are available only to users who have appropriate privileges. See *fs*(4) for information about **fs_minfree**.

When **df** is used on NFS file systems, the number of inodes is displayed as –1 . This is due to superuser access restrictions over NFS.

EXAMPLES

Report the number of free disk blocks for all mounted file systems:

```
df
```

Report the number of free disk blocks for all mounted HFS file systems:

```
df -F hfs
```

Report the number of free files for all mounted NFS file systems:

```
df -F nfs -e
```

Report the total allocated block figures and the number of free blocks, for all mounted file systems:

```
df -t
```

Report the total allocated block figures and the number of free blocks, for the file system mounted as /usr:

```
df -t /usr
```

WARNINGS

df does not account for:

- Disk space reserved for swap space,
- Space used for the HFS boot block (8K bytes, 1 per file system),
- HFS superblocks (8K bytes each, 1 per disk cylinder),
- HFS cylinder group blocks (1K-8K bytes each, 1 per cylinder group),
- Inodes (currently 128 bytes reserved for each inode).

Non-HFS file systems may have other items that this command does not account for.

The **-b** option, from prior releases, has been replaced by the **-B** option.

FILES

/dev/dsk/*	File system devices.
/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table

SEE ALSO

du(1), df(1M), fsck(1M), fstab(4), fstyp(1M), statvfs(2), fs(4), mnttab(4).

STANDARDS CONFORMANCE

df: SVID2, XPG2, XPG3

NAME

df (vxfs) - report number of free disk blocks on a VxFS file system

SYNOPSIS

/usr/bin/df [**-F vxfs**] [**-V**] [**-egiklnvtfb**] [**-o** *specific_options*]
[*special* | *directory* ...]

DESCRIPTION

The **df** command prints the number of free 512-byte blocks and free inodes available for file systems by examining the counts kept in the superblock or superblocks. If a *special* or a *directory* is not specified, the free space on all of the mounted file systems is printed. If the arguments to **df** are pathnames, **df** produces a report on the file system containing the named file. If the argument to **df** is a *special*, the file system can be an unmounted or mounted file system.

On a Version 1 or 2 disk, layout extents smaller than 8 Kbytes may not be usable for all types of allocation, so **df** does not count free blocks in extents below 8 Kbytes when reporting the total number of free blocks.

On a Version 2 or 3 disk layout, VxFS dynamically allocates inodes from the pool of free blocks, so the number of free inodes and blocks reported by **df** is an estimate based on the number of free extents and the current ratio of allocated inodes to allocated blocks. Allocating additional blocks may therefore decrease the count of free inodes, and vice versa.

Options

df recognizes the following options:

-b Report only the number of kilobytes free.

-e Report the number of files free.

-f Report only an actual count of the blocks in the free list (free inodes are not reported). When this option is specified, **df** reports on raw devices.

-F vxfs Specifies the file system type (**vxfs**).

-g Report the entire statvfs(2) structure.

-i Report the total number of inodes, the number of free inodes, number of used inodes and the percentage of inodes in use.

-k Report the allocation in Kbytes.

-l Report on local file systems only.

-n Report the file system name. If invoked with no other options this option prints a list of mounted file system types.

-o *specific_options*
Specifies options specific to the vxfs file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for the vxfs-specific module of the command.

The available option is

s Print the number of free extents of each size. Free extents are always an integral power of 2 in length, ranging from a minimum of 1 block to the maximum extent size supported by the file system.

-t Report the total allocated block figures and the number of free blocks.

-v Report the percentage of blocks used, the number of blocks used and the number of blocks free. This option cannot be used with other options.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

There are a number of options that specify output formats, some combinations of which are incompatible. If an incompatible combination is specified, one of the options will override the other(s).

EXAMPLES

Report the number of free disk blocks for all mounted file systems:

```
df
```

Report the number of free extents of each size, for all mounted VxFS file systems:

```
df -F vxfs -o s
```

Report the number of free files for all mounted VxFS file systems:

```
df -F vxfs -e
```

Report the total allocated block figures and the number of free blocks, for all mounted file systems:

```
df -t
```

Report the total allocated block figures and the number of free blocks, for the file system mounted as **/usr**:

```
df -t /usr
```

FILES

/dev/vg00/*	File-system devices.
/dev/dsk/*	File-system devices.
/etc/fstab	Static information about the file systems.
/etc/mnttab	mounted-file-system table.

SEE ALSO

du(1), fsck(1M), fs(4), mnttab(4), statvfs(2), df(1M).

STANDARDS CONFORMANCE

df : SVID2, XPG2, XPG3

NAME

dhcpdb2conf - DHCP client database converter

SYNOPSIS

dhcpdb2conf [*dhcpdb2conf_options*] [*lan_interfaces*]

DESCRIPTION

dhcpdb2conf provides a means of translating a client DHCP database into a set of standard configuration file variables. A DHCP client database can contain settings for such items as, IP address, hostname, and default gateway. Using **dhcpdb2conf**, you can simply list the contents of the database to the screen, create a set of configuration staging files, or execute direct edits on existing configuration files using the values contained in the client database.

Options

dhcpdb2conf allows you to specify a list of interfaces on the command line (e.g. lan0 lan1 ...). If no lan interface is specified, **dhcpdb2conf** will process all entries referenced in the client database. The entries themselves are defined as a unique lan interface and a list of attributes which correspond to that interface. The attributes can be selected for processing by specifying one or more filter flags on the command line. Each filter flag may be combined with any other filter flag(s). If no filter flag is specified, all the attributes for a lan interface will be processed. The following options are supported:

-a	Using the results of the specified filter, directly apply the variable defintions to the existing configuration files (for example, **/etc/rc.config.d/netconf**).
-c	Create a set of staging files using the results of the selected filter(s). Each variable processed will be applied to its corresponding configuration file. Specifically, **dhcpdb2conf** will generate a copy of the existing configuration file. As an example, **/etc/rc.config.d/netconf** will be copied to **/etc/rc.config.d/netconf.dhcp**. Once this staging file has been created, the variable that is being processed will be applied to the newly created file. *WARNING*: Using the **-c** option will override any existing values which are currently set in the system's configuration files.
-d	Process the DNS variable set: [domain, nameserver]
-h	Process HOSTNAME
-i	Process the INTERFACE variable set: [IP_ADDRESS, SUBNET_MASK, BROADCAST_MASK, LANCONFIG_ARGS]
-n	Process the NIS variable set: [NISDOMAIN, YPSET_ADDR]
-p	Print results to the screen (stdout), this is the default action if neither **-c** or **-a** are specified
-r	Process the ROUTE variable set: [ROUTE_DESTINATION, ROUTE_GATEWAY, ROUTE_COUNT]
-s *set index*	Specify the variable set index
-t	Process NTPDATE_SERVER

Configuration Files and Variable Names

The files and variables which can be processed are the following:

/etc/rc.config.d/netconf

HOSTNAME
INTERFACE_NAME[index]
IP_ADDRESS[index]
SUBNET_MASK[index]
BROADCAST_MASK[index]
LANCONFIG_ARGS[index]
ROUTE_DESTINATION[index]
ROUTE_GATEWAY[index]
ROUTE_COUNT[index]

/etc/rc.config.d/namesvrs

NISDOMAIN
YPSET_ADDR

/etc/rc.config.d/netdaemons

NTPDATE_SERVER

/etc/resolv.conf

domain
nameserver

EXAMPLES

To list the entire contents of the DHCP client database type:

```
dhcpdb2conf
```

To list only the INTERFACE variable set for lan0 type:

```
dhcpdb2conf -i lan0
```

To list the INTERFACE and ROUTE variable sets for lan0 and lan1 type:

```
dhcpdb2conf -ir lan0 lan1
```

To apply the INTERFACE and ROUTE variable sets for lan0 to the existing configuration files type:

```
dhcpdb2conf -ira lan0
```

To apply all variable sets to the existing configuration files using lan0 and set index = 1 type:

```
dhcpdb2conf -a -s 1 lan0
```

WARNINGS

Using the **-c** option will override any existing values which are currently set in the system's configuration files.

FILES

/usr/lbin/dhcpdb2conf
/etc/dhcpclient.data

SEE ALSO

auto_parms(1M).

NAME

dhcptools - command line tool for DHCP elements of bootpd

SYNOPSIS

dhcptools -d

dhcptools -h fip=*first_IP_address* **no=***number_of_entries_to_generate* **sm=***subnet_mask* **hn=***hostname_template* [**dn=***domain_name*]

dhcptools -p ht=*hardware_type* **ha=***hardware_address* **sn=***subnet_identifier* [**lt=***lease_time*] [**rip=** *requested_IP_address*]

dhcptools -P ci=*client_identifier* **sn=***subnet_identifier* [**lt=***lease_time*] [**rip=***requested_IP_address*]

dhcptools -r ip=*IP_address* **ht=***hardware_type* **ha=***hardware_address*

dhcptools -R ip=*IP_address* **ci=***client_identifier*

dhcptools -t [**ct=***count*]

dhcptools -v [**bt=***bootptabfile*] [**dt=***dhcptabfile*]

DESCRIPTION

dhcptools is a command line tool that provides access to DHCP-related options for the bootpd server. The options provide control for dumping internal data structures, generating a hosts file, previewing client address assignment, reclaiming unused addresses, tracing packets, and validating configuration files.

Options

dhcptools supports the following options:

-d Dump internal bootpd data to output files. The dump output files are **/tmp/dhcp.dump.bootptab**, **/tmp/dhcp.dump.dhcptab**, and **/tmp/dhcp.dump.other**. The first file reports fixed address clients known to the currently active **bootpd** server. The second file reports **bootpd** global and group configuration. The third file reports miscellaneous **bootpd** internal data.

-h Generate a hosts file in **/etc/hosts** format; see *hosts*(4). The output file is **/tmp/dhcphosts**. The file can be incorporated into a name database in advance of **bootpd** server activation so that the server can automatically allocate a host name along with an IP address to a DHCP client. For IP address allocation to DHCP clients, the **bootpd** server uses *gethostbyaddr*(3N) to find the host name associated with a particular IP address. Each host entry in **dhcphosts** contains an IP address followed by a hostname. The IP address of the first entry is first_IP_address. The hostname of the first entry is derived from the hostname_template. Each subsequent host entry contains a unique IP address and hostname derived from the first_IP_address, subnet_mask, and hostname_template. The wildcards permitted in the hostname_template are ***#?**. A ***** means to use a character selected sequentially from the range [*a-z*,*0-9*]. A **#** means to use a digit selected sequentially from the range [*0-9*]. A **?** means to use a letter selected sequentially from the range [*a-z*]. A maximum of 3 wildcards can be specified. If a domain_name is specified, it will be appended to the hostname. The maximum number_of_entries_to_generate is 1000.

-p Preview a client's address assignment based on current conditions for the **bootpd** server. The output is written to stdout. The subnet-identifier tells **bootpd** the subnet for which the client is requesting an IP address. Optionally, the user may request a specific IP address and lease duration using the parameters lease-time and requested-IP-address. Use Internet address dot notation (see *inet*(3N) for the IP address and an integer number of seconds for the lease-time.

-P Preview a client's address assignment based on current conditions for the **bootpd** server. This option is the same as **-p** except that the client is identified by a unique client-identifer. See *bootpd*(1M).

-r Reclaim a client's IP address for re-use by the **bootpd** server. This option is intended for limited use by the **bootpd** administrator to return an allocated but unused IP address to a DHCP allocation pool. The option may be useful to clear the bootpd database of old entries (e.g. for clients retired from service while holding an unexpired IP address lease). Do not reclaim an address that belongs to an active client. See *bootpd*(1M). The IP_address, hardware_address, and hardware_type can be obtained from the bootpd database file.

-R Reclaim a client's IP address for re-use by the **bootpd** server. This option is the same as **-r** except that the client is identified by its unique client_identifier. See *bootpd*(1M). The IP_address and matching client_identifier can be obtained from the bootpd database file.

-t Establish packet tracing for **bootpd**. This will trace the inbound and outbound BOOTP/DHCP packets for the local **bootpd** server. The output file is **/tmp/dhcptrace**. The packet trace count can be a value from 0 to 100. To query the current count, use **dhcptools -t**. To turn off packet tracing use **dhcptools -t ct=0**.

-v Validate **bootpd** configuration files. The default configuration files that will be validated are **/etc/bootptab** and **/etc/dhcptab**. When a bootptabfile or dhpctabfile is specified, the full pathname is required. The output file for validate is **/tmp/dhcpvalidate**.

Only one of the **-d**, **-h**, **-t**, **-p**, **-P**, **-r**, **-R**, or **-v** options is allowed per **dhcptools** command.

RETURN VALUE

dhcptools returns zero upon successful completion or non-zero if the command failed, in which case an explanation is written to standard error.

EXAMPLES

Dump the active **bootpd** server's internal data to the dump output files:

```
dhcptools -d
```

Generate a **/tmp/dhcphosts** file with 10 entries:

```
dhcptools -h fip=192.11.22.0 no=10 sm=255.255.255.0 hn=workstation#?
```

Query the active **bootpd** daemon for the the current packet trace count:

```
dhcptools -t
```

Set the count to 10 packets:

```
dhcptools -t ct=10
```

Preview two clients' address assignments by hardware address:

```
dhcptools -p ht=1 ha=080009000001 sn=192.11.22.0 lt=infinite
dhcptools -p ht=1 ha=080009000002 sn=192.11.22.0 lt=600 rip=192.11.22.105
```

To preview a client's address assignment by client identifier, a unique client identifier value is needed. This information can be obtained for actual DHCP clients (provided they support a client identifier) from the manufacturer's documentation. See *bootpd*(1M) for more information about the client identifier. Assuming that **serial_number_12345678** is a valid client identifier, the preview command is:

```
dhcptools -P ci="serial_number_12345678" sn=192.11.22.0
```

To reclaim an IP address by hardware address:

```
dhcptools -r ip=192.11.22.149 ht=1 ha=080009000006
```

The parameter values were obtained from this sample entry in the dhcpdb file:

```
C 192.11.22.0: 192.11.22.149 00 1 080009000006 FFFFFFFF 00
```

To reclaim an IP address by client identifier (see earlier example of preview by client identifier):

```
dhcptools -R ip=192.11.22.110 ci="serial_number_12345678"
```

To validate a bootptab and dhcptab file:

```
dhcptools -v bt=/home/mydir/bootptab dt=/home/mydir/dhcptab
```

WARNINGS

The **dhcptools** operations of dump, packet trace, preview, and reclaim depend on communication with the local **bootpd** server. If the server is not running, you may encounter an error.

AUTHOR

dhcptools was developed by HP.

FILES

/tmp/dhcphosts	hostgen output file in /etc/hosts format
/tmp/dhcptrace	packet trace output file
/tmp/dhcpvalidate	validate output file
/tmp/libdhcp.sl	library file
/tmp/dhcp.dump.bootptab	dump output file
/tmp/dhcp.dump.dhcptab	dump output file
/tmp/dhcp.dump.other	dump output file
/etc/bootptab	default bootptab file for validate
/etc/dhcptab	default dhcptab file for validate
/tmp/dhcpfifo.root	FIFO file for dhpctools to bootpd(1M) communication
/tmp/dhcpfifo.any	FIFO file for dhcptools to bootpd(1M) communication
/tmp/dhcpfifo	FIFO file for bootpd(1M) to dhcptools communication

SEE ALSO

bootpd(1M), bootpquery(1M); DARPA Internet Request For Comments RFC1541, RFC1542, RFC1533, RFC1534, Assigned Numbers

NAME

diskinfo - describe characteristics of a disk device

SYNOPSIS

/usr/sbin/diskinfo [**-b**|**-v**] *character_devicefile*

DESCRIPTION

The **diskinfo** command determines whether the character special file named by *character_devicefile* is associated with a SCSI, CS/80, or Subset/80 disk drive. If so, **diskinfo** summarizes the disk's characteristics.

The **diskinfo** command displays information about the following characteristics of disk drives:

Vendor name	Manufacturer of the drive (SCSI only)
Product ID	Product identification number or ASCII name
Type	CS/80 or SCSI classification for the device
Disk	Size of disk specified in bytes
Sector	Specified as bytes per sector

Both the size of disk and bytes per sector represent formatted media.

Options

The **diskinfo** command recognizes the following options:

-b Return the size of the disk in 1024-byte sectors.

-v Display a verbose summary of all of the information available from the device. (Since the information returned by CS/80 drives and SCSI drives differs, the associated descriptions also differ.)

- CS/80 devices return the following:
  Device name
  Number of bytes/sector
  Geometry information
  Interleave
  Type of device
  Timing information
- SCSI disk devices return the following:
  Vendor and product ID
  Device type
  Size (in bytes and in logical blocks)
  Bytes per sector
  Revision level
  SCSI conformance level data

DEPENDENCIES

General

The **diskinfo** command supports only CS/80, subset/80, and HP SCSI disk devices.

SCSI Devices

The SCSI specification provides for a wide variety of device-dependent formats. For non-HP devices, **diskinfo** may be unable to interpret all of the data returned by the device. Refer to the drive operating manual accompanying the unit for more information.

AUTHOR

diskinfo was developed by HP.

SEE ALSO

lsdev(1M), disktab(4), disk(7).

NAME

disksecn - calculate default disk section sizes

SYNOPSIS

disksecn [**-p** | **-d**] [**-b** *block_size*] [**-n** *disk_name*]

DESCRIPTION

disksecn is used to calculate the disk section sizes based on the Berkeley disk partitioning method.

disksecn recognizes the following options:

-p	Produce tables suitable for inclusion in the device driver.
-d	Produce tables suitable for generating the disk description file **/etc/disktab**.
-b *block_size*	When generating the above tables, use a sector size of *block_size* bytes, where *block_size* can be **256**, **512**, **1024**, or **2048**. Defaults to DEV_BSIZE (defined in <**sys/param.h**>) if not specified.
-n *disk_name*	Specifies the disk name to be used in calculating sector sizes; for example, **hp7912** or **hp7945**. If an unknown disk name is specified, *disksecn* prompts the user for the necessary disk information.

If neither **-p** nor **-d** table selection switches are specified a default table of the section sizes and range of cylinders used is output.

Disk section sizes are based on the total amount of space on the disk as given in the table below (all values are supplied in units of 256-byte sectors). If the disk is smaller than approximately 44 Mbytes, *disksecn* aborts and returns the message **disk too small, calculate by hand**.

Section	44-56MB	57-106MB	107-332MB	333+MB
0	97120	97120	97120	97120
1	39064	39064	143808	194240
3	39064	39064	78128	117192
4	unused	48560	110096	429704
6	7992	7992	7992	7992
10	unused	unused	unused	516096

NOTE:
It is important to note the difference between the block size passed into *disksecn* via the **-b** switch argument and the sector size the user is asked to input when an unknown disk name is passed to *disksecn* via the **-n** switch argument.

The block size is the sector size that *disksecn* assumes the disk to have when it prints the requested tables. All information printed in the tables is adjusted to reflect this assumed sector size (block size) passed in by the user. The sector size requested by *disksecn* when an unknown disk name is passed does not necessarily have to be the same as the assumed sector size (block size) passed in by the **-b** switch argument.

For example, a user wants to see the device driver tables for the disk named **hp7945** with an assumed sector size (block size) of 256 bytes. The user has the following information about the **hp7945** disk:

```
Disk type = winchester
Sector size = 512
Number of sectors per track (512 byte sectors) = 16
Number of tracks = 7
Number of cylinders = 968
Revolutions per minute = 3600
```

The user invokes *disksecn* by typing the following command:

```
disksecn -p -b 256 -n hp7945
```

Assuming that **hp7945** is an unknown disk name, *disksecn* prompts the user for the necessary disk information. The user should input the information as shown above, reflecting a sector size of 512 bytes. All the information will be adjusted within *disksecn* to reflect the assumed sector size (block size) of 256 bytes, passed as the argument of the **-b** switch, before the requested device driver table is output.

This adjustment also takes place when the disk name is known and an assumed sector size (block size) is passed in as the argument of the **-b** switch which is not DEV_BSIZE bytes, the assumed sector size (block size) used to create the **etc/disktab** file.

RETURN VALUE

disksecn returns the following values:

0	Successful completion.
1	Usage error.
2	User did not input parameters for an unknown disk.
3	Disk too small or an invalid block size.

disksecn aborts and prints an error message under the following conditions:

- *disksecn* was invoked without specifying a disk name.
- Requested both **-p** and **-d** switch.
- Illegal block size requested.
- Unknown disk name was specified and user did not supply disk information.
- Disk's maximum storage space is less than approximately 44 MB.

WARNINGS

Alternate names are not included in the output when the **-d** switch is used.

Blanks are required in the command line between each of the switches when invoking *disksecn*.

A blank is required between the **-n** switch and the disk name argument to that switch. For example:

```
disksecn -p -b 1024 -n hp9712
```

disksecn does not save the block size used to generate the **/etc/disktab** disk description file. The system assumes that the block size used was DEV_BSIZE when it reads the information stored in the **etc/disktab** file.

AUTHOR

disksecn was developed by the University of California, Berkeley.

FILES

/etc/disktab

SEE ALSO

disktab(4).

NAME

diskusg - generate disk accounting data by user ID

SYNOPSIS

/usr/sbin/acct/diskusg [*options*] [*files*]

DESCRIPTION

diskusg generates intermediate disk accounting information from data in *files*, or the standard input if omitted. **diskusg** outputs lines on the standard output, one per user, in the following format:

uid login #blocks

where:

uid	User's numerical user ID,
login	User's login name, and
#blocks	Total number of disk blocks allocated to this user.

diskusg normally reads only the inodes of file systems for disk accounting. In this case, *files* are the special filenames of these devices.

Options

diskusg recognizes the following options:

-s	Input data is already in **diskusg** output format. **diskusg** combines all lines for a single user into a single line.
-v	verbose. Print a list on standard error of all files that are charged to no one.
-i *fnmlist*	Ignore the data on those file systems whose file system name is in *fnmlist*. *fnmlist* is a list of file system names, separated by commas or enclosed within quotes. **diskusg** compares each name in this list with the file system name stored in the volume ID if it exists.
-p *file*	Use *file* as the name of the password file to generate login names. **/etc/passwd** is used by default.
-u *file*	Write records to *file* of files that are charged to no one. Records consist of the special file name, the inode number, and the user ID.

The output of **diskusg** is normally the input to **acctdisk** (see *acct*(1M)) which generates total accounting records that can be merged with other accounting records. **diskusg** is normally run in **dodisk** (see *acctsh*(1M)).

EXAMPLES

The following generates daily disk accounting information:

```
for i in /dev/rp00 /dev/rp01 /dev/rp10 /dev/rp11; do
    diskusg $i > dtmp.‘basename $i‘ &
done
wait
diskusg -s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

FILES

/etc/passwd	used for user-ID-to-login-name conversions

SEE ALSO

acct(1M), acctsh(1M), volcopy(1M), acct(4), vxdiskusg(1M).

STANDARDS CONFORMANCE

diskusg: SVID2, SVID3

NAME

dlf - download firmware to an HP SCSI disk array

SYNOPSIS

dlf -f *firmware_file device_file*

DESCRIPTION

dlf downloads a new set of controller firmware to the HP SCSI disk array associated with device file *device_file*. The *firmware_file* must be a binary file with a special format.

RETURN VALUE

dlf returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

ERROR MESSAGES

Errors can originate from problems with:

- **dlf**
- SCSI (device level) communications
- system calls

Error messages generated by dlf:

usage: dlf -f <firmware file> <special>
An error in command syntax has occurred. Enter command again with all required arguments, in the order shown.

dlf: Binary file has bad format
The binary file could not be read in properly by the utility.

dlf: device busy
To ensure that **dlf** does not modify a disk array that is being used by another process, **dlf** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

dlf: LUN # too big
The LUN number, which is derived from the device file name, is out of range.

dlf: Not a raw file
Utilities must be able to open the device file for raw access.

dlf: Not an HP SCSI disk array
The device being addressed is not an

dlf: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount. HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

dlf uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **dlf** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To download the special-format binary file **new_firmware** to the HP SCSI disk array **/dev/rdsk/c2t0d0** on a series 800:

```
dlf -f new_firmware /dev/rdsk/c2t0d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

dlf was developed by HP.

NAME

dmesg - collect system diagnostic messages to form error log

SYNOPSIS

/usr/sbin/dmesg [**-**] [**core**] [**system**]

DESCRIPTION

dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed by the system when unusual events occur (such as when system tables overflow or the system crashes). If the **-** argument is specified, **dmesg** computes (incrementally) the new messages since the last time it was run and places these on the standard output. This is typically used with **cron** (see *cron*(1)) to produce the error log **/var/adm/messages** by running the command:

```
/usr/sbin/dmesg - >> /var/adm/messages
```

every 10 minutes.

The arguments **core** and **system** allow substitution for the defaults **/dev/kmem** and **/stand/vmunix** respectively, where **core** should be a file containing the image of the kernel virtual memory saved by the *savecore*(1M) command and **system** should be the corresponding kernel. If the system is booted with a kernel other than /stand/vmunix say /stand/vmunix_new, **dmesg** must be passed this name, the command must be,

```
/usr/sbin/dmesg [-] /dev/kmem /stand/vmunix_new
```

WARNINGS

The system error message buffer is of small, finite size. **dmesg** is run only every few minutes, so there is no guarantee that all error messages will be logged.

AUTHOR

dmesg was developed by the University of California, Berkeley.

FILES

/var/adm/messages	error log (conventional location)
/var/adm/msgbuf	memory scratch file for **-** option
/dev/kmem	special file containing the image of kernel virtual memory
/stand/vmunix	the kernel, system name list

SEE ALSO

savecore(1M).

NAME

dpp - dedicated ports parser used by DDFA software

SYNOPSIS

dpp *dp_file* [**-c**] [**-k**] [**-l** *log_file*] [**-p** *ocd_program*]

DESCRIPTION

The Dedicated Ports Parser command (**dpp**) is part of the Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software. It parses the Dedicated Ports file (**dp**) and spawns an Outbound Connection Daemon (**ocd**) for each valid entry in the **dp** file.

dpp can be run from the shell or it can be included in a system initialization script to automatically run the DDFA software each time the system is booted.

See *ddfa*(7) for more information on how to configure the DDFA software and for an explanation of how it works.

Options and Arguments

dpp recognizes the following options and arguments:

dp_file It must be the first argument. The **dp** file (*dp_file*) defines the link between a terminal server port and the device file used by applications to access the port. Its contents must meet the specifications given in *dp*(4). If it is modified, **dpp** must be run again to activate the changes.

-c Specify that the **dp** file should be parsed and that all incorrect entries should be logged without invoking any **ocd** processes. This option is useful for debugging the **dp** file before running it properly. The **-p** option is ignored if the **-c** option is used.

-k Specify that the device file corresponding to each valid entry in the **dp** file should be removed before launching **ocd** for each valid entry. Removing the device file eventually causes an **ocd** process (if any is running) to shutdown. If this option is omitted, no device files will be removed and, therefore, only newly added valid entries in the **dp** file will have **ocd** launched.

ocd normally creates and removes devices files. However, if the process is killed incorrectly, such as with **kill -9**, the device file may remain. If the system is rebooted, the **-k** option can be specified to restart all **dp** file entries correctly.

If a corresponding **ocd** no longer exists, the device file is removed by any following invocation of **ocd** that requires the same device file.

In order to shutdown every **ocd** running without restarting them, the following command can be executed:

```
kill -15 ‘ps -e | grep ocd | awk ’{print $1}’‘
```

-l*log_file* Specify where to log error messages. If this option is omitted, all error messages are logged to standard output.

If the specified file does not already exist, it is created. The file must be nonexecutable and readable by **dpp**.

-p*ocd_program* Specify the path for an outbound connection daemon. The default path for is **/usr/sbin/ocd**. The daemon must be executable.

DIAGNOSTICS

Error messages are logged for bad arguments, bad file entries, and **ocd** creation errors. By default, they are logged to standard output. If the **-l** option is used, they are appended to the specified log file.

(0) ERROR: dp file is mandatory
(1) ERROR: dp file must be the first argument
(2) ERROR: Cannot read dp file (*filename***)**

The **dp** file either does not exist or cannot be accessed with the current access privileges.

(3) ERROR: No log file defined (-l option)
(4) ERROR: Cannot create log file (-l *filename***)**

The log file cannot be created, either because of an invalid path or because of insufficient access privileges.

(5) ERROR: Cannot access log file (-l *filename***)**

The log file cannot be accessed, either because of an invalid path or because of insufficient access privileges. The log file must be readable by everyone.

(6) ERROR: No ocd file defined in program option
(7) ERROR: Cannot execute ocd program (-p *pathname***)**

The **ocd** program specified in the **-p** option either does not exist or is not an executable file with the current access privileges.

(8) ERROR: Cannot purge device file (/dev/*filename***)**

The **-k** option has been specified and the device file exists, but it cannot be purged because of insufficient access privileges.

(9) ERROR: Cannot execute default program (/usr/sbin/ocd)

The default **ocd** cannot be executed, either because of insufficient access privileges or because it has not been correctly installed.

(10) ERROR: Entry ignored (Bad IP address)

The **dp** file entry specified does not have a valid IP address.

(11) ERROR: Entry ignored (no port/board info)
(12) ERROR: Entry ignored (Bad port number)

The port specified is either not a decimal value or a string composed of **x** or **X** characters.

(13) ERROR: Entry ignored (Bad board number)

The board specified is either not a decimal value or a string composed of **x** or **X** characters.

(14) ERROR: No more processes available on system

The **ocd** program specified cannot be started because there are no processes available on the system.

(15) ERROR: Entry ignored (no device_name)
(16) ERROR: Entry ignored (Bad device_name)

The device file specified cannot be created, either because of an invalid path or because of insufficient access privileges.

(17) ERROR: Entry ignored (Bad config name)

The specified configuration file cannot be read, either because of an invalid path or because of insufficient access privileges.

(18) ERROR: Entry ignored (Invalid log level)

The specified logging level is not in the range 0 to 3.

(19) ERROR: Entry ignored (Bad node name)

The specified node name does not exist or does not have an entry in a name database.

WARNINGS

To ensure that commands (such as *ps*) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, to ensure that commands (such as **w**, **passwd**, **finger**, and **wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
stty exta <&1 2>/dev/null
```

The printer interface scripts reside in the directory **/etc/lp/interface**. The line must be added just prior to the final 'exit' command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

Finally, **ocd** should be killed using **kill -15**. Do not use **kill -9** for this purpose as it does not remove the device file. **ocd** verifies the validity of an existing pseudonym before trying to use it. **dpp** and **ocd** use data stored in the file **/var/adm/utmp.dfa** to verify whether a process still owns a pseudonym before taking it over. If **ocd** finds an unowned pseudonym, it uses it.

FILES

/usr/examples/ddfa/dp
/usr/examples/ddfa/pcf
/usr/sbin/dpp
/usr/sbin/ocd
/usr/sbin/ocdebug
/var/adm/dpp_login.bin
/var/adm/utmp.dfa

SEE ALSO

ocd(1M), ocdebug(1M), dp(4), pcf(4), ddfa(7).

NAME

dsp - display status of an HP SCSI disk array

SYNOPSIS

dsp -p [**-h**|**-d**] *device_file*

dsp -l [**-h**|**-d**] *device_file*

DESCRIPTION

dsp displays the status of the LUN (in an HP SCSI disk array) that is associated with the device file *device_file*. **dsp** displays the status of physical drives in an array (when the **-p** option is specified), or the status of LUNs in an array (when the **-l** option is specified). This information can be displayed in interpreted form, or in raw hexadecimal or raw decimal format.

Options

-p Display physical drive status. The **-p** option displays the status of a LUN's physical drives, regardless of their LUN ownership. This information is retrieved the array physical page (Mode Page 2A), and inquiry data.

-l Display LUN status. The **-l** option displays information about the state of the LUN including it's RAID level, block and segment sizes, reconstruction information, and so on. This information is retrieved from the array logical page (Mode Page 2B), and inquiry data.

By default, data is displayed in interpreted form; if raw data is desired, one of the following options can be used:

-h Raw hex format. Displays the data in raw hex format in rows, each of which contains the ASCII representation of 16 hexadecimal data bytes, separated by spaces.

-d Raw decimal format. Displays the data in raw decimal format in rows, each of which contains the ASCII representation of 16 decimal data bytes, separated by spaces.

RETURN VALUE

dsp returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **dsp**
- SCSI (device level) communications
- system calls

Error messages generated by dsp:

usage: dsp <-p | -l> [-h | -d] <special>
An error in command syntax has occurred. Enter the command again with all required arguments.

dsp: Arg out of range
One of the arguments is larger than its allowed maximum value (or smaller than its allowed minimum value), or is incorrect in form. Check the size and form of each argument and make appropriate corrections.

dsp: LUN # too big
The LUN number, which is derived from the device special file name, is out of range.

dsp: Not a raw file
Utilities must be able to open the device file for raw access.

dsp: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

dsp: LUN does not exist
The requested LUN is not among those known to the controller.

dsp: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:
Sense data associated with the failed operation is printed.

Error messages generated by system calls:
dsp uses the following system calls:

stat(), **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **dsp** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To display the status of the drives on the HP SCSI disk array **/dev/rdsk/c2t4d0** on a Series 700:

```
dsp -p /dev/rdsk/c2t4d0
```

To display the status of the LUN associated with the HP SCSI disk array **/dev/rdsk/c2t0d0** on a Series 800 in raw hex format:

```
dsp -l -h /dev/rdsk/c2t0d0
```

To display the status of the drives on the HP SCSI disk array **/dev/rdsk/c2t5d0** in raw decimal format on a Series 700:

```
dsp -p -d /dev/rdsk/c2t5d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

dsp was developed by HP.

NAME

dump, rdump - incremental file system dump, local or across network

SYNOPSIS

/usr/sbin/dump [*option* [*argument* ...] *filesystem*]

/usr/sbin/rdump [*option* [*argument* ...] *filesystem*]

DESCRIPTION

The **dump** and **rdump** commands copy to magnetic tape all files in the *filesystem* that have been changed after a certain date. This information is derived from the files **/var/adm/dumpdates** and **/etc/fstab**. *option* specifies the date and other options about the dump. *option* consists of characters from the set **0123456789bdfnsuWw**. The **dump** and **rdump** commands work only on file systems of type **hfs**. If the given file system is not of type **hfs**, **dump** and **rdump** will abort after printing an error message.

Options

0-**9** This number is the "dump level". All files modified since the last date stored in file **/var/adm/dumpdates** for the same file system at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed. Thus, the option **0** causes the entire file system to be dumped.

b The blocking factor is taken from the next argument (default is 10 if not specified). Block size is defined as the logical record size times the blocking factor. **dump** writes logical records of 1024 bytes. When dumping to tapes with densities of 6250 BPI or greater without using the **b** option, the default blocking factor is 32.

d The density of the tape (expressed in BPIs) is taken from the next *argument*. This is used in calculating the amount of tape used per reel. The default value of 1600 assumes a reel tape.

f Place the dump on the next *argument* file instead of the tape. If the name of the file is **-**, **dump** writes to the standard output. When using **rdump**, this option should be specified, and the next argument supplied should be of the form *machine***:***device*.

n Whenever **dump** and **rdump** require operator attention, notify all users in group **operator** by means similar to that described by *wall*(1).

s The size of the dump tape is specified in feet. The number of feet is taken from the next *argument*. When the specified size is reached, **dump** and **rdump** wait for reels to be changed. The default tape size value of 2300 feet assumes a reel tape.

u If the dump completes successfully, write on file **/var/adm/dumpdates** the date when the dump started. This file records a separate date for each file system and each dump level. The format of **/var/adm/dumpdates** is user-readable and consists of one free-format record per line: file system name, increment level, and dump date in *ctime*(3C) format. The file **/var/adm/dumpdates** can be edited to change any of the fields if necessary.

W For each file system in **/var/adm/dumpdates**, print the most recent dump date and level, indicating which file systems should be dumped. If the **W** option is set, all other options are ignored and **dump** exits immediately.

w Operates like **W**, but prints only file systems that need to be dumped.

If no arguments are given, *option* is assumed to be **9u** and a default file system is dumped to the default tape.

Sizes are based on 1600-BPI blocked tape; the raw magnetic tape device must be used to approach these densities. Up to 32 read errors on the file system are ignored. Each reel requires a new process; thus parent processes for reels already written remain until the entire tape is written.

The **rdump** command creates a server, **/usr/sbin/rmt** or **/etc/rmt**, on the remote machine to access the tape device.

dump and **rdump** require operator intervention for any of the following conditions:

- end of tape,
- end of dump,
- tape-write error,
- tape-open error, or
- disk-read error (if errors exceed threshold of 32).

In addition to alerting all operators implied by the **n** option, **dump** and **rdump** interact with the control terminal operator by posing questions requiring **yes** or **no** answers when it can no longer proceed or if something is grossly wrong.

Since making a full dump involves considerable time and effort, **dump** and **rdump** each establish a checkpoint at the start of each tape volume. If, for any reason, writing that volume fails, **dump** and **rdump** will, with operator permission, restart from the checkpoint after the old tape has been rewound and removed and a new tape has been mounted.

dump and **rdump** periodically report information to the operator, including typically low estimates of the number of blocks to write, the number of tapes it will require, the time needed for completion, and the time remaining until tape change. The output is verbose to inform other users that the terminal controlling **dump** and **rdump** is busy and will be for some time.

Access Control Lists (ACLs)

The optional entries of a file's access control list (ACL) are not backed up with **dump** and **rdump**. Instead, the file's permission bits are backed up and any information contained in its optional ACL entries is lost (see *acl*(5)).

EXAMPLES

In the following example, assume that the file system **/mnt** is to be attached to the file tree at the root directory, (**/**). This example causes the entire file system (**/mnt**) to be dumped on **/dev/rmt/c0t0d0BEST** and specifies that the density of the tape is 6250 BPI.

```
/usr/sbin/dump 0df 6250 /dev/rmt/c0t0d0BEST /mnt
```

WARNINGS

dump will not backup a file system containing large files.

Tapes created from file systems containing files with UID/GIDs greater than 60,000 will have a new magic number in the header to prevent older versions of *restore*(1M) from incorrectly restoring ownerships for these files.

AUTHOR

dump and **rdump** were developed by the University of California, Berkeley.

FILES

/dev/rdsk/c0d0s0	Default file system to dump from.
/dev/rmt/0m	Default tape unit to dump to.
/var/adm/dumpdates	New format-dump-date record.
/etc/fstab	Dump table: file systems and frequency.
/etc/group	Used to find group **operator**.

SEE ALSO

restore(1M), rmt(1M), fstab(4), acl(5).

NAME
dumpfs - dump file system information

SYNOPSIS
/usr/sbin/dumpfs *rootdir* | *special*

DESCRIPTION
The **dumpfs** command prints the super block and cylinder group information for an HFS file system to the standard output. The file system may be specified by its root directory or the name of the device special file on which it resides. The information is very long and detailed. This command can be used to find file system information such as the file system block size or the minimum free space percentage.

DEPENDENCIES
The **dumpfs** command can only be used on HFS file systems.

AUTHOR
dumpfs was developed by the University of California, Berkeley.

SEE ALSO
fsck(1M), mkfs(1M), newfs(1M), tunefs(1M), disktab(4), fs(4).

NAME

edquota - edit user disk quotas

SYNOPSIS

/usr/sbin/edquota [**-p** *proto-user*] *username* ...

/usr/sbin/edquota -t

DESCRIPTION

The **edquota** command is the quota editor. One or more user names can be specified on the command line. For each *username*, a temporary file is created with a textual representation of the current disk quotas for that user, and an editor is invoked on the file. The quotas can then be modified, new quotas added, etc. Upon leaving the editor, **edquota** reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is specified by the **EDITOR** environment variable. It defaults to **vi** (see *vi*(1)).

In order for quotas to be established on a file system, the root directory of the file system must contain a file named **quotas**. See *quota*(5) for details.

Quotas can be established only for users whose user ID is less than 67,000,000. Attempts to establish quotas for other users will result in an error message. This restriction will be removed in a future version of HP-UX.

Only users who have appropriate privileges can edit quotas.

Options

-p *proto_user* Duplicate the quotas of the user name *proto_user* for each *username*. This is the normal mechanism used to initialize quotas for groups of users.

-t Edit the time limits for each file system. Time limits are set for file systems, not users. When a user exceeds the *soft* limit for blocks or inodes on a file system, a countdown timer is started and the user has an amount of time equal to the time limit in which to reduce usage to below the soft limit (the required action is given by the **quota** command). If the time limit expires before corrective action is taken, the quota system enforces policy as if the *hard* limit had been exceeded. The default time limit of 0 is interpreted to mean the value in **<sys/quota.h>**, or one week (7 days). Time units of sec(onds), min(utes), hour(s), day(s), week(s), and month(s) are understood. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one.

Temporary File Formats

Here is an example of the temporary file created for editing user block and inode quotas:

```
fs /mnt blocks (soft = 100, hard = 120) inodes (soft = 0, hard = 0)
fs / blocks (soft = 1000, hard = 1200) inodes (soft = 200, hard = 200)
```

Here is the format for editing quota time limits:

```
fs /mnt blocks time limit = 10.00 days, files time limit = 20.00 days
fs / blocks time limit = 0 (default), files time limit = 0 (default)
```

When editing **(default)** values, it is not necessary to remove the **(default)** string. For example, to change the **blocks time limit** for **/**, changing the **0** to **4 days** is sufficient.

WARNINGS

When establishing quotas for a user who has had none before, (for either blocks or inodes), the quota statistics for that user do not include any currently occupied file system resources. Therefore, it is necessary to run **quotacheck** (see *quotacheck*(1M)) to collect statistics for that user's current usage of that file system. See *quota*(5) for a detailed discussion of this topic.

edquota will only edit quotas on local file systems.

AUTHOR

edquota was developed by the University of California, Berkeley, and by Sun Microsystems, Inc.

FILES

/etc/fstab Static information about the file systems.

`/etc/mnttab`	Mounted file system table
*directory***`/quotas`**	Quota statistics static storage for a file system, where *directory* is the root of the file system as specified to the **`mount`** command (see *mount*(1M)).

SEE ALSO

vi(1), quota(1), quotacheck(1M), quotacheck_hfs(1M), quota(5).

NAME

eisa_config - EISA configuration tool

SYNOPSIS

eisa_config

eisa_config [**-a**]

eisa_config [**-c** *cfgfile*]

eisa_config [**-n** *scifile*]

DESCRIPTION

eisa_config is a specialized program for configuring EISA and ISA (referred to collectively as E/ISA) I/O boards on HP-UX workstations equipped with EISA backplanes. It is used each time the E/ISA configuration is to be changed in any way; i.e., whenever an EISA or ISA board is added to the system, removed from the system, or moved to a different location in the system. **eisa_config** should be run before any physical board configuration or installation changes are made. (This is not necessary in some cases -- see automatic mode below.)

eisa_config interprets information stored in configuration files and uses it to configure system resources needed to properly interact with E/ISA boards. Even though they may be physically present in the computer, E/ISA boards cannot be used by the HP-UX operating system until configuration by **eisa_config** is complete.

The **eisa_config** command takes one of four forms:

eisa_config	Use interactive commands to examine or modify configuration. **eisa_config** prompts for a command, executes it, reports the results of command execution, then prompts for the next command.
eisa_config -a	Attempt to automatically add new EISA boards to the configuration. This option is used by **/sbin/bcheckrc** but should not be used elsewhere. ISA boards cannot be added with this option.
eisa_config -c *cfgfile*	Check configuration (CFG) file (discussed below). This option is used mostly by E/ISA board developers. It simply checks the specified CFG file to verify that it follows correct grammar and can be used by **eisa_config**. This option does not affect current configuration in any way.
eisa_config -n *scifile*	Non-target mode. This option uses the contents of *scifile* instead of non-volatile memory (NVM) to set up E/ISA configuration, and is most commonly used for creating identical configurations on multiple workstations.

Assigning Resources

Depending on their design, internal capabilities, and their role in system operation, E/ISA boards use various combinations of one or more system resources such as DMA channels, interrupt lines, memory, etc. Also, given boards do not always use a full set of system resources; for example, EISA provides 11 interrupt lines, but a given board might be able to use only lines 3, 5, and 6. Thus a means for the board to determine what resources are to be used must be provided.

ISA boards use physical switches or jumpers on the board to specify what resources are to be used. The person installing the board sets the switches or jumpers as specified by the board's manufacturer and based on system needs. There are thousands of different kinds of ISA boards, but unfortunately there are no standard conventions for switch and jumper usage. This results in much confusion and numerous configuration problems. For example, it is easy to inadvertently assign a given resource to two different boards, but often very difficult to diagnose the problem.

EISA boards usually have no switches or jumpers for resource assignment. Instead, each EISA board has a corresponding configuration (CFG) file that tells the system how the board can be used and what resources it needs. **eisa_config** is the HP-UX system program that interprets the various CFG files for all boards in the system, then builds a conflict-free configuration.

Configuration Files

All EISA boards have a corresponding CFG file. ISA boards, when used in HP-UX systems, must also have a corresponding CFG file. Although **eisa_config** cannot automatically configure an ISA board, it can use the contents of the CFG file to determine what switch or jumper settings on an ISA board can be used to prevent resource conflicts.

eisa_config expects to find a CFG file for each E/ISA board connected to the workstation. The administrator is responsible for making sure that these CFG files are present in directory **/sbin/lib/eisa**. CFG files corresponding to boards being used should always be kept in this directory. Do not remove them after **eisa_config** is run the first time, because they will be needed every time the configuration is changed, such as when a new board is added or one is removed. Do not change the file names of the CFG files. The file name has a specific format which is used by **eisa_config** to automatically match a board with its CFG file.

CFG files are normally supplied by the E/ISA board manufacturer. Two scenarios apply:

- If the E/ISA board is supplied by HP, the CFG file corresponding to the board is loaded into **/sbin/lib/eisa** as part of normal operating system installation. It should never be removed.
- If the E/ISA board is not supplied by HP, install both the CFG file and the software driver for the board from HP-UX-readable media supplied by the board manufacturer. Copy the CFG file to directory **/sbin/lib/eisa** where it must remain as long as the card is present in the system.

All CFG files must follow a grammar specified in the EISA bus specification. The most basic building block in the CFG grammar is the *board*. Each board has several attributes including board ID (to match with a board's ID register), manufacturer, ASCII text describing what the board does, what kinds of slots the board can go in, whether the board has a readable ID register, and various other capability attributes.

Each file can also contain lists of board-wide resources (such as I/O registers, switches, and jumpers) and how they should be initialized.

A board can be treated as a set of one or more *functions* where a given board contains a single function or multiple functions. An example of a two-function board is one having both a serial port and a parallel printer port. Each function has a separate block in that board's CFG file. Each function has a name, a type, and a set of configuration *choices*.

Each *choice* block has a name and a set of attributes. These attributes include what resources the choice requires and whether the function is enabled or disabled by that choice. Initialization is also usually specified within a choice. A given choice might require that certain registers be initialized to a specified value and that switches be set in a certain way.

Configuration Processing

E/ISA configuration is handled as follows:

- **eisa_config** builds a conflict-free configuration, then saves the configuration in EISA non-volatile memory (NVM).
- Appropriate drivers and device files must be installed before rebooting the system.
- Next time the operating system is rebooted, the HP-UX kernel initializes the specified E/ISA boards according to the contents of NVM.

If a board is currently present in the system, but has no corresponding configuration data in NVM, the EISA board cannot be used until the **eisa_config** program is run again and the new board is accounted for in NVM. A newly installed or existing E/ISA board is not usable until **eisa_config** has added it and the system has been rebooted with the necessary drivers and device special files installed. See EXAMPLES for an illustration of how to add a new board to the system.

It is possible to add EISA boards that do not have switches or jumpers to the configuration without running **eisa_config** interactively. The **/sbin/bcheckrc** script invokes **eisa_config** with automatic mode during each system initialization. If a board has been added since the last time **eisa_config** was executed, **eisa_config** attempts to add the new board to the configuration. If the new board is successfully added, the system may need to be rebooted (**/sbin/bcheckrc** does this automatically). If the new board could not be added to the configuration, a warning is written to the system console and **/etc/eisa/config.err**.

In addition to writing to NVM, **eisa_config** also automatically saves the current configuration to an SCI file called **/etc/eisa/system.sci**. SCI files can also be created by the interactive **save** command (see below). The E/ISA subsystem can also be initialized from an SCI file, rather than from NVM by using the **eisa_config -n** command form discussed earlier. SCI files are quite useful when a site has several identically-configured workstations. Run **eisa_config** on one system and save the configuration in an SCI file. Copy this file to other systems, then use it to initialize those systems. Remember that the configuration must be saved to NVM and the system rebooted before the E/ISA boards can be used.

Drivers and Device Files

Running **eisa_config** is not the only task necessary when adding an E/ISA board to a system. Corresponding I/O drivers must be added to the kernel and appropriate device files must be created. These steps are the same as is required for any I/O card, and can be performed either before or after running **eisa_config**. The important thing to remember is that the E/ISA board cannot be used until *all* necessary tasks are complete.

Interactive Commands

If the command form **eisa_config** is used, **eisa_config** runs in interactive mode. Interactive mode conducts configuration changes by using a series of keyboard commands. **eisa_config** prompts for a command, executes it, displays the results of executing the command, then prompts for the next command. Interactive commands are broadly grouped into five categories:

action	Alter the configuration in some way.
display	Show current configuration.
cfg	Manage CFG files.
comments	Display help and comments information found in CFG files.
help	Help for using **eisa_config** interactive commands

The *action* commands are:

add *cfgfile slotnum* Adds a board to the current configuration. *cfgfile* specifies which CFG file corresponds to the board and *slotnum* identifies the slot where the board resides.

remove *slotnum* Remove a board from the current configuration. *slotnum* identifies the slot where the board currently resides.

move *curslotnum newslotnum*
Move a board that is currently configured in one slot to a different slot. *curslotnum* and *newslotnum* specify the current and new slot numbers, respectively.

change *slotnum functionnum choicenum*
Change the choice used for a given function. All three arguments, *slotnum*, *functionnum*, and *choicenum* are required. The function number (*functionnum*) and choice number (*choicenum*) can be obtained by using the **show board** command on the slot in question. Function numbers are of the format **F***num* and choice numbers are of the format **CH***num*. Note that a board must already be part of the configuration before the change command can be used.

When **eisa_config** adds a board, it selects a choice for each function. Generally, the first choice for each function is selected (the default). However, in order to resolve conflicts, **eisa_config** may select a different choice for a given function. When specifying a choice for a particular function by use of the **change** command, **eisa_config** always uses that choice; it does not select a different one, even when a conflict needs to be resolved.

save [*filename*] Save the current configuration. If the current configuration is not conflict-free, a warning is produced and the save is not done. If you specify a file name, the save is done to that file; otherwise, the save is done to NVM (and the **/etc/eisa/system.sci** file). Note that the **quit** command also (optionally) saves the configuration to NVM (and file **/etc/eisa/system.sci**).

When the configuration is saved to NVM, a log file is created that provides a brief desription of the new configuration. The log file is named **/etc/eisa/config.log**, and contains information generated by a **show** command, followed by a **show board** command, followed by a **show switch** command.

init [*filename*] Initialize the configuration. The initial configuration is retrieved from a file if one has been specified. Otherwise, it is retrieved from NVM. Note that an implicit **init** is done when **eisa_config** is first started. This command should only be used when the current configuration **eisa_config** is dealing with is incorrect. For example, if you make some changes that you decide you do not want, you can use this command to start over.

quit Leave **eisa_config**. If the configuration is conflict-free and has been changed, you are asked if you want to save the configuration (to NVM). If any switches or jumpers have to be changed as a result of this new configuration, you are notified of these changes prior to saving the configuration. Be sure that all switches and jumpers match what **eisa_config** has specified before booting the system.

When the configuration is saved to NVM, a log file is created that provides a brief desription of the new configuration. The log file is named **/etc/eisa/config.log**, and contains information generated by a **show** command, followed by a **show board** command, followed by a **show switch** command.

The *show* (display) commands are:

show List all slots and their current status; i.e., whether occupied by a particular board, or empty.

show slots *cfgfile*
List all of the slots that could accept the board corresponding to the CFG file *cfgfile*.

show board [*cfgfile* | *slotnum*]
List the basic attributes for the selected board or boards. Includes a list of all the functions on the board and a list of all available choices for each function. If the board is currently part of the configuration, the currently selected choice is marked. The default choice is the first choice listed for each function. If a board is not specified (either by CFG file name or slot number), information is displayed for each of board installed and configured in the system.

show switch [**changed**] [*slotnum*]
List the switch and jumper settings (both default and required) for the boards in the configuration. If the keyword **changed** is used, only those switches and jumpers that were changed from the previous configuration are displayed. If a slot number is specified, only switches and jumpers on the board in that slot are displayed. Note that **show switch** supports all combinations of **changed** and *slotnum*.

There are two kinds of *cfg* commands:

cfgtypes List the types of boards that have CFG files in directory **/sbin/lib/eisa** and how many CFG files in **/sbin/lib/eisa** are of each type.

cfgfiles [*type*] List all CFG files that are currently available for use in the **/sbin/lib/eisa** directory. If a specific board *type* is specified, only CFG files of that type are displayed.

comment commands extract the help and comments text provided in the specified CFG file or files. Both help and comments are displayed if they are available. Each command form accepts as an argument either a CFG file or a slot number identifying which board you want help for.

comment board [*cfgfile* | *slotnum*]
Display board-level help and comments.

comment function [*cfgfile* | *slotnum*]
Display function-level help and comments.

comment choice [*cfgfile* | *slotnum*]
Display choice-level help.

comment switch [*cfgfile* | *slotnum*]
Display help and comments for switches and/or jumpers as appropriate.

Note that all arguments (except the type of comments requested) are optional. If no optional argument is specified, all available comments for the specified file or board are extracted. For example:

comment board 1
Display help and comments available for the board currently configured in slot 1.

comment board	Display help and comments available for *all* currently configured boards.

The *help* commands explain how to use the **eisa_config** interactive commands. If no other arguments are given, help is displayed for all of the interactive commands. Alternatively, any valid command can be used as a argument to the help command. Help is then given for the specified command only.

help	Display a brief explanation of all valid **eisa_config** interactive commands.
help [*cmdname*]	Display an explanation of the command specified.

EXAMPLES

Add a new E/ISA board to the system:

1. Load the CFG file (from media provided by the manufacturer) into directory **/sbin/lib/eisa** if the file is not already present.
2. Run **eisa_config**. **eisa_config** reads the contents of NVM to obtain current system configuration.
3. Use the interactive **add** command to add the new board. **eisa_config** reads the corresponding CFG file to obtain needed configuration information.
4. Exit **eisa_config**, noting any required switch or jumper settings. **eisa_config** generates a new configuration and writes it to NVM. The required switch and jumper settings are also saved in the log file **/etc/eisa/config.log**.
5. Add the correct software drivers for the board (and board devices) to the kernel, and use *mknod*(1M) to create any needed device special files.
6. Shut down and disconnect power to the system.
7. Install the E/ISA board after changing any switch or jumper settings required by **eisa_config**.
8. Reboot the system. When the system is running again, the contents of NVM will match the E/ISA boards present in the system, and the newly added board can be used immediately.

This procedure can also be used to add multiple new boards at the same time. Simply use the **add** command once for each board and alter the other steps as appropriate.

If the board to be added is an EISA board that does not have switches or jumpers, the board can be added via automatic mode; that is, steps 2-4 above can be skipped.

AUTHOR

eisa_config was developed by HP and Compaq.

FILES

/sbin/lib/eisa/!XXX0000.CFG	CFG files
/etc/eisa/config.err	errors encountered in automatic mode
/etc/eisa/config.log	log file containing current E/ISA configuration
/etc/eisa/system.sci	mirror image of configuration saved to NVM

SEE ALSO

config(1M), mknod(1M).

e

e

NAME

envd - system physical environment daemon

SYNOPSIS

/usr/sbin/envd [**-f** *configfile*]

DESCRIPTION

The **envd** daemon provides a means for the system to respond to environmental conditions detected by hardware. Such responses are typically designed to maintain file system integrity and prevent data loss. The environmental condition currently recognized by **envd** is over-temperature.

envd logs messages and then executes actions when a supported environmental event is detected. Whether to do message logging and what actions to perform for a given environmental event are determined by *configfile* (default is **/etc/envd.conf**). If no **-f** option was specified and the default configfile **/etc/envd.conf** does not exist, **envd** fails. A recommended default *configfile* is available in **/usr/newconfig/etc/envd.conf**. The *configfile* (or **/etc/envd.conf**) is only examined when the daemon is started or when it receives a **SIGHUP** signal to restart and re-initialize the daemon itself.

envd uses the **syslog** message logging facility to log warning messages. If *configfile* specifies messages to be logged, the destination of the warning messages is determined by the configuration of the **LOG_DAEMON** facility of the **syslogd** daemon (see *syslogd*(1M) and *syslog*(3C) for details) and various **syslog** priorities defined below for the corresponding environmental events. Warning messages are written to the console if **envd** is unable to send to **syslogd**.

The *configfile* is composed of event lines, each of which followed by zero or more action lines. Comment lines can be interspersed at any point. No more than one event line can be specified for a given event.

Event — Event lines consist of an event keyword and a message indicator, separated by a colon (**:**). Valid event keywords are **OVERTEMP_CRIT** and **OVERTEMP_EMERG**. Valid message indicators are **y** and **n**. An example is **OVERTEMP_EMERG:y**, indicating that warning messages are to be sent for the OVERTEMP_EMERG event.

Event keywords must start in the first column, and only one event and one message indicator are allowed on a given line.

Action — Action lines can consist of a sequence of any valid **/usr/bin/sh** commands or pipelines. Lines from one event line to the next event line, or to the end of the file, are part of the action lines for the preceding event, and are passed intact to the shell to execute upon detecting the event. The action for an event can span across several lines, but the syntax of every line must be understood by **/usr/bin/sh**. There are no default actions for any events if no action lines are specified.

No parsing or syntax checking is performed on the action lines; system administrators are responsible for verifying the correctness of the action syntax.

Comments — Lines beginning with the **#** character in the first column are comment lines, and all characters up to the subsequent new-line character are ignored.

Blank lines are ignored as comment lines.

Here is an example **/etc/envd.conf** file:

```
# The example below configures envd to log the warning message and
# to rcp critical applications to a remote machine at OVERTEMP_CRIT.
# It configures envd to log emergency messages and to perform
# system shutdown at OVERTEMP_EMERG, in order to preserve
# the data integrity.

OVERTEMP_CRIT:y
    /usr/bin/rcp critical_appl_files \
        remote_machine:/backup

OVERTEMP_EMERG:y
    /usr/sbin/reboot -qh
```

Only users with appropriate privileges can invoke **envd**.

Over-temperature Handling

Over-temperature handling is supported only on systems equipped with over-temperature sensing hardware. Over-temperature limits may vary, depending on the hardware. Each system processor defines its own safest threshold for supported equipment combinations. The table below shows four levels of temperature states. For the temperature range specific to your system configuration, refer to any of the following documents for your system: *Site Planning and Preparation Guide*, *Installation and Configuration Guide*, or *Operator Handbook*.

State	State Description
NORMAL	Within normal operating temperature range
OVERTEMP_CRIT	Temperature has exceed the normal operating range of the system, but it is still within the operating limit of the hardware media.
OVERTEMP_EMERG	Temperature has exceeded the maximum specified operating limit of hardware media; power loss is imminent. A minimum of about 60 seconds is guaranteed between the OVERTEMP_MID state and the OVERTEMP_POWERLOSS (power loss) state.
OVERTEMP_POWERLOSS	Hardware will disconnect all power from all cards in the system chassis.

The **syslog** priorities mapped to two over-temperature events are: **LOG_EMERG** (for **OVERTEMP_EMERG**) and **LOG_CRIT** (for **OVERTEMP_CRIT**).

Any non-shutdown activities (e.g. file transfer) should be performed at **OVERTEMP_CRIT**. It is important to configure only critical activities for **OVERTEMP_CRIT** because the over-temperature might rise dramatically fast to **OVERTEMP_EMERG**. It is recommended to perform a quick shutdown using **/usr/sbin/reboot -qh** at **OVERTEMP_EMERG** to preserve file system data integrity. If the hardware enters the **OVERTEMP_POWERLOSS** state and the system has not been shut down, the sudden loss of power could result in data loss. Note that power-fail recovery functionality is not available in this case. When the hardware powers down, no warning messages are produced, and no action is taken by the system.

Whenever the temperature rises from one level to another (such as from **NORMAL** to **OVERTEMP_CRIT** or from **OVERTEMP_CRIT** to **OVERTEMP_EMERG**, the warning message, if specified, and the corresponding specified over-temperature action is executed once, and only once, per state change.

AUTHOR

envd was developed by HP.

FILES

/usr/sbin/envd	**envd** executable file
/etc/envd.conf	default **envd** configuration file
/etc/syslog.conf	default **syslog** configuration file
/var/tmp/envd.action[123]	**envd** work files

SEE ALSO

reboot(1M), shutdown(1M), syslogd(1M), syslog(3C).

HP-UX System Administration manuals.

NAME

exportfs - export and unexport directories to NFS clients

SYNOPSIS

/usr/sbin/exportfs [**-auv**]

/usr/sbin/exportfs [**-uv**] [*dir* ...]

/usr/sbin/exportfs -i [**-o** *options*] [**-v**] [*dir* ...]

DESCRIPTION

The **exportfs** command makes a local directory or file available to NFS clients for mounting over the network. Directories and files cannot be NFS-mounted unless they are first exported by **exportfs**.

exportfs is normally invoked at boot time by the **/sbin/init.d/nfs.server** script, and uses information contained in the **/etc/exports** file to export the file or file system named by each *dir*, which must be specified as a full path name.

If no options or arguments are specified in the command line, **exportfs** displays a list of the currently exported directories and files on standard output.

A superuser can run **exportfs** at any time to alter the list or characteristics of exported directories and files.

Options

exportfs recognizes the following options:

-a Export all directories listed in **/etc/exports**. If **-u** is also specified, unexport all of the currently exported directories.

-i Ignore the options in **/etc/exports**. Normally, **exportfs** consults **/etc/exports** for the options associated with the exported directory.

-u Unexport the indicated directories.

-v Verbose. Print each directory or file name as it is exported or unexported.

-o *options* Specify a comma-separated list of optional characteristics for the directory being exported. The list of *options* can include any of the following:

async All NFS Protocol Version 2 mounts will be asynchronous. This option is ignored for NFS PV3. Refer to *exports*(4) for warnings when using this option.

ro Export the directory read-only. If not specified, the directory is exported read-write.

rw=*hostname*[**:***hostname*]...
Export the directory read/write only to the listed clients. No other systems can access the directory. Up to 256 *hostnames* can be specified.

anon=*uid* If a request comes from an unknown user, use *uid* as the effective user ID.

Root users (user ID 0) are always treated as user **unknown** by the NFS server unless they are included in the **root** option below.

If the client is a UNIX system, only root users are considered **unknown**. All other users are recognized even if they are not in **/etc/passwd**.

The default value for *uid* is the user ID of user **nobody**. If user **nobody** does not exist, the value –2 is used. Setting the value of **anon** to –1 disables anonymous access.

root=*hostname*[**:***hostname*]...
Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access. Up to 256 *hostname*s can be specified.

access=*client*[**:***client*]...
Give mount access to each *client* listed. A *client* can either be a host name, or a netgroup (see *netgroup*(4)). **exportfs** checks for each *client* in the list first in file **/etc/hosts**, then in **/etc/netgroup**. The default value allows any machine to mount the given directory.

DIAGNOSTICS

If an NFS-mounted directory is unexported by **exportfs**, any access by the client to the directory causes an **NFS stale file handle** error. However, if **exportfs** is used to remove a client from the access list of an exported directory, an **NFS stale file handle** error does not result from any access by the client to the directory.

EXAMPLES

The following invocation of **exportfs** lists currently exported directories and files:

```
exportfs
```

Export entries in **/etc/exports**

```
exportfs -a
```

Unexport all exported files and directories:

```
exportfs -ua
```

Unexport all exported files and directories and print each directory or file name as it is unexported:

```
exportfs -uav
```

Export **/usr** to the world, ignoring options in **/etc/exports**:

```
exportfs -i /usr
```

Export **/usr/bin** and **/var/adm** read-only to the world:

```
exportfs -i -o ro /usr/bin /var/adm
```

Export **/usr/bin** read-write only to systems **polk** and **vanness**:

```
exportfs -i -o rw=polk:vanness /usr/bin
```

Export root access on **/var/adm** only to the system named **pine**, and mount access to both **pine** and **geary**:

```
exportfs -i -o root=pine,access=pine:geary /var/adm
```

WARNINGS

You cannot export a directory that resides within the same file system and is either a parent or subdirectory of a directory that is currently exported. For example, **/usr** and **/usr/local** cannot both be exported if they reside in the same disk partition.

If you unexport a directory, remove a client from the access list, then export again, the client still has access to the directory until the client unmounts the directory. Removing a client from the **root** or **rw** list takes effect immediately.

/etc/xtab is a system file that contains a list of currently exported directories and files. This file is maintained by **exportfs**. To ensure that this file is always synchronous with current system data structures, do not attempt to edit **/etc/xtab** by hand.

FILES

/etc/exports	Static export information
/etc/netgroup	List of network groups
/etc/xtab	Current state of exported directories

SEE ALSO

showmount(1M), exports(4), netgroup(4).

NAME

extendfs (generic) - extend a file system size

SYNOPSIS

/usr/sbin/extendfs [**-F FStype**] [**-q**] [**-v**] [**-s** *size*] *special*

DESCRIPTION

If the original file system image created on *special* does not make use of all of the available space, **extendfs** can be used to increase the capacity of a file system by updating the file system structure to include the extra space.

The command-line parameter *special* specifies the device special file of either a logical volume or a disk partition. The *special* must be un-mounted before **extendfs** can be run (see *mount*(1M)).

Options

extendfs recognizes the following options:

-F FStype
: Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/default/fs**.

-q Query the size of *special*. No file system extension will be done.

-v Verbose flag.

-s *size* Specifies the number of **DEV_BSIZE** blocks to be added to the file system. If *size* is not specified, the maximum possible size is used.

EXAMPLES

To increase the capacity of a file system created on a logical volume, enter:

umount /dev/vg00/lvol1

lvextend -L *larger_size* **/dev/vg00/lvol1**

extendfs -F hfs /dev/vg00/rlvol1

mount /dev/vg00/lvol1 *mount_directory*

SEE ALSO

fstyp(1M), lvextend(1M), mkfs(1M), mount(1M), umount(1M), fs(4), fs_wrapper(5).

NAME

extendfs (hfs) - extend an HFS file system size

SYNOPSIS

/usr/sbin/extendfs [**-F hfs**] [**-q**] [**-v**] [**-s** *size*] *special*

DESCRIPTION

If the original HFS file system image created on *special* does not make use of all of the available space, the **extendfs** command can be used to increase the capacity of an HFS file system by updating the file system structure to include the extra space.

The command-line parameter *special* specifies the character device special file of either a logical volume or a disk partition. The *special* must be unmounted before the **extendfs** command can be run (see *mount*(1M)).

Options

extendfs recognizes the following options:

-F hfs	Specify the HFS file system type.
-q	Query the size of *special*. No file system extension will be done.
-v	Verbose flag.
-s *size*	Specifies the number of **DEV_BSIZE** blocks to be added to the file system. If the number of blocks is not specified, the maximum possible size is used.

EXAMPLES

To increase the capacity of a file system created on a logical volume, enter:

```
umount /dev/vg00/lvol1
lvextend -L larger_size /dev/vg00/lvol1
extendfs -F hfs /dev/vg00/rlvol1
mount /dev/vg00/lvol1 mount_directory
```

WARNINGS

The root file system cannot be extended using the **extendfs** command because the root file system is always mounted, and the **extendfs** command only works on unmounted file systems.

extendfs will fail if used on a file system, on a logical volume, where the logical block size of the logical volume is greater than the file system's fragment size. The logical block size, of a logical volume changes, when additional disks with larger sector size are added.

RETURN VALUE

extendfs returns the following values:

0	No errors were detected and file system was successfully extended.
1	Command aborted.

SEE ALSO

extendfs(1M), lvextend(1M), mkfs(1M), mount(1M), umount(1M), fs(4).

NAME

extendfs (vxfs) - extend a VxFS file system size

SYNOPSIS

/usr/sbin/extendfs [**-F vxfs**] [**-q**] [**-v**] [**-s** *size*] *special*

DESCRIPTION

If the original VxFS file system image created on *special* does not make use of all of the available space, **extendfs** can be used to increase the capacity of a VxFS file system by updating the file system structure to include the extra space.

The command-line parameter *special* specifies the device special file of either a logical volume or a disk partition. If *special* refers to a mounted file system, *special* must be unmounted before **extendfs** can be run (see *mount*(1M)).

Options

extendfs recognizes the following options:

-F vxfs
Specify the VxFS file system type.

-q Query the size of *special*. No file system extension will be done.

-v Verbose flag.

-s *size* Specifies the number of **DEV_BSIZE** blocks to be added to the file system. If *size* is not specified, the maximum possible size is used.

EXAMPLES

To increase the capacity of a file system created on a logical volume, enter:

```
umount /dev/vg00/lvol1
lvextend -L larger_size /dev/vg00/lvol1
extendfs -F vxfs /dev/vg00/rlvol1
mount /dev/vg00/lvol1 mount_directory
```

SEE ALSO

extendfs(1M), lvextend(1M), mkfs(1M), mount(1M), umount(1M), fs(4).

NAME

fbackup - selectively back up files

SYNOPSIS

/usr/sbin/fbackup -f *device* [**-f** *device*] ... [**-0-9**] [**-nsuvyAEl**] [**-i** *path*] [**-e** *path*] [**-g** *graph*] [**-d** *path*] [**-I** *path*] [**-V** *path*] [**-c** *config*]

/usr/sbin/fbackup -f *device* [**-f** *device*] ... [**-R** *restart*] [**-nsuvyAEl**] [**-d** *path*] [**-I** *path*] [**-V** *path*] [**-c** *config*]

DESCRIPTION

fbackup combines features of **dump** and **ftio** to provide a flexible, high-speed file system backup mechanism (see *dump*(1M) and *ftio*(1)). **fbackup** selectively transfers files to an output device. For each file transferred, the file's contents and all the relevant information necessary to restore it to an equivalent state are copied to the output device. The output device can be a raw magnetic tape drive, the standard output, a DDS-format tape, a rewritable magneto-optical disk or a file.

The selection of files to backup is done by explicitly specifying trees of files to be included or excluded from an **fbackup** session. The user can construct an arbitrary graph of files by using the **-i** or **-e** options on the command line, or by using the **-g** option with a graph file. For backups being done on a regular basis, the **-g** option provides an easier interface for controlling the backup graph. **fbackup** selects files in this graph, and attempts to transfer them to the output device. The selectivity depends on the mode in which **fbackup** is being used; i.e., full or incremental backup.

When doing full backups, all files in the graph are selected. When doing incremental backups, only files in the graph that have been modified since a previous backup of that graph are selected. If an incremental backup is being done at level 4 and the **-g** option is used, the database file is searched for the most recent previous backup at levels 0-3. If a file's modification time is before the time when the last appropriate session began and the i-node change time is before the time that same session ended, the file is not backed up. Beginning at HP-UX Release 8.0, all directories lying on the path to a file that qualifies for the incremental backup will also be on the backup media, even if the directories do not qualify on their own status.

If **fbackup** is used for incremental backups, a database of past backups must be kept. **fbackup** maintains this data in the text file **/var/adm/fbackupfiles/dates**, by default. Note that the directory **/var/adm/fbackupfiles** must be created prior to the first time **fbackup** is used for incremental backups. The **-d** option can be used to specify an alternate database file. The user can specify to update this file when an **fbackup** session completes successfully. Entries for each session are recorded on separate pairs of lines. The following four items appear on the first line of each pair: the graph file name, backup level, starting time, and ending time (both in *time*(2) format). The second line of each pair contains the same two times, but in *strftime*(3C) format. These lines contain the local equivalent of **STARTED:**, the start time, the local equivalent of **ENDED:**, and the ending time. These second lines serve only to make the dates file more readable; **fbackup** does not use them. All fields are separated by white space. Graph file names are compared character-by-character when checking the previous-backup database file to ascertain when a previous session was run for that graph. Caution must be exercised to ensure that, for example, **graph** and **./graph** are not used to specify the same graph file because **fbackup** treats them as two different graph files.

The general structure of a **fbackup** volume is the same, no matter what type of device is used. There are some small specific differences due to differing capabilities of devices. The general structure is as follows:

- Reserved space for ASCII tape label (1024 bytes)
- **fbackup** specific volume label (2048 bytes)
- session index (size in field of volume label)
- data

Each file entry in the index contains the volume number and the pathname of the file. At the beginning of every volume, **fbackup** assumes that all files not already backed up will fit on that volume; an erroneous assumption for all but the last volume. Indices are accurate only for the previous volumes in the same set. Hence, the index on the last volume may indicate that a file resides on that volume, but it may not have actually been backed up (for example, if it was removed after the index was created, but before **fbackup** attempted to back it up). The only index guaranteed to be correct in all cases is the on-line index (**-I** option), which is produced after the last volume has been written. Specific minor differences are listed below:

- When using 9-track tape drives or DDS-format tape drives several small differences exist. The main blocks of information are separated by EOF. **fbackup** checkpoints the media periodically to enhance error recovery. If a write error is detected, the user normally has two options: First, a new volume can be mounted and that volume rewritten from the beginning. Second, if the volume is not too severely damaged, the good data before the error can be saved, and the write error is treated as a normal end-of-media condition. The blocks of data with their checkpoint records are also separated by EOF. In addition if the DDS-format drive supports **Fast Search Marks** these will be used to enhance recovery speed by placing them between blocks of files.

- For a magneto-optical device, a disk, a file, or standard output, there are no special marks separating the information pieces. Using standard output results in only one volume.

fbackup provides the ability to use UCB-mode tape drives. This makes it possible to overlap the tape rewind times if two or more tape drives are connected to the system.

f

Set-up

There are several things the user will want to consider when setting **fbackup** up for regular use. These include type of device and media, full versus incremental frequency, amount of logging information to keep on-line, structure of the graph file, and on-line versus off-line backup.

The type of device used for backups can affect such things as media expenses, ability to do unattended backup and speed of the backup. Using 9-track tapes will probably result in the highest performance, but require user intervention for changing tapes. A magneto-optical autochanger can provide an unattended backup for a large system and long life media, however the media cost is high. The lowest cost will probably be achieved through DDS-format devices, but at the lowest performance.

It is also important to consider how often full backups should be made, and how many incremental backups to make between full backups. Time periods can be used, such as a full backup every Friday and incrementals on all other days. Media capacities can be used if incremental backups need to run unattended. The availability of personnel to change media can also be an important factor as well as the length of time needed for the backup. Other factors may affect the need for full and incremental backup combinations such as contractual or legal requirements.

If backup information is kept online; i.e., output from the **-V** or **-I** options, the required storage space must also be considered. Index file sizes are hard to predict in advance because they depend on system configuration. Each volume header file takes less than 1536 bytes. Of course the more information that is kept on-line, the faster locating a backup media for a recovery will be.

There are several ways to structure the graph file or files used in a system backup. The first decision involves whether to use one or more than one graph files for the backup. Using one file is simpler, but less flexible. Using two or more graph files simplifies splitting backups into logical sets. For example, one graph file can be used for system disks where changes tend to be less frequent, and another graph file for the users area. Thus two different policies can be implemented for full and incremental backups.

fbackup was designed to allow backups while the system is in use by providing the capability to retry an active file. When absolute consistency on a full backup is important, the system should probably be in single-user mode. However, incremental backups can be made while the system is in normal use, thus improving system up-time.

Options

-c *config* *config* is the name of the configuration file, and can contain values for the following parameters:

- Number of 1024-byte blocks per record,
- Number of records of shared memory to allocate,
- Number of records between checkpoints,
- Number of file-reader processes,
- Maximum number of times **fbackup** is to retry an active file,
- Maximum number of bytes of media to use while retrying the backup of an active file,
- Maximum number of times a magnetic tape volume can be used,
- Name of a file to be executed when a volume change occurs. This file must exist and be executable.
- Name of a file to be executed when a fatal error occurs. This file must exist and be executable.
- The number of files between the **Fast Search Marks** on DDS-format tapes. The cost of these marks are negligible in terms of space on the DDS-format tape. Not all DDS-format devices support fast search marks.

Each entry in the configuration file consists of one line of text in the following format: identifier, white space, argument. In the following sample configuration file, the number of blocks per record is set to 16, the number of records is set to 32, the checkpoint frequency is set to 32, the number of file reader processes is set to 2, the maximum number of retries is set to 5, the maximum retry space for active files is set to 5,000,000 bytes, the maximum number of times a magnetic tape volume can be used is set to 100, the file to be executed at volume change time is **/var/adm/fbackupfiles/chgvol**, the file to be executed when a fatal error occurs is **/var/adm/fbackupfiles/error**, and the number of files between fast search marks is set to 200.

```
blocksperrecord    16
records            32
checkpointfreq     32
readerprocesses    2 (maximum of 6)
maxretries         5
retrylimit         5000000
maxvoluses         100
chgvol             /var/adm/fbackupfiles/chgvol
error              /var/adm/fbackupfiles/error
filesperfsm        200
```

Each value listed is also the default value, except **chgvol** and **error**, which default to null values.

-d *path* This specifies a path to a database for use with incremental backups. It overrides the default database file **/var/adm/fbackupfiles/dates**.

-e *path* *path* specifies a tree to be excluded from the backup graph. This tree must be a subtree of part of the backup graph. Otherwise, specifying it will not exclude any files from the graph. There is no limit on how many times the **-e** option can be specified.

-f *device* *device* specifies the name of an output file. If the name of the file is **-**, **fbackup** writes to the standard output. There is no default output file; at least one must be specified. If more than one output file is specified, **fbackup** uses each one successively and then repeats in a cyclical pattern. Patterns can be used in the device name in a manner resembling file name expansion as done by the shell (see *sh-bourne*(1) and other shell manual entries. The patterns must be protected from expansion by the shell by quoting them. The expansion of the pattern results in all matching names being in the list of devices used.

There is slightly different behavior if remote devices are used. A device on the remote machine can be specified in the form *machine*:*device*. **fbackup** creates a server process from **/usr/sbin/rmt** on the remote machine to access the tape device. If **/usr/sbin/rmt** does not exist on the remote system, **fbackup** creates a server process from **/etc/rmt** on the remote machine to access the tape device. Only half-inch 9-track magnetic tapes or DDS-format tapes can be remote devices. The fast search and save set marks capabilities are not used when remote DDS-format devices are used.

-g *graph* *graph* defines the graph file. The graph file is a text file containing the list of file names of trees to be included or excluded from the backup graph. These trees are interpreted in the same manner as when they are specified with the **-i** and **-e** options. Graph file entries consist of a line beginning with either **i** or **e**, followed by white space, and then the path name of a tree. Lines not beginning with **i** or **e** are treated as an error. There is no default graph file. For example, to backup all of **/usr** except for the subtree **/usr/lib**, a file could be created with the following two records:

```
i /usr
e /usr/lib
```

-i *path* *path* specifies a tree to be included in the backup graph. There is no limit on how many times the **-i** option can be specified.

-n Cross NFS mount points. By default **fbackup** does not cross NFS mount points, regardless of paths specified by the **-i** or **-g** options.

-l Includes LOFS files specified by the backup graph. By default, **fbackup** does not cross LOFS mount points. If **-l** is specified, and the backup graph includes files which are also in a LOFS that is in the backup graph, then those files will backed up twice.

-s Backup the object that a symbolic link refers to. The default behavior is to backup the symbolic link.

-u Update the database of past backups so that it contains the backup level, the time of the beginning and end of the session, and the graph file used for this **fbackup** session. For this update to take place, the following conditions must exist: Neither the **-i** nor the **-e** option can be used; the **-g** option must be specified exactly once (see below); the **fbackup** must complete successfully.

-v Run in verbose mode. Generates status messages that are otherwise not seen.

-y Automatically answer **yes** to any inquiries.

-A Do not back up optional entries of access control lists (ACLs) for files. Normally, all mode information is backed up including the optional ACL entries. With the **-A** option, the summary mode information (as returned by **stat()**) is backed up. Use this option when backing up files from a system that contains ACL to be recovered on a system that does not understand ACL (see *acl*(5)).

-E Do not back up extent attributes. Normally, all extent attributes that have been set are included with the file. This option only applies to file systems which support extent attributes.

-I *path* *path* specifies the name of the on-line index file to be generated. It consists of one line for each file backed up during the session. Each line contains the volume number on which that file resides and the file name. If the **-I** option is omitted, no index file is generated.

-V *path* The volume header information is written to *path* at the end of a successful **fbackup** session. The following fields from the header are written in the format *label***:***value* with one pair per line.

Magic Field	On a valid **fbackup** media it contains the value **FBACKUP_LABEL** (HP-UX release 10.20 and beyond). Before HP-UX release 10.20, it contained the value **FBACKUP LABEL**.
Machine Identification	This field contains the result of **uname -m**.
System Identification	This field contains the result of **uname -s**.
Release Identification	This field contains the result of **uname -r**.
Node Identification	This field contains the result of **uname -n**.
User Identification	This field contains the result of **cuserid()** (see *cuserid*(3S)).
Record Size	This field contains the maximum length in bytes of a data record.
Time	This field contains the clock time when **fbackup** was started.
Media Use	This field contains the number of times the media has been used for backup. Since the information is actually on the media, this field will always contain the value 0.
Volume Number	This field contains a **#** character followed by 3 digits, and identifies the number of volumes in the backup.
Checkpoint Frequency	This field contains the frequency of backup-data-record checkpointing.
Index Size	This field contains the size of the index.
Backup Identification Tag	This field is composed of two items: the process ID (pid) and the start time of that process.
Language	This field contains the language used to make the backup.

-R *restart* Restart an **fbackup** session from where it was previously interrupted. The *restart* file contains all the information necessary to restart the interrupted session. None of the **-**[**ieg0**-**9**] options can be used together with the restart option.

-0-9 This single-digit number is the backup level. Level **0** indicates a full backup. Higher levels are generally used to perform incremental backups. When doing an incremental backup of a particular graph at a particular level, the database of past backups is searched to find the date of the most recent backup of the same graph that was done at a lower level. If no such entry is found, the beginning of time is assumed. All files in the graph that have been modified since this date are backed up.

Access Control Lists (ACLs)

If a file has optional ACL entries, the **-A** option is required to enable its recovery on a system whose access control lists capability is not present.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which files are stored in the backup device and the order output by the **-I** option.

LC_TIME determines the format and contents of date and time strings.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_COLLATE** and **LC_TIME** and **LC_MESSAGES** are not all specified in the environment or if either is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **fbackup** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

fbackup returns one of the following values:

0 upon normal completion.

1 if it is interrupted but allowed to save its state for possible restart.

2 if any error conditions prevent the session from completing.

4 if any warning conditions are encountered.

If warnings occur, the operator should verify the fbackup logs to justify the sanity of the backup taken.

EXAMPLES

In the following two examples, assume the graph of interest specifies all of **/usr** except **/usr/lib** (as described in the **g** key section above).

The first example is a simple case where a full backup is done but the database file is not updated. This can be invoked as follows:

```
/usr/sbin/fbackup -0i /usr -e /usr/lib -f /dev/rmt/c0t0d0BEST
```

The second example is more complicated, and assumes the user wants to maintain a database of past **fbackup** sessions so that incremental backups are possible.

If sufficient on-line storage is available, it may be desirable to keep several of the most recent index files on disk. This eliminates the need to recover the index from the backup media to determine if the files to be recovered are on that set. One method of maintaining on-line index files is outlined below. The system administrator must do the following once before **fbackup** is run for the first time (creating intermediate level directories where necessary):

- Create a suitable configuration file called **config** in the directory **/var/adm/fbackupfiles**
- Create a graph file called **usr-usrlib** in the directory **/var/adm/fbackupfiles/graphs**
- Create a directory called **usr-usrlib** in the directory **/var/adm/fbackupfiles/indices**

A shell script that performs the following tasks could be run for each **fbackup** session:

- Build an index file path name based on both the graph file used (passed as a parameter to the script) and the start time of the session (obtained from the system). For example:

```
/var/adm/fbackupfiles/indices/usr-usrlib/871128.15:17
```

(for Nov 28, 1987 at 3:17 PM)

- Invoke **fbackup** with this path name as its index file name. For example:

```
cd /var/adm/fbackupfiles
/usr/sbin/fbackup -0uc config -g graphs/usr-usrlib\
   -I indices/usr-usrlib/871128.15:17\
      -f /dev/rmt/c0t0d0BEST
```

When the session completes successfully, the index is automatically placed in the proper location.

Note that **fbackup** should be piped to **tcio** when backing up to a CS/80 cartridge tape device see *tcio*(1)). The following example copies the entire contents of directory **/usr** to a cartridge tape:

```
/usr/sbin/fbackup i /usr -f - | tcio -oe /dev/rct/c0d1s2
```

f

WARNINGS

With release 10.20, HP-UX supports large files (greater than 2GB) and increased UID/GIDs (greater than 60,000). Archives containing files with these attributes would cause severe problems on systems that do not support the increased sizes. For this reason, **fbackup** creates tapes with a new magic number ("FBACKUP_LABEL"). This prevents **fbackup** tape archives from being restored on pre-10.20 HP-UX systems. **frecover** still reads both tape formats so that **fbackup** tape archives created on pre-10.20 HP-UX systems can be restored.

Starting with HP-UX Release 8.0, **fbackup** does not back up network special files because RFA networking is obsolete. A warning message is issued if a network special file is encountered in the backup graph and the file is skipped.

The use of **fbackup** for backing up NFS mounted file systems is not guaranteed to work as expected if the backup is done as a privileged user. This is due to the manner in which NFS handles privileged-user access by mapping user **root** and uid **0** to user **nobody**, usually uid **-2**, thus disallowing root privileges on the remote system to a root user on the local system.

The utility set comprised of **fbackup** and **frecover** was originally designed for use on systems equipped with not more than one gigabyte of total file system storage. Although the utilities have no programming limitations that restrict users to this size, complete backups and recoveries of substantially larger systems can cause a large amount system activity due to the amount of virtual memory (swap space) used to store the indices. Users who want to use these utilities, but are noticing poor system-wide performance due to the size of the backup, are encouraged to backup their systems in multiple smaller sessions, rather than attempting to backup the entire system at one time.

Due to present file-system limitations, files whose inode data, but not their contents, are modified while a backup is in progress might be omitted from the next incremental backup of the same graph. Also, **fbackup** does not reset the inode change times of files to their original value.

fbackup allocates resources that are not returned to the system if it is killed in an ungraceful manner. If it is necessary to kill **fbackup**, send it a **SIGTERM**; not a **SIGKILL**.

For security reasons, configuration files and the **chgvol** and **error** executable files should only be writable by their owners.

If sparse files are backed up without using data compression, a very large amount of media can be consumed.

fbackup does not require special privileges. However, if the user does not have access to a given file, the file is not backed up.

fbackup consists of multiple executable objects, all of which are expected to reside in directory **/usr/sbin**.

fbackup creates volumes with a format that makes duplication of volumes by **dd** impossible (see *dd*(1)). Copying an **fbackup** volume created on one media type to another media type does not produce a valid **fbackup** volume on the new media because the formats of volumes on 9-track tape, backup to a file, rewritable optical disks and DDS-format tapes are not identical.

When configuring the parameter **blocksperrecord** (see **-c** option), the record size is limited by the maximum allowed for the tape drive. Common maximum record sizes include 16 1-Kbyte blocks for tape drive models HP 7974 and HP 7978A, 32 blocks for the HP 7978B, 60 blocks for the HP 7980, and 64 blocks for DDS tape drives. Note also that the *blocksize* used in earlier releases (7.0 and before) was 512 bytes, whereas it is now 1024 bytes. This means that the same value specified in *blocksperrecord* in an earlier release creates blocks twice their earlier size in the current release (i.e., a *blocksperrecord* parameter of 32 would create 16-Kbyte blocks at Release 7.0, but now creates 32-Kbyte blocks). If *blocksperrecord* exceeds the byte count allowed by the tape drive, the tape drive rejects the write, causing an error to be communicated to **fbackup** which **fbackup** interprets as a bad tape. The resulting write error message resembles the following:

```
fbackup (3013): Write error while writing backup at tape block 0.
Diagnostic error from tape 11...... SW_PROBLEM  (printed by driver on console)
fbackup (3102): Attempting to make this volume salvageable.
```
etc.

DEPENDENCIES

NFS

Access control lists of networked files are summarized (as returned in **st_mode** by **stat()**), but not copied to the new file (see *stat*(2)).

Series 800

On NIO-bus machines there can be problems when a CS/80 cartridge tape device is on the same interface card as hard disk devices. If writes longer than 16K bytes are made to the tape device, it is possible to have disk access time-out errors. This happens because the tape device has exclusive access to the bus during write operations. Depending on the system activity, this problem may not be seen. The default write size of **fbackup** is 16 Kbytes.

Series 700/800

fbackup does not support QIC-120, and QIC-150 formats on QIC devices. If **fbackup** is attempted for these formats, **fbackup** fails and the following message is displayed :

```
mt lu X: Write must be a multiple of 512 bytes in QIC 120 or QIC 150
```

AUTHOR

fbackup was developed by HP.

FILES

/var/adm/fbackupfiles/dates   database of past backups

SEE ALSO

cpio(1), ftio(1), tcio(1), dump(1M), frecover(1M), restore(1M), rmt(1M), stat(2), acl(5), mt(7).

NAME

fcmsutil - Fibre Channel Mass Storage Utility Command for the Fibre Channel Mass Storage Host Bus Adapters.

SYNOPSIS

/opt/fc/bin/fcmsutil *device_file*

/opt/fc/bin/fcmsutil *device_file* **echo** *remote-N-Port-ID* [*data-size*]

/opt/fc/bin/fcmsutil *device_file* **test** *remote-N-Port-ID* [*data-size*]

/opt/fc/bin/fcmsutil *device_file* **read** *offset*

/opt/fc/bin/fcmsutil *device_file* **write** *offset value*

/opt/fc/bin/fcmsutil *device_file* **lb plm**|**tachyon**

/opt/fc/bin/fcmsutil *device_file* **get local**|**fabric**

/opt/fc/bin/fcmsutil *device_file* **get remote** *N-Port-ID*

/opt/fc/bin/fcmsutil *device_file* **get_lgn** *N-Port-ID*

/opt/fc/bin/fcmsutil *device_file* **reset** [**check**|**clear**]

/opt/fc/bin/fcmsutil *device_file* **stat**

/opt/fc/bin/fcmsutil *device_file* **stat_els**

/opt/fc/bin/fcmsutil *device_file* **clear_stat**

/opt/fc/bin/fcmsutil *device_file* **read_cr**

/opt/fc/bin/fcmsutil *device_file* **lgninfo_all**

DESCRIPTION

The **fcmsutil** command is a diagnostic tool to be used for the Fibre Channel Mass Storage Host Bus Adapters. This command provides the ability to perform Fibre Channel Test and Echo functionality, provides the ability to read and write to the card's registers, etc. This command requires the use of a device file to indicate the interface over which the requested command needs to be performed. **fcmsutil** can be used only by users who have an effective user ID of 0. Some of the options require detailed knowledge of the device specific adapter.

Options

fcmsutil recognizes the following options as indicated in SYNOPSIS. All keywords are case-insensitive and are position dependent.

device_file Can be used alone or with other options.

When used without any options it provides information such as the N_Port ID, Node World Wide Name and Port World Wide Name, Topology of the Fabric, the Speed of the Link, the Hard Physical Address of the Card, the Driver State, the number of Active Outbound Exchanges and number of Active Logins.

echo This option requires two parameters, the *remote-N-Port-ID* and *data-size* (size of packet to send).

A Fibre Channel Echo packet of the specified size is sent to the remote node. The command completes successfully when an echo response is received from the remote node. The command times out if a response is not received in twice RA_TOV time.

Note: Packet size specified must be a multiple of 4.

test This option requires two parameters, the *remote-N-Port-ID* and *data-size* (size of packet to send).

A Fibre Channel Test packet of the specified size is sent to the remote node. The command completes successfully and immediately on sending the test packet.

Note: Packet size specified must be a multiple of 4.

read This option requires one parameter, the *offset* of the register to read from. The *offset* can be specified in either hex or in decimal format. The *offset* specified is an offset from the base of the Tachyon Memory Map. The user of this command is therefore expected to have internal knowledge of the chip.

write This option requires two parameters, the *offset* of the register to write to and the *value* to be written (can be specified in either hex or in decimal format). The *offset* specified is an offset from the base of the Tachyon Memory Map. The user of this command is therefore expected to have internal knowledge of the chip.

lb This option requires one parameter, **plm** or **tachyon**. This command performs an internal loopback test when the **plm** option is specified and performs an external loopback test when the **tachyon** option is specified. The Fibre Channel Chip is programmed in either internal loopback mode (**plm**) or external loopback mode (**glm**) based on the parameter specified. The self test then involves sending a packet and receiving back the packet and checking its integrity.

This is a **destructive** test and data loss during the execution of this test may occur.

get The **get** option is used to obtain Fibre Channel login parameters of either the **local** port, the **fabric** port or of a **remote** port.

get_lgn The **get_lgn** option is used to obtain detailed information maintained in the login block associated with each N_Port that this N_Port has communicated with. The *remote-N-Port-ID* is a required parameter for this option.

reset This option is used to **reset** the card. This is a **destructive** test and communication to all nodes will be terminated till the **reset** process is completed.

stat This option is used to obtain detailed statistics maintained by the driver.

stat_els This option is used to obtain detailed statistics maintained on Extended Link Services from the driver.

clear_stat This option is used to initialize all of the statistics maintained by the driver to zero.

read_cr This option can be used to read all of the readable registers on the card and format the detailed information.

lgninfo_all
This option is used to obtain a comprehensive list of nodes to which a successful login has been established.

AUTHOR

/opt/fc/bin/fcmsutil was developed by HP.

f

NAME

fcutil - Fibre Channel Utility Command for the J2389 Fibre Channel Adapter

SYNOPSIS

/opt/fc/bin/fcutil *device_file*

/opt/fc/bin/fcutil *device_file* **echo** *remote-N-Port-ID* [*data-size*]

/opt/fc/bin/fcutil *device_file* **test** *remote-N-Port-ID* [*data-size*]

/opt/fc/bin/fcutil *device_file* **read** *offset*

/opt/fc/bin/fcutil *device_file* **write** *offset value*

/opt/fc/bin/fcutil *device_file* **lb plm**|**tachyon**

/opt/fc/bin/fcutil *device_file* **get local**|**fabric**

/opt/fc/bin/fcutil *device_file* **get remote** *N-Port-ID*

/opt/fc/bin/fcutil *device_file* **get_lgn** *N-Port-ID*

/opt/fc/bin/fcutil *device_file* **reset** [**check**|**clear**]

/opt/fc/bin/fcutil *device_file* **stat**

/opt/fc/bin/fcutil *device_file* **stat_els**

/opt/fc/bin/fcutil *device_file* **clear_stat**

/opt/fc/bin/fcutil *device_file* **read_cr**

/opt/fc/bin/fcutil *device_file* **lgninfo_all**

DESCRIPTION

The **fcutil** command is a diagnostic tool to be used for the J2389 Fibre Channel Adapter card. This command provides the ability to perform Fibre Channel Test and Echo functionality, provides the ability to read and write to the cards registers, etc. This command requires the use of a device file to indicate the interface over which the requested command needs to be performed. **fcutil** can be used only by users who have an effective user ID of 0. Some of the options require detailed knowledge of the device specific adapter.

Options

fcutil recognizes the following options as indicated in *SYNOPSIS*. All keywords are case-insensitive and are position dependent.

device_file Can be used alone or with other options.

When used without any options it provides information such as the N_Port ID, Node World Wide Name and Port World Wide Name, Topology of the Fabric, the Speed of the Link, the Hard Physical Address of the Card, the Driver State, the number of Active Outbound Exchanges and number of Active Logins.

echo This option requires two parameters, the *remote-N-Port-ID* and *data-size* (size of packet to send).

A Fibre Channel Echo packet of the specified size is sent to the remote node. The command completes successfully when an echo response is received from the remote node. The command times out if a response is not received in twice RA_TOV time.

Note: Packet size specified must be a multiple of 4.

test This option requires two parameters, the *remote-N-Port-ID* and *data-size* (size of packet to send).

A Fibre Channel Test packet of the specified size is sent to the remote node. The command completes successfully and immediately on sending the test packet.

Note: Packet size specified must be a multiple of 4.

read This option requires one parameter, the *offset* of the register to read from. The *offset* can be specified in either hex or in decimal format. The *offset* specified is an offset from the base of the Tachyon Memory Map. The user of this command is therefore expected to have internal knowledge of the chip.

write This option requires two parameters, the *offset* of the register to write to and the *value* to be written (can be specified in either hex or in decimal format). The *offset* specified is an offset from the base of the Tachyon Memory Map. The user of this command is therefore expected to have internal knowledge of the chip.

lb This option requires one parameter, **plm** or **tachyon**. This command performs an internal loopback test when the **plm** option is specified and performs an external loopback test when the **tachyon** option is specified. The Fibre Channel Chip is programmed in either internal loopback mode (**plm**) or external loopback mode (**glm**) based on the parameter specified. The self test then involves sending a packet and receiving back the packet and checking its integrity.

This is a **destructive** test and data loss during the execution of this test may occur.

get The **get** option is used to obtain Fibre Channel login parameters of either the **local** port, the **fabric** port or of a **remote** port.

get_lgn The **get_lgn** option is used to obtain detailed information maintained in the login block associated with each N_Port that this N_Port has communicated with. The *remote-N-Port-ID* is a required parameter for this option.

reset This option is used to **reset** the card. This is a **destructive** test and communication to all nodes will be terminated till the **reset** process is completed.

stat This option is used to obtain detailed statistics maintained by the driver. A normal user should use **netstat** and **lanadmin** to obtain statistics.

stat_els This option is used to obtain detailed statistics maintained on Extended Link Services from the driver.

clear_stat This option is used to initialize all of the statistics maintained by the driver to zero.

read_cr This option can be used to read all of the readable registers on the card and format the detailed information.

lgninfo_all
This option is used to obtain a comprehensive list of nodes to which a successful login has been established.

AUTHOR

/opt/fc/bin/fcutil was developed by HP.

SEE ALSO

netstat(1M), lanadmin(1M).

f

NAME

fddiinit - initialize FDDI network interface; connect to FDDI network

SYNOPSIS

/usr/sbin/fddiinit [**-l** *download_file*] [**-s**] *device_file*

DESCRIPTION

fddiinit:

- Downloads firmware to the FDDI network interface and connects the interface to the FDDI network.
- Must be executed for each interface present on a machine.
- Is also executed from within the **fddi initialization script** during network initialization.
- Is also used to reinitialize and reconnect the interface after the interface has been reset. Use the **fddistop** command to reset the interface (see *fddistop*(1M)).

Options and Command-Line Arguments

fddiinit recognizes the following options and command-line arguments:

-l *download_file*	Specifies the firmware download file. See DEPENDENCIES for machine-dependent details.
-s	(silent) Suppress the progress message. While **fddiinit** is running, it periodically prints a series of dots on the terminal screen to indicate that the firmware download is in progress. When the **-s** option is specified, the dots will not be printed.
device_file	Specifies the device special file associated with the FDDI interface. By convention, device special files are kept in the **/dev** directory. Each device file has a name and a device number to uniquely identify the interface. See DEPENDENCIES for a description of how to create device files.

RETURN VALUE

Upon successful completion, **fddiinit** returns 0; otherwise, it returns 1.

ERRORS

fddiinit fails and the firmware is not downloaded if any of the following conditions are encountered:

- Command used incorrectly – usage message is returned.
- Invalid device file – returns message **Can't open device file**. Check the device file. See DEPENDENCIES for description of how to create device files.
- Invalid download file – returns **Can't open download file** or **Invalid file format**. Contact your HP Customer Support representative.
- Hardware or driver error – download was unsuccessful because of a hardware or firmware problem. Check to ensure that hardware is correctly connected. If the download is still unsuccessful, replace the card with a known-good unit if one is available, and retry the command. Otherwise, contact your HP customer support representative.

DEPENDENCIES

Series 700 Built-In FDDI:

Each device file has a name and a device number to uniquely identify the interface. To create the Built-In FDDI device file manually (instead of through SAM), specify the applicable major and minor numbers in the HP-UX **/usr/sbin/mknod** command. Built-In FDDI device files have the following major and minor numbers:

Major	Minor	Card Instance number
111	**0xYY0000**	YY

The following example uses **/usr/sbin/mknod** to create the Built-In FDDI device special file **/dev/lan1** on the FDDI Built-In card device with a card instance of 1:

```
/usr/sbin/mknod /dev/lan1 c 111 0x010000
```

If the FDDI interface card is configured using SAM (see *sam*(1M)), SAM creates the device file automatically and the name corresponds to the network interface name and unit. For example, device files **`/dev/lan1`** and **`/dev/lan2`** are for network interfaces **`lan1`** and **`lan2`** respectively. You can also use the **`/usr/sbin/lanscan`** command to display information about the network interfaces on the system.

`fddiinit` does not require a download file for the Built-In FDDI card.

Series 800:

Device files for HPPB FDDI are created automatically by **`/usr/sbin/insf`** (see *insf*(1M)) when the system is rebooted after installing the HPPB FDDI driver and adapter card. The device file name is of the form **`/dev/lan`***X* where *X* >= 0.

The major number for HPPB FDDI device files is **`191`**. The minor number containing the card instance number is assigned based on the configuration of the HPPB FDDI card in the HPPB backplane relative to other LAN cards. Each LAN card has a unique minor device number.

To determine the device special file corresponding to a particular FDDI adapter, first use the **`/usr/sbin/lanscan`** command (see *lanscan*(1M)) to obtain the card instance number that matches the hardware path of that adapter. Then use the **`/usr/sbin/lssf`** command (see *lssf*(1M)) on those files in the **`/dev`** directory that have a major number of **`191`** to find a file that has a matching card instance number.

`mksf` (see *mksf*(1M)) can be used to manually create a device file for the HPPB FDDI interface.

The default download file is **`/usr/lib/fddi_dnld`**. This download file is used for the HPPB FDDI card.

AUTHOR

`fddiinit` was developed by HP.

FILES

`/usr/lib/fddi_dnld`	default FDDI download file.

SEE ALSO

fddistop(1M), fddistat(1M), fddinet(1M), mknod(1M), lanscan(1M).

NAME

fddinet - display logical FDDI ring map information

SYNOPSIS

/usr/bin/fddinet [**-n**] [**-d** *station_address*] *device_file*

DESCRIPTION

fddinet displays logical connection information for the reachable nodes connected to the same FDDI ring.

Options and Command-Line Arguments

fddinet recognizes the following options and command-line arguments:

-n Use FDDI native form when displaying address information. The default is the canonical form.

-d *station_address*
Specifies the MAC Address of the node that is to be first in the display of the logical ring map. If the **-n** option is used in the command line, the MAC Address is a 12-character, hexadecimal-digit string in FDDI native form; otherwise, the default canonical form is used. It can start with or without the usual **0x** prefix. For example, both **0x080009091219** and **080009091219** are valid MAC Addresses.

device_file Device special file associated with the FDDI interface. By convention, device files are kept in the **/dev** directory. Each device file has a name and a device number to uniquely identify the interface. See the DEPENDENCIES section of *fddiinit*(1M) for a description of how to create device files.

RETURN VALUE

Upon successful completion, **fddinet** returns 0; otherwise it returns 1.

ERRORS

fddinet fails if any of the following conditions is encountered:

- Command used incorrectly - Usage message is returned.
- Invalid device file - returns **Can't open device file**. Check the device file. See DEPENDENCIES section of *fddiinit*(1M) for a description of how to create device files.
- Hardware or driver error - hardware failed to respond to the request. Ensure that the hardware is correctly connected, then use **fddiinit** to reinitialize the interface if necessary (see *fddiinit*(1M)). If the same failure happens after interface reinitialization, replace the interface with a known-good unit, if one is available, and retry the command. Otherwise, contact your HP Customer Support representative.

EXAMPLES

```
MAC_Address      Node_Type                UNA              Topology
0x080009091319   SAS Station(1 MAC)       0x080009091329   Wrapped
0x080009091329   SAS Station(1 MAC)       0x08006A0D0225   Wrapped
0x08006A0D0225   Concentrator(6 Port)     0x08000909133F   Rooted
0x08000909133F   SAS Station(1 MAC)       0x080009091319   Wrapped
```

Fields are defined as follows:

MAC_Address Specifies the 48-bit MAC Address of the node in hexadecimal format. The default is canonical form. FDDI native form is used if the **-n** option appears in the command line.

Node_Type Specifies whether the node is a Single Attachment Station (SAS), Dual Attachment Station (DAS), or Concentrator. SAS and DAS station types include the MAC count displayed inside parentheses after the node type; concentrator station types include the number of master ports inside parentheses after the node type.

UNA Specifies the MAC Address of the upstream neighbor in hexadecimal format. The default is canonical form. FDDI native form is used if the **-n** option appears in the command line.

Topology Displays the topology of the station. Possible values are:

Wrapped	Set when the station's attachment state is **Wrap_A**, **Wrap_B**, **Wrap_S**, or **Wrap_AB**.
Unrooted	Set when a concentrator has no active A, B or S Port.
Twisted A-A	Set when an A-A connection is detected in the station.
Twisted B-B	Set when a B-B connection is detected in the station.
Rooted	Set when the station does not have an A or B or S Port active in tree mode.
SRF	Set if the station supports the Status Report (SRF) protocol.

AUTHOR

fddinet was developed by HP.

SEE ALSO

fddiinit(1M), fddistop(1M), fddistat(1M), mknod(1M).

NAME

fddipciadmin - show PCI FDDI interface status

SYNOPSIS

/usr/bin/fddipciadmin *interface_name*

DESCRIPTION

fddipciadmin displays information about the status of the PCI FDDI interface. The **fddipciadmin** utility first shows summary information about the PCI FDDI interface. It then displays a menu that allows the user to refresh statistics and display other interface attributes.

Command-Line Argument

fddipciadmin requires the command-line argument:

interface_name Specifies the interface name for the PCI FDDI device (for example, lan2) or the name of the character device file for the PCI FDDI device (for example, /dev/lan2).

RETURN VALUE

Upon successful completion, **fddipciadmin** returns 0; otherwise it returns 1.

ERRORS

fddipciadmin fails if any of the following conditions is encountered:

- Command used incorrectly - Usage message is returned.
- Device is not a PCI FDDI device - returns "**/dev/***interface_name* is not a valid device". Check the interface name. Use **lanscan** to display interface name and type. Use **ioscan** to determine the type of FDDI device. Check that the driver was properly installed (use the **dmesg** command or **what /stand/vmunix** and look for the PCI FDDI driver, fddi4).
- Device does not exist - returns "device file **/dev/***interface_name* does not exist". Check the interface name. Use **lanscan** to display interface name and type.
- Hardware or driver error - returns "failed *ioctl_request*, check the hardware," where *ioctl_request* is the name of the **ioctl** operation requested. Check the LEDs on the card. Run **lanscan** and check the hardware state.

When **fddipciadmin** starts, it reads the current statistics from the PCI FDDI driver. From the **fddipciadmin** menu you can force **fddipciadmin** to re-read (refresh) these statistics. The **fddipciadmin** menu also has options to re-display the summary information, display SMT (FDDI Station Management) attributes, display FDDI MAC (Media Access Control) attributes, display Port A attributes, display Port B attributes, display path attributes (attributes of the logical segment of the FDDI ring that passes through this station), display the link-level multicast addresses configured for this station, display driver statistics, display link statistics, and exit the program.

EXAMPLES

```
                    << INTERFACE STATUS SUMMARY >>

 MAC Address:         0x0060B0580E19     Wire Format:         0x00060D1A7098
 Up Stream Neighbor:  0x080009455DD5     Wire Format:         0x100090A2BAAB
 Down Stream Neighbor:0x0060B0580E03     Wire Format:         0x00060D1A70C0
 RMT State:           Ring_Op            CF State:            C_Wrap_A
 Frame Count:         00322395           Token Count:         57108118
 Receive Count:       00000339           Transmit Count:      342
 Lost Count:          0                  Error Count:         0
 RingOp Count:        1
 LER Estimate A:      10**-15            LER Estimate B:      10**-15
 T_Req (ms):          7.9873             T_Neg (ms):          5.0001
 Number of multicast addresses configured: 0
```

Fields are defined as follows:

MAC Address Medium Access Control (unit) Address. Specifies the 48-bit MAC Address of the node in canonical (Least Significant Bit) hexadecimal format. The *Wire Format* shows the MAC address in Most Significant Bit order.

Up Stream Neighbor
Upstream Neighbor's (MAC) Address. Specifies the MAC Address of the upstream neighbor in canonical hexadecimal format. The *Wire Format* shows the MAC address in Most Significant Byte order.

Down Stream Neighbor
Downstream Neighbor's (MAC) Address. Specifies the MAC Address of the downstream neighbor in canonical hexadecimal format. The *Wire Format* shows the MAC address in Most Significant Byte order.

RMT State Ring Management State. Indicates whether the state is: Isolated, Non_Op, Ring_Op, Detect, Non_Op_Dup, Ring_Op_Dup, Directed, Trace or Unknown. The normal state is Ring_Op. Isolated usually indicates that the interface is not connected to the network. Refer to the RMT description in the ANSI FDDI/SMT specification for more details.

CF State (Attachment) Configuration State of the station. Indicates whether the state is: Isolated, Local_A, Local_B, Local_S, Wrap_A, Wrap_B, Wrap_AB, Wrap_S, C_Wrap_A, C_Wrap_B, C_Wrap_S, Thru or Unknown. The normal state is Thru. The Isolated state indicates that there is no internal connection between MAC (Media Access Control) and PHY (Physical Layer Protocol) modules. Usually indicates the card is not cabled to ring. The Wrap_A, Wrap_B, Wrap_AB, C_Wrap_A and C_Wrap_B states indicate that there is only a single data ring. Data for the ring is transmitted and received through the same port (A or B). This usually indicates that a dual ring is wrapped (a failure was detected and a surrounding node is "wrapping" and using either the A or B port to send and receive data) or this is a transitory startup state indicating that the A or B port is ready to be incorporated into the ring. Refer to the CF_State variable description in the ANSI FDDI/SMT specification for more details.

Frame Count Specifies the total number of frames received with End Delimiter by the station. This count includes void frames, token frames, beacon frames, claim frames, SMT frames and LLC frames.

Token Count The number of times a token (both restricted and non-restricted tokens) has been received. Useful for determining the network load.

Receive Count Specifies the total number of non-MAC frames (SMT or LLC frames) with an address recognized by the station and successfully received by the station.

Transmit Count
Specifies the total number of non-MAC transmit frames originated by this station.

Lost Count Specifies the total number of frames received with format error detected. When the station detects a frame with a format error, it strips the rest of the frame from the ring and replaces it with Idle symbols.

Error Count Specifies the total number of frames received with the End Delimiter not set (not 'S').

RingOp Count Ring Operational Count. Indicates the number of times the ring has entered an operational state from a non-operational state. (This value is not required to be exact and the actual count may be greater than the number shown.)

LER Estimate A
Link Error Rate Estimate. Specifies the estimated long term average link error rate for Port A.

LER Estimate B
Link Error Rate Estimate. Specifies the estimated long term average link error rate for Port B.

T_Req Token Request Time. Specifies the requested Target Token Rotation Time (TTRT) by the local station in the claim token process in milliseconds. Refer to the T_Req value description in the ANSI FDDI/SMT specification for more details.

T_Neg Negotiated Target Token Rotation Time (TTRT). Specifies the target rotation time being used by all the stations on the ring. This value is negotiated during the claim token process. The value of *T_Neg* is in milliseconds. Refer to the *T_Neg* value description in the ANSI FDDI/SMT specification for more details.

Number of multicast addresses configured
Specifies the number of link-level multicast addresses configured for this interface.

AUTHOR

fddipciadmin was developed by HP.

SEE ALSO

fddi(7), netstat(1), mknod(1M).

NAME

fddisetup - initialize and connect all system FDDI network interfaces

SYNOPSIS

/usr/sbin/fddisetup

DESCRIPTION

fddisetup:

- Scans the kernel I/O system data structures for all FDDI interface cards installed on the system. It invokes **fddiinit** with default parameters for every FDDI interface card found (see *fddiinit*(1M)).
- The **fddisetup** command must be present in the **fddi initialization script** and the **/etc/local/powerfail** file. It is invoked at system start-up to download all FDDI cards in the system before IP addresses are assigned. The entry in the **/etc/local/powerfail**(1M) file must be present after the **/sbin/dasetup** entry. It is invoked when the system recovers from a power-failure condition to reinitialize the FDDI interface cards.
- It can also be invoked manually. However, this is not recommended as it reinitializes all FDDI interface cards present on the system. When it is necessary to reinitialize and reconnect a specific interface, use **fddiinit** interactively.

Command-Line Arguments

fddisetup does not support any options or arguments.

RETURN VALUE

Upon successful completion, **fddisetup** returns 0; otherwise, it returns 1.

ERRORS

If **fddisetup** fails, the firmware may be downloaded on some FDDI cards on the system and not on others. If this happens, use **lanscan** to determine which FDDI interface cards have been downloaded properly and which ones were not (see *lanscan*(1M)). **lanscan** shows an **UP** hardware state for cards that have been downloaded properly. Use **fddiinit** interactively to manually download FDDI interface cards.

The error message **fddisetup: Can't read I/O configuration** indicates an access permissions problem. Log in as super-user and try the command again.

fddisetup can also produce error messages returned by the **fddiinit** command (see *fddiinit*(1M)).

AUTHOR

fddisetup was developed by HP.

SEE ALSO

fddistop(1M), fddistat(1M), fddinet(1M), mknod(1M), lanscan(1M).

NAME

fddistat - show FDDI interface status

SYNOPSIS

/usr/sbin/fddistat [**-n**] *device_file*

DESCRIPTION

fddistat displays information about the status of the FDDI interface.

Options and Command-Line Arguments

fddistat recognizes the following options and command-line arguments:

-n Use FDDI native form when displaying address information. The default is canonical form.

device_file Specifies the device special file associated with the FDDI interface. By convention, device files are kept in the **/dev** directory. Each device file has a name and a device number to uniquely identify the interface. See DEPENDENCIES section of *fddiinit*(1M) for a description of how to create device files.

RETURN VALUE

Upon successful completion, **fddistat** returns 0; otherwise it returns 1.

ERRORS

fddistat fails if any of the following conditions is encountered:

- Command used incorrectly - Usage message is returned.
- Invalid device file - returns **Can't open device file**. Check the device file. See DEPENDENCIES section of *fddiinit*(1M) for a description of how to create device files.
- Hardware or driver error - hardware failed to respond to the request. Ensure that hardware is correctly connected, and use the **fddiinit** command to reinitialize the interface if it is necessary (see *fddiinit*(1M)). If the same failure occurs after the interface is reinitialized, replace the interface with a known-good unit, if available, then retry the command. Otherwise, contact your HP Customer Support representative.

EXAMPLES

```
MAC_Address   0x080009091335
UNA           0x080009091189
RMT           Ring_Op
CF_State      Wrap_S
Frame_Ct      5000
Receive_Ct    3500
Transmit_Ct   4000
Lost_Ct       12
Error_Ct      1
LER_Estimate  10**-15
T_Req (ms)    150
T_Neg (ms)    150
```

Fields are defined as follows:

MAC_Address Medium Access Control (unit) Address. Specifies the 48-bit MAC Address of the node in hexadecimal format. The default is canonical form. FDDI native form is used if the **-n** option is specified in the command line.

UNA Upstream Neighbor's (MAC) Address. Specifies the MAC Address of the upstream neighbor in hexadecimal format. The default is canonical form. FDDI native form is used if the **-n** option appears in the command line.

RMT Ring Management State. Indicates whether the state is: Isolated, Non_Op, Ring_Op, Detect, Non_Op_Dup, Ring_Op_Dup, Directed, or Trace. Refer to the RMT description in the ANSI FDDI/SMT specification for more details.

CF_State (Attachment) Configuration State of the station. Indicates whether the state is: Isolated, Wrap_S, Wrap_A, Wrap_B, Wrap_AB, or Thru. Only the Isolated and the

Wrap_S states are valid for single attachment station (SAS). Refer to the CF_State variable description in the ANSI FDDI/SMT specification for more details.

Frame_Ct	Frame Count. Specifies the total number of frames received with End Delimiter by the station. This count includes void frames, token frames, beacon frames, claim frames, SMT frames and LLC frames.
Receive_Ct	Receive Count. Specifies the total number of SMT or LLC frames successfully received by the station.
Transmit_Ct	Transmit Count. Specifies the total number of transmit frames originated by this station.
Lost_Ct	Lost Count. Specifies the total number of frames received with format error detected. When the station detects a frame with a format error, it strips the rest of the frame from the ring and replaces it with Idle symbols.
Error_Ct	Error Count. Specifies the total number of frames received with the End Delimiter not set (not 'S').
LER_Estimate	Link Error Rate Estimate. Specifies the long term average link error rate. It ranges from 10^{-4} to 10^{-15}.
T_Req	Token Request Time. Specifies the requested Target Token Rotation Time (TTRT) by the local station in the claim token process. The value of *T_Req* is in milliseconds. Refer to the T_Req value description in the ANSI FDDI/SMT specification for more details.
T_Neg	Negotiated Target Token Rotation Time (TTRT). Specifies the target rotation time being used by all the stations on the ring. This value is negotiated during the claim token process. The value of *T_Neg* is in milliseconds. Refer to the *T_Neg* value description in the ANSI FDDI/SMT specification for more details.

AUTHOR

fddistat was developed by HP.

SEE ALSO

netstat(1), fddiinit(1M), fddistop(1M), fddinet(1M), mknod(1M).

NAME

fddistop - stop and reset the FDDI interface

SYNOPSIS

/usr/sbin/fddistop *device_file*

DESCRIPTION

fddistop disconnects the interface from the FDDI ring and resets the firmware to the reset state. Use the **fddiinit** command to reinitialize the interface and reconnect to the FDDI network (see *fddiinit*(1M)).

Command-Line Arguments

fddistop recognizes the following command-line argument:

device_file	Specifies the device special file for the FDDI interface. By convention, device files are kept in the **/dev** directory. Each device file has a name and a device number to uniquely identify the interface. See DEPENDENCIES section of *fddiinit*(1M) for a description of how to create device files.

RETURN VALUE

Upon successful completion, **fddistop** returns 0; otherwise it returns 1.

ERRORS

fddistop fails if any of the following conditions are encountered:

- Command used incorrectly – usage message is returned.
- Invalid device file – returns **Can't open device file**. Check the device file. See DEPENDENCIES section of *fddiinit*(1M) for a description of how to create device files.

AUTHOR

fddistop was developed by HP.

SEE ALSO

fddiinit(1M), fddistat(1M), fddinet(1M), mknod(1M).

NAME

fddisubagtd - FDDI SNMP subagent daemon

SYNOPSIS

/usr/sbin/fddisubagtd

DESCRIPTION

fddisubagtd starts the FDDI SNMP subagent which handles the GET and SET requests for FDDI MIB.

The **fddisubagtd** provides RFC 1512 defined network management functionality. It works within the HP OpenView network management framework. The SNMP Master Agent sends requests for Management Information Base (MIB) values to **fddisubagtd**. The subagent replies with the information requested.

The **fddisubagtd** provides functionality to GET and SET various SMT 7.2 statistics as defined in RFC 1512.

AUTHOR

fddisubagtd was developed by HP.

SEE ALSO

snmpd(1M).

RFC 1512.

NAME

fdetach - detach a STREAMS-based file descriptor from a filename

SYNOPSIS

fdetach *path*

DESCRIPTION

The **fdetach** command detaches or disassociates a file descriptor for an open STREAMS device or pipe from its filename in the file system. The *path* argument is the *path* that was previously associated with the file descriptor by the **fattach()** function.

Operations on *path* will subsequently affect the file system node, not the STREAMS device or pipe. The permissions and status of the node are returned to the state that they were in before the STREAMS device or pipe was attached. Any other paths that the STREAMS device or pipe may be attached to are not affected.

To successfully issue the **fdetach** command, the user must be superuser or must be the owner of the file and have write permission.

RETURN VALUE

fdetach returns 0 (zero) on success. If **fdetach** fails, it returns 1 and prints a message to **stderr**.

EXAMPLES

To detach the file descriptor for the STREAMS file **/tmp/streamfile** from its associated file system node, enter:

```
fdetach /tmp/streamfile
```

FILES

/usr/lib/nls/C/fdetach.cat	NLS catalog for **fdetach**.

SEE ALSO

fattach(3c), fdetach(3c), streamio(7).

NAME

ff(generic) - list file names and statistics for a file system

SYNOPSIS

/usr/sbin/ff [**-F** *FStype*] [**-o** *specific_options*] [**-V**] *special* ...

DESCRIPTION

The **ff** command reads the i-list and directories of each *special* file, assuming it to be a file system, saving i-node data for files that match the selection criteria. Output consists of the path name for each saved i-node, plus any other file information requested with the **-o** option. Output fields are positional. The output is produced in i-node order; fields are separated by tabs. The default line produced by **ff** includes the path name and i-number fields.

Options and Arguments

ff recognizes the following options and arguments:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching each *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-o *specific_options*
Specify options specific to each file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for a specific *FStype*-specific module of the command. See the file-system-specific man pages for a description of the *specific_options* supported, if any.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

List the path names and i-numbers of all files in the file system **/dev/dsk/c1d2s0**:

```
ff /dev/dsk/c1d2s0
```

Execute the **ff** command on HFS file system **/dev/dsk/c1d2s0**:

```
ff -F hfs /dev/dsk/c1d2s0
```

Display a completed command line without executing the command:

```
ff -V /dev/dsk/c1d2s0
```

FILES

/etc/default/fs	File that specifies the default system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

find(1), ff_*FStype*(1M), fstyp(1M), ncheck(1M), fstab(4), fs_wrapper(5).

f

NAME

ff(hfs) - list file names and statistics for HFS file system

SYNOPSIS

/usr/sbin/ff [**-F hfs**] [**-a** *num*] [**-c** *num*] [**-i** *inode-list*] [**-I**] [**-l**] [**-m** *num*] [**-n** *file*]
[**-p** *prefix*] [**-s**] [**-u**] [**-V**] *special* ...

DESCRIPTION

The **ff** command reads the i-list and directories of each special file *special*, assuming it to be an HFS file system, saving i-node data for files that match the selection criteria. Output consists of the path name for each saved i-node, plus any other file information requested using the print options below. Output fields are positional. The output is produced in i-node order; fields are separated by tabs. The default line produced by **ff** contains the path name and i-number fields. With all options specified, the output fields include path name, i-number, size, and user ID.

The *num* parameter in the options descriptions is a decimal number, where **+***num* means more than *num*, **-***num* means less than *num*, and *num* means exactly *num*. A day is defined as a 24-hour period.

ff lists only a single path name out of many possible ones for an i-node with more than one link, unless you specify the **-l** option. With **-l**, **ff** applies no selection criteria to the names listed. All possible names for every linked file on the file system are included in the output. On very large file systems, memory may run out before **ff** completes execution.

Options and Arguments

ff recognizes the following options and arguments:

-a *num*	Select a file if the i-node has been accessed in *num* days.
-c *num*	Select a file if the i-node has been changed in *num* days.
-F hfs	Specify the HFS file system type.
-i *inode-list*	Generate names for any i-node specified in the *inode-list*.
-I	Do not display the i-node number after each path name.
-l	Generate a list of all path names for files with more than one link.
-m *num*	Select a file associated with an i-node if it has been modified in *num* days.
-n *file*	Select a file associated with an i-node if it has been modified more recently than the specified *file*.
-p *prefix*	Add the specified *prefix* to each path name. The default prefix is **.** (dot).
-s	Write the file size, in bytes, after each path name.
-u	Write the owner's login name after each path name.
-V	Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

List the path names and i-numbers of all files in the file system **/dev/dsk/c1d2s0**:

```
ff /dev/dsk/c1d2s0
```

Same as above, but suppress the printing of i-numbers:

```
ff -I /dev/dsk/c1d2s0
```

List files on the same file system that have been modified recently, displaying the path name, i-number, and owner's user name (the **-u** option). List only files that have been modified within the last two days (the **-m -2** option):

```
ff -m -2 -u /dev/dsk/c1d2s0
```

List all files on the same file system, including the path name and i-number of each file, that was last accessed more than 30 days ago (**-a +30**):

```
ff -a +30 /dev/dsk/c1d2s0
```

Find all path names associated with i-nodes **451** and **76** (the **-l** option):

```
ff -l -i 451,76 /dev/dsk/c1d2s0
```

Execute the **ff** command on an HFS file system **/dev/dsk/c1d2s0**:

```
ff -F hfs /dev/dsk/c1d2s0
```

FILES

/etc/fstab	Static information about the file systems.

SEE ALSO

find(1), ff(1M), ncheck(1M), fstab(4).

NAME

ff (vxfs) - fast find: list file names and statistics for a VxFS file system

SYNOPSIS

/usr/sbin/ff [**-F vxfs**] [**-VIlsu**] [**-p** *prefix*] [**-a** *num*] [**-m** *num*] [**-c** *num*] [**-n** *file*]
[**-i** *inode-list*] [**-o** *specific_options*] *special* ...

DESCRIPTION

The **ff** command reads the i-list and directories of the special file *special*, assuming it to be a VxFS file system, printing i-node data for files that match the selection criteria. Output consists of the pathname for each saved i-node, plus any other file information requested using the print options below. Output fields are positional. The output is produced in i-node order; fields are separated by tabs. The default line produced by the **ff** command includes the path name and i-number fields. With all options specified, the output fields include path name, i-number, size, and user ID.

f

The *num* parameter in the options descriptions is a decimal number, where **+***num* means more than *num* days, **-***num* means less than *num* days, and *num* means exactly *num* days. A day is defined as a 24 hour period.

Options

ff recognizes the following options:

-a *num*	Select a file if the i-node has been accessed in *num* days.
-c *num*	Select a file if the i-node has been changed in *num* days.
-F vxfs	Specify the VxFS file system type.
-i *inode-list*	Generate names for any i-nodes specified in the *inode-list*.
-I	Does not display the i-node number after each path name.
-l	Generates a list of all path names for files with more than one link.
-m *num*	Select a file associated with the i-node if it has been modified in *num* days.
-n *file*	Select a file associated with an i-node if it has been modified more recently than the specified *file*.
-p *prefix*	Adds the specified *prefix* to each path name. The default prefix is **.** (dot).
-o *specific_options*	Specify options specific to the VxFS file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for the VxFS specific module of the command. The available option is: **s** Display only special files and files with set-user-ID mode.
-s	Writes the file size, in bytes, after each path name.
-u	Writes the owner's login name after each path name.
-V	Echo the completed command line, but performs no other action. The command line is generated by incorporating the user specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

List the path names and i-numbers of all files in the file system **/dev/vg01/rlvol1**:

```
ff /dev/vg01/rlvol1
```

Same as above, but suppress the printing of i-numbers:

```
ff -I /dev/vg01/rlvol1
```

List files on the same file system that have been modified recently, displaying the path name, i-number, and owner's user name (**-u** option). List only files that have been modified within the last two days (**-m -2** option):

```
ff -m -2 -u /dev/vg01/rlvol1
```

List all files on the same file system, including the path name and i-number of each file, that was last accessed more than 30 days ago (**`-a +30`**):

```
ff -a +30 /dev/vg01/rlvol1
```

Find all path names associated with i-nodes **`451`** and **`76`** (**`-l`** option):

```
ff  -l  -i  451,76  /dev/vg01/rlvol1
```

Execute the **`ff`** command on a VxFS file system **`/dev/vg01/rlvol1`**:

```
ff -F vxfs /dev/vg01/rlvol1
```

FILES

`/etc/fstab`	Static information about the file systems.

SEE ALSO

ff(1M), find(1), fstab(4), ncheck(1M).

NAME

fingerd - remote user information server

SYNOPSIS

/usr/lbin/fingerd [**-r**]

DESCRIPTION

fingerd is the server for the RFC 742 Name/Finger protocol. It provides a network interface to **finger**, which gives a status report of users currently logged in on the system or a detailed report about a specific user (see *finger*(1)). The Internet daemon executes **fingerd** when it receives a service request at the port listed in the services data base for "finger" using "tcp" protocol; see *inetd*(1M) and *services*(4).

To start **fingerd** from **inetd**, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
finger stream tcp nowait bin /usr/lbin/fingerd fingerd
```

Once a remote host is connected, **fingerd** reads a single "command line" terminated by a carriage-return and line-feed. It uses this command line as the arguments to an invocation of **finger**. **fingerd** sends the output of **finger** to the remote host and closes the connection.

If the command line is null (contains only a carriage-return and line-feed pair), **finger** returns a report that lists all users logged in on the system at that moment.

If a user name is specified on the command line (for example, *user*<CR><LF>), the response lists more extended information for only that particular user, whether logged in or not. See *finger*(1) for the details of this extended information.

If **fingerd** is run with the **-r** option, it allows remote user names on the command line (for example, *user@host*<CR><LF>). Otherwise, if the command line contains a remote user name, **fingerd** prints the error message **Remote finger not allowed** and closes the connection.

AUTHOR

fingerd was developed by the University of California, Berkeley and HP.

SEE ALSO

finger(1), inetd(1M), services(4),
RFC 742 for the Name/Finger protocol.

NAME

fixman - fix manual pages for faster viewing with man(1)

SYNOPSIS

/usr/sbin/fixman [**-A** *alt-path*]

DESCRIPTION

The **fixman** command is a shell script that processes man pages in the **cat*** directories to unexpand spaces to tabs where possible, and to remove all character-backspace pairs (which usually exist to cause overstriking or underscoring for printer output). Removal of unnecessary character sequences improves the speed of *man*(1), and reduces disk space consumption. The **fixman** command should be run after using **catman** to create formatted, **cat**-able manual entries from unformatted, *nroff*(1)-compatible source files (see *catman*(1M)).

By default, **fixman** searches for **cat*** subdirectories in the following parent directories in the order indicated:

- **/usr/share/man**
- **/usr/contrib/man**
- **/usr/local/man**

If the **MANPATH** environment variable is set, the directory paths specified by **MANPATH** are searched instead of the default. See *environ*(5) for a description of the **MANPATH** environment variable.

The **fixman** command does not remove duplicate blank lines. Thus, all files remain a multiple of one page (66 lines) long and can still be passed directly to **lp** (see *lp*(1)). (Note that *man*(1) normally uses **more -s** to accomplish this removal.)

To ensure success, **fixman** should be run by a user who has appropriate privileges. It will take awhile to complete depending on system speed, load, memory size, etc. As a side-effect, file ownerships and permissions may be changed.

Options

-A *alt-path*
Perform actions based on the given alternate root. With this option, *alt-path* will be prepended to all directory paths, including default paths or the paths defined by **MANPATH**.

EXTERNAL INFLUENCES

Environment Variables

MANPATH, if set, defines the directories to be searched for **cat**-able manual entries.

WARNING

If the value of **MANPATH** is not the same while **fixman** is running as it was when **catman** was run or when manpage files were installed, some files may be missed and not processed (see *catman*(1M)).

EXAMPLES

Run fixman from a server to fix the manual pages on a diskless under the alternate root **/export/shared_roots/OS_700**:

```
fixman -A /export/shared_roots/OS_700
```

This will fix manpages in **cat*** directories under:

```
/export/shared_roots/OS_700/usr/share/man/
/export/shared_roots/OS_700/usr/contrib/man/
/export/shared_roots/OS_700/usr/local/man/
```

FILES

/usr/share/man/cat∗[.Z]	Directories containing [compressed] *nroff*(1)-formatted versions of manual entries
/usr/local/man/cat∗[.Z]	
/usr/contrib/man/cat∗[.Z]	

AUTHOR

fixman was developed by HP.

SEE ALSO

catman(1M), chmod(1), expand(1), lp(1), man(1), mv(1), sed(1), environ(5).

NAME

format - format an HP SCSI disk array LUN

SYNOPSIS

`format` *device_file*

DESCRIPTION

`format` formats one LUN of the HP SCSI disk array associated with device file, *device_file*. The format will usually be a soft or zeroing format, in which the controller writes zeroes to the data area and parity area, if any, of the LUN.

NOTE: The above should always be true of a sub-LUN, but the controller might decide, based on certain conditions, to do a full format of a regular LUN, which consists of sending a mode select and a media initialization command to the physical drive(s) in question, followed by zeroing the data and parity area, if any. The conditions which will cause a full format to be done are as follows:

1. The controller received a Mode Select command which requires a drive sector size change.
2. The controller received a Mode Select command which changed a parameter in the Format Device Page (0x03).
3. The LUN contains one or more failed drives. In this case only a certain subset of the drives containing the failed drives will be formatted.
4. Either the FmtData or the CmpLst bit in the Format Unit CDB is set.

f

RETURN VALUE

`format` returns the following values:

0	Successful completion.
-1	Command failed.

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **`format`**
- SCSI (device level) communications
- system calls

Error messages generated by format:

`usage: format <special>`

An error in command syntax has occurred. Enter command again with all required arguments, in the order shown.

`format: device busy`

To ensure that **`format`** does not modify a disk array that is being used by another process, **`format`** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**`device busy`**" error message is returned by the driver. To eliminate the "**`device busy`**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before formatting array LUNs (see *vgchange*(1M)).

`format: LUN # too big`

The LUN number, which is derived from the device file name, is out of range.

`format: LUN does not exist`

The addressed LUN is not configured, and thus is not known to the array controller.

`format: Not a raw file`

Utilities must be able to open the device file for raw access.

`format: Not an HP SCSI disk array`

The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

format uses the following system calls:

> **malloc()**, **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **format** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To format the HP SCSI disk array LUN **/dev/rdsk/c2t0d0** on a Series 800:

```
format /dev/rdsk/c2t0d0
```

WARNING

The **format** command will destroy all user data on the addressed LUN.

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

format was developed by HP.

NAME

frecover - selectively recover files

SYNOPSIS

/usr/sbin/frecover -r [**-hmosvyAFNOX**] [**-c** *config*] [**-f** *device*] [**-S** *skip*] [**-E** *extarg*]

/usr/sbin/frecover -R *path* [**-f** *device*]

/usr/sbin/frecover -x [**-hmosvyAFNOX**] [**-c** *config*] [**-e** *path*] [**-f** *device*] [**-g** *graph*] [**-i** *path*] [**-S** *skip*] [**-E** *extarg*]

/usr/sbin/frecover -I *path* [**-vy**] [**-f** *device*] [**-c** *config*]

/usr/sbin/frecover -V *path* [**-vy**] [**-f** *device*] [**-c** *config*]

f

DESCRIPTION

frecover reads media written by the *fbackup*(1M) command. Its actions are controlled by the selected function **-r**, **-R**, **-x**, **-V**, or **-I**.

The function performed by **frecover** is specified by one of the following letters:

-r The backup media is read and the contents are loaded into the directories from which they were backed up. This option should only be used to recover a complete backup onto a clear directory or to recover an incremental backup after a full level-zero recovery (see *fbackup*(1M)). This is the default behavior.

-x The files identified by the **-i**, **-e**, and **-g** options (see below) are extracted or not extracted from the backup media. If a file to be extracted matches a directory whose contents have been written to the backup media, and the **-h** option is not specified, the directory is recursively extracted. The owner, modification time, and access control list (including optional entries, unless the **-A** option is specified) are recovered. If no file argument is given (including an empty graph file), all files on the backup media are extracted, unless the **-h** option is specified.

-I *path* The index on the current volume is extracted from the backup media and is written to *path*.

-V *path* The volume header on the current volume is extracted from the backup media and is written to *path*. The following fields from the header are extracted in the format *label*:*value* with one pair per line.

Magic Field	On a valid **fbackup** media it contains the value **FBACKUP_LABEL**. On a pre-10.20 **fbackup** media it contains **FBACKUP LABEL**.
Machine Identification	This field contains the result of **uname -m**.
System Identification	This field contains the result of **uname -s**.
Release Identification	This field contains the result of **uname -r**.
Node Identification	This field contains the result of **uname -n**.
User Identification	This field contains the result of *cuserid*(3S).
Record Size	This field contains the maximum length in bytes of a data record.
Time	This field contains the time **fbackup** was started.
Media Use	This field contains the number of times the media has been used for backup.
Volume Number	This field contains a **#** character followed by 3 digits, and identifies the current volume in the backup.
Checkpoint Frequency	This field contains the frequency of backup-data-record checkpointing.
Fast Search Mark Frequency	This field contains the number of files between fast search marks for backups made with DDS tape drives.
Index Size	This field contains the size of the index.

Backup Identification Tag
This field is composed of 2 items: the process ID (pid), and the start time of that process.

Language This field contains the language used to make the backup.

-R *path* An interrupted full recovery can be continued using this option. **frecover** uses the information in file *path* to continue the recovery from where it was interrupted. The only command line option used by **frecover** with this option is **-f**. The values in *path* override all other options to **frecover**. Note also that only full recoveries are restarted with this option, because no history of include or exclude lists is stored in the restart file. If a partial recovery (i.e., using the **-x** option) is interrupted then restarted with this option, **frecover** continues recovering where the partial recovery left off, but restores all files on the backup media beyond this point.

The following characters can be used in addition to the letter that selects the desired function:

-c *config* *config* specifies the name of a configuration file to be used to alter the behavior of **frecover**. The configuration file allows the user to specify the action to be taken on all errors, the maximum number of attempts at resynchronizing on media errors (**-S** option), and changing media volumes. Each entry of a configuration file consists of an action identifier followed by a separator followed by the specified action. Valid action identifiers are **error**, **chgvol**, and **sync**. Separators can be either tabs or spaces. In the following sample configuration file, each time an error is encountered, the script **/var/adm/fbackupfiles/frecovererror** is executed. Each time the backup media is to be changed, the script **/var/adm/fbackupfiles/frecoverchgvol** is executed. The maximum number of resynchronization attempts is five.

```
error  /var/adm/fbackupfiles/frecovererror
chgvol /var/adm/fbackupfiles/frecoverchgvol
sync 5
```

-e *path* *path* is interpreted as a graph to be excluded from the recovery. There is no limit on how many times the **-e** option can be specified.

-f *device* *device* identifies the backup device to be used instead of the default **/dev/rmt/0m**. If *device* is **-**, **frecover** reads from standard input. Thus *fbackup*(1M) and **frecover** can be used in a pipeline to backup and recover a file system as follows:

```
fbackup -i /usr -f - | (cd /mnt; frecover -Xrf -)
```

If more than one output file is specified, **frecover** uses each one successively and then repeats in a cyclical pattern. Patterns can be used in the device name in a way similar to file name expansion as done by *sh*(1). The expansion of the pattern results in all matching names being in the list of devices used. A device on the remote machine can be specified in the form *machine*:*device*. **frecover** creates a server process, **/usr/sbin/rmt**, on the remote machine to access the tape device. If **/usr/sbin/rmt** does not exist on the remote system, **frecover** creates a server process from **/etc/rmt** on the remote machine to access the tape device. The pattern matching capability does not apply to remote devices. Only half-inch 9-track magnetic tapes or DDS-format tapes can be remote devices. The fast search capability is not used when accessing remote DDS-format devices.

-g *graph* *graph* defines a graph file. Graph files are text files and contain the list of file names (graphs) to be recovered or skipped. Files are recovered using the **-i** option; thus if the user wants to recover all of **/usr**, the graph file contains one record:

```
i /usr
```

It is also possible to skip files by using the **-e** option. For instance, if a user wants to recover all of **/usr** except for the subgraph **/usr/lib**, the graph file contains two records:

```
i /usr
e /usr/lib
```

If the graph file is missing, **frecover** exits with an error message. An empty graph file results in recovering all files on the media.

-h Extract the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the backup media.

-i *path* *path* is interpreted as a graph to be included in the recovery. There is no limit on how many times the **-i** option can be specified.

-m Print a message each time a file marker is encountered. Using this option, **frecover** prints a message each time either a DDS setmark, a file marker, or a checkpoint record is read. Although useful primarily for troubleshooting, these messages can also be used to reassure the user that the backup is progressing during long, and otherwise silent, periods during the recovery.

-o Recover the file from the backup media irrespective of age. Normally **frecover** does not overwrite an existing file with an older version of the file.

-s Attempt to optimize disk usage by not writing null blocks of data to sparse files.

-v Normally **frecover** works silently. The **-v** (verbose) option causes it to display the file type and name of each file it treats.

-y Automatically answer **yes** to any inquiries.

-A Do not recover any optional entries in access control lists (ACLs). Normally, all access control information, including optional ACL entries, is recovered. This option drops any optional entries and sets the permissions of the recovered file to the permissions of the backed up file. Use this option when recovering files backed up from a system with ACLs on a system for which ACLs are not desired (see *acl*(5)).

-F Recover files without recovering leading directories. For example, this option would be used if a user wants to recover **/usr/bin/vi**, **/usr/bin/sh**, and **/etc/passwd** to a local directory without creating each of the graph structures.

-E *extarg* Specifies the handling of any extent attributes backed up by *fbackup*(1M). The **-E** option takes the following keywords as arguments:

- **warn** Issues a warning message if extent attributes cannot be restored, but restore the file anyway.
- **ignore** Do not restore extent attributes.
- **force** Issue an error message and do not restore the file if extent attributes cannot be restored.

  Extent attributes cannot be restored if the files are being restored to a file system which does not support extent attributes or if the file system's block size is incompatible with the extent attributes. If **-E** is not specified, *extarg* defaults to **warn**.

-N (no recovery) Prevent **frecover** from actually recovering any files onto disk, but read the backup as if it was, in fact, recovering the data from the backup, producing the same output that it would on a normal recovery. This option is useful for verifying backup media contents in terms of validity (block checksum errors are reported), and contents (a listing of files can be produced by using the **-N** and **-v** options together). Note that the listing of files produced with the **-N** and **-v** options requires the reading of the entire backup, but is therefore a more accurate reflection of the backup's contents than the index stored at the beginning of the backup (which was created at the start of the backup session, and is not changed during the course of the backup).

-O Use the effective uid and gid for the owner and group of the recovered file instead of the values on the backup media.

-S *skip* **frecover** does not ask whether it should abort the recovery if it gets a media error. It tries to skip the bad block or blocks and continue. Residual or lost data is written to the file named by *skip*. The user can then edit this file and recover otherwise irretrievable data.

-X Recover files relative to the current working directory. Normally **frecover** recovers files to their absolute path name.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which **frecover** expects files to be stored in the backup device and the order in which file names are output by the **-I** option.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_COLLATE** and **LC_MESSAGES** are not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **frecover** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

For incremental backups created prior to installing HP-UX Release 8.0, or for recoveries that do not begin with the first volume (such as when, reading tape 3 first), it is possible for the preceding directories to a recoverable file to not be on the media. This can happen, for example, if the directories did not change since the last full backup. If **frecover** encounters a file on the backup that should be recovered, but it has not recovered the file's parent directories from the backup, it prints a message stating that the recovery will continue with that file, and attempts to create the file's parent directories as needed.

Use of **frecover** does not require special privileges. However, if a user does not have access permission to a given file, the file is not recovered.

Network special files are obsolete. Therefore, **frecover** cannot restore these files. A warning message is issued if an attempt is made to recover a network special file, and the file is skipped.

When using a DDS tape written with the current release of **fbackup** to do a partial recovery, **frecover** attempts to use the DDS fast-search capability to find files on the tape more quickly. In order to do this, however, **frecover** needs to create an in-memory copy of the index, and mark the files on that index which it needs to recover before actually reading through the tape to find the files. This is done when the first index is read from the tape, and accounts for a period of time just after recovery is begun where the tape is inactive while this in-memory index is constructed. The larger the index is, the longer this period lasts.

The utility set comprised of *fbackup* and **frecover** was originally designed for use on systems equipped with not more than one gigabyte of total file system storage. Although the utilities have no programming limitations that restrict users to this size, complete backups and recoveries of substantially larger systems can cause a large amount system activity due to the amount of virtual memory (swap space) used to store the indices. Users who want to use these utilities, but are noticing poor system-wide performance due to the size of the backup, are encouraged to back up their systems in multiple smaller sessions, rather than attempting to back up the entire system at one time.

Note that when recovering files with access-control lists, the ACL entries are stored on the backup as user login names. If a login name cannot be found in the password file, the file is recovered without its ACL, and an error is printed. In order to fully recover files backed up with ACLs, the password file (**/etc/passwd**) must be recovered before attempting to recover any desired ACLs.

Care should be taken to match the names specified by the include and exclude options with the names in the index on the tape. Since the files are stored on the backup in lexographic order as defined by the **LANG** or **LC_COLLATE** environment variable, **frecover** uses the exact path names to determine when a partial recovery is complete, and when an earlier tape needs to be loaded. If a user's specification of a file to be recovered is misspelled, this may cause confusing messages, such as **frecover** asking for the previous volume, when volume one is mounted.

DEPENDENCIES

SS Series 700/800 **frecover** is not supported on QIC devices with QIC-120, and QIC-150 formats. If **frecover** is attempted for these formats, **frecover** fails and the following message is displayed :

```
mt lu X:Read must be a multiple of 512 bytes in QIC 120 and QIC 150
```

AUTHOR

frecover was developed by HP.

FILES

`/dev/rmt/0m`	Default backup device.

SEE ALSO

cpio(1M), dump(1M), fbackup(1M), restore(1M), rmt(1M), tcio(1M), acl(5).

NAME

freedisk - recover disk space

SYNOPSIS

`freedisk` [**`-a`** *n*] [**`-v`**]

DESCRIPTION

The **`freedisk`** command is an interactive script that finds and optionally removes filesets that do not appear to have been used since they were originally installed by **`swinstall`** (see *swinstall*(1M)). NOTE: Familiarity with **`swremove`** (see *swremove*(1M)) is required for successful use of this tool.

The **`freedisk`** command has two phases, any combination of which can be executed or skipped.

The first phase analyzes the regular files in all filesets to discover filesets that have remained unused since installation. Use the **`-a`** option to specify a usage time other than "since installation."

Filesets that appear to be entirely unused, but which are dependencies of other filesets that are in use, are treated by **`freedisk`** as though they were "in use" and are not presented as candidates for removal.

At the end of the first phase, the **`swremove`** command is invoked interactively with the filesets that are candidates for removal already selected. During the **`swremove`** session any, all, or none of the pre-selected filesets can be removed.

The second phase of **`freedisk`** optionally removes filesets that are used only for building kernels. These filesets are identified by containing a control file named **`freedisk_rmvbl`**. This removal occurs regardless of when the filesets were last used. This phase should be executed only if you are sure you will not need to rebuild a kernel for any reason. The interactive interface provides more information on this capability.

You can reload kernel build filesets removed during this phase by using **`/var/adm/sw/krn_rmvd.log`** as the argument to the **`-f`** option of **`swinstall`**.

Options

`freedisk` supports the following options:

`-a` *n* Check access of files only in the previous *n* days instead of the default of checking access since the fileset installation date. The *n* value should be a positive integer. It is passed to **`find`** (see *find(1))* as **`-atime -n`**.

`-v` Provide very verbose output. Useful when detailed information is required as to which specific files have been used in each fileset.

If you prefer to track the operation of the utility in a scrollable and easily viewable form, redirect the output to a file (see the example below) and use an editor on that file.

RETURN VALUE

The following are exit values of **`freedisk`**:

`0` Successful completion.
`1` One or more critical errors occurred.

DIAGNOSTICS

Error messages are self-explanatory.

EXAMPLES

Use the verbose option of **`freedisk`** to identify individual files used in each fileset and keep a copy of the output in a file for later use:

```
/opt/contrib/bin/freedisk -v 2>&1 | tee filename
```

Find filesets that have not been used in the past 90 days:

```
/opt/contrib/bin/freedisk -a 90
```

WARNINGS

Removing the kernel build filesets in phase two can result in unresolved fileset dependencies. This means that **`swverify`** (see *swverify*(1M)) will indicate errors, unless the appropriate options are used to ignore missing dependencies.

Be careful when using the **-a** *n* option. Small values of *n* might cause infrequently used filesets to be discovered as unused.

AUTHOR

freedisk was developed by the Hewlett-Packard Company.

FILES

/var/adm/sw/krn_rmvd.log	log of removed kernel-build filesets
/var/adm/sw/swremove.log	log of **swremove** actions
/var/adm/sw/swagent.log	log of **swagent** actions

SEE ALSO

find(1), swinstall(1M), swmodify(1M), swremove(1M), swverify(1M), and the manual *Managing HP-UX Software with SD-UX*.

f

NAME

fsadm (generic) - a file system administration command

SYNOPSIS

/usr/sbin/fsadm [**-F** *FStype*] [**-V**] [**-o** *specific_options*] *special*

DESCRIPTION

The **fsadm** command is designed to perform selected administration tasks on file systems. These tasks may differ between file system types. *special* is a device file containing an unmounted file system. However, if the file system is of the type that provides online administration capabilities the *special* could be a *directory*. *directory* must be the root of a mounted file system.

Only a superuser can invoke **fsadm**.

Options

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching each *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-o *specific_options*
Specify options specific to each file system type. *specific_options* is a list of comma separated suboptions and/or keyword/attribute pairs intended for a specific *FStype*-specific module of the command. See the file system specific manual entries for a description of the *specific_options* supported, if any.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Convert a **HFS** file system from a **nolargefiles** file system to a **largefiles** file system:

```
fsadm -F hfs -o largefiles /dev/vg02/lvol1
```

Display **HFS** relevant file system statistics:

```
fsadm -F hfs /dev/vg02/lvol1
```

FILES

/etc/fstab Static information about the systems

SEE ALSO

fsadm_*FStype*(1M), fsck(1M), fstab(4), fs_wrapper(5).

NAME

fsadm (hfs) - an HFS file system administration command

SYNOPSIS

/usr/sbin/fsadm [**-F hfs**] [**-V**] [**-o** *specific_options*] *special*

DESCRIPTION

The **fsadm** command is designed to perform selected administration tasks on a HFS file systems. *special* is a device file containing an unmounted file system.

Only a superuser can invoke **fsadm**.

Options

-F *hfs* Specify the HFS file system type.

-o *specific_options*
Specify a list of comma separated suboptions and/or keyword/attribute pairs from the list below. The following *specific_options* are valid on HFS file systems.

largefiles Converts a **nolargefiles** file system to a **largefiles** file system. The file system should be unmounted and must be in a clean state (see *fsck*(1M)). A **largefiles** file system supports file sizes greater than 2 gigabytes.

nolargefiles
Converts a **largefiles** file system to a **nolargefiles** file system. The file system should be umounted and must be in a clean state (see *fsck*(1M)). All **largefiles** should be purged from the file system for the conversion to succeed.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

DIAGNOSTICS

Error and warning messages may originate from **fsadm** and **fsck**. See *fsadm*(1M) or *fsck*(1M) to interpret the error and warning messages.

EXAMPLES

Convert a **nolargefiles** HFS file system to a **largefiles** HFS file system:

```
fsadm -F hfs -o largefiles /dev/vg02/rlvol1
```

Convert a **largefiles** HFS file system to a **nolargefiles** file system:

```
fsadm -F hfs -o nolargefiles /dev/vg02/rlvol1
```

Display relevant HFS file system statistics:

```
fsadm -F hfs /dev/vg02/rlvol1
```

WARNINGS

The size of a file system will impact the performance of the **fsadm** command.

During conversion from largefiles file system to a nolargefiles file system **fsadm** scans the entire file system for a large file. This functionality degrades the performance of the **fsadm** command.

FILES

/etc/fstab	Static information about the systems

SEE ALSO

fsadm(1M), fsck(1M), fstab(4), fs_wrapper(5).

NAME

fsadm (vxfs) - resize or reorganize a VxFS file system

SYNOPSIS

/usr/sbin/fsadm [**-F vxfs**] [**-V**] [**-D**] *mount_point*

/usr/sbin/fsadm [**-F vxfs**] [**-V**] [**-E**] *mount_point*

/usr/sbin/fsadm [**-F vxfs**] [**-V**] [**-e**] [**-E**] [**-s**] [**-v**] [**-l** *largesize*]
[**-a** *days*] [**-t** *time*] [**-p** *passes*] [**-r** *rawdev*] *mount_point*

/usr/sbin/fsadm [**-F vxfs**] [**-V**] [**-d**] [**-D**] [**-s**] [**-v**] [**-a** *days*] [**-t** *time*]
[**-p** *passes*] [**-r** *rawdev*] *mount_point*

/usr/sbin/fsadm [**-F vxfs**] [**-V**] [**-b** *newsize*] [**-r** *rawdev*] *mount_point*

/usr/sbin/fsadm [**-F vxfs**] [**-V**] [**-o largefiles**|**nolargefiles**] *mount_point*|*special*

f

DESCRIPTION

fsadm is designed to perform various online administration functions on VxFS file systems. The current version supports querying or changing the file-system compatibility bits, file-system resizing, extent reorganization, and directory reorganization. **fsadm** operates on file systems mounted for read/write access. The **-o** option can also operate on a device containing a clean, unmounted file system. Only a privileged user can change compatibility bits on a mounted file system or resize or reorganize a file system.

Options

-F vxfs	Specify the VxFS file system type.
-V	Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options. This option allows the user to verify the command line.
-b *newsize*	Resize the file system to *newsize* sectors.
-D	Report on directory fragmentation. If specified in conjunction with the **-d** option, the fragmentation report is produced both before and after the directory reorganization.
-E	Report on extent fragmentation. If specified in conjunction with the **-e** option, the fragmentation report is produced both before and after the extent reorganization.
-d	Reorganize directories. Directory entries are reordered to place subdirectory entries first, then all other entries in decreasing order of time of last access. The directory is also compacted to remove free space.
-e	Extent reorganization. Attempt to minimize fragmentation. Aged files are moved to the end of the allocation units to produce free space. Other files are reorganized to have the minimum number of extents possible.
-s	Print a summary of activity at the end of each pass.
-v	Verbose. Report reorganization activity.
-a *days*	Consider files not accessed within the specified number of *days* as aged files. The default is 14 days. Aged files are moved to the end of the directory by the **-d** option, and reorganized differently by the **-e** option.
-l *largesize*	Large file size in file system blocks. Indicates the size of files to be considered as large files. The value must be between 8 and 2048 blocks. The default is 64 blocks.
-p *passes*	Maximum number of *passes* to run. The default is 5 passes. Reorganizations are processed until reorganization is complete, or the specified number of *passes* have been run.
-r *rawdev*	Pathname of raw device to read to determine file layout and fragmentation. This option can be used when **fsadm** cannot determine what the raw device should be.
-t *time*	Maximum time to run. Reorganizations are processed until reorganization is complete, or the time limit has expired. *time* is specified in seconds.
-o *specific_options*	Specifies options specific to the vxfs file system type. *specific_options* is a list of suboptions pairs intended for the vxfs-specific module of the command.

The following *specific_options* are valid on a VxFS file system:

`largefiles`
: Set the *largefile compatibility bit* for the file system. When this bit is set large files (greater than 2 Gbyte) can be created on the file system.

`nolargefiles`
: Clear the *largefile compatibility bit* for the file system. When this bit is not set, large files cannot be created on the file system. An attempt to clear the bit will fail if any large files exist on the file system.

The **`-o largefiles`**, **`-o nolargefiles`**, **`-b`**, **`-D`**, **`-E`**, **`-d`**, and **`-e`** options determine what function will be performed. If none of these options is specified **`fsadm`** will print the current compatibility-bit settings and exit. Otherwise it will perform the function(s) defined by the option(s). The **`-b`**, **`-o largefiles`**, and **`-o nolargefiles`** options cannot be specified if any other of these options are given. If both **`-e`** and **`-d`** are specified, **`fsadm`** will perform the directory reorganization first. It will perform the extent reorganization after the directory reorganization has been completed.

File-System Compatibility Bits

The **`-o largefiles`** and **`-o nolargefiles`** options can be used to change the *largefile compatibility bit*. When invoked without options **`fsadm`** prints the current state of the compatibility bits.

VxFS 3.0 has some new features that are incompatible with earlier versions of HP-UX and with old applications. These features are large files (file sizes greater than 2 Gbyte), and hierarchical storage management via the DMAPI (Data Management Applications Programming Interface).

Large files are available only with the Version 3 disk layout, available in VxFS 3.0 and above, so an old version of HP-UX will never be exposed to them (the file-system **`mount`** would fail). But many existing applications will break if confronted with large files, so a compatibility bit is provided that allows or prevents the creation of large files on the file system. If the *largefile compatibility bit* is set, large files may be created on the file system. If it is not set, any attempt to create a large file on the file system will fail.

An attempt to set the bit via the **`-o largefiles`** option will succeed only if the file system has the Version 3 disk layout (see the *vxupgrade*(1M) manual page to upgrade a file system from the Version 2 disk layout to the Version 3 disk layout). An attempt to clear the bit via the **`-o nolargefiles`** option will succeed only if the bit is set and there are no large files present on the file system. (Also see the *mount_vxfs*(1M) manual page).

The **`-o largefiles`** and **`-o nolargefiles`** options are the only **`fsadm`** options that can be used on an unmounted file system. An unmounted file system can be specified by invoking **`fsadm`** with a special device rather than a mount point. If an unmounted file system is specified, it must be clean.

The *DMAPI compatibility bit* cannot be changed by **`fsadm`**; it can only be queried. If set, it indicates that the file system is mounted, or has been mounted, under the control of the Hierarchical Storage Management software and cannot be mounted unless that software is active on the system.

Defragmentation

For optimal performance, the kernel-extent allocator must be able to find large extents when it wants them. To maintain file-system performance, **`fsadm`** should be run periodically against all VxFS file systems to reduce fragmentation. **`fsadm`** should be run somewhere between once a day and once a month against each file system. The frequency depends on file system usage and activity patterns, and the importance of performance. The **`-v`** option can be used to examine the amount of work performed by **`fsadm`**. The frequency of reorganization can be adjusted based on the rate of file system fragmentation.

There are two options that are available to control the amount of work done by **`fsadm`**. The **`-t`** option is used to specify a maximum length of time to run. The **`-p`** option is used to specify a maximum number of passes to run. If both are specified, the utility exits if either of the terminating conditions is reached. By default, **`fsadm`** will run 5 passes. If both the **`-e`** and **`-d`** options are specified, the utility will run all the directory reorganization passes before any extent reorganization passes.

`fsadm` uses the file **`.fsadm`** in the **`lost+found`** directory as a lock file. When **`fsadm`** is invoked, it opens the file **`lost+found/.fsadm`** in the root of the file system specified by *mount_point*. If the file does not exist, it is created. The *fcntl*(2) system call is used to obtain a write lock on the file. If the write lock fails, **`fsadm`** will assume that another **`fsadm`** is running and will fail. **`fsadm`** will report the process ID of the process holding the write lock on the **`.fsadm`** file.

File System Resizing

If the **-b** option is specified, **fsadm** will resize the file system whose mount point is *mount_point*. If *newsize* is larger than the current size of the file system, the file system will be expanded to *newsize* sectors. Similarly, if *newsize* is smaller than the current size of the file system, an attempt will be made to shrink the file system to *newsize* sectors.

Reducing the size of a file system will fail if there are file-system resources currently in use within the sectors to be removed from the file system. In this case, a reorganization may help free those busy resources and allow a subsequent reduction in the size of the file system.

Reporting on Directory Fragmentation

As files are allocated and freed, directories tend to grow and become sparse. In general, a directory is as large as the largest number of files it ever contained, even if some files have been subsequently removed.

The command line to obtain a directory fragmentation report is:

fsadm [**-D**] [**-r** *rawdev*] *mount_point*

The following is some example output from the **fsadm -D** command:

```
# fsadm -D /lhome

Directory Fragmentation Report

             Dirs        Total      Immed    Immeds   Dirs to   Blocks to
             Searched    Blocks     Dirs     to Add   Reduce    Reduce
au    0            15          3        12        0         0           0
au    1             0          0         0        0         0           0
total              15          3        12        0         0           0
```

The column labeled "Dirs Searched" contains the total number of directories. A directory is associated with the extent-allocation unit containing the extent in which the directory's inode is located. The column labeled "Total Blocks" contains the total number of blocks used by directory extents.

The column labeled "Immed Dirs" contains the number of directories that are immediate, meaning that the directory data is in the inode itself, as opposed to being in an extent. Immediate directories save space and speed up pathname resolution. The column labeled "Immeds to Add" contains the number of directories that currently have a data extent, but that could be reduced in size and contained entirely in the inode.

The column labeled "Dirs to Reduce" contains the number of directories for which one or more blocks could be freed if the entries in the directory are compressed to make the free space in the directory contiguous. Since directory entries vary in length, it is possible that some large directories may contain a block or more of total free space, but with the entries arranged in such a way that the space cannot be made contiguous. As a result, it is possible to have a nonzero "Dirs to Reduce" calculation immediately after running a directory reorganization. The **-v** (verbose) option of directory reorganization reports occurrences of failure to compress free space.

The column labeled "Blocks to Reduce" contains the number of blocks that could be freed if the entries in the directory are compressed.

Measuring Directory Fragmentation

If the totals in the columns labeled "Dirs to Reduce" are substantial, a directory reorganization should improve performance of pathname resolution. The directories that fragment tend to be the directories with the most activity. A small number of fragmented directories may account for a large percentage of name lookups in the file system.

Directory Reorganization

If the **-d** option is specified, **fsadm** will reorganize the directories on the file system whose mount point is *mount_point*. Directories are reorganized in two ways: compression and sorting.

For compression, the valid entries in the directory are moved to the front of the directory and the free space is grouped at the end of the directory. If there are no entries in the last block of the directory, the block is released and the directory size is reduced.

If the directory entries are small enough, the directory will be placed in the inode immediate data area.

The entries in a directory are also sorted to improve pathname lookup performance. Entries are sorted based on the last access time of the entry. The **-a** option is used to specify a time interval; 14 days is the default if **-a** is not specified. The time interval is broken up into 128 buckets, and all times within the same bucket are considered equal. All access times older than the time interval are considered equal, and those entries are placed last. Subdirectory entries are placed at the front of the directory and symbolic links are placed after subdirectories, followed by the most-recently-accessed files.

The directory reorganization runs in one pass across the entire file system.

The command line to reorganize directories of a file system is:

fsadm -d [**-s**] [**-v**] [**-p** *passes*] [**-t** *timeout*] [**-r** *rawdev*] [**-D**] *mount_point*

The following example illustrates the output of the command **fsadm -d -s** command:

```
# fsadm -d -s /lhome

Directory Reorganization Statistics

              Dirs        Dirs        Total     Failed    Blocks    Blocks    Immeds
              Searched    Changed     Ioctls    Ioctls    Reduced   Changed   Added
au     0        2343        1376       2927          1       209      3120       72
au     1         582         254        510          0        47       586       28
au     2         142          26         38          0        21        54       16
au     3          88          24         29          1         5        36        2
total           3155        1680       3504          2       282      3796      118
```

The column labeled "Dirs Searched" contains the number of directories searched. Only directories with data extents are reorganized. Immediate directories are skipped. The column labeled "Dirs Changed" contains the number of directories for which a change was made.

The column labeled "Total Ioctls" contains the total number of VX_DIRSORT ioctls performed. Reorganization of directory extents is performed using this ioctl.

The column labeled "Failed Ioctls" contains the number of requests that failed for some reason. The reason for failure is usually that the directory being reorganized is active. A few failures should be no cause for alarm. If the **-v** option is used, all ioctl calls and status returns are recorded.

The column labeled "Blocks Reduced" contains the total number of directory blocks freed by compressing entries. The column labeled "Blocks Changed" contains the total number of directory blocks updated while sorting and compressing entries.

The column labeled "Immeds Added" contains the total number of directories with data extents that were compressed into immediate directories.

Reporting on Extent Fragmentation

As files are created and removed over time, the free extent map for an allocation unit will change from having one large free area to having many smaller free areas. This process is known as fragmentation. Also, when files are grown — particularly when growth occurs in small increments — small files could be allocated in multiple extents. In the ideal case, each file that is not sparse would have exactly one extent (containing the entire file), and the free-extent map would be one continuous range of free blocks.

Conversely, in a case of extreme fragmentation, there can be free space in the file system, none of which can be allocated. For example, on Version 2 VxFS file systems, the indirect-address extent size is always 8K long. This means that to allocate an indirect-address extent to a file, an 8K extent must be available. For example, if no extent of 8K byes or larger is available, even though more than 8K of free space is available, an attempt to allocate a file into indirect extents will fail and return ENOSPC.

Determining Fragmentation

To determine whether fragmentation exists for a given file system, the free extents for that file system need to be examined. If a large number small extents are free, then there is fragmentation. If more than half of the amount of free space is taken up by small extents (smaller than 64 blocks), or there is less than 5 percent of total file system space available in large extents, then there is serious fragmentation.

Running the Extent-Fragmentation Report

The extent-fragmentation report can be run to acquire detailed information about the degree of fragmentation in a given file system.

The command line to run an extent-fragmentation report is

fsadm -E [**-l** *largesize*] [**-r** *rawdev*] *mount_point*

The extent reorganizer has the concept of an immovable extent: if the file already contains large extents, reallocating and consolidating these extents will not improve performance, so they are considered immovable. **fsadm**'s notion of how large an extent must be to qualify as immovable can be controlled by the **-l** option. By default, *largesize* is 64 blocks, meaning that any extent larger than 64 blocks is considered to be immovable. For the purposes of the extent-fragmentation report, the value chosen for *largesize* will affect which extents are reported as being immovable extents.

The following is an example of the output generated by the **fsadm -E** command:

```
# fsadm -E /lhome

Extent Fragmentation Report
          Files with          Total              Total
          Extents          Extents    Blocks        Distance
 au   0     14381           18607      30516        4440997
 au   1      2822            3304      24562         927841
 au   2      2247            2884      22023        1382962
 au   3       605             780      24039         679867
 total      19992           25575      101140       7431667
            Consolidatable              Immovable
         Extents      Blocks         Extents     Blocks

 au   0            928        2539               0          0
 au   1            461        5225              99      13100
 au   2            729        8781              58      11058
 au   3            139        1463              49      17258
 total            2257       18008             206      41416
          Free Extents By Size
au   0  Free Blocks 217, Smaller Than 8 - 48%, Smaller Than 64 - 100%
     1:        15      2:         15      4:        15      8:        14
    16:         0     32:          0     64:         0    128:         0
   256:         0    512:          0   1024:         0   2048:         0
   096:         0   8192:          0  16384:         0
au   1  Free Blocks 286, Smaller Than 8 - 41%, Smaller Than 64 - 100%
     1:        16      2:         21      4:        15      8:        13
    16:         4     32:          0     64:         0    128:         0
   256:         0    512:          0   1024:         0   2048:         0
  4096:         0   8192:          0  16384:         0
au    2 Free Blocks 510, Smaller Than 8 - 15%, Smaller Than 64 - 100%
     1:        10      2:         14      4:        10      8:        14
    16:         8     32:          6     64:         0    128:         0
   256:         0    512:          0   1024:         0   2048:         0
  4096:         0   8192:          0  16384:         0
au    3 Free Blocks 6235,  Smaller Than 8 - 3%,  Smaller Than 64 - 15%
     1:        29      2:         33      4:        27      8:        30
    16:        18     32:          8     64:         4    128:         3
   256:         2    512:          2   1024:         1   2048:         1
  4096:         0   8192:          0  16384:         0
au    4 Free Blocks 8551,  Smaller Than 8 - 2%,  Smaller Than 64 - 22%
     1:        29      2:         33      4:        30      8:        38
    16:        28     32:         29     64:        26    128:        11
   256:         8    512:          3   1024:         0   2048:         0
  4096:         0   8192:          0  16384:         0
total   Free Blocks 15799, Smaller Than 8 - 4%,  Smaller Than 64 - 24%
     1:        99      2:        116      4:        97      8:       109
    16:        58     32:         43     64:        30    128:        14
   256:        10    512:          5   1024:         1   2048:          1
  4096:         0   8192:          0  16384:         0
```

The numbers in the column labeled "Files with Extents" indicate the total number of files that have data extents. A file is considered to be in the extent-allocation unit that contains the extent holding the file's inode.

The column labeled "Total Extents" contains the total number of extents belonging to files in the allocation unit. The extents themselves are not necessarily in the same allocation unit.

The column labeled "Total Blocks" contains the total number of blocks used by files in the allocation unit. If the total number of blocks is divided by the total number of extents, the resulting figure is the average extent size.

The column labeled "Total Distance" contains the total distance between extents in the allocation unit. For example, if a file has two extents, the first containing blocks 100 through 107 and the second containing blocks 110 through 120, the distance between the extents is 110-107, or 3. In general, a lower number means that files are more contiguous. If an extent reorganization is run on a fragmented file system, the value for Total Distance should be reduced.

The column labeled "Consolidatable Extents" contains the number of extents that are candidates to be consolidated. *Consolidation* means merging two or more extents into one combined extent. For files that are entirely in direct extents, the extent reorganizer will attempt to consolidate extents into extents up to size *largesize*. All files of size *largesize* or less will typically be contiguous in one extent after reorganization. Since most files are small, this will usually include about 98 percent of all files.

The column labeled "Consolidatable Blocks" contains the total number of blocks in Consolidatable Extents. The column labeled "Immovable Extents" contains the total number of extents that are considered to be immovable. In the report, an immovable extent appears in the allocation unit of the extent itself, as opposed to in the allocation unit of its inode. This is because the extent is considered to be immovable, and thus permanently fixed in the associated allocation unit.

The column labeled "Immovable Blocks" contains the total number of blocks in immovable extents. The figures under the heading "Free Extents By Size" indicate per-allocation unit totals for free extents of each size. The totals are for free extents of size 1, 2, 4, 8, 16, ... up to a maximum of the number of data blocks in an allocation unit. The totals should match the output of **df -o s** unless there has been recent allocation or deallocation activity (as this utility acts on mounted file systems). These figures give an indication of fragmentation and extent availability on a per-allocation-unit basis. For each allocation unit, and for the complete file system, the total free blocks and total free blocks by category are shown. The figures are presented as follows:

- The figure labeled "Free Blocks" indicates the total number of free blocks.
- The figure labeled "Smaller Than 8" indicates the percentage of free blocks that are in extents less than 8 blocks in length.
- The figure labeled "Smaller Than 64" indicates the percentage of free blocks that are in extents less than 64 blocks in length.

In the preceding example, 4 percent of free space is in extents less than 8 blocks in length, and 24 percent of the free space is in extents less than 64 blocks in length. This represents a typical value for a mature file system that is regularly reorganized. The total free space is about 10 percent.

Extent Reorganization

If the **-e** option is specified, **fsadm** will reorganize the data extents on the file system whose mount point is *mount_point*. The primary goal of extent reorganization is to defragment the file system.

To reduce fragmentation, extent reorganization tries to place all small files in one contiguous extent. The **-l** option is used to specify the size of a file that is considered large. The default is 64 blocks. Extent reorganization also tries to group large files into large extents of at least 64 blocks. In addition to reducing fragmentation, extent reorganizations improves performance. Small files can be read or written in one I/O operation. Large files can approach raw-disk performance for sequential I/O operations.

Extent reorganization also tries to improve the locality of reference on the file system. Extents are moved into the same allocation unit as their inode. Within the allocation unit, small files and directories are migrated to the front of the allocation unit. Large files and inactive files are migrated towards the back of the allocation unit. (A file is considered inactive if the access time on the inode is more than 14 days old. The time interval can be varied using the **-a** option.) Extent reorganization should reduce the average seek time by placing inodes and frequently used data closer together.

fsadm will try to perform extent reorganization on all inodes on the file system. Each pass through the inodes will move the file system closer to the organization considered optimal by **fsadm**. The first pass might place a file into one contiguous extent. The second pass might move the file into the same allocation unit as its inode. Then, since the first file has been moved, a third pass might move extents for a file in another allocation unit into the space vacated by the first file during the second pass.

When the file system is more than 90% full, **fsadm** shifts to a different reorganization scheme. Instead of attempting to make files contiguous, extent reorganization tries to defragment the free-extent map into chunks of at least 64 blocks or the size specified by the *-l* option.

The command line to perform extent reorganization is

fsadm -F vxfs -e [**-sv**] [**-p** *passes*] [**-t** *time*] [**-a** *days*] [**-l** *largesize*] [**-r** *rawdev*] *mount_point*

The following example illustrates the output from the **fsadm -F vxfs -e -s** command:

```
# fsadm -F vxfs -e -s

Allocation Unit 0, Pass 1 Statistics
           Extents    Consolidations Performed     Total Errors
          Searched   Number   Extents   Blocks File Busy   Not Free
 au    0      2467       11        30      310         0          0
 au    1         0        0         0        0         0          0
 au    2         0        0         0        0         0          0
 au    3         0        0         0        0         0          0
 au    4         0        0         0        0         0          0
 total        2467       11        30      310         0          0

              In Proper Location  Moved to Proper Location
              Extents     Blocks    Extents     Blocks
 au    0         1379       8484        794      10925
 au    1            0          0          0          0
 au    2            0          0          0          0
 au    3            0          0          0          0
 au    4            0          0          0          0
 total           1379       8484        794      10925

         Moved to Free Area        In Free Area     Could not be Moved
          Extents     Blocks     Extents    Blocks    Extents     Blocks
 au    0      231       4851           4       133          0          0
 au    1        0          0           0         0          0          0
 au    2        0          0           0         0          0          0
 au    3        0          0           0         0          0          0
 au    4        0          0           0         0          0          0
 total        231       4851           4       133          0          0


Allocation Unit 0, Pass 2 Statistics
           Extents    Consolidations Performed     Total Errors
          Searched   Number   Extents   Blocks File Busy   Not Free
 au    0      2467        0         0        0         0          0
 au    1         0        0         0        0         0          0
 au    2         0        0         0        0         0          0
 au    3         0        0         0        0         0          0
 au    4         0        0         0        0         0          0
 total        2467        0         0        0         0          0

              In Proper Location  Moved to Proper Location
              Extents     Blocks    Extents     Blocks
 au    0         2173      19409        235       4984
 au    1            0          0          0          0
 au    2            0          0          0          0
 au    3            0          0          0          0
 au    4            0          0          0          0
 total           2173      19409        235       4984

         Moved to Free Area        In Free Area       Could not be Moved
          Extents    Blocks     Extents    Blocks     Extents      Blocks
 au    0         0         0           0         0           0           0
 au    1         0         0           0         0           0           0
 au    2         0         0           0         0           0           0
 au    3         0         0           0         0           0           0
 au    4         0         0           0         0           0           0
 total           0         0           0         0           0           0
```

Note that the default five passes were scheduled, but the reorganization finished in two passes.

This file system had not had much activity since the last reorganization, with the result that little reorganization was required. The time it takes to complete extent reorganization varies, depending on fragmentation and disk speeds. However, in general, extent reorganization may be expected to take approximately one minute for every 10 megabytes of disk space used.

In the preceding example, the column labeled "Extents Searched" contains the total number of extents examined. The column labeled "Number" (located under the heading "Consolidations Performed") contains the total number of consolidations or merging of extents performed. The column labeled "Extents" (located under the heading "Consolidations Performed") contains the total number of extents that were consolidated. (More than one extent may be consolidated in one operation.) The column labeled "Blocks" (located under the heading "Consolidations Performed") contains the total number of blocks that were consolidated.

The column labeled "File Busy" (located under the heading "Total Errors") contains the total number of reorganization requests that failed because the file was active during reorganization. The column labeled "Not Free" (located under the heading "Total Errors") contains the total number of reorganization requests that failed because an extent that the reorganizer expected to be free was allocated at some time during the reorganization.

The column labeled "In Proper Location" contains the total extents and blocks that were already in the proper location at the start of the pass. The column labeled "Moved to Proper Location" contains the total extents and blocks that were moved to the proper location during the pass.

The column labeled "Moved to Free Area" contains the total number of extents and blocks that were moved into a convenient free area in order to free up space designated as the proper location for an extent in the allocation unit being reorganized. The column labeled "In Free Area" contains the total number of extents and blocks that were in areas designated as free areas at the beginning of the pass.

The column labeled "Could not be Moved" contains the total number of extents and blocks that were in an undesirable location and could not be moved. This occurs when there is not enough free space to allow sufficient extent movement to take place. This often occurs on the first few passes for an allocation unit if a large amount of reorganization needs to be performed.

If the next to the last pass of the reorganization run indicates extents that cannot be moved, then the reorganization fails. A failed reorganization may leave the file system badly fragmented, since free areas are used when trying to free up reserved locations. To lessen this fragmentation, extents are not moved into the free areas on the final two passes of the extent reorganizer, and the last pass of the extent reorganizer only consolidates free space.

Notes

The online reorganization and online resize features of **fsadm** are available only with the Advanced VxFS package.

FILES

lost+found/.fsadm lock file **/dev/rdsk/**∗ file-system devices

SEE ALSO

mkfs_vxfs(1M), fcntl(2), vxfsio(7).

NAME

fscat (vxfs) - cat a VxFS file system

SYNOPSIS

/usr/sbin/fscat [**-F vxfs**] [**-V**] [**-o** *offset*] [**-l** *length*] [**-b** *block_size*] *special*

DESCRIPTION

The **fscat** utility provides an interface to a VxFS snapshot file system similar to that provided by the **dd** utility invoked on the special file of other VxFS file systems. On most VxFS file systems, the block or character special file for the file system provides access to a raw image of the file system for purposes such as backing up the file system to tape. On a snapshot file system, access to the corresponding block or character special provides little useful information. The **fscat** utility, however, provides a stream of bytes representing the file system snapshot. This stream can be processed several ways, such as being processed in a pipeline, being written to a tape, and so on. **fscat** will work when executed on the device special file of any VxFS file system.

By default, the output is a stream of bytes that starts at the beginning of the file system and continues to the last byte. On a snapshot file system, data is read from the file system using the VX_SNAPREAD ioctl on the mount point. On other VxFS file systems, data is read from the specified *special*. Data is written to standard output.

All numbers entered as option arguments may have **0** prepended to indicate octal, or **0x** prepended to indicate hexadecimal. A **b** may be appended to indicate the value is in 512-byte blocks, a **k** to indicate the value is in kilobytes, or an **m** to indicate the value is in megabytes. An appended letter may be separated from the number by a space, in which case the letter and number should be enclosed in a set of quotes (for example, " **512 b** ").

Options

-F vxfs	Specifies the VxFS file system type.
-V	Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options. This option allows the user to verify the command line.
-o *offset*	Specify the transfer start offset within the file system, in bytes.
-l *length*	Specify the transfer length, in bytes. A *length* of **0** includes the remainder of the file system after the specified offset.
-b *block_size*	Specify the output block size, in bytes. *block_size* must be less than or equal to 1 megabyte.

NOTES

fscat is only available with the VxFS Advanced package.

A snapshot file system cannot be written to. A snapshot file system exists only as long as it is mounted; once unmounted, the special file no longer contains a snapshot file system.

SEE ALSO

dd(1), vxfsio(7).

NAME

fsck (generic) - file system consistency check and interactive repair

SYNOPSIS

/usr/sbin/fsck [**-F** *FSType*] [**-m**] [**-V**] [*special* ...]

/usr/sbin/fsck [**-F** *FSType*] [**-o** *FSspecific-options*] [**-V**] [*special* ...]

DESCRIPTION

The **fsck** command audits and interactively repairs inconsistent conditions for HP-UX file systems on mass storage device files identified by *special*. If the file system is consistent, the number of files on that file system and the number of used and free blocks are reported. If the file system is inconsistent, **fsck** provides a mechanism to fix these inconsistencies, depending on which form of the **fsck** command is used.

special represents a special device (e.g., **/dev/rdsk/c1d0s8**).

Options

fsck recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-m Perform a sanity check only. **fsck** will return 0 if the file system is suitable for mounting. If the file system needs additional checking, the return code is 32. If the file system is mounted, the return code is 33. Error codes larger than 33 indicate that the file system is badly damaged.

-o *FSspecific-options*
Specify options specific to each file system type. *FSspecific-options* is a list of suboptions and/or keyword/attribute pairs intended for a file-system-specific version of the command. See the file-system-specific manual entries for a description of the *specific_options* supported, if any.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

RETURN VALUES

The following values are returned by the **-m** option to **fsck**:

0	Either no errors were detected or all errors were corrected.
32	The file system needs additional checking.
33	The file system is mounted.

Return values greater that **33** indicate that file system is badly corrupted. File system specific versions of **fsck** will have their own additional return values (see fsck_*FSType*(1M)).

WARNINGS

This command may not be supported for all file system types.

FILES

/etc/default/fs	Specifies the default file system type
/etc/fstab	Default list of file systems to check

STANDARDS CONFORMANCE

fsck: SVID3

SEE ALSO

fsck_*FSType*(1M), mkfs(1M), newfs(1M), fstab(4), fs_wrapper(5).

NAME

fsck (hfs) - HFS file system consistency check and interactive repair

SYNOPSIS

/usr/sbin/fsck [**-F hfs**] [**-m**] [**-V**] [**-b** *blocknum*] [*special* ...]

/usr/sbin/fsck [**-F hfs**] [**-c** *size*] [**-f**] [**-p**|**-P**] [**-V**] [*special* ...]

/usr/sbin/fsck [**-F hfs**] [**-b** *blocknum*] [**-c** *size*] [**-f**] [**-n**|**-N**|**-y**|**-Y**] [**-q**] [**-V**] [*special* ...]

DESCRIPTION

The **fsck** command audits and repairs inconsistent conditions for HFS file systems on mass storage device files identified by *special*. If the file system is consistent, the number of files on that file system and the number of used and free blocks are reported. If the file system is inconsistent, **fsck** provides a mechanism to fix these inconsistencies, depending on which form of the **fsck** command is used.

f

special represents a special device (e.g., **/dev/rdsk/c1d0s8**).

If the target device is a swap device, **fsck** does not continue to process. **fsck** also checks the target device to ensure a mounted file system is not being checked. If a mounted device is specified but the **-f** option is omitted, **fsck** prompts the user for a response.

If the **-p**|**-P** option is used and *special* is not specified, **fsck** reads the pass numbers in **/etc/fstab** to determine which groups of disks to inspect in parallel, taking maximum advantage of I/O overlap to process the file systems as quickly as possible. The **-p**|**-P** option is normally used in the script **/sbin/bcheckrc** during automatic reboot.

Normally, the root file system is checked on pass 1, and other "root" (section 0) file systems on pass 2. Other small file systems are checked on separate passes (such as the section 4 file systems on pass 3 and the section 7 file systems on pass 4), and finally the large user file systems are checked on the last pass (for example, pass 5). A pass number of 0 in **/etc/fstab** causes a file system not to be checked. If the optional fields are not present on a line in **/etc/fstab**, **fsck** processes the file system on such lines sequentially after all eligible file systems with positive pass numbers have been processed.

The inconsistencies that **fsck** with the **-p**|**-P** option corrects are shown below. These are inconsistencies that are correctable without data loss. If it encounters other inconsistencies, it exits with an abnormal return status. For each corrected inconsistency, one or more lines are printed identifying the file system on which the correction will take place and the nature of the correction. Correctable inconsistencies are limited to the following:

- Unreferenced inodes
- Unreferenced continuation inodes (see *inode*(4))
- Unreferenced pipes and FIFOs
- Link counts in inodes too large
- Missing blocks in the free list
- Blocks in the free list also in files
- Counts in the superblock wrong.

The **-P** option operates in the same manner as the **-p** option except that cleanly unmounted file systems are not checked (see *fsclean*(1M)). This can greatly decrease the amount of time required to reboot a system that was brought down cleanly.

If the **-p**|**-P** option is not specified, the pass numbers are ignored and the file systems are checked interactively in the order they are listed in **/etc/fstab**.

Without the **-p**|**-P** option, **fsck** prompts for concurrence before each correction is attempted when the file system is inconsistent. It should be noted that some corrective actions result in a loss of data. The amount and severity of data loss can be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond **yes** or **no**. If the operator does not have write permission, **fsck** defaults to a **-n** action.

Options

fsck recognizes the following options:

-F hfs Specify the HFS file system.

-c *size* Set the size of the buffer cache which **fsck** uses to cache disk blocks. *size* is the number of cache blocks, and is between 0 and 100 inclusive. The most common use of this option is

-c 0 to disable all caches, thus reducing memory usage.

-b *blocknum*
Use the specified *blocknum* as the superblock for the file system. An alternate superblock can usually be found at block **((SBSIZE+BBSIZE)/DEV_BSIZE)**, typically block 16. **DEV_BSIZE** is defined in **<sys/param.h>**. You can also find a list of alternate superblocks in **/var/adm/sbtab** (see *mkfs*(1M)).

-f Force **fsck** to check a mounted file system.

-m Perform a sanity check only. Verify whether *special* is mounted, or needs additional checking. Refer to the RETURN VALUE section for more information.

-n|**-N** Assume a **no** response to all questions asked by **fsck** about repairing a file system. Do not open the file system for writing.

-p "Preen" the file system. Proceed to process and repair file systems without user interaction, as described above. Exit immediately if there is a problem requiring intervention.

-P Same as **-p** except that cleanly unmounted file systems are not checked.

-q Quiet. Do not print size-check messages in Phase 1. Unreferenced fifos are silently removed. If **fsck** requires it, counts in the superblock and cylinder groups are automatically fixed.

-V Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-y|**-Y** Assume a **yes** response to all questions asked by **fsck** about repairing a file system. This should be used with great caution, because this is a free license to continue after essentially unlimited trouble has been encountered.

In all cases, **fsck** checks the following inconsistencies:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Size checks:
  - Directory size not of proper format.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks:
  - File pointing to unallocated inode.
  - Inode number out of range.
- Superblock checks:
  - More blocks for inodes than there are in the file system.
- Bad free block list format.
- Total free block and/or free inode count incorrect.
- Invalid continuation inode number in a primary inode.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. The only restriction is that the directory **lost+found** must have empty slots in which entries can be made. This is accomplished by copying a number of files to the directory, then removing them before **fsck** is executed.

Unreferenced continuation inodes are removed with the **-p** option, since they do not refer back to the primary inode. When a primary inode contains an invalid continuation inode number, the continuation inode number should be cleared (that is, set to 0). This is not done automatically (with the **-p** option), because access control list information may have been lost and should be corrected.

After **fsck** has checked and fixed the file system, it stores the correct **fs_clean** flag in the superblock if it is not already there. For a nonroot file system, **FS_CLEAN** is stored there. For the root file system, which is mounted at the time of the **fsck**, no changes are required to the superblock if no problems were found and **FS_OK** was already set.

Checking the raw device is almost always faster.

RETURN VALUE

fsck returns the following values:

0	Either no errors were detected or all errors were corrected.
1	A syntax error or other operational error occurred when invoked with the -V option.
4	Root file system errors were corrected. The system must be rebooted.
8	Some uncorrected errors exist on one or more of the file systems checked, there was a syntax error, or some other operational error occurred.
12	A signal was caught during processing.
32	The file system is unmounted and needs additional checking.
33	The file system is mounted.
34	The file system is damaged.

WARNINGS

fsck should not be run on mounted file systems or on the root device. If you do run on mounted file systems, be sure the system is in single-user state (see *shutdown*(1M)).

The special case of the **-c** option, **-c 0**, will disable all internal caches, which will reduce memory usage but may impact performance.

The **-F** option, from prior releases, has been replaced by the **-f** option.

AUTHOR

fsck was developed by HP, AT&T, the University of California, Berkeley.

FILES

/etc/fstab	Default list of file systems to check.
/var/adm/sbtab	List of locations of the superblocks for file systems. The **mkfs** command appends entries to this file.

STANDARDS CONFORMANCE

fsck: SVID3

SEE ALSO

fsck(1M), dumpfs(1M), fsclean(1M), mkfs(1M), newfs(1M), shutdown(1M), fstab(4), fs(4), inode(4), fs_wrapper(5), acl(5).

NAME

fsck (vxfs) - check and repair a VxFS file system

SYNOPSIS

/usr/sbin/fsck [**-F vxfs**] [**-V**] [**-pPmnNyY**] [**-o full,nolog**] [*special*...]

DESCRIPTION

The **fsck** utility checks VxFS file systems for consistency. Since VxFS records pending file system updates in an intent log, **fsck** typically runs an intent log replay, rather than a full structural file system check on a VxFS file system.

If *special* is not specified, **fsck** reads the table in **/etc/fstab**, using the first field to determine which file system to check.

Options

-F vxfs Specify the VxFS file system type.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-y|**Y** Assume a "yes" response to all questions asked by **fsck**. Additionally, if the file system requires a full file system check after the log replay, or if the **nolog** suboption causes the log replay to be skipped and the file system is not clean, then a full file system check is performed.

-m Check whether or not the file system is marked clean. This option does not validate the file system. If the file system is corrupt for some reason, a subsequent mount may fail and a full **fsck** may be required to clean it. **fsck -n** may be used to test for file system corruption.

-n|**N** Assume a "no" response to all questions asked by **fsck**; do not open the file system for writing. Log replay is not performed. A full file system check is performed.

-p Cause **fsck** to produce messages that identify the device being checked.

-P With VxFS, **-P** is used by **fsck** by default; it does not provide any functionality. With other file system types, **-P** may be used for optional functionality.

-o Specify VxFS file system specific options. These options can be a combination of the following in a comma-separated list:

full
Perform a full file system check. The default is to perform an intent log replay only. Since the VxFS file system maintains an intent log, a complete check is generally not required. If the file system detects damage or the log replay operation detects damage, an indication that a complete check is required is placed in the super-block, and a full check is performed.

nolog
Do not perform log replay. This option may be used if the log area was physically damaged.

When a full check is performed, the following inconsistencies are checked:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode outside the range of the file system.
- Incorrect link counts.
- Size checks:
  - Incorrect number of blocks.
  - Directory entry format.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks:
  - File pointing to unallocated inode.
  - Inode number out of range.
  - Linkage to parent directory.

  - Hash chain linkage.
  - Free space count.
- Super-block checks:
  - Checksum mismatch.
  - More blocks for inodes than there are in the file system.
- Structural Files:
  - Fileset headers.
  - Object Location Table (OLT).
  - Inode list files.
  - Inode allocation summary files.
  - Attribute files (including Access Control Lists).
  - Attribute link counts.
- Bad free block list format.
- Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the user's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. The only restriction is that the directory **lost+found** must already exist in the root of the file system being checked.

OUTPUT

Structural errors discovered during a full check are displayed on standard output. Responses required during a full check are read from standard input.

The following return codes are used for the **-m** (generic) option for all devices other than the one used by the root file system:

0	The file system is unmounted and clean.
32	The file system is unmounted and needs checking.
33	The file system is mounted.
34	The stat of the device failed.
Other	The state could not be determined because of an error.

The following return codes are used for the **-m** (generic) option for the device used by the root file system:

0	The root file system is mounted read-only and is clean, or the root file system is mounted read/write and therefore doesn't need checking.
32	The root file system is mounted read-only and needs checking.
34	The stat of the device failed.
Other	The state could not be determined because of an error.

ERROR/DIAGNOSTICS

All error messages that relate to the contents of a file system produced during a log replay are displayed on standard output. All I/O failures and exit messages are displayed on standard error output.

NOTES

Checking the raw device is almost always faster.

A full file-system check will always perform any pending extended-inode operations, generating various messages, without operator interaction. If a structural flaw is detected, the **VX_FULLFSCK** flag will be set on the file system, without operator interaction. If **fsck** was not invoked with the **-y** option, it must be reinvoked with the **-y** or **-o full** option to perform a full **fsck**.

If the **-o full** flag is used on a clean file system, **fsck** will perform a log replay first, and since the **VX_FULLFSCK** flag is set, it will not update the inode and extent maps before performing the full **fsck**, so it will report inconsistencies. Use the **-n** option to verify file-system inconsistency.

FILES

/etc/fstab	Default list of file systems to check.

SEE ALSO

fsck(1M), mkfs(1M), ncheck(1M).

NAME

fsclean - determine the shutdown status of HFS file systems

SYNOPSIS

/sbin/fsclean [**-q**] [**-v**] [*special* ...]

DESCRIPTION

The **fsclean** command determines the shutdown status of the HFS file system specified by *special* or, in the absence of *special*, the file systems listed in **/etc/fstab** of type **hfs** with the **rw**, **default**, or **ro** options set. All optional fields in **/etc/fstab** must be present for **fsclean** to be able to check each file system.

fsclean reads the superblock to determine whether the file system's last shutdown was done correctly, and returns one of the following values:

0 All of the checked file systems were shut down correctly.

1 One or more checked file systems were not shutdown correctly, implying that **fsck** should be run (see *fsck*(1M)).

2 Other error (such as **cannot open the specified device file**).

The **fsclean** command is usually silent.

Options:

-q Check quotas. Instead of checking the file system shutdown status, **fsclean** checks the validity of disk quota statistics. This option is useful for determining whether **quotacheck** should be run (see *quotacheck*(1M)). If *special* is not provided, then all file systems in **/etc/fstab** of type **hfs** with the **rw** (or **default**) and **quota** options are checked.

-v Be verbose. Prints the status of each file system checked.

DEPENDENCIES

fsclean only operates on HFS file systems.

AUTHOR

fsclean was developed by HP.

FILES

/etc/fstab	Default list of file systems to check

SEE ALSO

dumpfs(1M), fsck(1M), fsck_hfs(1M), mount(1M), quotacheck(1M), quotacheck_hfs(1M), reboot(1M), fstab(4).

NAME

fsdb - file system debugger (generic)

SYNOPSIS

/usr/sbin/fsdb [**-F** *FStype*] [**-o** *specific_options*] [**-V**] *special*

Remarks

Always execute the **fsck** command (see *fsck*(1M)) after running **fsdb**.

DESCRIPTION

The **fsdb** command can be used to patch up a damaged file system after a crash. It is intended for experienced users only. The file system type to be debugged is specified as *FStype*. Each file system type has a unique structure requiring different debugging capabilities. The manual entries for the file-system-specific **fsdb** should be consulted before attempting any debugging or modifications.

Options and Arguments

fsdb recognizes the following options and arguments:

special The file name of the special file containing the file system.

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-o *specific_options*
Specify suboptions specific to each file system type. *specific_options* is a comma-separated list of suboptions and/or keyword/attribute pairs supported by the specific *FStype*.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from the **/etc/fstab** file. This option allows the user to verify the command line.

EXAMPLES

Invoke the file system debugger on HFS file system **/dev/dsk/c1d2s0**:

```
fsdb -F hfs /dev/dsk/c1d2s0
```

Display a completed command line without executing the debugger:

```
fsdb -V /dev/dsk/c1d2s0
```

The previous command might display:

```
fsdb -F hfs /dev/dsk/c1d2s0
```

WARNINGS

Only experienced users should use **fsdb**. The failure to fully understand the usage of **fsdb** and the file system's internal organization can lead to complete destruction of the file system and total loss of data.

AUTHORS

fsdb was developed by HP and AT&T.

FILES

/etc/default/fs	Specifies the default file system type
/etc/fstab	Static information about the file systems

SEE ALSO

fsck(1M), fsdb_*FStype*(1M), fstyp(1M), stat(2), fs_wrapper(5).

STANDARDS CONFORMANCE

fsdb: SVID3

NAME

fsdb - HFS file system debugger

SYNOPSIS

/usr/sbin/fsdb [**-F hfs**] [**-V**] *special* [**-b** *blocknum*] [**-**]

Remarks

Always execute the **fsck** command (see *fsck*(1M)) after running **fsdb**.

DESCRIPTION

The **fsdb** command can be used to patch up a damaged file system after a crash.

Options and Arguments

fsdb recognizes the following options and arguments.

special	The file name of the special file containing the file system.
-	Initially disable the error-checking routines that are used to verify the inode and fragment addresses. See the **O** symbol. If used, this option must follow *special* on the command line.
-b *blocknum*	Use *blocknum* as the superblock for the file system. If used, this option must follow *special* on the command line.
-F hfs	Specify the HFS file system type.
-V	Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from the **/etc/fstab** file. This option allows the user to verify the command line.

Operation

fsdb normally uses the first superblock for the file system, located at the beginning of the disk section, as the effective superblock. An alternate superblock can always be found at block **((SBSIZE+BBSIZE)/DEV_BSIZE)**, typically block 16. The **-b** option can be used to specify the superblock location.

fsdb deals with the file system in terms of block fragments, which are the unit of addressing in the file system and the minimum unit of space allocation. To avoid possible confusion, *fragment* is used to mean that, and *block* is reserved for the larger true block. **fsdb** has conversions to translate fragment numbers and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control block entries or descending the file system tree.

fsdb contains several error-checking routines to verify inode and fragment addresses. These can be disabled if necessary by invoking **fsdb** with the optional **-** argument, or by using the **O** symbol.

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. Hexadecimal numbers must be prefixed with **0x**. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

fsdb reads a fragment at a time. A buffer management routine is used to retain commonly used fragments of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding fragment.

Symbols

The following symbols are recognized by **fsdb**:

!	Escape to shell
#	Absolute address
+	Address arithmetic
-	Address arithmetic
<	Restore an address
>	Save an address
=	Numerical assignment
=+	Incremental assignment
=-	Decremental assignment

="	Character string assignment
b	Convert from fragment number to disk address (historically "block")
d	Directory slot offset
f	File print facility
i	Convert from i-number to inode address; for continuation inodes as well as primary inodes (see *inode*(4))
p	General print facility
q	Quit
B	Byte mode
D	Double-word mode
O	Error checking flip-flop
W	Word mode
X	Hexadecimal flip-flop

Dots, tabs, and spaces can be used as function delimiters, but are not necessary. A line with just a newline character increments the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry, or inode, allowing the user to step through a region of a file system.

Information is printed in a format appropriate to the data type. If the **X** toggle is off, bytes, words, and double words are printed in the form:

octal-address **:** *octal-value* **(***decimal-value***)**

If the **X** toggle is on, bytes, words, and double words are printed in the form:

hex-address **:** *hex-value*

If the **B** (byte) or **D** (double-word) mode is in effect, the colon (**:**) shown above is preceded by **.B** or **.D**, respectively.

Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name.

Inodes are printed with labeled fields describing each element.

Print Facilities

The print facilities generate a formatted output in various styles. Octal numbers are prefixed with a zero. Hexadecimal numbers are prefixed with **0x**. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the interrupt character. If a number follows the **p** symbol, that many entries are printed. A check is made to detect fragment boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current fragment are printed. The print options available are:

b	Print as octal bytes
c	Print as characters
d	Print as directories
e	Print as decimal words
i	Print as inodes (primary or continuation)
o	Print as octal words
x	Print as hexadecimal words

The **f** symbol prints data fragments associated with the current inode. If followed by a number, that fragment of the file is printed. (Fragments are numbered from zero). The desired print option letter follows the fragment number, if present, or the **f** symbol. This print facility works for small as well as large files except for special files such as FIFOs, and device special files.

Inode and Directory Mnemonics

The following mnemonics are used for inode examination and refer to the current working inode:

a*num*	Data block numbers (*num* is in the range 0 – 14)
at	Time last accessed
ci	Continuation inode number
ct	Last time inode changed
gid	Group ID number
ln	Link count

maj	Major device number
md	Mode
min	Minor device number
mt	Time last modified
sz	File size in byte unit
uid	User ID number

The following mnemonics are used for directory examination:

di	I-number of the associated directory entry
nm	Name of the associated directory entry

EXAMPLES

386i	Print i-number 386 in an inode format. This now becomes the current working inode.
ln=4	Change the link count for the working inode to 4.
ln=+1	Increment the link count by 1.
fc	Print in ASCII fragment zero of the file associated with the working inode.
2i.fd	Print the first fragment-size piece of directory entries for the root inode of this file system.
d5i.fc	Change the current inode to that associated with the fifth directory entry (numbered from zero) found from the above command. The first fragment's worth of bytes of the file are then printed in ASCII.
1b.px	Print the first fragment of the superblock of this file system in hexadecimal.
2i.a0b.d7=3	Change the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.
d7.nm="newname"	Change the name field in the directory slot to the given string. Quotes are optional if the first character of the name field is alphabetic.
a2b.p0d	Print the third fragment of the current inode as directory entries.

WARNINGS

Only experienced users should use **fsdb**. The failure to fully understand the usage of **fsdb** and the file system's internal organization can lead to complete destruction of the file system and total loss of data.

AUTHOR

fsdb was developed by HP and AT&T.

FILES

/etc/fstab Static information about the file systems

SEE ALSO

dumpfs(1M), fsck(1M), fsdb(1M), stat(2), dir(4), fs(4).

STANDARDS CONFORMANCE

fsdb: SVID3

f

NAME

fsdb (vxfs) - VxFS file system debugger

SYNOPSIS

/usr/sbin/fsdb [**-F vxfs**] [**-V**] [**-z** *inumber*] *special*

DESCRIPTION

The **fsdb** command can be used to patch up a damaged VxFS file system after a crash. A special device *special* is used to indicate the file system to be debugged. The **fsdb** command is intended for experienced users only.

The **fsdb** command has conversions to translate block and inumbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control block entries or descending the file system tree.

By default, numbers are considered decimal. Octal numbers must be prefixed with **0.** Hedecimal numbers must be prefixed with **0x**. When using hexadecimal numbers, it is preferable to follow the number with a space, since a number of commands are letters that are also hexadecimal digits. In this document a pound sign (**#**) is used to indicate that a number is to be specified.

The **fsdb** command reads a block at a time and works with raw and block I/O. All I/O is unbuffered, so changes made to the file system are immediate and changes made by other processes or by the kernel are immediately seen by the **fsdb** command.

Options

-F vxfs	Specifies the VxFS file-system type.
-V	Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.
-z *inumber*	Clear the inode identified by *inumber* (non-interactive). Multiple **-z** options accumulate.

The following symbols are recognized by the **fsdb** command:

h [mod | print]	Print summary of commands that display [modify | format] the file system.
? [mod | print]	Print summary of commands that display [modify | format] the file system.
help [mod | print]	Print summary of commands that display [modify | format] the file system.
!	Escape to shell.
|	Pipe output of fsdb command to a shell command.
q	Quit.
"string"	A character string. Inside a character string, a NULL character may be specified with "\0"; a double quote may be specified with "\""; and a backslash may be specified with "\\ ".
+ - ∗ / %	Add, subtract, multiply, divide, and modulus.
=	Assignment
i	An inode in the primary inode list.
ai	An inode in the attribute inode list.
au	An allocation unit.
b	A block.
im	The immediate data area of an inode. Small directories and symbolic link files (96 bytes or less) are stored directly in the inode itself, in the area normally occupied by data block numbers and extent sizes.
attr	An attribute inode.
cdb	Current directory block.
d	A directory entry.
a	An inode address entry.
B	A byte.
H	A half-word (2 bytes)
W	A word (4 bytes)
D	A double-word (8 bytes)
p	General print facility
calc	Simple calculator and base converter
find	Find a matching pattern in the file system

fset	A fileset.
iau	An inode allocation unit in the primary inode list.
aiau	An inode allocation unit in the attribute inode list.
cut	The current usage table.
olt	The object location table.
mapi	Map logical file offset to an inode extent.
reset	Reset device.

The print facility recognizes the following print formats:

S	Print as a super-block.
A	Print as an allocation-unit header.
AS	Print as an auxilliary super-block.
L	Print as intent-log records.
I	Print as inodes.
T	Print as typed extent descriptors.
dent	Print as directory entries.
db	Print as a directory block.
dh	Print as a directory header.
o	Print as octal words.
oB oH oW oD	Print as octal bytes, half-words, words, or double-words.
x	Print as hexadecimal words.
xB xH xW xD	Print as hexadecimal bytes, half-words, words, or double-words.
e	Print as decimal words.
eB eH eW eD	Print as decimal bytes, half-words, words, or double-words.
c	Print as characters.
F	Print as fileset headers.
C	Print as current usage table entries.
IA	Print as an inode allocation unit header.
oltext	Print as an object location table extent.
Q	Print as a BSD quota record.
DV	Print as a device record.

Changes to inode fields may be made symbolically. The following symbols represent inode fields:

md	Inode mode field
ln	Inode link count field
uid	Inode user ID Number field
gid	Inode group ID Number field
szlo	Low-order word of inode file size field
szhi	High-order word of inode file size field
sz	Inode file size field
de#	Inode direct extent data block numbers (0 - 9)
des#	Inode direct extent sizes (0 - 9)
ie#	Inode indirect extent data block numbers (0 - 1)
ies	Inode indirect extent size
at	Inode access time field (seconds)
ats	Inode access time field (microseconds).
ct	Inode change time field (seconds).
cts	Inode change time field (microseconds).
mt	Inode modification time field (seconds).
mts	Inode modification time field (microseconds).
af	Inode allocation flags field.
gen	Inode generation count field.
org	Inode mapping type field.
fe	Inode fixed extent size field.
bl	Inode blocks held field.
eopflg	Inode extended operation flag field.
eopdat	Inode extended operation data field.
rdev	If device, inode device number.
maj	If device, inode major number.
min	If device, inode minor number.
pd	If directory, inode parent directory.
res	If regular file, inode reservation.

verhi	Inode high-order word of serial number.
verlo	Inode low-order word of serial number.
fsindex	Referencing fileset ID.
matching	Inode number of matching inode.
iano	Indirect attribute inode.

Changes to directory block fields may be made symbolically. The following symbols represent directory block fields:

tfree	Total free space (only if in a data block).
hash#	Hash chain start (0 through 31, only if in a data block).
d#	Directory entry (variable number of entries).
nhash	Number of hash chains.

Changes to directory entry fields may be made symbolically. The following symbols represent directory entry fields:

ino	Inode number
nm	Entry name
nmlen	Name length
reclen	Record length (only if in a data block)
hnext	Name hash next (only if in a data block)

It is preferable to separate each token on a command line with a space. Although the command parser does not insist on space separation, there is no ambiguity in the command language if each token is separated with a space. For example, the command 0x23b b sets the current position to block 0x23b hexadecimal. The command 0x23bb is invalid, since the command is parsed as simply a hexadecimal number. The command 23b positions to block 23 decimal, since the command is not ambiguous.

Commands are separated by new lines, or multiple commands may be placed on one line, separated by a period (.) or a semicolon (;). When multiple commands are placed on one line, generally only the last command displays results. This allows positioning commands to be followed by printing commands or change commands without intermediate printing.

The **fsdb** command maintains several positions in the file system: the **current position**, the **current primary-inode position (i)**, the **current attribute-inode position (ai)**, the **current inode type (i or ai)**, the **current fileset-header position (fset)**, the **current allocation-unit position (au)**, the **current primary-inode allocation-unit (iau) position**. the **current inode allocation-unit type (iau or aiau)**. the **current attribute-inode allocation-unit (aiau) position**. These are used by various **fsdb** commands. (The au positions are supported through Version 3, but not beyond.)

The following commands are supported:

# B|H|W|D	Set current position in the file system to the specified offset in bytes, half-words, words, or double-words. If the last command on a line, print the byte, half-word, word, or double-words in hexadecimal.
+|- # B|H|W|D	Set current position to specified relative offset in bytes, half-words, words, or double-words. If the last command on a line, print the byte, half-word, word, or double-words in hexadecimal.
# au	Set current position in the file system to the specified allocation unit (au) position. Set current allocation unit position to the resulting offset. If the last command on a line, print the allocation unit header.
+|- # au	Set current position in the file system to the specified position relative to the current allocation unit (au) position. Set current allocation unit position to the resulting offset. If the last command on a line, print the allocation unit header.
au	Set current position in the file system to the current allocation unit position. If the last command on a line, print the allocation unit header.
# b	Set current position in the file system to the specified offset in blocks. Set current block position to the resulting offset. The block size is the block size of the file system. If the last command on a line, print the first word in the block in hexadecimal.
+|- # b	Set current position to specified relative offset in blocks. Set current block position to the resulting offset. If the last command on a line, print the first word in the block in hexadecimal.
b	Set current position to current block position (the block specified by the last [+|-] # b operation). If the last command on a line, print the first word in the block in hexadecimal.

cut	Set current position to the current usage table (cut). If the last command on a line, print the first current usage table entry.
dev	Set current position to the primary device's configuration record. If the last command on a line, print the device-configuration record.
# fset	Set current position in the file system to the fileset header entry for the specified fileset index. Set current fileset position to the resulting offset. If the last command on a line, print the specified fileset header.
+ | - # fset	Set current position in the file system to the fileset header entry for the specified position relative to the current fileset position. Set current fileset position to resulting offset. If the last command on a line, print the specified fileset header.
fset	Set current position in the file system to the current fileset position. If the last command on a line, print the fileset header for the current fileset.
# aiau	Set current position in the file system to the specified attribute inode allocation unit (aiau) in a fileset. Set the current attribute inode allocation unit position to the resulting offset. If the last command on a line, print the attribute inode allocation unit header.
+ | - # aiau	Set the current position in the file system to the specified position relative to the current attribute inode allocation unit (aiau) position. Set the current attribute inode allocation unit position to the resulting offset. If the last command on a line, print the attribute inode allocation unit header.
aiau	Set the current position in the file system to the current attribute inode allocation unit (aiau) position. If the last command on a line, print the attribute inode allocation unit header.
# iau	Set current position in the file system to the specified inode allocation unit (iau) in a fileset. Set the current inode allocation unit position to the resulting offset. If the last command on a line, print the inode allocation unit header.
+ | - # iau	Set the current position in the file system to the specified position relative to the current inode allocation unit (iau) position. Set the current inode allocation unit position to the resulting offset. If the last command on a line, print the inode allocation unit header.
iau	Set the current position in the file system to the current inode allocation unit (iau) position. If the last command on a line, print the inode allocation unit header.
# ai	Set current position in the current fileset to the ilist entry for the specified attribute inode. Set current attribute inode position to the resulting offset. If the last command on a line, print the ilist entry for the inode.
+ | - # ai	Set current position in the current fileset to the ilist entry for the specified relative attribute inode. Set current attribute inode position to the resulting offset. If the last command on a line, print the ilist entry for the inode.
ai	Set current position in the current fileset to the current attribute inode position. If the last command on a line, print the ilist entry for the inode.
# i	Set current position in the current fileset to the ilist entry for the specified inode. Set current inode position to the resulting offset. If the last command on a line, print the ilist entry for the inode.
+ | - # i	Set current position in the current fileset to the ilist entry for the specified relative inode. Set current inode position to the resulting offset. If the last command on a line, print the ilist entry for the inode.
i	Set current position in the current fileset to the current inode position. If the last command on a line, print the ilist entry for the inode.
a#	Set current position to specified offset in blocks specified by the inode address #. Addresses 0 through 9 are for direct extents (de). Addresses 10-11 are for indirect extents (ie). The addresses are displayed when printing an ilist entry. Set current block position to the resulting offset. If the last command on a line, print the first word in the block in hexadecimal.
im	Set current position to immediate data area of the current inode. Set current block position to the resulting offset. If the last command on a line, print the first word of the area in hexadecimal.
attr	Set current position to attribute data area of the current inode. Set current block position to the resulting offset. If the last command on a line, print the first word in the block in hexadecimal.

B | H | W | D =# [#]

Set the current position and change the number at the specified offset to the given number. If a double-word offset is specified, then two numbers separated by a space are required. The resulting value is printed in hexadecimal.

+|-# B|H|W|D =# [#]
: Set the current position and change the number at the specified relative offset to the given number. If a double-word offset is specified, then two numbers separated by a space are required. The resulting value is printed in hexadecimal.

B|H|W|D = "*string*"
: Set the current position and change the characters at the specified offset to the given string. The resulting value is printed as a character string.

+|- # B|H|W|D = "*string*"
: Set the current position and change the characters at the specified relative offset to the given string. The resulting value is printed as a character string.

olt
: Set the current position to the object location table (olt). If the last command on a line, print the object location table.

p [#] format
: Print the contents of the file system at the current offset as the specified number of entries of a given format. The allowable print formats are specified above. If a number of entries to print is not specified, one entry is printed.

inode_field = #
: Set the contents of the given inode field to the specified number. The current inode specifies the inode list entry to be modified. The symbols representing inode fields are previously listed.

directory_block_field = #
: Set the contents of the given directory block field to the specified number. The current block is treated as a directory block and the offset in that block which is represented by the given field is changed. The symbols representing directory block fields are listed above.

d#
: Set the current directory entry to the specified number. The current block is treated as a directory block. If the current block is an immediate data area for an inode, then the block is treated as containing immediate directory entries. If the last command on a line, the directory entry at the resulting offset is printed.

directory_entry_field = #
: Set the contents of the given directory field to the specified number. The current directory entry specifies where the directory entry is located. The resulting value is printed in hexadecimal.

nm = "*string*"
: Set the directory name field of the current directory entry to the specified string. The resulting value is printed as a character string.

calc # [+|-|*|/ #]
: Take a number or the sum, difference, product or dividend of two numbers and print in decimal, octal, hexadecimal and character format.

find # B|H|W|D [#]
: Search for the given numeric pattern in the file system. The size of the object to match is specified. If a double-word is specified, then two numbers must be given. The search is performed forward from the current offset. A maximum number of blocks to search may be specified. If found, the location and value are printed in hexadecimal.

find "*string*" [#]
: Search for the given character string in the file system. The search is performed forward from the current offset. A maximum number of blocks to search may be specified. If found the location and string are printed.

fmtlog
: Format all intent log entries. A completely formatted intent log can be quite lengthy. It is a good idea use the **fsdb** command as a filter and redirect the output to a file or pager to look at a complete log format.

listfset
: List all filesets by their indexes and names.

mapi #
: Treat the number as a logical offset in the file described by the current inode, and print the extent that it maps to.

reset
: Does the equivalent of exiting **fsdb** and restarting on same device.

The following help commands are supported:

h|help	Display primary help screen.
h mod	Display modification-commands help screen.
h print	Display print-commands help screen.

EXAMPLES

386i	Prints inumber 386 in an inode format. This now becomes the current working inode.
ln=4	Changes the link count for the working inode to 4.
1024.p S	Prints the super-block of this file system symbolically.
2i.a0b.d7.ino = 3	Changes the inumber for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.
d7.nm = "*foo*"	Changes the name field in the directory slot to "*foo*".
23i.im.pdb	Prints the immediate area of inode 23 as a directory block.
23i.im.d5	Prints the sixth directory entry in the immediate area of inode 23.

WARNINGS

Always execute *fsck*(1M) after using the **fsdb** command to modify a file system (use **fsck -o full,nolog**).

SEE ALSO

fsck(1M), fsdb(1M).

NAME

fsirand - install random inode generation numbers

SYNOPSIS

/usr/sbin/fsirand [**-p**] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special*, and also installs a filesystem ID in the superblock. This process increases the security of filesystems exported by NFS.

Use **fsirand** only on an unmounted filesystem that was checked with **fsck** (see *fsck*(1M)). The only exception is that it can be used on the root filesystem in single-user mode if the system is immediately re-booted afterwards using **reboot -n**.

The **-p** option prints the generation numbers for all inodes.

WARNINGS

fsirand should not be run on mounted filesystems. If executing **fsirand** on the root filesystem, the system should be in single-user mode and should be re-booted immediately afterwards using **reboot -n**.

AUTHOR

fsirand was developed by Sun Microsystems, Inc.

SEE ALSO

statfs(2).

NAME

fstyp - determine file system type

SYNOPSIS

/usr/sbin/fstyp [**-v**] *special*

DESCRIPTION

The **fstyp** command allows the user to determine the file system type of a mounted or unmounted file system. *special* represents a device special file (for example: **/dev/dsk/c1t6d0**).

The file system type is determined by reading the superblock of the supplied *special* file. If the superblock is read successfully, the command prints the file system type identifier on the standard output and exits with an exit status of **0**. If the type of the file system cannot be identified, the error message **unknown_fstyp** (no matches) is printed and the exit status is **1**. Exit status **2** is not currently returned, but is reserved for the situation where the file system matches more than one file system type. Any other error will cause exit status **3** to be returned.

The file system type is determined by reading the superblock of the supplied *special* file.

f

Options

-v	Produce verbose output. The output contains information about the file system's superblock.

RETURN VALUE

fstyp returns the following values:

0	Successful completion.
1	Unknown file system type.
2	File system matches more than one type.
3	Usage error or access problem.

EXAMPLES

Find the type of the file system on a disk, **/dev/dsk/c1t6d0**:

```
fstyp /dev/dsk/c1t6d0
```

Find the type of the file system on a logical volume, **/dev/vg00/lvol6**:

```
fstyp /dev/vg00/lvol6
```

Find the file system type for a particular device file and also information about its super block:

```
fstyp -v /dev/dsk/c1t6d0
```

SEE ALSO

stat(2), statvfsdev(2).

NAME

ftpd - DARPA Internet File Transfer Protocol server

SYNOPSIS

/usr/lbin/ftpd [**-l**] [**-p**] [**-v**] [**-t** *timeout*] [**-P**] [**-T** *maxtimeout*] [**-u** *umask*] [**-B** *size*]

DESCRIPTION

ftpd is the DARPA Internet File Transfer Protocol server. It expects to be run by the Internet daemon (see *inetd*(1M) and *inetd.conf*(4)). **inetd** runs **ftpd** when a service request is received at the port indicated in the **ftp** service specification in **/etc/services** (see *services*(4)). **ftpd** recognizes the following options and command-line arguments.

-l	Causes each FTP session to be logged in the syslog file. For anonymous FTP sessions, other information is also logged in the syslog file. This information includes what files are stored and retrieved and what directories are created.
-p	The default action of ftpd does not allow usage of reserved ports as the originating port on the client's system i.e., the PORT command cannot specify a reserved port. This option allows the client to specify a reserved port. Note, allowing usage of reserved ports can result in the misuse of ftpd. The security ramifications should be understood before the option is turned on.
-v	Logs other information in the syslog file. This information is what is normally logged for anonymous FTP sessions. This information includes what files are stored and retrieved and what directories are created.
-t *timeout*	Causes **ftpd** to timeout inactive sessions after *timeout* seconds. By default, **ftpd** terminates an inactive session after 15 minutes.
-P	Enables third party transfer.
-T *maxtimeout*	A client can also request a different timeout period. The **-T** option sets to *maxtimeout* the maximum timeout that client can request, in seconds. By default, the maximum timeout is 2 hours.
-u *umask*	Change default **ftpd** umask from 027 to *umask*.
-B *size*	Sets the buffer size of the data socket to *size* blocks of 1024 bytes. The valid range for *size* is from 1 to 64 (default is 56). **NOTE:** A large buffer size will improve the performance of **ftpd** on fast links (e.g. FDDI), but may cause long connection times on slow links (e.g. X.25).

ftpd currently supports the following commands (uppercase and lowercase are interpreted as equivalent):

Command	**Description**
ABOR	Abort previous command
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	Append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	Give list files in a directory (**ls -l**)
MKD	Make a directory
MDTM	Show last modification time of file
MODE	Specify data transfer *mode*
NLST	Give name list of files in directory
NOOP	Do nothing
PASS	Specify password
PASV	Prepare for server-to-server transfer
PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
REST	Restart incomplete transfer

RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from file name
RNTO	Specify rename-to file name
SITE	Non-standard commands (see next section)
SIZE	Return size of file
STAT	Return status of server
STOR	Store a file
STOU	Store a file with a unique name
STRU	Specify data transfer *structure*
SYST	Show operating system type of server system
TYPE	Specify data transfer *type*
USER	Specify user name
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The following non-standard or HP-UX specific commands are supported by the **SITE** command:

Command	**Description**
UMASK	Change umask. (e.g., **SITE UMASK 002**)
IDLE	Set idle-timer. (e.g., **SITE IDLE** 60)
CHMOD	Change mode of a file. (e.g., **SITE CHMOD 755** filename)
HELP	Give help information. (e.g., **SITE HELP**)

The remaining FTP requests specified in Internet RFC 959 are recognized, but not implemented. **MDTM** and **SIZE** are not specified in RFC 959, but are expected in the next updated **FTP RFC**.

The FTP server aborts an active file transfer only when the **ABOR** command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959. If **ftpd** receives a **STAT** command during a data transfer, preceded by a Telnet IP and Synch, it returns the status of the transfer.

ftpd interprets file names according to the "globbing" conventions used by *csh*(1). This allows users to utilize the metacharacters *****, **.**, **[**, **]**, **{**, **}**, **~**, and **?**.

ftpd authenticates users according to three rules:

- The user name must be in the password data base, **/etc/passwd**, and not have a null password. The client must provide the correct password for the user before any file operations can be performed.
- The user name must not appear in the file **/etc/ftpusers** (see *ftpusers*(4)).
- The user must have a standard shell returned by **getusershell()**.

Optionally, a system administrator can permit public access or "anonymous FTP." If this has been set up, users can access the anonymous FTP account with the user name **anonymous** or **ftp** and any non-null password (by convention, the client host's name). **ftpd** does a **chroot()** to the home directory of user **ftp**, thus limiting anonymous FTP users' access to the system. If the user name is **anonymous** or **ftp**, an anonymous FTP account must be present in the password file (user **ftp**). In this case the user is allowed to log in by specifying any password (by convention this is given as the user's e-mail address).

In order to permit anonymous FTP, there must be an entry in the *passwd*(4) database for an account named **ftp**. The password field should be *****, the group membership should be **guest**, and the login shell should be **/usr/bin/false**. For example (assuming the **guest** group ID is **10**):

```
ftp:*:500:10:anonymous ftp:/home/ftp:/usr/bin/false
```

The anonymous FTP directory should be set up as follows:

~ftp The home directory of the FTP account should be owned by user **root** and mode 555 (not writable). Since **ftpd** does a **chroot()** to this directory, it must have the following subdirectories and files:

~ftp/usr/bin
This directory must be owned by root and mode 555 (not writable). The file **/sbin/ls** should be copied to **~ftp/usr/bin.** This is needed to support directory listing by

ftpd. The command should be mode 111 (executable only). If the FTP account is on the same file system as **/sbin**, **˜ftp/usr/bin/ls** can be hard link, but it may not be a symbolic link, because of the **chroot()**. The command must be replaced when the system is updated.

~ftp/etc
This directory must be owned by root and mode 555 (not writable). It should contain versions of the files *passwd*, *group*, and *logingroup*. See *passwd*(4) and *group*(4). These files must be owned by root and mode 444 (readable only). These are needed to map user and group ids in the **LIST** command, and to support (optional) sub-logins of anonymous FTP. Sub-logins can sometimes be used to allow access to particular files by only specific remote users (who know the sub-login password) without giving those remote users logins on the system. A sub-login user would access the system via anonymous FTP, then use **USER** and **PASS** to change to the sub-login user.

~ftp/etc/passwd
This file should contain entries for the **ftp** user and any other users who own files under the anonymous **ftp** directory. Such entries should have ***** for passwords. **~ftp/etc/passwd** should also contain entries for any desired anonymous FTP sub-logins. The sub-logins must have passwords, which must be encrypted as in *passwd*(4). Group IDs must be listed in the anonymous FTP group file, **˜ftp/etc/group**. The path names of home directories in **˜ftp/etc/passwd** must be with respect to the anonymous FTP home directory. A sub-login home directory should be owned by the sub-login user ID. The shell field is ignored, and can be empty.

For example, the anonymous FTP sub-login name **subftp** would have an entry in the FTP **passwd** file that resembles:

```
subftp:bAg6vI82aq5Yt:501:10:ftp sub-login:/subftp:
```

FTP sub-login IDs do not need to be present in the system **/etc/passwd** file. Assuming the anonymous FTP directory is **/home/ftp**, the sub-login home directory in the example would be created by user **root** as follows:

```
cd /home/ftp
mkdir subftp
chmod 700 subftp
chown 501 subftp
chgrp guest subftp
```

File **˜ftp/etc/group** should contain the group names associated with any group IDs in file **˜ftp/etc/passwd** and any group IDs of files in the anonymous FTP subdirectories. In the above example, **˜ftp/etc/group** would require an entry for **guest**, and the associated group ID would have to be the same as in the system's **/etc/group** file.

~ftp/etc/logingroup
Permits anonymous ftp sub-logins to be members of multiple groups. Can be a hard link to FTP **˜ftp/etc/group**.

~ftp/pub (optional)
This directory is used by anonymous FTP users to deposit files on the system. It should be owned by user **ftp** and should be mode 777 (readable and writable by all).

~ftp/dist (optional)
Directories used to make files available to anonymous ftp users should be mode 555 (not writable), and any files to be distributed should be owned by root and mode 444 (readable only) so that they cannot be modified or removed by anonymous FTP users.

DIAGNOSTICS

ftpd replies to FTP commands to ensure synchronization of requests and actions during file transfers, and to indicate the status of **ftpd**. Every command produces at least one reply, although there may be more than one. A reply consists of a three-digit number, a space, some text, and an end of line. The number is useful for programs; the text is useful for users. The number must conform to this standard, but the text can vary.

The first digit of the message indicates whether the reply is good, bad, or incomplete. Five values exist for the first digit. The values and the interpretations of the values are:

1	The requested action is being initiated; expect another reply before proceeding with a new command.
2	The requested action is complete. The server is ready for a new request.
3	The command has been accepted, but the requested action requires more information.
4	The command was not accepted, the requested action failed, but the error condition is temporary and the action can be requested again.
5	The command was not accepted, the requested action failed, and the error condition would most likely occur again if the same command sequence is repeated.

The second digit indicates the functional area that the message addresses. The values of the second digit and the interpretations of these values are:

0	Syntax. A message with a 0 for the second digit indicates that a syntax error occurred.
1	Information. A message with a 1 as the second digit indicates that the message is in reply to a request for information.
2	Connections. A message with a 2 as the second digit indicates that the message is a reply to a request for control and data connection information.
3	Authentication and accounting. A message with a 3 as the second digit indicates that the message is a reply to a login or accounting procedure.
4	Not currently specified.
5	File system. A message with a 5 as the second digit indicates that the text following the number contains information concerning the status of the server file system.

The third digit provides a further clarification of the information supplied by the second digit. Following are several examples of messages. Note that **ftpd**'s replies match the number but not the text.

110	Restart marker reply. MARK *yyyy*=*mmmm* where *yyyy* is a user process data stream marker, and *mmmm* is **ftpd**'s equivalent marker
120	Service ready in *nnn* minutes
200	Command okay
211	System status, or system help reply
212	Directory status
230	User logged in, proceed
250	Requested file action okay, completed
331	User name okay, need password
350	Requested file action pending further information
425	Cannot open data connection
451	Requested action aborted: local error in processing
500	Syntax error, command unrecognized or command line too long
530	Not logged in
550	Requested action not taken; file unavailable, not found, no access

WARNINGS

The password is sent unencrypted through the socket connection.

Anonymous FTP is inherently dangerous to system security.

DEPENDENCIES

Pluggable Authentication Modules (PAM)

PAM is an Open Group standard for user authentication, password modification, and validation of accounts. In particular, **pam_authenticate()** is invoked to perform all functions related to login. This includes retrieving the password, validating the account, and displaying error messages.

AUTHOR

ftpd was developed by the University of California, Berkeley.

SEE ALSO

ftp(1), inetd(1M), chroot(2), getusershel(3C), inetd.conf(4), ftpusers(4), passwd(4), group(4), pam_authenticate(3).

NAME

ftpd - DARPA Internet File Transfer Protocol server

SYNOPSIS

/usr/lbin/ftpd [**-l**] [**-p**] [**-v**] [**-t** *timeout*] [**-P**] [**-T** *maxtimeout*] [**-u** *umask*] [**-A**] [**-B** *size*]

DESCRIPTION

ftpd is the DARPA Internet File Transfer Protocol server. It expects to be run by the Internet daemon (see *inetd*(1M) and *inetd.conf*(4)). **inetd** runs **ftpd** when a service request is received at the port indicated in the **ftp** service specification in **/etc/services** (see *services*(4)).

Options

ftpd recognizes the following options and command-line arguments.

-l Causes each FTP session to be logged in the syslog file. For anonymous FTP sessions, other information is also logged in the syslog file. This information includes what files are stored and retrieved and what directories are created.

-p The default action of ftpd does not allow usage of reserved ports as the originating port on the client's system i.e., the PORT command cannot specify a reserved port. This option allows the client to specify a reserved port. Note, allowing usage of reserved ports can result in the misuse of ftpd. The security ramifications should be understood before the option is turned on.

-v Logs other information in the syslog file. This information is what is normally logged for anonymous FTP sessions. This information includes what files are stored and retrieved and what directories are created.

-t *timeout* Causes **ftpd** to timeout inactive sessions after *timeout* seconds. By default, **ftpd** terminates an inactive session after 15 minutes.

-P Enables third party transfer.

-T *maxtimeout*
A client can also request a different timeout period. The **-T** option sets to *maxtimeout* the maximum timeout that client can request, in seconds. By default, the maximum timeout is 2 hours.

-u *umask* Change default **ftpd** umask from 027 to *umask*.

-A Applicable only in a secure environment based on Kerberos V5. Causes access to be denied if network authentication fails. See *sis*(5).

-B *size* Sets the buffer size of the data socket to *size* blocks of 1024 bytes. The valid range for *size* is from 1 to 64 (default is 56). *NOTE*: A large buffer size will improve the performance of **ftpd** on fast links (e.g. FDDI), but may cause long connection times on slow links (e.g. X.25).

ftpd currently supports the following commands (uppercase and lowercase are interpreted as equivalent):

Command	**Description**
ABOR	Abort previous command
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	Append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	Give list files in a directory (**ls -l**)
MKD	Make a directory
MDTM	Show last modification time of file
MODE	Specify data transfer *mode*
NLST	Give name list of files in directory
NOOP	Do nothing
PASS	Specify password
PASV	Prepare for server-to-server transfer

PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
REST	Restart incomplete transfer
RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from file name
RNTO	Specify rename-to file name
SITE	Non-standard commands (see next section)
SIZE	Return size of file
STAT	Return status of server
STOR	Store a file
STOU	Store a file with a unique name
STRU	Specify data transfer *structure*
SYST	Show operating system type of server system
TYPE	Specify data transfer *type*
USER	Specify user name
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The following commands are supported when **ftpd** is operating in a secure environment which is based on Kerberos V5 (see *sis*(5)).

Command	**Description**
AUTH	Authentication/security mechanism
ADAT	Authentication/security data
CCC	Clear command channel
ENC	Privacy protected command
MIC	Integrity protected command
PROT	Data channel protection level (level 'C' only)
PBSZ	Protection buffer size (has no effect)

These commands are described in draft 8 of the FTP security extensions.

The following non-standard or HP-UX specific commands are supported by the **SITE** command:

Command	**Description**
UMASK	Change umask. (e.g., **SITE UMASK 002**)
IDLE	Set idle-timer. (e.g., **SITE IDLE** 60)
CHMOD	Change mode of a file. (e.g., **SITE CHMOD 755** filename)
HELP	Give help information. (e.g., **SITE HELP**)

The remaining FTP requests specified in Internet RFC 959 are recognized, but not implemented. **MDTM** and **SIZE** are not specified in RFC 959, but are expected in the next updated **FTP RFC**.

The FTP server aborts an active file transfer only when the **ABOR** command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959. If **ftpd** receives a **STAT** command during a data transfer, preceded by a Telnet IP and Synch, it returns the status of the transfer.

ftpd interprets file names according to the "globbing" conventions used by *csh*(1). This allows users to utilize the metacharacters *****, **.**, **[**, **]**, **{**, **}**, **~**, and **?**.

ftpd authenticates users according to three rules:

- The user name must be in the password data base, **/etc/passwd**, and not have a null password. The client must provide the correct password for the user before any file operations can be performed.
- The user name must not appear in the file **/etc/ftpusers** (see *ftpusers*(4)).
- The user must have a standard shell returned by **getusershell()**.

Optionally, a system administrator can permit public access or "anonymous FTP." If this has been set up, users can access the anonymous FTP account with the user name **anonymous** or **ftp** and any non-null password (by convention, the client host's name). **ftpd** does a **chroot()** to the home directory of user

ftp, thus limiting anonymous FTP users' access to the system. If the user name is **anonymous** or **ftp**, an anonymous FTP account must be present in the password file (user **ftp**). In this case the user is allowed to log in by specifying any password (by convention this is given as the user's e-mail address).

In order to permit anonymous FTP, there must be an entry in the *passwd*(4) database for an account named **ftp**. The password field should be *****, the group membership should be **guest**, and the login shell should be **/usr/bin/false**. For example (assuming the **guest** group ID is **10**):

```
    ftp:*:500:10:anonymous ftp:/home/ftp:/usr/bin/false
```

The anonymous FTP directory should be set up as follows:

˜ftp The home directory of the FTP account should be owned by user **root** and mode 555 (not writable). Since **ftpd** does a **chroot()** to this directory, it must have the following subdirectories and files:

~ftp/usr/bin
This directory must be owned by root and mode 555 (not writable). The file **/sbin/ls** should be copied to **˜ftp/usr/bin.** This is needed to support directory listing by **ftpd**. The command should be mode 111 (executable only). If the FTP account is on the same file system as **/sbin**, **˜ftp/usr/bin/ls** can be hard link, but it may not be a symbolic link, because of the **chroot()**. The command must be replaced when the system is updated.

~ftp/etc
This directory must be owned by root and mode 555 (not writable). It should contain versions of the files *passwd*, *group*, and *logingroup*. See *passwd*(4) and *group*(4). These files must be owned by root and mode 444 (readable only). These are needed to map user and group ids in the **LIST** command, and to support (optional) sub-logins of anonymous FTP. Sub-logins can sometimes be used to allow access to particular files by only specific remote users (who know the sub-login password) without giving those remote users logins on the system. A sub-login user would access the system via anonymous FTP, then use **USER** and **PASS** to change to the sub-login user.

~ftp/etc/passwd
This file should contain entries for the **ftp** user and any other users who own files under the anonymous **ftp** directory. Such entries should have ***** for passwords. **~ftp/etc/passwd** should also contain entries for any desired anonymous FTP sub-logins. The sub-logins must have passwords, which must be encrypted as in *passwd*(4). Group IDs must be listed in the anonymous FTP group file, **˜ftp/etc/group**. The path names of home directories in **˜ftp/etc/passwd** must be with respect to the anonymous FTP home directory. A sub-login home directory should be owned by the sub-login user ID. The shell field is ignored, and can be empty.

For example, the anonymous FTP sub-login name **subftp** would have an entry in the FTP **passwd** file that resembles:

```
subftp:bAg6vI82aq5Yt:501:10:ftp sub-login:/subftp:
```

FTP sub-login IDs do not need to be present in the system **/etc/passwd** file. Assuming the anonymous FTP directory is **/home/ftp**, the sub-login home directory in the example would be created by user **root** as follows:

```
    cd /home/ftp
    mkdir subftp
    chmod 700 subftp
    chown 501 subftp
    chgrp guest subftp
```

File **˜ftp/etc/group** should contain the group names associated with any group IDs in file **˜ftp/etc/passwd** and any group IDs of files in the anonymous FTP subdirectories. In the above example, **˜ftp/etc/group** would require an entry for **guest**, and the associated group ID would have to be the same as in the system's **/etc/group** file.

~ftp/etc/logingroup
Permits anonymous ftp sub-logins to be members of multiple groups. Can be a hard link to FTP **˜ftp/etc/group**.

~ftp/pub (optional)
This directory is used by anonymous FTP users to deposit files on the system. It should be owned by user **ftp** and should be mode 777 (readable and writable by all).

~ftp/dist (optional)
Directories used to make files available to anonymous ftp users should be mode 555 (not writable), and any files to be distributed should be owned by root and mode 444 (readable only) so that they cannot be modified or removed by anonymous FTP users.

DIAGNOSTICS

ftpd replies to FTP commands to ensure synchronization of requests and actions during file transfers, and to indicate the status of **ftpd**. Every command produces at least one reply, although there may be more than one. A reply consists of a three-digit number, a space, some text, and an end of line. The number is useful for programs; the text is useful for users. The number must conform to this standard, but the text can vary.

The first digit of the message indicates whether the reply is good, bad, or incomplete. Five values exist for the first digit. The values and the interpretations of the values are:

1 The requested action is being initiated; expect another reply before proceeding with a new command.

2 The requested action is complete. The server is ready for a new request.

3 The command has been accepted, but the requested action requires more information.

4 The command was not accepted, the requested action failed, but the error condition is temporary and the action can be requested again.

5 The command was not accepted, the requested action failed, and the error condition would most likely occur again if the same command sequence is repeated.

The second digit indicates the functional area that the message addresses. The values of the second digit and the interpretations of these values are:

0 Syntax. A message with a 0 for the second digit indicates that a syntax error occurred.

1 Information. A message with a 1 as the second digit indicates that the message is in reply to a request for information.

2 Connections. A message with a 2 as the second digit indicates that the message is a reply to a request for control and data connection information.

3 Authentication and accounting. A message with a 3 as the second digit indicates that the message is a reply to a login or accounting procedure.

4 Not currently specified.

5 File system. A message with a 5 as the second digit indicates that the text following the number contains information concerning the status of the server file system.

The third digit provides a further clarification of the information supplied by the second digit. Following are several examples of messages. Note that **ftpd**'s replies match the number but not the text.

110	Restart marker reply. MARK *yyyy*=*mmmm* where *yyyy* is a user process data stream marker, and *mmmm* is **ftpd**'s equivalent marker
120	Service ready in *nnn* minutes
200	Command okay
211	System status, or system help reply
212	Directory status
230	User logged in, proceed
250	Requested file action okay, completed
331	User name okay, need password
350	Requested file action pending further information
425	Cannot open data connection
451	Requested action aborted: local error in processing
500	Syntax error, command unrecognized or command line too long
530	Not logged in
550	Requested action not taken; file unavailable, not found, no access

WARNINGS

The password is sent unencrypted through the socket connection.

Anonymous FTP is inherently dangerous to system security.

DEPENDENCIES

Pluggable Authentication Modules (PAM)

PAM is an Open Group standard for user authentication, password modification, and validation of accounts. In particular, **`pam_authenticate()`** is invoked to perform all functions related to login. This includes retrieving the password, validating the account, and displaying error messages.

AUTHOR

`ftpd` was developed by the University of California, Berkeley.

SEE ALSO

ftp(1), inetd(1M), chroot(2), getusershell(3C), ftpusers(4), group(4), inetd.conf(4), passwd(4), pam_authenticate(3), sis(5).

NAME

fuser - list processes using a file or file structure

SYNOPSIS

/usr/sbin/fuser [**-c**|**-f**] [**-ku**] *file* ... [[**-**] [**-c**|**-f**] [**-ku**] *file* ...] ...

DESCRIPTION

The **fuser** command lists the process IDs of processes that have each specified *file* open. For block special devices, all processes using any file on that device are listed. The process ID can be followed by a letter, identifying how the *file* is being used.

c *file* is its current directory.

r *file* is its root directory, as set up by the **chroot** command (see *chroot*(1M)).

o It has *file* open.

m It has *file* memory mapped.

t *file* is its text file.

f

Options

You can specify the following options:

-c Display the use of a mount point and any file beneath that mount point. Each *file* must be a file system mount point.

-f Display the use of the named file only, not the files beneath it if it is a mounted file system.

-u Display the login user name in parentheses following each process ID.

-k Send the **SIGKILL** signal to each process using each *file*.

You can re-specify options between groups of files. The new set of options replaces the old set. A dash (**-**) by itself cancels all options currently in force.

The process IDs associated with each file are printed to standard output as a single line separated by spaces and terminated with a single newline. All other output — the file name, the letter, and the user name — is written to standard error.

You must be superuser to use **fuser**.

NETWORKING FEATURES

You can use **fuser** with NFS file systems or files. If the file name is in the format used in **/etc/mnttab** to identify an NFS file system, **fuser** will treat the NFS file system as a block special device and identify any process using that file system.

If contact with an NFS file system is lost, **fuser** will fail, since contact is required to obtain the file system identification. Once the NFS file system is re-contacted, stale file handles from the previous contact can be identified, provided that the NFS file system has the same file system identification.

EXAMPLES

Terminate all processes that are preventing disk drive 1 from being unmounted, listing the process ID and login name of each process being killed.

```
fuser -ku /dev/dsk/c201d1s?
```

List process IDs and login names of processes that have the password file open.

```
fuser -u /etc/passwd
```

Combine both the above examples into a single command line.

```
fuser -ku /dev/dsk/c201d1s? - -u /etc/passwd
```

If the device **/dev/dsk/c201d1s7** is mounted on directory **/home**, list the process IDs and login names of processes using the device. Alternately, if **/home** is the mount point for an NFS file system, list process IDs and login names of processes using that NFS file system.

```
fuser -cu /home
```

If **machine1:/filesystem/2mount** is an NFS file system, list all processes using any file on that file system. If it is not an NFS file system, treat it as a regular file.

```
fuser machine1:/filesystem/2mount
```

SEE ALSO

ps(1), mount(1M), kill(2), signal(2).

STANDARDS CONFORMANCE

`fuser`: SVID2, SVID3

NAME

fwtmp, wtmpfix - manipulate connect accounting records

SYNOPSIS

/usr/sbin/acct/fwtmp [**-ic**]

/usr/sbin/acct/wtmpfix [*files*]

DESCRIPTION

fwtmp

fwtmp reads from the standard input and writes to the standard output, converting binary records of the type found in **wtmp** to formatted ASCII records. The ASCII version is useful to enable editing, via *ed*(1), bad records or general purpose maintenance of the file.

The argument **-ic** is used to denote that input is in ASCII form, and output is to be written in binary form. (The arguments **i** and **c** are independent, respectively specifying ASCII input and binary output, thus **-i** is an ASCII to ASCII copy and **-c** is a binary to binary copy).

wtmpfix

wtmpfix examines the standard input or named files in **wtmp** format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A **-** can be used in place of *files* to indicate the standard input. If time/date corrections are not performed, *acctcon1* will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records is written to **/var/adm/wtmp**. The first record is the old date denoted by the string **old time** placed in the line field and the flag **OLD_TIME** placed in the type field of the <**utmp.h**> structure. The second record specifies the new date, and is denoted by the string **new time** placed in the line field and the flag **NEW_TIME** placed in the type field. *wtmpfix* uses these records to synchronize all time stamps in the file. *wtmpfix* nullifies date change records when writing to the standard output by setting the time field of the <**utmp.h**> structure in the old date change record equal to the time field in the new date change record. This prevents *wtmpfix* and *acctcon1* from factoring in a date change record pair more than once.

In addition to correcting time/date stamps, *wtmpfix* checks the validity of the name field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it changes the login name to **INVALID** and write a diagnostic to the standard error. This minimizes the risk that *acctcon1* will fail when processing connect accounting records.

DIAGNOSTICS

wtmpfix generates the following diagnostics messages:

Cannot make temporary: xxx failed to make temp file
Input truncated at offset: xxx missing half of date pair
New date expected at offset: xxx missing half of date pair
Cannot read from temp: xxx some error reading
Bad file at offset: xxx ut_line entry not digit, alpha, nor | or **{** (first character only checked)
Out of core: *malloc* fails. (Saves table of date changes)
No dtab: software error (rarely seen, if ever)

FILES

/usr/include/utmp.h
/var/adm/wtmp

SEE ALSO

acct(1M), acctcms(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), ed(1), runacct(1M), acct(2), acct(4), utmp(4).

BUGS

fwtmp generates no errors, even on garbage input.

STANDARDS CONFORMANCE

fwtmp: SVID2, SVID3

wtmpfix: SVID2, SVID3

NAME
gated - gateway routing daemon

SYNOPSIS
gated [**-c**] [**-C**] [**-n**] [**-N**] [**-t***trace_options*] [**-f** *config_file*] [*trace_file*]

DESCRIPTION
gated is a routing daemon that handles multiple routing protocols and replaces routed, egpup, and any routing daemon that speaks the HELLO routing protocol. **gated** currently handles the RIP, BGP, EGP, HELLO, and OSPF routing protocols. The **gated** process can be configured to perform all routing protocols or any subset of them (see *WARNINGS* below).

Options
The command-line options are:

-c Specifies that the configuration file will be parsed for syntax errors and then **gated** will exit. **gated** will leave a dump file in **/var/tmp/gated_dump** if there were no errors. **gated** does not need to be run as the superuser to use the **-c** option but it may not be possible to read the kernel forwarding table and interface configuration if not run as superuser. The **-c** option implies **-tgeneral**. All *trace_option* clauses in the configuration file will be ignored.

-C Specifies that the configuration file will just be parsed for syntax errors. **gated** will exit with a status 1 if there were any errors and 0 (zero) if there were not. **gated** does not need to be run as the superuser to use the **-C** option but it may not be possible to read the kernel forwarding table and interface configuration if not run as the superuser.

-n Specifies that **gated** will not modify the kernel forwarding table. This is used for testing **gated** configurations with actual routing data.

-N Specifies that **gated** will not daemonize. Normally, if tracing to stderr is not specified **gated** will daemonize if the parent process ID is not 1. This allows the use of an **/etc/inittab**-like method of invoking **gated** that does not have a PID of 1.

-t*trace_options*

Specifies a comma separated list of trace options to be enabled on startup. If no flags are specified, **general** is assumed. No space is allowed between this option and it's arguments.

This option must be used to trace events that take place before the configuration file is parsed, such as determining the interface configuration and reading routes from the kernel.

See the *GateD Configuration Guide* for valid trace options and a more detailed explanation of tracing.

-f *config_file*

Use an alternate config file. By default, **gated** uses **/etc/gated.conf**.

trace_file Trace file in which to place trace information.

If a trace file is specified on the command line, or no trace flags are specified on the command line, **gated** detaches from the terminal and runs in the background. If trace flags are specified without specifying a trace file, **gated** assumes that tracing is desired to stderr and remains in the foreground.

Signal Processing
The following signals may be used to control **gated**:

SIGHUP Re-read configuration. A **SIGHUP** causes **gated** to reread the configuration file. **gated** first performs a clean-up of all allocated policy structures. All BGP and EGP peers are flagged for deletion and the configuration file is re-parsed.

If the re-parse is successful, any BGP and EGP peers that are no longer in the configuration are shut down, and new peers are started. **gated** attempts to determine if changes to existing peers require a shutdown and restart. OSPF is not capable of reconfiguring; it is shutdown and restarted during a reconfiguration. This may have an adverse impact on the routing system.

It should also be possible to enable/disable any protocol without restarting **gated**.

SIGINT Snap-shot of current state.

The current state of all **gated** tasks, timers, protocols and tables are written to **/var/tmp/gated_dump**.

On systems supporting **fork()**, this is done by forking a subprocess to dump the table information so as not to impact **gated**'s routing functions. On systems where memory management does not support copy-on-write, this will cause the **gated** address space to be duplicated; this may cause a noticeable impact on the system. On system not supporting **fork()**, the main process immediately processes the dump, which may impact **gated**'s routing functions.

SIGTERM Graceful shutdown.

On receipt of a **SIGTERM**, **gated** attempts a graceful shutdown. All tasks and protocols are asked to shutdown. Most will terminate immediately, the exception being EGP peers which wait for confirmation. It may be necessary to repeat the **SIGTERM** once or twice if it this process takes too long.

All protocol routes are removed from the kernel's routing table on receipt of a **SIGTERM**. Interface routes, routes with RTF_STATIC set (from the route command where supported) and static routes specifying **retain** will remain. To terminate **gated** with the exterior routes intact, use **SIGKILL**.

SIGUSR1 Toggle tracing.

On receipt of a **SIGUSR1**, **gated** will close the trace file. A subsequent **SIGUSR1** will cause it to be reopened. This will allow the file to be moved regularly.

It is not possible to use **SIGUSR1** if a trace file has not been specified, or tracing is being performed to stderr.

SIGUSR2 Check for interface changes.

On receipt of a **SIGUSR2**, **gated** will rescan the kernel interface list looking for changes.

WARNINGS

gated contains provisions for BGP protocol, but it is not officially supported by HP at the present time. Some RIP version 2 features (RFC1388) are not currently supported: MIB and route tag. The optional OSPF version 2 (RFC1247) feature of TOS (type of service) based routing is not supported. The route aggregation, generating a more general route from compressing the specific routes through the explicit configuration, is not supported in this release.

AUTHORS

gated was primarily developed by Cornell University which includes code from the Regents of the University of California and the University of Maryland.

This software and associated documentation is Copyright 1990, 1991, 1992 by Cornell University.

SEE ALSO

gated.conf(4), arp(1M), fork(2), gdc(1M), ifconfig(1M), netstat(1), ospf_monitor(1M), ripquery(1M), *GateD Documentation*, *GateD Configuration Guide*.

RFC 891	DCN Local-Network Protocols (HELLO)
RFC 904	Exterior Gateway Protocol Formal Specification
RFC 1058	Routing Information Protocol
RFC 1163	A Border Gateway Protocol (BGP)
RFC 1164	Application of the Border Gateway Protocol in the Internet
RFC 1247	OSPF Specification, Version 2.

NAME

gdc - operational user interface for gated

SYNOPSIS

gdc [**-q**] [**-n**] [**-c** *coresize*] [**-f** *filesize*] [**-m** *datasize*] [**-s** *stacksize*] [**-t** *seconds*] *command*

DESCRIPTION

gdc provides a user-oriented interface for the operation of the *gated*(1M) routing daemon. It provides support for starting and stopping the daemon, for the delivery of signals to manipulate the daemon when it is operating, for the maintenance and syntax checking of configuration files, and for the production and removal of state dumps and core dumps.

gdc can reliably determine **gated**'s running state and produces a reliable exit status when errors occur, making it advantageous for use in shell scripts which manipulate **gated**. Commands executed using **gdc** and, optionally, error messages produced by the execution of those commands, are logged via the same *syslogd*(1M) facility which **gated** itself uses, providing an audit trail of operations performed on the daemon.

If installed as a setuid root program **gdc** will allow non-root users who are members of a trusted group (by default the **gdmaint** group) to manipulate the routing daemon while denying access to others. The name of the user is logged along via *syslogd*(1M) along with an indication of each command executed, for audit purposes.

The command-line options are:

-n	Run without changing the kernel forwarding table. Useful for testing, and when operating as a route server which does no forwarding.
-q	Run quietly. With this option informational messages which are normally printed to the standard output are suppressed and error messages are logged via *syslogd*(1M) instead of being printed to the standard error output. This is often convenient when running **gdc** from a shell script.
-t *seconds*	Specifies the time in seconds which **gdc** will spend waiting for **gated** to complete certain operations, in particular at termination and startup. By default this value is set to 10 seconds.

These additional command-line options may be present, depending on the options used to compile **gdc**:

-c *coresize*	Sets the maximum size of a core dump a **gated** started with **gdc** will produce. Useful on systems where the default maximum core dump size is too small for **gated** to produce a full core dump on errors.
-f *filesize*	Sets the maximum file size a **gated** started with **gdc** will produce. Useful on systems where the default maximum file dump size is too small for **gated** to produce a full state dump when requested.
-m *datasize*	Sets the maximum size of the data segment of a **gated** started with **gdc**. Useful on systems where the default data segment size is too small for **gated** to run.
-s *stacksize*	Sets the maximum size of stack of a **gated** started with **gdc**. Useful on systems where the default maximum stack size is too small for **gated** to run.

The following commands cause signals to be delivered to **gated** for various purpose:

COREDUMP	Sends an abort signal to **gated**, causing it to terminate with a core dump.
dump	Signal **gated** to dump its current state into the file **/usr/tmp/gated_dump**.
interface	Signal **gated** to recheck the interface configuration. **gated** normally does this periodically in any event, but the facility can be used to force the daemon to check interface status immediately when changes are known to have occurred.
KILL	Cause **gated** to terminate ungracefully. Normally useful when the daemon has hung.
reconfig	Signal **gated** to reread its configuration file, reconfiguring its current state as appropriate.
term	Signal **gated** to terminate after shutting down all operating routing protocols gracefully. Executing this command a second time should cause **gated** to terminate even if some protocols have not yet fully shut down.

toggletrace
: If **gated** is currently tracing to a file, cause tracing to be suspended and the trace file to be closed. If **gated** tracing is current suspended, cause the trace file to be reopenned and tracing initiated. This is useful for moving trace files.

By default **gated** obtains its configuration from a file normally named **/etc/gated.config**. The **gdc** program also maintains several other versions of the configuration file, in particular named:

/etc/gated.conf+ The *new* configuration file. When **gdc** is requested to install a new configuration file, this file is renamed **/etc/gated.conf**.

/etc/gated.conf- The *old* configuration file. When **gdc** is requested to install a new configuration file, the previous **/etc/gated.conf** is renamed to this name.

/etc/gated.conf-- The *really old* configuration file. **gdc** retains the previous *old* configuration file under this name.

The following commands perform operations related to configuration files:

checkconf Check **/etc/gated.conf** for syntax errors. This is usefully done after changes to the configuration file but before sending a **reconfig** signal to the currently running **gated**, to ensure that there are no errors in the configuration which would cause the running **gated** to terminate on reconfiguration. When this command is used, **gdc** issues an informational message indicating whether there were parse errors or not, and if so saves the error output in a file for inspection.

checknew Like **checkconf** except that the *new* configuration file, **/etc/gated.conf+**, is checked instead.

newconf Move the **/etc/gated.conf+** file into place as **/etc/gated.conf**, retaining the older versions of the file as described above. **gdc** will decline to do anything when given this command if the *new* configuration file doesn't exist or otherwise looks suspect.

backout Rotate the configuration files in the **newer** direction, in effect moving the *old* configuration file to **/etc/gated.conf**. The command will decline to perform the operation if **/etc/gated.conf-** doesn't exist or is zero length, or if the operation would delete an existing, non-zero length **/etc/gated.conf+** file.

BACKOUT Perform a **backout** operation even if **/etc/gated.conf+** exists and is of non-zero length.

modeconf Set all configuration files to mode 664, owner root, group gdmaint. This allows a trusted non-root user to modify the configuration files.

createconf If **/etc/gated.conf+** does not exist, create a zero length file with the file mode set to 664, owner root, group gdmaint. This allows a trusted non-root user to install a new configuration file.

The following commands provide support for starting and stopping **gated**, and for determining its running state:

running Determine if **gated** is currently running. This is done by checking to see if **gated** has a lock on the file containing its pid, if the pid in the file is sensible and if there is a running process with that pid. Exits with zero status if **gated** is running, non-zero otherwise.

start Start **gated**. The command returns an error if **gated** is already running. Otherwise it executes the **gated** binary and waits for up to the delay interval (10 seconds by default, as set with the **-t** option otherwise) until the newly started process obtains a lock on the pid file. A non-zero exit status is returned if an error is detected while executing the binary, or if a lock is not obtained on the pid file within the specified wait time.

stop Stop **gated**, gracefully if possible, ungracefully if not. The command returns an error (with non-zero exit status) if **gated** is not currently running. Otherwise it sends a terminate signal to **gated** and waits for up to the delay interval (10 seconds by default, as specified with the **-t** option otherwise) for the process to exit. Should **gated** fail to exit within the delay interval it is then signaled again with a second terminate signal. Should it fail to exit by the end of the second delay interval it is signaled for a third time with a kill signal. This should force immediate termination unless something is very broken. The command terminates with zero exit status when it detects that **gated** has terminated, non-zero otherwise.

restart	If **gated** is running it is terminated via the same procedure as is used for the **stop** command above. When the previous **gated** terminates, or if it was not running prior to command execution, a new **gated** process is executed using the procedures described for the **start** command above. A non-zero exit status is returned if any step in this procedure appears to have failed.

The following commands allow the removal of files created by the execution of some of the commands above:

rmcore	Removes any existing **gated** core dump file.
rmdump	Removes any existing **gated** state dump file.
rmparse	Removes the parse error file generated when a **checkconf** or **checknew** command is executed and syntax errors are encountered in the configuration file being checked.

FILES

Many of default filenames listed below contain the string %s, which is replaced by the name with which gated is invoked. Normally this is **gated**, but if invoked as **gated-test**, **gated** will by default look for **/etc/gated-test.conf**. These paths may all be changed at compilation time.

/usr/sbin/gated	The **gated** binary.
/etc/gated.conf	Current **gated** configuration file.
/etc/gated.conf+	Newer configuration file.
/etc/gated.conf-	Older configuration file.
/etc/gated.conf--	Much older configuration file.
/var/run/gated.pid	Where **gated** stores its pid.
/var/tmp/gated_dump	**gated**'s state dump file.
/var/tmp/gated_parse	Where config file parse errors go.
/var/tmp	Where **gated** drops its core file.

AUTHOR

gdc was developed by Dennis Ferguson and Cornell University.

SEE ALSO

gated(1M), ospf_monitor(1M), ripquery(1M), syslogd(1M), gated.conf(4), *GateD Documentation*, *GateD Configuration Guide*.

BUGS

Many commands only work when **gated** is installed in the system directory it was configured with.

There is not yet any way to tell **gdc** about systems which name their core dump other than **core** (**core.gated** is a less common possibility).

NAME

geocustoms - configure system language on multi-language systems

SYNOPSIS

geocustoms [**-l** *locale*]

DESCRIPTION

The **geocustoms** utility manages default selection and retention/removal of multiple languages installed on ignited systems. The geocustoms program is executed at first boot on ignited (Instant Ignition) systems with multiple languages available. On subsequent sessions, the command **/usr/sbin/geocustoms** starts **geocustoms**.

Options:

-l *locale* Sets the **LANG** variable (and all other appropriate dependencies, if applicable) to the value of *locale*. If the *locale* argument is not a valid option for that system, the User Interface (UI) will appear as if the option had not been used.

An additional locale value can be used in this context; **SET_NULL_LOCALE** can be the argument to the **-l** option, the result of which will be setting locale variables to **NULL** by default. A null locale will allow programs to execute without using localized message catalogs. This can increase system performance. All HP-UX messages appear in English if the locale is set to **NULL**.

EXTERNAL INFLUENCES

Environment Variables

geocustoms writes default values to system configuration files regarding the following environmental variables: **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_COLLATE**, **LC_MONETARY**, **LC_NUMERIC**, **LC_TIME**, **LC_MESSAGES**.

International Code Set Support

Native Language Support (NLS):
If the standard message catalogs exist, then they are in **/usr/lib/nls**. The **geocustoms** command will use the standard message catalogs, if they are on the system. If the standard message catalogs are not on the system, then the messages appear in English. (This is in accordance with standard NLS behavior). All European languages for CDE will be supported. For HP-UX 10.30, this includes English, French, German, Italian, Spanish and Swedish. All prompts and logging messages will be localized.

Locale (Language Variant) names are always localized in accordance with standard NLS behavior.

NLS is extended to allow multiple "fonts" on the initial screen at the same time through use of bitmapped images.

RETURN VALUES

0 Successful completion and/or clean exit from program.

1 Program was unable to complete all objectives.

DIAGNOSTICS

Errors:

geocustoms writes to **stderr**, and to **/var/adm/sw/lang.log**.

Standard Output

geocustoms does not write to **stdout**.

Standard Error

geocustoms only writes to **stderr** in case of command line error or request for syntax. Any UI error messages appear via an error window.

Logging

Both interactive and non-interactive sessions log summary events at:

/var/adm/sw/lang.log.

EXAMPLES

To set the default system language non-interactively to German:

```
/usr/sbin/geocustoms -l de_DE.iso88591
```

DEPENDENCIES

ObAM 4.2

SD-UX 10.30

HP-UX 10.30

Compatability

This product is designed for compatibility with HP-UX 10.30 with a Common Desktop Environment (CDE). No attempt has been made to support the Visual User Environment (VUE).

Notes

If geocustoms is invoked by the user, it may be necessary to log out and log in again for language changes to take effect.

If language bundles have been marked for removal, that will occupy the **swagentd()** for some minutes at the next system boot.

Limitations

geocustoms does *not* do the following:

- Manage languages at the codeset level.
- Provide a user interface for Asian languages.
- Manage keyboard selection.
- Create or remove locale definitions.
- Provide a special interface for restoring or adding languages to the system from separate media.

AUTHOR

geocustoms was developed by HP.

FILES:

geocustoms creates a text file **/var/adm/sw/lang.log**.

geocustoms creates, if necessary, and modifies **/etc/dt/config/Xconfig** and **/etc/rc.config.d/LANG**.

geocustoms will read NLS files, as discussed in Native Language Support above.

SEE ALSO:

locale(1), swinstall(1M), swlist(1M), swremove(1M), setlocale(3C).

STANDARDS CONFORMANCE

POSIX.2, UNIX95 (SPEC1170 and XPG4).

NAME

getext (vxfs) - get extent attributes

SYNOPSIS

/usr/sbin/getext [**-F vxfs**] [**-V**] [**-f**] [**-s**] *file*...

DESCRIPTION

The **getext** command displays extent attribute information associated with a set of files.

Options

-F vxfs	Specifies the VxFS file system type.
-V	Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options. This option allows the user to verify the command line.
-f	Do not print the filenames for which extent attributes are displayed.
-s	Do not print output for files that do not have fixed extent sizes or reservations.

g

OUTPUT

file1: **Bsize 1024 Reserve 36 Extent Size 3 align noextend**

The above line indicates a file with 36 blocks of reservation, a fixed extent size of 3 blocks, all extents aligned to 3 block boundaries, and the file cannot be extended once the current reservation is exhausted. In this case, the file system block size is 1024 bytes. Reservation and fixed extent sizes are allocated in units of the file system block size.

NOTES

Only the **align** and **noextend** allocation flags (set through **setext(1M)** or the VX_SETEXT ioctl) are persistent attributes of the file and therefore visible via **getext** or the VX_GETEXT ioctl. **trim** is also visible, although it is cleared and the reservation is reduced on the final close of the file.

SEE ALSO

setext(1M), vxfsio(7).

NAME

getty - set terminal type, modes, speed, and line discipline

SYNOPSIS

/usr/sbin/getty [**-h**] [-t *timeout*] *line* [*speed* [*type* [*linedesc*]]]

/usr/sbin/getty -c *file*

DESCRIPTION

getty is a program that is invoked by *init*(1M). It is the second process in the series, (*init-getty-login-shell*) that ultimately connects a user with the HP-UX system. Initially, if **/etc/issue** exists, *getty* prints its contents to the user's terminal, followed by the login message field for the entry it is using from **/etc/gettydefs**. *getty* reads the user's login name and invokes the *login*(1) command with the user's name as argument. While reading the name, *getty* attempts to adapt the system to the speed and type of terminal being used.

Configuration Options and Arguments

getty recognizes the following arguments:

line Name of a tty line in **/dev** to which *getty* is to attach itself. *getty* uses this string as the name of a file in the **/dev** directory to open for reading and writing. By default *getty* forces a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed. However, when *getty* is run on a direct port, *getty* does not force a hangup on the line since the driver ignores changes to zero speed on ports open in direct mode (see *modem*(7)).

-h Tells *getty* not to force a hangup on the line before setting the speed to the default or specified speed.

-t *timeout* *getty* exits if the open on the line succeeds and no one types anything within *timeout* seconds.

speed A label to a speed and tty definition in the file **/etc/gettydefs**. This definition tells *getty* at what speed to initially run, what the login message should look like, what the initial tty settings are, and what speed to try next should the user indicate that the speed is inappropriate (by typing a *break* character). The default *speed* is 300 baud.

type A character string describing to *getty* what type of terminal is connected to the line in question. *getty* understands the following types:

none	default
vt61	DEC vt61
vt100	DEC vt100
hp45	Hewlett-Packard HP2645
c100	Concept 100

The default terminal is **none**; i.e., any crt or normal terminal unknown to the system. Also, for terminal type to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled in the default condition.

linedesc A character string describing which line discipline to use when communicating with the terminal. Hooks for line disciplines are available in the operating system, but there is only one presently available — the default line discipline, **LDISC0**.

When given no optional arguments, *getty* sets the *speed* of the interface to 300 baud, specifies that raw mode is to be used (awaken on every character), that echo is to be suppressed, either parity allowed, new-line characters will be converted to carriage return-line feed, and tab expansion performed on the standard output. It types the login message before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pushing the "break" key. This causes *getty* to attempt the next *speed* in the series. The series that *getty* tries is determined by what it finds in **/etc/gettydefs**.

The user's name is terminated by a new-line or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see *ioctl*(2)).

The user's name is scanned to see if it contains any lowercase alphabetic characters; if not, and if the name is non-empty, the system is told to map any future uppercase characters into the corresponding lowercase characters.

getty also understands the "standard" ESS2 protocols for erasing, killing and aborting a line, and terminating a line. If *getty* sees the ESS erase character, _, or kill character, **$**, or abort character, **&**, or the ESS line terminators, **/** or **!**, it arranges for this set of characters to be used for these functions.

Finally, *login* is called with the user's name as an argument. Additional arguments can be typed after the login name. These are passed to *login*, which places them in the environment (see *login*(1)).

Check Option

A check option is provided. When *getty* is invoked with the **-c** option and *file*, it scans *file* as if scanning **/etc/gettydefs** and prints the results on the standard output. If there are any unrecognized modes or improperly constructed entries, *getty* reports these. If the entries are correct, *getty* prints out the values of the various flags. See *ioctl*(2) for an interpretation of values. Note that some values are added to the flags automatically.

DEPENDENCIES

HP 2334 MultiMux:

The modem control parameter *MRTS* must be present in the **/etc/gettydefs** file when using *getty* in conjunction with an HP 2334 or HP 2335 MultiMux to ensure that the RTS modem control signal is asserted correctly.

Example:

9600# B9600 HUPCL PARENB **MRTS** # B9600 SANE PARENB ISTRIP IXANY #login: #19200

MRTS is not intended for use with devices other than the HP 2334 or HP 2335 MultiMux.

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

ct(1), login(1), init(1M), ioctl(2), gettydefs(4), inittab(4), modem(7), termio(7).

BUGS

While *getty* does understand simple single character quoting conventions, it is not possible to quote the special control characters that *getty* uses to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. Therefore it is not possible to log in by means of *getty* and type a **#**, **@**, **/**, **!**, _, backspace, **^U**, **^D**, or **&** as part of your login name or arguments. They will always be interpreted as having their special meaning as described above.

NAME

getx25 - get x25 line

SYNOPSIS

/usr/sbin/getx25 *line* *speed* *pad-type*

DESCRIPTION

getx25 is functionally very similar to **getty** (see *getty*(1M)) but is used only for incoming lines that are connected to an X.25 PAD. It performs special functions such as setting up an initial PAD configuration. It also logs the number of the caller in **/var/uucp/.Log/LOGX25**. The third parameter is the name of the PAD being used. HP 2334A is the only one supported at this time. A typical invocation would be:

```
/usr/sbin/getx25 x25.1 2 HP2334A
```

AUTHOR

getx25 was developed by HP.

SEE ALSO

login(1), uucp(1), getty(1M).

NAME

groupadd - add a new group to the system

SYNOPSIS

groupadd [**-g** *gid* [**-o**]] *group*

DESCRIPTION

The **groupadd** command creates a new group on the system by adding the appropriate entry to the **/etc/group** file. The **groupadd** command expects the *group* argument, which is the name of the new group. The name consists of a string of printable characters that may not include a colon (:) or newline (\n).

Options

The **groupadd** command may be used with the following options:

-g *gid* Specifies the group ID for the new group. *gid* must be a non-negative decimal integer less than MAXUID as defined in the <**param.h**> header file. By default the next available unique group ID in the valid range is allocated. Group IDs in the range 0-99 are reserved.

-o Allow the *gid* to be non-unique (i.e., a duplicate).

NETWORKING FEATURES

The **groupadd** command is aware of NIS user entries. Only local groups may be added with this command. Attempts to add an NIS group will result in an error. NIS groups must be administered from the NIS server. If **groupadd** is used on a system where NIS is installed, it may fail with the error

```
group x is not unique
```

(return value 9) if the group specified is not present in the local **/etc/group** file, but is an NIS group (see *group*(4)). NIS groups are also checked when verifying uniqueness of the new *gid*, which may result in the error

```
GID # is not unique
```

(return value 4).

RETURN VALUE

The **groupadd** command exits with one of the following values:

0	No error.
2	Invalid command syntax.
3	Invalid argument supplied to an option.
4	*gid* is not unique (when **-o** is not used).
9	*group* is not unique.
10	Cannot modify the **/etc/group** file.
11	**/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.
12	Unable to open **/etc/ptmp** file or **/etc/passwd** file is non-existent.

EXAMPLES

Add the group **project1** to the **/etc/group** file.

```
groupadd project1
```

Add the group **project12** to the **/etc/group** file with the group ID **111** as long as no group currently exists with a group ID of **111**.

```
groupadd -g 111 project12
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **groupadd** terminates.

FILES

/etc/group
/etc/ptmp

SEE ALSO

users(1), groupdel(1M), groupmod(1M), logins(1M), useradd(1M), userdel(1M), usermod(1M), group(4).

STANDARDS CONFORMANCE

groupadd: SVID3

NAME

groupdel - delete a group from the system

SYNOPSIS

groupdel *group*

DESCRIPTION

The **groupdel** command deletes a group from the system by removing the appropriate entry from the **/etc/group** file.

The **groupdel** command must be used with the *group* argument. *group* is the name of the group to be deleted, consisting of a string of printable characters.

NETWORKING FEATURES

This command is aware of NIS user entries. Only local groups may be deleted with **groupdel**. Attempts to delete an NIS group will result in an error. NIS groups must be administered from the NIS server. If **groupdel** is used on a system where NIS is installed, it may fail with the error

group *x* **does not exist**

(return value 6), if the group specified is an NIS group (see *group*(4)).

RETURN VALUE

groupdel exits with one of the following values:

0	No error.
2	Invalid command syntax.
3	Invalid argument supplied to an option.
6	*group* does not exist.
10	Cannot modify the **/etc/group** file.
11	**/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.
12	Unable to open **/etc/ptmp** or **/etc/passwd** file is non-existent.

EXAMPLES

Delete the group **project1** from the **/etc/group** file if it exists:

```
groupdel project1
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **groupdel** terminates.

FILES

/etc/group
/etc/ptmp

SEE ALSO

users(1), groupadd(1M), groupmod(1M), logins(1M), useradd(1M), userdel(1M), usermod(1M), group(4).

STANDARDS CONFORMANCE

groupdel: SVID3

NAME

groupmod - modify a group on the system

SYNOPSIS

groupmod [**-g** *gid* [**-o**]] [**-n** *name*] *group*

DESCRIPTION

The **groupmod** command modifies a group on the system by altering the appropriate entry in the **/etc/group** file.

The **groupmod** command must be used with the *group* argument, which is the name of the group to be modified.

Options

The **groupmod** command may be used with the following options:

-g *gid*	Change the value of the group ID to *gid*. *gid* must be a non-negative decimal integer less than MAXUID as defined in the <**param.h**> header file.
-o	Allow the *gid* to be non-unique (i.e., a duplicate).
-n *name*	Change the name of the group to *name*. *name* consists of a string of printable characters that may not include a colon (:) or newline (\n).

NETWORKING FEATURES

This command is aware of NIS user entries. Only local groups may be modified with **groupmod**. Attempts to modify an NIS group will result in an error. NIS groups must be administered from the NIS server. If **groupmod** is used on a system where NIS is installed, it may fail with the error

group *x* **does not exist**

(return value 6) if the group specified is an NIS group (see *group*(4)). However, NIS groups are checked when verifying uniqueness of the new *gid* or new group name, which may result in the above error, or the error

GID # is not unique

(return value 4).

RETURN VALUES

groupmod exits with one of the following values:

0	No error.
2	Invalid command syntax.
3	Invalid argument supplied to an option.
4	*gid* is not unique (when **-o** is not used).
6	*group* does not exist.
9	*group* is not unique.
10	Cannot modify the **/etc/group** file.
11	**/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.
12	Unable to open **/etc/ptmp** file or the **/etc/passwd** file is non-existent.

EXAMPLES

Change the group ID of the group **project2** to **111** in the file **/etc/group** if the group **project2** exists. This is done even if the group ID **111** is already in use.

```
groupmod -g 111 -o project2
```

Change the name of **project2** to **project22** in the file **/etc/group** if the group **project22** does not already exist.

```
groupmod -n project22 project2
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **groupmod** terminates.

FILES

`/etc/group`
`/etc/ptmp`

SEE ALSO

users(1), groupadd(1M), groupdel(1M), logins(1M), useradd(1M), userdel(1M), usermod(1M), group(4).

STANDARDS CONFORMANCE

`groupmod`: SVID3

NAME

hosts_to_named - translate host table to name server file format

SYNOPSIS

hosts_to_named -d *domain* **-n** *network-number* [*options*]

DESCRIPTION

hosts_to_named translates the host table, **/etc/hosts**, into files that are usable by the name server *named*(1M). The format of these files is defined in RFC1035. The files are created in the current directory. Once the host table is translated, the name server files can be maintained directly, or the translation can be repeated after each change to the host table.

If a line in the host table contains no domain names, all names on the line are assumed to be in the default domain. The first *domain* listed is the "default domain". If data is being created for more than 1 domain or if certain options are used, there must be domain names in the host table to determine which names belong in which domain.

The name server data is referred to as "resource records".

Options are:

-a *network-number*
: Add the information about hosts in the local domain from network *network-number*. This is the same as the **-n** option except that no pointer (PTR) data is created. This is useful when there are multiple domains on a network and a different server is handling the address-to-name mapping for *network-number*.

-b *bootfile*
: Name the boot file *bootfile*. The default is **named.boot** in the current directory.

-c *subdomain*
: Create alias (CNAME) records for hosts in *subdomain* of the default domain. When a subdomain is delegated, it is useful to create aliases for the old names in the default domain that point to the new names in the *subdomain*. After creating the alias (CNAME) records, ignore lines in the host table that contain names in the *subdomain*. This option can be used more than once on the command line. This option requires domain names in the host table. When the old names in this *domain* are no longer used, they can be ignored with the **-e** option. If the *subdomain* name does not have dots, the default domain is appended to *subdomain*.

-d *domain*
: Create data for *domain*. This option can be used more than once on the command line if data is being created for more than 1 domain. The first *domain* listed is the "default domain". This option requires domain names in the host table for all hosts in domains except the default domain.

-e *subdomain*
: Eliminate lines from the host table that contain names in the *subdomain* before translating. If the *subdomain* name does not have dots, the default domain is appended. This option may be used more than once on the command line. This option requires domain names in the host table.

-f *file*
: Read command line options from *file*. The **-f** option is not allowed within a file.

-h *host*
: Declare *host* to be the host in the start of authority (SOA) record that the name server data was created on. Also use *host* for the electronic mail address of the responsible user in the SOA record. The default is the host this command is run on.

-m *weight:mailhub*
: For each canonical hostname from the host table, create mail exchanger (MX) records with the specified weight and mail hub. The weight is a positive integer. The mail hub is a hostname. If the mail hub name has no dots, the default domain is appended. This option can be used more than once on the command line.

-n *network-number*[*:mask*]
: Create data for *network-number*. See below for description of *network-number*. If only one *domain* is listed with **-d**, all data for *network-number* is assumed to be in *domain*. The optional subnet mask *mask* can be used instead of supplying each *network-number* for a subnet using multiple **-n** options. *mask* must be in dot notation.

-o *refresh:retry:expire:min*
: Set the values in the start-of-authority (SOA) record to those specified. See below for

description of the start-of-authority (SOA) record.

-p *domain* Create only pointer (PTR) data for hosts in *domain*. This is useful when there are multiple domains on a network and a different server is responsible for *domain*, but this server is responsible for the address-to-name mapping. This option can be used more than once on the command line. This option requires domain names in the host table.

-q Run quietly. No messages are printed.

-r Create name server data indicating that the name server is authoritative for **.** (the root of the domain tree). The file created is **db.root**. Use this only when your network is isolated from the Internet. If other root servers exist for the isolated network, they must be added manually.

-s *server* Create name server (NS) records that declare *server* is an authoritative name server for all of the domains created. If more than 1 server is authoritative, each needs to be declared. If the server name does not have any dots in it, the default domain is appended. The default server is the host this script is run on. This option can be used more than once on the command line.

-t Create text (TXT) records from the comments that appear with host data. The comments will all be in lower case because the host table is translated to lower case. If **[no smtp]** appears in a comment, it is omitted. The **[no smtp]** is used to control mail exchanger (MX) data.

-u *user* Declare *user* to be the electronic mail address of the person responsible for this domain. This is used in the start of authority (SOA) record. The format required in the name server data is *user***.***host* (host must be a domain name). If given as *user*, the host on which this script is run is appended. If given as *user@host*, the @ is replaced with a dot (**.**). The default user is **root**.

-w Create well known services (WKS) data declaring that the host provides the SMTP service. This is done only when mail exchanger (MX) data is also being created and only for hosts without **[no smtp]** in a comment.

-z *internet-address*
Create a secondary boot file, **boot.sec.save**, from the primary boot file listing *internet-address* as the server to load the data from. The boot file has the server back up the data on disk. The *internet-address* defaults to the value used with **-Z**. This option can be used more than once.

-A Do not create name server data for aliases in the host table.

-C *file* Create resource records from strings in the comment field of the host table. Each string in the comment field (except **[no smtp]**) is searched for in *file*. The format of *file* is a string, a colon, and a resource record. If the string in the comment field matches the string before the colon in *file*, a resource record is added consisting of the name of the host followed by everything after the colon from the matching line in *file*. For example, host information (HINFO) records can be created by adding **360:IN HINFO hp9000s360 hp-ux** to *file* and adding 360 to comments in the host table.

-D Do not create name server data for domain names in the host table.

-F By default, the serial number is incremented for a domain only if the data has changed (pointer (PTR) data only). This option forces the serial number to be incremented, even if the data has not changed.

-H *host-file* Use *host-file* instead of **/etc/hosts**.

-M Do not create mail exchanger (MX) records for hosts in the host table.

-N *mask* Apply the default subnet mask *mask* to each *network-number* specified with **-n** except for ones with their subnet masks already provided. *mask* must be in dot notation. This is the same as supplying each *network-number* for a subnet using multiple **-n** options.

-S *server* This option is the same as the **-s** option, but it only applies to the last *domain* specified with **-d** or the last *network-number* specified with **-n**. This option is for when *server* is backing up some, but not all, of the domains.

-Z *internet-address*
: Create a secondary boot file, **boot.sec**, from the primary boot file listing *internet-address* as the server to load the data from. The boot file does not have the server back up the data on disk. The *internet-address* defaults to value used with **-z**. This option can be used more than once.

-1
: This option is obsolete.

hosts_to_named translates the host table to lower case to help eliminate duplicate data. Since the name server treats uppercase and lowercase as equivalent, names that differ only in case are considered the same.

Alias (CNAME) records are created for *subdomains* delegated with **-c**. Lines from the host table that contain names in *subdomains* from **-c** and **-e** are removed from the lowercase copy of the host table.

The host table is then used to create the name server data for each *network-number* declared on the command line. Do not include the trailing 0's in the network number. No distinction is made between class A, B, or C addresses nor is there any understanding of subnets unless a subnet mask is supplied. Example network numbers are: 10 (for all addresses of the form 10.*.*.*), 10.1 (for addresses of the form 10.1.*.*), or 10.2.2 (for addresses of the form 10.2.2.*).

h

Address (A) records are created for mapping hostnames to IP addresses. Alias (CNAME) records are created for aliases of hosts that are not multi-homed. The data are placed in a file named **db.***DOMAIN* where *DOMAIN* is the first part of the domain from the command line. For the domain **div.inc.com**, the file is named **db.div**. All other name server data goes in this file except the pointer (PTR) records described below.

Pointer (PTR) records are created for mapping IP addresses to host names. PTR records are placed in a file named **db.***NET* where *NET* is the network number from the command line. Network 10 data is placed in **db.10**. Network 10.1 data are placed in "db.10.1".

Mail exchanger (MX) records are created unless the **-M** option is used. The default MX record has a weight of 10 with the host itself as its mail exchanger. No default MX record is created for a host if **[no smtp]** is in the comment section of that line in the host table. MX records for each mail hub declared with the **-m** option are added for each host even if **[no smtp]** is in the comment section.

Well known services (WKS) records are created for each host that handles SMTP mail (does not have **[no smtp]**) if **-w** is used. The only service listed is SMTP.

Text (TXT) records are created for comments associated with hosts in the host table if **-t** is used. The comments do not include **[no smtp]**.

For each domain, a start of authority (SOA) record is created. The SOA record requires 2 domain names: the host that the data is created on and the electronic mail address of the person responsible. The **-h** and **-u** options influence the names. In addition, the SOA record requires 5 values: a serial number, a refresh time, a retry time, an expire time, and a minimum ttl (time to live). The first time the data is created, the serial number is set to 1, the refresh time is set to 3 hours, the retry time is set to 1 hour, the expire time is set to 1 week, and the minimum ttl is set to 1 day. The **-o** option changes these values except for the serial number. Each subsequent time **hosts_to_named** is run, the serial number is incremented. If any of the other fields in the SOA record are modified, the changed values are retained.

If there are files named **spcl.***DOMAIN* or **spcl.***NET* in the current directory, **$INCLUDE** directives are added to the corresponding **db.***DOMAIN* or **db.***NET* file for the **spcl** file. In this way, special data can be added to the data generated by *hosts_to_named*.

The first time **hosts_to_named** is run, it creates a default boot file for a primary name server. Each subsequent time **hosts_to_named** is run, the boot file is updated if necessary. New entries are made in the boot file for any additional networks or domains not already in the boot file. No entries are deleted from the boot file.

The boot file for a caching-only server, **boot.cacheonly**, is created if it does not exist. The boot files for secondary servers, **boot.sec.save** and **boot.sec**, are created if the **-z** or **-Z** options are used. The boot files for secondary servers are created new each time from the primary server boot file so that they are equivalent.

EXAMPLES

Create name server data for networks 15.19.8 and 15.19.9 in **div.inc.com.**

```
hosts_to_named -d div.inc.com -n 15.19.8 -n 15.19.9
```

Create name server data for networks 15.19.8 and 15.19.9 in **div.inc.com.** Ignore aliases in the host table and include 2 mail hubs - **aaa.div.inc.com** and **bbb.mkt.inc.comk**. Put all of the options in a file.

```
hosts_to_named -f option_file
```

Option_file contains the following lines:

```
-d div.inc.com
-n 15.19.8 -n 15.19.9
-m 20:aaa
-m 30:bbb.mkt.inc.com
-A
```

Network 15.19.15 has hosts in the **xx.inc.com** domain and the **div.inc.com** domain. Create name server data for **xx.inc.com**. Create only pointer (PTR) data for hosts in **div.inc.com** on network 15.19.15 (this requires the hosts in **div.inc.com** to have the canonical name or an alias of the form **x.div.inc.com**).

```
hosts_to_named -d xx.inc.com -n 15.19.15 -p div.inc.com
```

Create name server data for network 15.19.8 in **div.inc.com**. Include **div.inc.com** data from network 15.19.15 but do not create pointer (PTR) data for 15.19.15 since that is being handled by the **xx.inc.com** server.

```
hosts_to_named -d div.inc.com -n 15.19.8 -a 15.19.15
```

AUTHOR

hosts_to_named was developed by HP.

FILES

/etc/hosts	The host table
named.boot	Primary server boot file
boot.cacheonly	Caching only server boot file
boot.sec.save	Secondary server boot file
boot.sec	Secondary server boot file
db.127.0.0	Pointer information for 127.0.0.1
db.cache	Stub cache file for root server addresses
db.root	Data for servers for the root domain
db.DOMAIN	Address and other data for a domain
db.DOMAIN.in-addr	Pointer data for all network-numbers
db.NET	Pointer data for a network-number

SEE ALSO

named(1M), RFC1034, RFC1035

h

NAME

hpux - HP-UX bootstrap

SYNOPSIS

hpux [**-F**] [**-lm**] [**-a**[**C**|**R**|**S**|**D**] *devicefile*] [**-f***number*] [**-i***string*] [**boot**] [*devicefile*]
hpux ll [*devicefile*] (same as **hpux ls -aFln**)
hpux ls [**-aFiln**] [*devicefile*]
hpux set autofile *devicefile string*
hpux show autofile [*devicefile*]
hpux -v
hpux restore *devicefile* (Series 700 only; see DEPENDENCIES.)

DESCRIPTION

hpux is the HP-UX specific secondary system loader (SSL) utility for bootstrap (see *isl*(1M) for the initial system loader). It supports the operations summarized below, as shown in the SYNOPSIS and detailed later in this DESCRIPTION.

boot	Loads an object file from an HP-UX file system or raw device and transfers control to the loaded image. (Note, the **boot** operation is position dependent).
ll	Lists the contents of HP-UX directories in a format similar to **ls -aFln**. (See *ls*(1); **ls** only works on a local disk with a HFS file system).
ls	Lists the contents of HP-UX directories. (See *ls*(1); **ls** only works on a local disk with a HFS file system).
show autofile	Displays the contents of the **autoexecute** file.
set autofile	Changes the contents of the **autoexecute** file to that specified by *string*.
-v	Displays the release and version numbers of the **hpux** utility.
restore	Recovers the system from a properly formatted bootable tape. (Series 700 specific; see DEPENDENCIES.)

hpux commands can be given interactively from the keyboard, or provided in an *isl* **autoexecute** file.

hpux is limited to operations on the interface initialized by *pdc*(1M). In most cases, operations are limited to the boot device interface.

Notation

hpux accepts numbers (numeric constants) in many of its options. Numbers follow the C language notation for decimal, octal, and hexadecimal constants. A leading 0 (zero) implies octal and a leading 0x or 0X implies hexadecimal. For example, 037, 0x1F, 0X1f, and 31 all represent the same number, decimal 31.

hpux boot, **ll**, **ls**, **set autofile**, **show autofile**, and **restore** operations accept *devicefile* specifications, which have the following format:

*manager***(***w***/***x***.***y***.***z***;***n***)***filename*

The *devicefiles* specification is comprised of a device name and a file name. The device name (*manager***(***w***/***x***.***y***.***z***;***n***)**), consists of a generic name of an I/O system *manager* (device or interface driver) such as **disc**, a hardware path to the device, and minor number. The *manager* name can be omitted entirely if the default is used. *w***/***x***.***y***.***z* is the physical hardware path to the device, identifying bus converters, slot numbers, and hardware addresses. For Series 700 machines, there are a set of mnemonics that can be used instead of the hardware paths. The *n* is the minor number that controls *manager*-dependent functionality. The file name part, *filename*, is a standard HP-UX path name. Some **hpux** operations have defaults for particular components. A *devicefile* specification containing a device part only specifies a raw device. A *devicefile* specification containing a file name implies that the device contains an HP-UX file system, and that the *filename* resides in that file system.

A typical boot *devicefile* specification is

```
disc(2/4.0.0;0)/stand/vmunix
```

The *manager* is **disc**, the hardware path to the disk device is **2/4.0.0**, the minor number shown as **0** by default, and the **/stand/vmunix** is the *filename* for the boot device.

hpux now supports a consolidated list of managers: **disc**, **tape**, and **lan**. The manager **disc** manages all CS/80 disks connected via HP-IB (formerly **disc0**); CS/80 disks connected via the HP 27111 interface (formerly **disc2**); CS/80 disks connected via NIO HP-IB (formerly **disc1**); all disks connected via SCSI, (formerly **disc3**), and all *autochanger* disk devices (formerly **disc30**). The manager **lan** manages remote boot through the HP 28652A NIO based LAN interface (formerly **lan1**). Remote boot is currently supported on this card only and not on any CIO-based LAN card. The manager **tape** manages the HP 7974, HP 7978, and HP 7980 tape drives via HP-IB (formerly **tape1**) and tape drives via SCSI (formerly **tape2**).

The hardware path in a *devicefile* specification is a string of numbers, each suffixed by slash, (**/**), followed by a string of numbers separated by dots (**.**), each number identifying a hardware component notated sequentially from the bus address to the device address. A hardware component suffixed by a slash indicates a bus converter and may not be necessary on your machine. For example, in *w***/***x***.***y***.***z* *w* is the address of the bus converter, *x* is the address of the MID-BUS module, *y* is the CIO slot number, and *z* is the HP-IB address or HP 27111 bus address.

The minor number, *n*, in a *devicefile* specification controls driver-dependent functionality. (See the manual, *Configuring HP-UX for Peripherals*, for minor-number bit assignments of specific drivers).

File names are standard HP-UX path names. No preceding slash (**/**) is necessary and specifying one will not cause problems.

Defaults

Default values chosen by **hpux** to complete a command are obtained through a sequence of steps. First, any components of the command specified explicitly are used. If the command is not complete, **hpux** attempts to construct defaults from information maintained by **pdc** (see *pdc*(1M)). If sufficient information to complete the command is unavailable, the **autoexecute** file is searched. If the search fails, any remaining unresolved components of the command are satisfied by hard-coded defaults.

There is no hard-coded default choice for a *manager*; if none can be chosen, **hpux** reports an error.

When the hardware path to the boot device is not specified, **hpux** defaults to information maintained by *pdc*. The hardware path element has no hard-coded default.

If the minor number element is not supplied, **hpux** takes its default from the **autoexecute** file. Failing that, the hard-coded default of 0 is used.

For the **boot** command, a *devicefile* specification without a file name indicates that the boot device does not contain an HP-UX file system. **hpux** interprets this as a NULL (instead of missing) file name and does not search for a default. If the entire *devicefile* specification is missing, **hpux** searches for a default; either the **autoexecute** file contents or the hard-coded default is chosen.

There are two possible hard-coded default *devicefile* specifications. One hard-coded default *devicefile* specification is **/vmunix**. The other hard-coded default *devicefile* specification is **/stand/vmunix**.

If you have a LVM system where the boot volume and the root volume are on different logical volumes, the kernel would be **/vmunix**. This is because the boot volume will be mounted under /stand when the system is up.

For all other configurations, the kernel would be **/stand/vmunix**.

The search order for the hard-coded defaults is **/stand/vmunix** and then **/vmunix.**

boot Operation

The **boot** operation loads an object file from an HP-UX file system or raw device as specified by the optional *devicefile*. It then transfers control to the loaded image.

Any missing components in a specified *devicefile* are supplied with a default. For example, a *devicefile* of **vmunix.new** would actually yield:

```
disc(8.0.0;0)vmunix.new
```

and a *devicefile* of **(8.0.1)/stand/vmunix**, for booting from the disk at HP-IB address 1, would yield

```
disc(8.0.1;0)/stand/vmunix
```

Regardless of how incomplete the specified *devicefile* may be, **boot** announces the complete *devicefile* specification used to find the object file. Along with this information, **boot** gives the sizes of the **TEXT**, **DATA**, and **BSS**, segments and the entry offset of the loaded image, before transferring control to it.

The **boot** operation accepts several options. Note that **boot** options *must* be specified positionally as shown in the syntax statement in the SYNOPSIS. Options for the **boot** operations are as follows:

-a[C|R|S|D] *devicefile*	Accept a new location (as specified by *devicefile)* and pass it to the loaded image. If that image is an HP-UX kernel, the kernel will erase its predefined I/O configuration, and configure in the specified *devicefile*. If the **C**, **R**, **S**, or **D** option is specified, the kernel configures the *devicefile* as the **console**, **root**, **swap**, or **dump** device, respectively. Note that **-a** can be repeated multiple times.
-f*number*	Use the number and pass it as the flags word to the loaded image.
-i*string*	Set the initial *run-level* for **init** (see *init*(1M)) when booting the system. The *run-level* specified will override any *run-level* specified in an *initdefault* entry in **/etc/inittab** (see *inittab*(4)).
-lm	Boot the system in LVM maintenance mode, configure only the root volume, and then initiate single user mode.
-F	Use with SwitchOver/UX software. Ignore any locks on the boot disk. The **-F** option should be used only when it is known that the processor holding the lock is no longer running. (If this option is not specified and a disk is locked by another processor, the kernel will not boot from it, to avoid the corruption that would result if the other processor were still using the disk).

boot places some restrictions on object files it can load. It accepts only the HP-UX magic numbers **EXEC-MAGIC** (0407), **SHAREMAGIC** (0410), and **DEMANDMAGIC** (0413). See *magic*(4). The object file must contain an Auxiliary Header of the **HPUX_AUX_ID** type and it must be the first Auxiliary Header (see *a.out*(4)).

ll and ls Operations

The **ll** and **ls** operations list the contents of the HP-UX directory specified by the optional *devicefile*. The output is similar to that of **ls -aFl** command, except the date information is not printed.

The default *devicefile* is generated just as for **boot**, defaulting to the current directory.

set autofile Operation

The **set autofile** operation overwrites the contents of the **autoexecute** file, *autofile*, with the string specified (see **autoexecute** in the EXAMPLES section).

show autofile Operation

The **show autofile** operation displays the contents of the **autoexecute** file, *autofile* (see **autoexecute** in the EXAMPLES section).

DIAGNOSTICS

If an error is encountered, **hpux** prints diagnostic messages to indicate the cause of the error. These messages fall into the General, Boot, Copy, Configuration, and System Call categories. System Call error messages are described in *errno*(2). The remaining messages are listed below.

General

bad minor number in devicefile spec
The minor number in the *devicefile* specification is not recognized.

bad path in devicefile spec
The hardware path in the *devicefile* specification is not recognized.

command too complex for parsing
The command line contains too many arguments.

no path in devicefile spec
The *devicefile* specification requires (but does not contain) a hardware path component.

panic (in hpuxboot): (display==*number***, flags==***number***)** *string*
A severe internal **hpux** error has occurred. Report to your nearest HP Field Representative.

Boot

`bad magic`
The specified object file does not have a recognizable magic number.

`bad number in flags spec`
The flags specification in the **`-f`** option is not recognized.

`Exec failed: Cannot find /stand/vmunix or /vmunix.`
Neither /stand/vmunix or /vmunix could be found.

`booting from raw character device`
In booting from a raw device, the *manager* specified only has a character interface, which might cause problems if the block size is incorrect.

`isl not present, please hit system RESET button to continue`
An unsuccessful **`boot`** operation has overlaid **`isl`** in memory. It is impossible to return control to **`isl`**.

`short read`
The specified object file is internally inconsistent; it is not long enough.

`would overlay`
Loading the specified object file would overlay **`hpux`**.

Configuration

`cannot add path, error` *number*
An unknown error has occurred in adding the hardware path to the I/O tree. The internal error number is given. Contact your HP Field Representative.

`driver does not exist`
The manager specified is not configured into **`hpux`**.

`driver is not a logical device manager`
The *manager* named is not that of a logical device manager and cannot be used for direct I/O operations.

`error rewinding device`
An error was encountered attempting to rewind a device.

`error skipping file`
An error was encountered attempting to forward-space a tape device.

`negative skip count`
The skip count, if specified, must be greater than or equal to zero.

`no major number`
The specified *manager* has no entry in the block or character device switch tables.

`path incompatible with another path`
Multiple incompatible hardware paths have been specified.

`path long`
The hardware path specified contains too many components for the specified *manager*.

`path short`
The hardware path specified contains too few components for the specified *manager*.

`table full`
Too many devices have been specified to **`hpux`**.

EXAMPLES

As a preface to the examples which follow, here is a brief overview of HP-UX system boot-up sequences.

Automatic Boot

Automatic boot processes on various HP-UX systems follow similar general sequences. When power is applied to the HP-UX system processor, or the system **Reset** button is pressed, processor-dependent code (firmware) is executed to verify hardware and general system integrity (see *pdc*(1M)). After checking the hardware, **pdc** gives the user the option to override the **autoboot** sequence by pressing the **Esc** key. At that point, a message resembling the following usually appears on the console.

```
(c) Copyright. Hewlett-Packard Company. 1994.
All rights reserved.

PDC ROM rev. 130.0
32 MB of memory configured and tested.

Selecting a system to boot.
To stop selection process, press and hold the ESCAPE key...
```

If no keyboard activity is detected, **pdc** commences the **autoboot** sequence by loading **isl** (see *isl*(1M)) and transferring control to it. Since an **autoboot** sequence is occurring, **isl** finds and executes the **autoexecute** file which, on an HP-UX system, requests that **hpux** be run with appropriate arguments. Messages similar to the following are displayed by **isl** on the console:

```
Booting from: scsi.6  HP 2213A
Hard booted.
ISL Revision A.00.09  March 27, 1990
ISL booting  hpux boot disk(;0)/stand/vmunix
```

hpux, the secondary system loader, then announces the operation it is performing, in this case **boot**, the *devicefile* from which the load image comes, and the **TEXT** size, **DATA** size, **BSS** size, and start address of the load image, as shown below, before control is passed to the image.

```
Booting disk(scsi.6;0)/stand/vmunix
966616+397312+409688 start 0x6c50
```

The loaded image then displays numerous configuration and status messages.

Interactive Boot

To use **hpux** interactively, **isl** must be brought up in interactive mode by pressing the **Esc** key during the interval allowed by **pdc**. **pdc** then searches for and displays all bootable devices and presents a set of boot options. If the appropriate option is chosen, **pdc** loads **isl** and **isl** interactively prompts for commands. Information similar to the following is displayed:

```
Selection process stopped.

Searching for Potential Boot Devices.
To terminate search, press and hold the ESCAPE key.

Device Selection    Device Path            Device Type
-------------------------------------------------------------
P0                  scsi.6.0               QUANTUM PD210S
P1                  scsi.1.0               HP      2213A
p2                  lan.ffffff-ffffff.f.f  hpfoobar

b)  Boot from specified device
s)  Search for bootable devices
a)  Enter Boot Administration mode
x)  Exit and continue boot sequence

Select from menu: b p0 isl

Trying scsi.6.0
Boot path initialized.
Attempting to load IPL.

Hard booted.
ISL Revision A.00.2G  Mar  27, 1994
ISL>
```

Although all of the operations and options of **hpux** can be used from **isl** interactively, they can also be executed from an **autoexecute** file. In the examples below, user input is the remainder of the line after each **ISL>** prompt shown. The remainder of each example is text displayed by the system. Before going over specific examples of the various options and operations of **hpux**, here is an outline of the steps taken in the automatic boot process. Although the hardware configuration and boot paths shown are for a single Series 800 machine, the user interfaces are consistent across all models. When the system **Reset** button is depressed, **pdc** executes self-test, and assuming the hardware tests pass, **pdc** announces itself, sends a BELL character to the controlling terminal, and gives the user 10 seconds to override the **autoboot** sequence by entering any character. Text resembling the following is displayed on the console:

```
Processor Dependent Code (PDC) revision 1.2
Duplex Console IO Dependent Code (IODC) revision 3

Console path        = 56.0.0.0.0.0.0   (dec)
                      38.0.0.0.0.0.0   (hex)

Primary boot path   = 44.3.0.0.0.0.0   (dec)
                     2c.00000003.0.0.0.0.0   (hex)

Alternate boot path = 52.0.0.0.0.0.0   (dec)
                      34.0.0.0.0.0.0   (hex)

32 MB of memory configured and tested.

Autosearch for boot path enabled

To override, press any key within 10 seconds.
```

If no keyboard character is pressed within 10 seconds, **pdc** commences the **autoboot** sequence by loading **isl** and transferring control to it. Because an **autoboot** sequence is occurring, **isl** merely announces itself, finds and executes the **autoexecute** file which, on an HP-UX system, requests that **hpux** be run with appropriate arguments. The following is displayed on the console.

```
10 seconds expired.
Proceeding with autoboot.

Trying Primary Boot Path
------------------------
Booting...
Boot IO Dependent Code (IODC) revision 2

HARD Booted.

ISL Revision A.00.2G Mar  20, 1994

ISL booting  hpux
```

hpux then announces the operation it is performing, in this case **boot**, the *devicefile* from which the load image comes, and the **TEXT** size, **DATA** size, **BSS** size, and start address of the load image. The following is displayed before control is passed to the image.

```
Boot
: disc3(44.3.0;0)/stand/vmunix
3288076 + 323584 + 405312 start 0x11f3e8
```

Finally, the loaded image displays numerous configuration and status messages, then proceeds to **init** *run-level* 2 for multiuser mode of operation.

isl must be brought up in interactive mode to use the operations and options of **hpux**. To do this, simply enter a character during the 10 second interval allowed by **pdc**. **pdc** then asks if the primary boot path is acceptable. Answering yes (**Y**) is usually appropriate. **pdc** then loads **isl** and **isl** interactively prompts for commands. The following lines show the boot prompt, the **Y** response, subsequent boot messages, and finally the Initial System Loader (ISL) prompt that are sent to the display terminal:

```
Boot from primary boot path (Y or N)?> y
Interact with IPL (Y or N)?> y

Booting...
Boot IO Dependent Code (IODC) revision 2

HARD Booted.

ISL Revision A.00.2G Mar  20, 1994

ISL>
```

Although all of the operations and options of **hpux** can be used from **isl** interactively, they can also be executed from an **autoexecute** file. In the examples below, all user input follows the **ISL>** prompt on the same line. Subsequent text is resultant messages from the ISL.

Default Boot

Entering **hpux** initiates the default boot sequence. The boot path read from **pdc** is **8.0.0**, the manager associated with the device at that path is **disc**, the minor number, in this case derived from the **autoexecute** file, is **4** specifying section 4 of the disk, and the object file name is **/stand/vmunix**.

```
ISL> hpux

Boot
: disc3(44.3.0;0)/stand/vmunix
3288076 + 323584 + 405312 start 0x11f3e8
```

Booting Another Kernel

In this example, **hpux** initiates a **boot** operation where the name of the object file is **vmunix.new**.

```
ISL> hpux vmunix.new

Boot
: disc3(44.3.0;0)/stand/vmunix.new
3288076 + 323584 + 405312 start 0x11f3e8
```

Booting From Another Section

In this example (shown for backward compatibility), a kernel is booted from another section of the root disk. For example, suppose kernel development takes place under **/mnt/azure/root.port** which happens to reside in its own section, section 3 of the root disk. By specifying a minor number of **3** in the above example, the object file **sys.azure/S800/vmunix** is loaded from **/mnt/azure/root.port**.

```
ISL> hpux (;3)sys.azure/S800/vmunix

Boot
: disc(8.0.0;0x3)sys.azure/S800/vmunix
966616+397312+409688 start 0x6c50
```

Booting From Another Disk

Only the hardware path and file name are specified in this example. All other values are boot defaults. The object file comes from the file system on another disk.

```
ISL> hpux (52.5.0.0)/stand/vmunix

Boot
: disc(52.5.0.0)/stand/vmunix
966616+397312+409688 start 0x6c50
```

Booting From LAN

This example shows how to boot a cluster client from the LAN. Though this example specifies a *devicefile*, you can also use default boot, as shown in a previous example. For a boot operation other than default boot, the file name must be specified and can be no longer than 11 characters. Booting to **isl** from a local disk then requesting an image to be loaded from the LAN is *not* supported.

```
ISL> hpux lan(32)/stand/vmunix

Boot
: lan(32;0x0)/stand/vmunix
966616+397312+409688 start 0x6c50
```

Booting To Single User Mode

In this example, the **-i** option is used to make the system come up in *run-level* **s**, for single user mode of operation.

```
ISL> hpux -is

Boot
: disc(8.0.0;0x0)/stand/vmunix
966616+397312+409688 start 0x6c50
```

(Kernel Startup Messages Omitted)

```
INIT: Overriding default level with level 's'

INIT: SINGLE USER MODE
WARNING:  YOU ARE SUPERUSER !!
#
```

h

Booting With A Modified I/O Configuration

Here, a tape driver is configured in at CIO slot 2, HP-IB address 0. Regardless of what was present in the kernel's original I/O configuration, the driver **tape** is now configured at that hardware path. Similarly, **mux0** is configured in at CIO slot 1 which is to be the console. The only other devices configured are the console and root device, which **boot** derived from **pdc**.

```
ISL> hpux -aC mux0(8.1) -a tape(8.2.0)

Boot
: disc(8.0.0;0x0)/stand/vmunix
: Adding mux0(8.1;0x0)...
: Adding tape(8.2.0;0x0)...
966616+397312+409688 start 0x6c50
Beginning I/O System Configuration.
cio_ca0 address = 8
   hpib0 address = 0
      disc0 lu = 0 address = 0
   mux0 lu = 0 address = 1
   hpib0 address = 2
      tape1 lu = 0 address = 0
I/O System Configuration complete.
```

(Additional Kernel Startup Messages Omitted)

Booting From A Raw Device

This example shows booting from a raw device (that is, a device containing no file system). Note that no file name is specified in the *devicefile*. The device is an HP 7974 tape drive, and therefore **tape** is the *manager* used. The tape drive is at CIO slot 2, HP-IB address 3. The first file on the tape will be skipped. The minor number specifies a tape density of 1600 BPI with no rewind on close. Depending on the minor number, **tape** requires the tape be written with 512 or 1024 byte blocks.

```
ISL> hpux tape(8.2.3;0xa0000)

Boot
: tape(8.2.3;0xa0000)
966616+397312+409688 start 0x6c50
```

Displaying The Autoexecute File

In this example, **show autofile** is used to print the contents of the **autoexecute** file residing in the boot LIF, on the device from which **hpux** was booted. Optionally, a *devicefile* can be specified in order to read the **autoexecute** file from the boot LIF of another boot device.

```
ISL> hpux show autofile
Show autofile
```

```
    : AUTO file contains (hpux)
```

Changing The Autoexecute File

This example shows how to change the contents of the **autoexecute** file. Once done, the system can be reset, and the new command will be used during any unattended boot.

```
    ISL> hpux set autofile "hpux /stand/vmunix.std"
    Set autofile
    : disk(2/0/1.3.0.0.0.0.0;0)
    : AUTO file now contains "(hpux /stand/vmunix.std)"
```

Listing Directory Contents

The contents of the directory (**/stand**) on the root disk are listed. The format shows the file protections, number of links, user id, group id, and size in bytes for each file in the directory. There are three available kernels to boot: **vmunix**, **vmunix.test**, and **vmunix.prev**. Listing the files over the LAN is not supported.

```
    ISL> hpux ll /stand

    Ls
    : disk(2/0/1.3.0.0.0.0.0;0)/stand
    dr-xr-xr-x    3 2        2              1024 ./
    drwxr-xr-x   17 0        0              1024 ../
    -rw-r--r--    1 0        3               191 bootconf
    drwxr-xr-x    2 0        0              1024 build/
    -rw-r--r--    1 0        0               632 ioconfig
    -rw-r--r--    1 0        3                82 kernrel
    -r--r--r--    1 0        3               426 system
    -rw-r--r--    1 0        3               437 system.prev
    -rwxr-xr-x    1 0        3           7771408 vmunix*
    -rwxr-xr-x    1 0        3           7771408 vmunix.prev*
```

Getting The Version

The **-v** option is used to get the version numbers of **hpux**.

```
    ISL> hpux -v

    Release: 10.00
    Release Version:
    @(#) X10.20.B HP-UX() #1: Dec  4 1995 16:55:08
```

DEPENDENCIES

Series 700 Only

The **restore** operation is provided as a recovery mechanism in the event that a disk becomes totally corrupted. It copies data from a properly formatted bootable tape to disk. When this tape contains a backup image of the disk, the entire disk is restored. To create a properly formatted tape (DDS ONLY), the following commands should be executed:

```
    dd if=/usr/lib/uxbootlf of=/dev/rmt/0mn bs=2k
    dd if=/dev/rdsk/1ss of=/dev/rmt/0m bs=64k
```

The first **dd** puts a boot area on the tape, making it a bootable image (see *dd*(1)). Once the boot image is on tape, the tape is *not* rewound. The next **dd** appends an image of the disk to the tape. The entire process takes about one hour for a 660 MB HP 2213 disk. To avoid later problems with **fsck** after the disk is restored, bring the system to single user mode and type **sync** a few times before doing the second **dd** (see *fsck*(1M)). Once created, the tape can be used to completely restore the disk:

1. Insert the tape into the tape drive.
2. Instruct the machine to boot to ISL from the tape. This is usually done by specifying **scsi.3** as the boot path.
3. Enter the following in response to the ISL prompt:

```
        ISL> hpux restore disk(scsi.1;0)
```

This restores the disk image from the tape to the actual disk at **scsi.1**. *Any existing data on the disk will be lost*. This command destroys the contents of the device specified by *devicefile*. The restoration process takes about one hour for a 660 MB drive.

NOTE: There is a 2 GB limit on the amount of data that can be restored. The tape and disk must be on the boot device interface.

Also, this command may be replaced in the future by superior installation and recovery mechanisms. At that time, this command will be removed.

SEE ALSO

boot(1M), fsck(1M), init(1M), isl(1M), pdc(1M), errno(2), a.out(4), inittab(4), magic(4).

NAME

i4admin - administer LicensePower/iFOR licensing

SYNOPSIS

i4admin [**-***Standard-X-Arguments*]

i4admin -a [**-n** *server-name*] [**-f** *filename*] [**-v** **"'***vendor-name***'** [*vendor-id vendor-password*]**"** **-p** **"'***product-name***'** **'***product-version***'** *license-password* [**'***license-annotation***'**]**"**]

i4admin -d [**-n** *server-name*] **-v** *vendor-name* **-p** *product-name* **-t** *timestamp*

i4admin -l s|**v**|**p** [**-i**] [**-n** **"***server-name...***"**] [**-v** **"'***vendor-name***'***...***"**] [**-p** **"'***product-name***'***...***"**] [**-u** **"***user-name...***"**]

i4admin -s [**-n** **"***server-name...***"**] [**-v** **"'***vendor-name***'***...***"**] [**-p** **"'***product-name***'***...***"**] [**-u** **"***user-name...***"**]

i4admin -r 1|**2**|**3**|**4**|**5** [**-e 1**|**234567**] [**-b** *start-date*] [**-g** *end-date*] [**-n** **"***server-name...***"**] [**-v** **"'***vendor-name***'***...***"**] [**-p** **"'***product-name***'***...***"**] [**-u** **"***user-name...***"**]

i4admin -x *before-date* **-n** **"***server-name...***"**

i4admin -h

i

DESCRIPTION

The LicensePower/iFOR Administration tool, **i4admin**, completely manages the LicensePower/iFOR licensing system. The tool can perform the following tasks:

- Perform basic license administration (e.g., adding and deleting licenses).
- Construct a single logical view of the license system from which current summary license usage and current detailed license usage reports can be generated.
- Generate detailed license event and license usage reports from logged server data.

The **i4admin** tool has a Graphical User Interface (GUI) and a Command Line Interface (CLI). If **i4admin** is invoked with non-X arguments, the CLI version is started, otherwise the GUI version is started.

A printable on-line administration guide is also available. (See the *FILES* section below.)

CLI Actions

The CLI is invoked with one of the following actions, and one or more action modifiers.

-a Add a product license to a specified license server. There are two ways to add a license to a license server.

If the license information has been provided in the form of a license certificate (a flat file describing the license), the license certificate can be added by specifying the *server-name* and the license certificate *filename*. If the server name is omitted, the license is added to the license server running on the local machine.

If the license information has not been provided in a license certificate, the parameters must be entered individually. All three vendor parameters are not always required. If the vendor for the product is already installed on the server, only the *vendor-name* must be specified, otherwise the *vendor-name*, *vendor-id* and *vendor-password* must be specified.

-d Delete a product license. To delete a compound password, or a use-once license, the license must have expired. If the server name is omitted, the license is deleted from the license server running on the local machine. The license *timestamp* must be specified to differentiate between licenses for the same product (same Vendor ID, Product ID, and Product version), which are installed on the same server. The license *timestamp* can be found using the list product details command:

```
i4admin -lp -i -p product-name
```

-l List installed license information. The command is qualified by the list type flag, **s**|**v**|**p**, to list servers, vendors, or products respectively.

The vendor list can be limited to specific servers by entering one or more *server-names*. If more than one *server-name* is entered, the list must be enclosed in double quotes.

By default the product list contains a summary of product information. Detailed product information can be queried by specifying the **-i** parameter. The product list can be filtered by server, vendor, and user. If more than one *vendor-name* is entered, the list of *vendor-names* must be enclosed in double quotes. Any *vendor-name* which contains white space must also be enclosed in single quotes.

Specify one or more *user-names* to limit the product list to products currently in use by the those users.

-s Generate a status report containing detailed current license usage. For each product, the report includes the number of licenses in use, the user of the product and when license was acquired. By default the status report is generated based on all active license servers in the cell. The scope of the report can be limited by specifying *server-names*, *vendor-names*, *product-names*, or *user-names*.

-r Generates reports which are based on license events logged by the license server. The command will generate one of five reports specified by the report-type flag (**1**|**2**|**3**|**4**|**5**).

1 Reports server log events. This command is further qualified by the event-flag which is described below.

2 For each product lists the number of requests for licenses, the number of licenses granted, and the percent of rejected requests.

3 Lists the same information as **2** but breaks out a separate entry for each user.

4 For each product, lists the maximum concurrent nodes, maximum concurrent users, and average time in use.

5 For each product, lists the number of times each user invoked the product and the average time the product was in use.

-x *before-date*
Delete all log entries on the servers specified by *server-names* which are timestamped on or before *before-date*

-h Display a synopsis of command-line options

CLI Action Modifiers

-b *start-date*
Specify the start date for generating log reports. By default the start date is Jan. 1 1970.

-e *event-type*
Specify an event filter for the standard event report (**-r1**). By default all events are listed.

1 All events (default)

2 License related events (license request, license release, etc.)

3 Vendor messages

4 License database modifications (license added, license deleted, etc.)

5 Error events (license request failed, vendor not found, etc.)

6 Server start/stop

7 Fatal error events (server out of memory, server file IO error, etc).

Error events 2-7 can be combined, e.g., **-e357** to list vendor messages, error events, and fatal error events.

-f *filename*
Specifies filename for adding a license certificate.

-g *end-date*
Specify the end date for generating log reports. By default the end date is current day.

-i Include license details (start date, end-date, multi-use rules, timestamp, etc.) when listing products.

-n **"***server-name...***"**
Specify a server when performing administrative actions (adding a license, deleting a license, cleaning the log file), or limit the scope of a listing, status report or event report to a particular server, or servers. If more than one *server-name* is specified to limit the scope of a listing or report, the entire argument must be enclosed in double quotes.

-p "'*product-name***' '***product-version***'** *license-password* [**'***license-annotation***'**]**"**
Specify a product when adding a license (**-a**) which is not defined in a license certificate. The entire argument must be enclosed in double quotes. If the *product-name*, *product-version*, or *license-annotation* contains white space the argument must be enclosed in single quotes.

-p "'*product-name***'***...***"**
Specify a product, or products to limit the scope of a product listing (**-lp**), a status report (**-s**), or a event report (**-r**). If multiple *product-name*s are specified, the entire argument must be enclosed in double quotes. If any *product-name* contains white space it must be enclosed in single quotes to differentiate the argument from multiple single-word product names.

-u "*user-name...***"**
Limit the scope of a status report, or event report to a specific user, or users. If more than one user is specified, the entire argument must be enclosed in double quotes.

-v "'*vendor-name***'** [*vendor-id vendor-password*]**"**
Specify a vendor when adding a product license manually. If another product for this vendor has been installed on an active license server in this cell, only the *vendor-name* must be specified. If a product for this vendor has not been previously installed on an active server in this cell, the *vendor-id* and the *vendor-password* must also be specified.

i

GUI Description

The **i4admin** GUI provides an intuitive dialog based interface to manage all aspects of the LicensePower/iFOR licensing system. The main window is divided into four functional areas:

- The menu bar contains pulldown menus which provide the interface to all administrative commands.
- The toolbar provides direct access to frequently used commands.
- All reports are displayed in the scrolling display area.
- When performing a task, the tool displays its progress in the status line at the bottom of the main window.

The GUI tool can perform the following tasks which will be described in detail in succeeding sections.

- Basic license administration which includes adding and deleting licenses.
- Extensive report generation based on current license usage and logged license events.

GUI Administrative Tasks

The Administrative tasks are adding licenses, deleting licenses, and cleaning up stale licenses. There are two ways to add a license. If the license information has been provided in the form of a license certificate (a flat file describing the license), follow the first procedure. If the license information has been provided in any other form, follow the second procedure.

Adding a license from a license certificate

1. Open the **Add** pulldown menu and select the **License...** menu item.
2. Select the server to add the license to from the **Server** drop-down listbox.
3. Select the **Read certificate...** button.
4. Enter the name of the license certificate in the **Selection** entry field. The **Filter** entry field and the **Filter** button can be used to limit the selection to a specific file or range of files.
5. Select **OK** to accept the file selection and close the dialog. Verify that the Vendor name, Product name, and Product version appear correctly on the **Add License** panel.
6. Select **OK** to add the license to the selected server and close the **Add license** dialog.

Adding a license manually

1. Open the **Add** pulldown menu and select the **License...** menu item.
2. Select the server to add the license to from the **Server** drop-down listbox.
3. Select the **Enter manually...** button.
4. Select the product's vendor from the drop down list of vendors which are displayed. If the product's vendor is not displayed, select the **New vendor** button to specify the vendor information.

5. Enter the Product name, Product version, License password, and optional License annotation (if provided) in the fields.
6. Select **OK** to accept the information and close the dialog. Verify that the Vendor name, Product name, and Product version appear correctly on the **Add license** dialog.

Deleting a license

1. Change to the **Product details** view. To change views select the desired view from the **View** pulldown menu.
2. Select a license to delete. Note that selected items which can be acted on are distinguished from plain text by the highlight color of the selection.
3. Select **Delete license** button from the **Selected** pulldown menu. The tool will ask for confirmation before deleting the license. Note that compound passwords, and use-once licenses cannot be deleted before their expiration date.

Cleaning up stale licenses

When a client application acquires a license from the license server, it also periodically checks back with the server to tell the server the application is still running. The interval between checks is referred to as the check-in period. The server does not automatically release licenses for applications which have missed their check-in period. However, if a client application attempts to acquire a license and none are available, the server will check all the outstanding licenses to make sure the respective clients have checked in. If a client has missed its check-in period, that client's license will be granted. The clean stale license command forces the server to iterate through the outstanding licenses, releasing the licenses which have not been checked.

To clean up stale licenses for a product or products:

1. Select one or more products from the **Product summary** view or the **Product status** view. Multiple entries can be selected by holding the Shift or Control key down while selecting.
2. Open the **Selected** menu and choose the **Clean stale licenses** menu item.

GUI Usage and Installed License Reporting

This set of reports are generated based on installed license details, and current usage information. The reports are generated based on a snapshot of the license system at a particular instant in time. Since the license system may be constantly changing, the information contained in these reports is only as current as the last snapshot.

These reports contain information which is summed across the license system. The **i4admin** tool constructs a single logical view of the license system from which these reports are generated. This logical view is referred to as a snapshot of the license system. There are three reports based on the snapshot. The reports are accessed via the **View** pulldown menu.

- The product summary is a terse view of a product's installed licenses and current license usage. >From this view the administrator can quickly identify problem areas, i.e., a product has 10 licenses installed, and 10 are in use.
- The product details view reports detailed installed product information, including the number of license installed, the start and expiration date of the licenses, and the server that the license is installed on. >From this view, the administrator can select delete a license.
- The product status view generates a detailed current usage report which includes; the number of licenses installed, the number of licenses currently checked out, who is using the license from what node, and how long the user has had the license.

By default these reports are based on all the installed products and licenses on all the servers contained in the current snapshot. The scope of any of these reports can be limited by applying one or more View Filters. The View filter allows the report to be scoped by server, vendor, product, or user. To change the View filter:

1. Select **Filter...** from the **View** pulldown menu.
2. From the **View filter** dialog select the type of filter to apply.
3. Select **OK** to close the individual filter selection dialog. Select **OK** to close the **View filter** dialog. The view will be immediately updated based on the new view when the **View filter** dialog is closed.

It is important to remember that these reports are only as current as the last snapshot. The snapshot can be updated manually or automatically.

To update the snapshot manually, select **Refresh now** from the **Snapshot** pulldown menu. The snapshot will be immediately updated,

To update the snapshot automatically, open the **Automatic refresh** dialog from the **Snapshot** pulldown menu. Select the **Automatic refresh** radio button, and enter a refresh interval in minutes.

GUI License Event Reporting

These reports are generated by querying information directly from a server or servers. Since the amount of logged event information may be extensive it is impractical to create a local snapshot of all the log information to generate reports from.

The reports can be filtered using the same View Filter as previously discussed. A log report can be scoped by server, vendor, product, or user. By default, the View filter dialogs allow the administrator to select from the servers, vendors, products, and users which are contained in the current snapshot. If the desired filter item is not contained in the current snapshot, the administrator can manually specify the name in an entry field on the filter dialog.

There are five log reports which are summarized below.

- License event log reports which reports logged server events without deriving additional information. There are seven categories of events which can be included in this reports.

  1. All events
  2. (default) License related events (license request, license release, etc.)
  3. Vendor messages
  4. License database modifications.
  5. Error events (license request failed, vendor not found, etc.)
  6. Server start/stop
  7. Fatal error events (server out of memory, server file IO error, etc.)

  Note that error events 2-7 can be combined.

- License requests by product. For each product lists the number of requests for licenses, the number of licenses granted, and the percent of rejected requests.
- License requests by user. Lists the same information and the previous reports, but breaks out a separate entry for each user.
- License use by product. For each product lists the maximum concurrent nodes, maximum concurrent users, and average time in use.
- License use by user. For each product, lists the number of times each user invoked the product and the average time the product was in use.

AUTHOR

i4admin a product of Isogon Corporation.

FILES

/opt/ifor/ls/conf/i4rpt.fmt	Report templates
/opt/ifor/ls/res/*.bmp	Icon bitmaps
/opt/ifor/ls/res/i4admin.pdl	Panel definitions
/opt/ifor/ls/doc/i4admin.pdf	LicensePower/iFOR Administrator's Guide (Adobe Acrobat format)
/opt/ifor/ls/doc/i4admin.ps	LicensePower/iFOR Administrator's Guide (postscript format)

SEE ALSO

LicensePower/iFOR Administrator's Guide, i4lmd(1M), i4start(1M), i4stop(1M), i4target(1M), i4tv(1M).

(LicensePower/iFOR Version 4.0)

NAME

i4lmd - starts the license server on a local node

SYNOPSIS

i4lmd [**-s**[**ecure**]] [**-l**[**ogname**]] [**-v**[**erbose**]] [**-z**[**debugging**]] [**-n**[**o**] *event_types*] [**-c**[**oldstart**]]

DESCRIPTION

The **i4lmd** command starts a license server on the local node. There is no graphic interface for this command, the shell script **i4config** is used to configure the license server. License servers should not be run manually.

A printable on-line administration guide is also available. (See the *FILES* section below.)

NOTE: Please refer to the release notes and **i4config** for information on how to automate the start-up of **i4lmd** on your specific platform.

Options

-s Secure mode. A LicensePower/iFOR license server running in secure mode will only permit modifications to its database from tools run locally (on the same node). Tools running on remote node are not permitted to modify the database.

-l *log_name* Redirects license server log entries to a file and location other than the default (**/opt/ifor/ls/conf/logdb***). The alternate log file specification (*filename*) must be fully qualified starting from the root directory (/).

-v *verbose* The verbose flag should only be used by administrators the event of a server failure. This command allows the administrator to review license calls and activity from the client programs. The **-v** option is used in conjunction with **-z**.

-z *debugging* The debugging flag allows the administrator to review all rpc communication between the clients and the server. The **-z** option is used in conjunction with **-v**.

-no Turns off logging of the events specified in *event_list*. Any combination of events is valid, but items in the list of events must not be separated by spaces or other characters. Following are the event types that you may specify:

- **l** License-grant and license-release events.
- **c** License checkin events. (Licensed products usually check in with the license server at regular intervals while a user is using the product).
- **w** Waiting events: these include wait events (a user was waiting for a license), waitgrant events (a user was waiting for and then was granted a license), and waitremove events (a user was waiting for a license and then asked to be removed from the queues before a license was granted).
- **v** Vendor events: a vendor was added, renamed or deleted.
- **p** Product events: a product was added, renamed, or deleted.
- **e** Error events.
- **t** License timeout events. (When a licensed product fails to check in with the license server, it may stop running after it "times out." The vendor of the product sets the timeout interval, which is how long a product may run after it has lost contact with the license server).
- **m** Message events.
- **s** License server start/stop events.

-c This option will delete all transactions records from the database and subsequently that cache during server startup.

EXAMPLES

Start a license server; do not log checkin, vendor, product, timeout, or message events:

```
i4lmd -no cvptm
```

Start a license server, deleting all transactions from the database:

```
i4lmd -c
```

Start a license server, overriding the default log file:

i4lmd -l */logs/license_server_log*

AUTHOR

i4lmd is a product of Isogon Corporation.

FILES

/opt/ifor/ls/bin/i4lmd	
/opt/ifor/ls/bin/i4config	
/opt/ifor/ls/doc/i4admin.pdf	LicensePower/iFOR Administrator's Guide (Adobe Acrobat format)
/opt/ifor/ls/doc/i4admin.ps	LicensePower/iFOR Administrator's Guide (postscript format)

SEE ALSO

LicensePower/iFOR Administrator's Guide, i4admin(1M), i4start(1M), i4stop(1M), i4target(1M), i4tv(1M).

NAME

i4start - LicensePower/iFOR server start tool

SYNOPSIS

`i4start`

DESCRIPTION

The **`i4start`** tool can be used to manually re-start a LicensePower/iFOR license server that has been stopped (for instance, with the **`i4stop`** tool). It will also start location brokers, if they are needed on the system. The settings of the tool are activated after the first invocation of **`i4config`**.

A printable on-line administration guide is also available. (See the *FILES* section below.)

EXAMPLES

`i4start`

FILES

`/opt/ifor/ls/bin/i4start`	
`/opt/ifor/ls/bin/i4config`	
`/opt/ifor/ls/doc/i4admin.pdf`	LicensePower/iFOR Administrator's Guide (Adobe Acrobat format)
`/opt/ifor/ls/doc/i4admin.ps`	LicensePower/iFOR Administrator's Guide (postscript format)

AUTHOR

`i4start` is a product of Isogon Corporation.

SEE ALSO

LicensePower/iFOR Administrator's Guide, i4admin(1M), i4lmd(1M), i4stop(1M), i4target(1M), i4tv(1M).

NAME

i4stop - LicensePower/iFOR server stop tool

SYNOPSIS

i4stop

DESCRIPTION

The **i4stop** tool can be used to manually stop a LicensePower/iFOR license server (and location brokers) if they are running on the system. Use this tool **on** the system that contains the active LicensePower/iFOR license server that you want to stop. The tool is located in **/opt/ifor/ls/bin**.

A printable on-line administration guide is also available. (See the *FILES* section below.)

EXAMPLES

i4stop

FILES

/opt/ifor/ls/bin/i4stop

/opt/ifor/ls/doc/i4admin.pdf	LicensePower/iFOR Administrator's Guide (Adobe Acrobat format)
/opt/ifor/ls/doc/i4admin.ps	LicensePower/iFOR Administrator's Guide (postscript format)

AUTHOR

i4stop is a product of Isogon Corporation.

SEE ALSO

LicensePower/iFOR Administrator's Guide, i4admin(1M), i4lmd(1M), i4start(1M), i4target(1M), i4tv(1M).

NAME

i4target - returns the local LicensePower/iFOR target id

SYNOPSIS

i4target

i4target [**-c**] [**-C**] [**-h**] [**-H**] [**-o**] [**-O**] [**-q**] [**-Q**] [**-v**] [**-V**]

DESCRIPTION

i4target is used to find the target ID that can be used by LicensePower/iFOR for locking licenses to a particular system.

To create LicensePower/iFOR licenses for an application, an application supplier will need the target ID of the machine where the LicensePower/iFOR licenses will be installed. The target ID tool (**i4target**) should be run on the machine where you want to identify a LicensePower/iFOR target ID. For server-based licensing, this will be the machine that is executing the license server (**i4lmd**) where you plan to install this application supplier's licenses. For nodelocked licensing, this will be the system where the application will be executing.

The algorithm that is used to identify a LicensePower/iFOR target ID may vary depending on operating system platform.

For example: On an HP-UX machine licenses managed by the **i4lmd** (concurrent and use once licenses), the LicensePower/iFOR target ID is derived from the link level address of the LAN card accessed by the device file **/dev/i4target** on the machine that is running the **i4lmd**. If **/dev/i4target** does not exist and the super-user is executing **i4target**, **i4target** will create **/dev/i4target**. On an HP 9000 Series 700 or 800, the device file will be for the lan0 LAN card. This is the same method used by the i4lmd for determining the LicensePower/iFOR ID of the machine on which it is executing.

On HP-UX, for LicensePower/iFOR nodelocked licenses, the LicensePower/iFOR ID is derived from:

- The LAN card accessed by **/dev/i4target**, or
- The built in SPU ID number, or
- An HIL ID Module.

A printable on-line administration guide is also available. (See the *FILES* section below.)

Options

-c -C	Change the permanent target ID value.
-h -H	Help. Display a list of options.
-o -O	Display operating system name.
-q -Q	Display target ID in quiet mode (without headers).
-v -V	Display a verbose list of the LicensePower/iFOR target IDs from each possible source. The list consist of the link level address of the installed LAN cards. A super-user can then use the address to change to an alternate LAN card. This lets you change the IO slot where a LAN card is installed without losing the use of LicensePower/iFOR licenses locked to that LAN card.

RETURN VALUE

i4target always returns 0.

DIAGNOSTICS

Messages displayed during execution are self-explanatory.

EXAMPLES

To find the current local LicensePower/iFOR target ID(s):

i4target

Examples for each of the options are shown below:

i4target -c or **i4target -C**

```
Current Permanent Target ID: 3e53d0
```

i

```
1. Target ID value: 3e53d0
        LAN card at logical unit 0

There is only one choice for the new Permanent Target ID.
Enter '1' to select it; enter any other character to abort: 1

New Permanent Target ID: 3e53d0

NOTE: i4lmd must be restarted for the new
      Permanent Target ID to take effect.
```

i4target -h or **i4target -H**

```
Usage:
  i4target  [options]
        options are:
                -[vV] : verbose mode; detailed output
                -[qQ] : quiet mode; no headers in output
                -[cC] : change Permanent Target ID;
                -[hH] : displays this message
                -[oO] : displays os name
```

i

i4target -o or **i4target -O**

```
HP-UX
```

i4target -q or **i4target -Q**

```
3e53d0
```

i4target -v or **i4target -V**

```
Permanent Target ID: 3e53d0

SPU Target ID: 70328251

The Permanent Target ID is derived from a permanent hardware source
on the system from which the i4target program is executed.
This target ID may be used for all license types.

The SPU ID is derived from a hardware identification number on the
SPU.  It is used as the Permanent Target ID when no higher-priority
sources for Permanent Target ID (i.e., LAN cards) are present.
```

AUTHOR

i4target is a product of Isogon Corporation.

FILES

/opt/ifor/ls/bin/i4target	
/opt/ifor/ls/doc/i4admin.pdf	LicensePower/iFOR Administrator's Guide (Adobe Acrobat format)
/opt/ifor/ls/doc/i4admin.ps	LicensePower/iFOR Administrator's Guide (postscript format)

SEE ALSO

LicensePower/iFOR Administrator's Guide, http://www.isogon.com for latest information on i4target. i4admin(1M), i4lmd(1M), i4start(1M), i4stop(1M), i4tv(1M).

NAME

i4tv - verify that LicensePower/iFOR License Servers are working

SYNOPSIS

i4tv [**-n** *hostname* | **-z** | **-v**] [**-h** | **-usage** | **-version**]

DESCRIPTION

The **i4tv** tool can be used after the license servers have been started to verify that that they are running properly. The **i4tv** program resides in the **/opt/ifor/ls/bin** directory. A message describing a completed license transaction and a list of all license servers will be displayed. Once a license server has been configured using **i4config**, the **i4tv** tool is used to quickly verify the status of the license server **i4lmd**.

Options

-n *hostname*	The **-n** option is used to check that the specified machine is running a license server. It returns **0** if the hostname is running **i4lmd** and it returns **1** if the hostname is not running **i4lmd**.
-z	The **-z** option turns on RPC tracing messages, which can be used to diagnose problems.
-v	Displays progress messages during the license request operation.
-h	Displays command usage information (same as **-usage**).
-usage	Displays command usage information (same as **-h**).
-version	Displays command version information.

If you can run **i4tv** successfully but are still having a problem with a licensed product, the problem is probably with the licenses, or possibly with the product itself: in this case, talk to the vendor of the licensed software product.

If you can not run **i4tv** successfully or it takes more than 10 seconds to retrieve a license, verify that **glbd** and **i4lmd** are running. Use the utility **lb_admin** to clean the database. Answer YES to all database entries that do not respond. If you receive one of the error messages listed below, use the explanation of the error to fix the problem. Then try running **i4tv** again.

If you can not run **i4tv** successfully and receive an error that's not listed below, it means there is a problem with the software on which LicensePower/iFOR ARK is layered (for example, TCP), or a hardware problem.

A printable on-line administration guide is also available. (See the *FILES* section below.)

ERROR MESSAGES

netls_no_svrs_found	No license servers are running or someone has deleted the LicensePower/iFOR Test Vendor from the license servers.
netls_license_not_found	Someone has deleted the Test Vendor licenses that each server automatically installs the first time that it starts. This prohibits anyone from using the test and verification tool (**i4tv**).
netls_not_authorized	Someone has edited the user file to restrict the use of **i4tv**.
netls_bad_timestamp	System clocks have not been synchronized to within 12 hours.

EXAMPLES

Run the **i4tv** test and verification tool:

```
i4tv
```

```
i4TV Version 4.0 -- LicensePower/iFOR Test and Verification Tool
A product of Isogon Corporation
Completed license transaction on node 3541b8 running LicensePower/iFOR 4.0
Active LicensePower/iFOR Servers:
   hp_snake.gradient.com (HP-UX) running LicensePower/iFOR Version 3.0.0
```

Check for the presence of the license server hp1030:

```
i4tv -n hp1030
```

```
A product of Isogon Corporation
hp1030 running
```

AUTHOR

i4tv is a product of Isogon Corporation.

FILES

/opt/ifor/ls/bin/i4tv

/opt/ifor/ls/doc/i4admin.pdf	LicensePower/iFOR Administrator's Guide (Adobe Acrobat format)
/opt/ifor/ls/doc/i4admin.ps	LicensePower/iFOR Administrator's Guide (postscript format)

SEE ALSO

LicensePower/iFOR Administrator's Guide, i4admin(1M), i4lmd(1M), i4start(1M), i4stop(1M), i4target(1M).

NAME

identd - TCP/IP IDENT protocol server

SYNOPSIS

/usr/lbin/identd [**-i**| **-w**|**-b**] [**-t***seconds*] [**-u***uid*] [**-g***gid*] [**-p***port*] [**-a***address*] [**-c***charset*] [**-n**] [**-o**] [**-e**] [**-l**] [**-V**] [**-m**] [**-N**] [**-d**] [*kernelfile* [*kmemfile*]]

DESCRIPTION

identd is a server which implements the TCP/IP proposed standard IDENT user identification protocol as specified in the RFC 1413 document.

identd operates by looking up specific TCP/IP connections and returning the user name of the process owning the connection.

Arguments

-i The **-i** flag, which is the default mode, should be used when starting the daemon from **inetd** with the "nowait" option in the **/etc/inetd.conf** file. Use of this mode will make **inetd** start one **identd** daemon for each connection request.

-w The **-w** flag should be used when starting the daemon from **inetd** with the "wait" option in the **/etc/inetd.conf** file. This is the preferred mode of operation since that will start a copy of **identd** at the first connection request and then **identd** will handle subsequent requests without having to do the nlist lookup in the kernel file for every request as in the **-i** mode above. The **identd** daemon will run either forever, until a timeout, as specified by the **-t** flag, occurs.

-b The **-b** flag can be used to make the daemon run in standalone mode without the assistance from **inetd**. This mode is the least preferred mode, and not supported by HP, since a bug or any other fatal condition in the server will make it terminate and it will then have to be restarted manually. Other than that is has the same advantage as the **-w** mode in that it parses the nlist only once.

-t*seconds*
The **-t***seconds* option is used to specify the timeout limit. This is the number of seconds a server started with the **-w** flag will wait for new connections before terminating. The server is automatically restarted by **inetd** whenever a new connection is requested if it has terminated. A suitable value for this is 120 (2 minutes), if used. It defaults to no timeout (ie, will wait forever, or until a fatal condition occurs in the server).

-u*uid* The **-u***uid* option is used to specify a user id number which the **ident** server should switch to after binding itself to the TCP/IP port if using the **-b** mode of operation.

-g*gid* The **-g***gid* option is used to specify a group id number which the **ident** server should switch to after binding itself to the TCP/IP port if using the **-b** mode of operation.

-p*port* The **-p***port* option is used to specify an alternative port number to bind to if using the **-b** mode of operation. It can be specified by name or by number. Defaults to the IDENT port (113).

-a*address*
The **-a***address* option is used to specify the local address to bind the socket to if using the **-b** mode of operation. Can only be specified by IP address and not by domain name. Defaults to the INADDR_ANY address which normally means all local addresses.

-V The **-V** flag makes **identd** display the version number and the exit.

-l The **-l** flag tells **identd** to use the System logging daemon **syslogd** for logging purposes.

-o The **-o** flag tells **identd** to not reveal the operating system type it is run on and to instead always return "OTHER".

-e The **-e** flag tells **identd** to always return "UNKNOWN-ERROR" instead of the "NO-USER" or "INVALID-PORT" errors.

-c*charset*
The **-c***charset* flags tells **identd** to add the optional (according to the IDENT protocol) character set designator to the reply generated. <charset> should be a valid character set as described in the MIME RFC in upper case characters.

-n	The **-n** flags tells **identd** to always return user numbers instead of user names if you wish to keep the user names a secret.
-N	The **-N** flag makes **identd** check for a file **.noident** in each homedirectory for a user which the daemon is about to return the user name for. It that file exists then the daemon will give the error HIDDEN-USER instead of the normal USERID response.
-m	The **-m** flag makes **identd** use a mode of operation that will allow multiple requests to be processed per session. Each request is specified one per line and the responses will be returned one per line. The connection will not be closed until the connecting part closes it's end of the line. *Please note that this mode violates the protocol specification as it currently stands*.
-d	The **-d** flag enables some debugging code that normally should *NOT* be enabled since that breaks the protocol and may reveal information that should not be available to outsiders.
kernelfile	*kernelfile* defaults to the normally running kernel file.
kmemfile	*kmemfile* defaults to the memory space of the normally running kernel.

INSTALLATION

identd is invoked either by the internet server (see *inetd*(1M)) for requests to connect to the IDENT port as indicated by the **/etc/services** file (see *services*(5)) when using the **-w** or **-i** modes of operation or started manually by using the **-b** mode of operation.

EXAMPLES

Since the server is located in **/usr/lbin/identd** one can put either:

```
ident stream tcp wait bin /usr/lbin/identd identd -w -t120
```

or:

```
ident stream tcp nowait bin /usr/lbin/identd identd -i
```

into the **/etc/inetd.conf** file.

To start it using the unsupported **-b** mode of operation one can put a line like this into the **/sbin/init.d/sendmail** file under the 'start' section:

```
/usr/lbin/identd -b -u2 -g2
```

This will cause **identd** to be started as daemon whenever **sendmail** is running. It will run in the background as user 2, group 2 (user 'bin', group 'bin').

SEE ALSO

inetd.conf(4).

NAME

ifconfig - configure network interface parameters

SYNOPSIS

ifconfig *interface address_family* [*address* [*dest_address*]] [*parameters*]

ifconfig *interface* [*address_family*]

DESCRIPTION

The first form of the **ifconfig** command assigns an address to a network interface and/or configures network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine. It can also be used at other times to redefine an interface's address or other operating parameters.

The second form of the command, without *address_family*, displays the current configuration for *interface*. If *address_family* is also specified, **ifconfig** reports only the details specific to that address family.

Only a user with appropriate privileges can modify the configuration of a network interface. All users can run the second form of the command.

Arguments

ifconfig recognizes the following arguments:

address Either a host name present in the host name database (see *hosts*(4)), or a DARPA Internet address expressed in Internet standard dot notation (see *inet*(3N)).

address_family Name of protocol on which naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. Therefore, it is necessary to specify the *address_family*, which may affect interpretation of the remaining parameters on the command line. The only address family currently supported is **inet** (DARPA-Internet family).

dest_address Address of destination system. Consists of either a host name present in the host name database (see *hosts*(4)), or a DARPA Internet address expressed in Internet standard dot notation (see *inet*(3N)).

interface A string of the form *name unit*, such as **lan0**. (See the Interface Naming subsection given below.)

parameters One or more of the following operating parameters:

- **up** Mark an interface "up". Enables interface after an **ifconfig down**. Occurs automatically when setting the address on an interface. Setting this flag has no effect if the hardware is "down".

- **down** Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface.

- **broadcast** (Inet only) Specify the address that represents broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

- **metric** *n* Set the routing metric of the interface to *n*. The default is 0. The routing metric is used by the routing protocol (see *gated*(1M)). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

- **netmask** *mask*
  (Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks or aggregating networks into supernets. *mask* can be specified as a single hexadecimal number with a leading **0x**, with a dot-notation Internet address, or with a pseudo-network name listed in the network table (see *networks*(4)). For subdividing networks into sub-networks, *mask* must include the network part of the local address, and the subnet part which is taken from the host field of the address. *mask* must contain 1's in the bit positions in the 32-bit address that are to be used for the network and subnet parts, and 0's in the host part. The 1's in the *mask* must be

contiguous starting from the leftmost bit position in the 32-bit field. *mask* must contain at least the standard network portion, and the subnet field must be contiguous with the network portion. The subnet field must contain at least 2 bits. The subnet part after performing a bit-wise AND operation between the *address* and the *mask* must not contain all 0's or all 1's. For aggregating networks into supernets, *mask* must only include a portion of the network part. *mask* must contain contiguous 1's in the bit positions starting from the leftmost bit of the 32-bit field.

arp Enable the user of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). If an interface already had the Address Resolution Protocol disabled, the user must "unplumb" the interface before it can be enabled for Address Resolution Protocol.

-arp Disable the use of the Address Resolution Protocol. If an interface already had the Address Resolution Protocol enabled, the user must "unplumb" the interface before it can be disabled for Address Resolution Protocol.

plumb Setup the Streams plumbing needed for TCP/IP for a primary interface name. (See the Interface Naming subsection given below.). By default, the **plumb** operation is done automatically when an IP address is specified for an interface.

unplumb Tear down the Streams plumbing for a primary interface name. (See the Interface Naming subsection given below.) Secondary interface does not require "plumbing" and it can be removed by assigning an IP address of 0.0.0.0.

Interface Naming

The *interface* name associated with a network card is composed of the *name* of the interface (e.g. **lan** or **snap**), the *ppa number* which identifies the card instance for this interface, and an optional *IP index number* which allows the configuration of multiple IP addresses for an interface. For LAN cards, the *interface* name **lan** will be used to designate Ethernet encapsulation and **snap** for IEEE 802.3 encapsulation. The **lanscan** command can be used to display the *interface* name and *ppa number* of each interface that is associated with a network card (see *lanscan*(1M)).

Multiple IP addresses assigned to the same *interface* may be in different subnets. An example of an interface name without an *IP index number* is **lan0**. An example of an interface name with a *IP index number* is **lan0:1**. Note: specifying **lan0:0** is equivalent to **lan0**.

Loopback Interface

The loopback interface (**lo0**) is automatically configured when the system boots with the TCP/IP software. The default IP address and netmask of the loopback interface are 127.0.0.1 and 255.0.0.0, respectively. The user is not permitted to change the address of the primary loopback interface (**lo0:0**).

Supernets

A supernet is a collection of smaller networks. Supernetting is a technique of using the netmask to aggregate a collection of smaller networks into a supernet.

This technique is particularly useful when the limit of 254 hosts per class C network is too restrictive. In those situations a netmask containing only a portion of the network part may be applied to the hosts in these networks to form a supernet. This supernet netmask should be applied to those interfaces that connect to the supernet using the *ifconfig* command. For example, a host can configure its interface to connect to a class C supernet, 192.6, by configuring an IP address of 192.6.1.1 and a netmask of 255.255.0.0 to its interface.

DIAGNOSTICS

Messages indicate if the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

AUTHOR

ifconfig was developed by HP and the University of California, Berkeley.

SEE ALSO

netstat(1), lanscan(1M), hosts(4), routing(7).

NAME

inetd - Internet services daemon

SYNOPSIS

/usr/sbin/inetd [**-c**]

/usr/sbin/inetd [**-k**]

/usr/sbin/inetd [**-l**]

DESCRIPTION

The **inetd** daemon is the Internet superserver, which invokes Internet server processes as needed. It must be running before other hosts can connect to the local host through **ftp**, **rcp**, **remsh**, **rlogin**, and **telnet**. The **inetd** daemon also supports services based on the Remote Procedure Call (RPC) protocol (NFS), such as **rwalld** and **rusersd**. If RPC servers are started by **inetd**, the **portmap** server (see *portmap*(1M)) must be started before **inetd**.

The **inetd** daemon is designed to invoke all the Internet servers as needed, thus reducing load on the system. It is normally started at system boot time. Only one **inetd** can run at any given time.

The **inetd** daemon starts servers for both stream and datagram type services. For stream services, **inetd** listens for connection requests on Internet stream sockets. When a connection is requested for one of its sockets, **inetd** decides which service the socket will support, forks a process, invokes an appropriate server for the connection, and passes the connected socket to the server as **stdin** and **stdout**. Then **inetd** returns to listening for connection requests.

For datagram services, **inetd** waits for activity on Internet datagram sockets. When an incoming datagram is detected, **inetd** forks a process, invokes an appropriate server, and passes the socket to the server as **stdin** and **stdout**. Then **inetd** waits, ignoring activity on that datagram socket, until the server exits.

The **inetd** daemon is normally started by the **/sbin/init.d/inetd** script, which is invoked during the boot-time initialization. Otherwise, **inetd** can be started only by the superuser.

The Internet daemon and the servers it starts inherit the **LANG** and **TZ** environment variables and the **umask** of the process that started **inetd**. If **inetd** is started by the superuser, it inherits the superuser's umask, and passes that umask to the servers it starts.

When invoked, **inetd** reads **/etc/inetd.conf** and configures itself to support whatever services are included in that file (see *inetd.conf*(4)). The **inetd** daemon also performs a security check if the file **/var/adm/inetd.sec** exists (see *inetd.sec*(4)). If the Internet daemon refuses a connection for security reasons, the connection is shut down. Most RPC-based services, if their first connection is refused, attempt to connect four more times at 5-second intervals before timing out. In such cases, **inetd** refuses the connection from the same service invocation five times. This is visible in the system log if **inetd** connection logging and **syslogd** logging for the **daemon** facility are both enabled (see *syslogd*(1M)).

The **inetd** daemon provides several "trivial" services internally by use of routines within itself. The services are **echo**, **discard**, **chargen** (character generator), **daytime** (human readable time), and **time** (machine readable time in the form of the number of seconds since midnight, January 1, 1900). The **inetd** daemon provides both TCP- and UDP-based servers for each of these services. See *inetd.conf*(4) for instructions on configuring internal servers.

Options

inetd recognizes the following options. These options can be used only by a superuser.

-c Reconfigure the Internet daemon; in other words, force the current **inetd** to reread **/etc/inetd.conf**. This option sends the signal **SIGHUP** to the Internet daemon that is currently running. Any configuration errors that occur during the reconfiguration are logged to the **syslogd** daemon facility.

-k Kill the current **inetd**. This option sends the signal **SIGTERM** to the Internet daemon that is currently running, causing it to exit gracefully. This option is the preferred method of killing **inetd**.

-l By default, **inetd** starts with connection logging disabled. If no **inetd** is running, the **-l** option causes the **inetd** to start with connection logging enabled. Otherwise the **-l** option causes **inetd** to send the signal **SIGQUIT** to the **inetd** that is already running, which causes it to toggle the state of connection logging.

When connection logging is enabled, the Internet daemon logs attempted connections to services. It also logs connection attempts which fail the security check. This information can be useful when trying to determine if someone is repeatedly trying to access your system from a particular remote system (in other words, trying to break into your system). Successful connection attempts are logged to the **syslogd** daemon facility at the **info** log level. Connection attempts failing the security check are logged at the **notice** log level. **inetd** also logs whether the connection logging has been enabled or disabled at the *info* log level.

DIAGNOSTICS

The following diagnostics are returned by the Internet daemon before it disconnects from the terminal.

An inetd is already running

An attempt was made to start an Internet daemon when one was already running. It is incorrect to call the Internet daemon a second time without the **-c**, **-k**, or **-l** option.

There is no inetd running

An attempt was made to reconfigure an Internet daemon when none was running.

Inetd not found

This message occurs if **inetd** is called with **-c** and another Internet daemon is running but cannot be reconfigured. This occurs if the original Internet daemon died without removing its semaphore.

Next step: Use the **inetd -k** command to remove the semaphore left by the previous Internet daemon; then restart the daemon.

The following diagnostics are logged to the **syslogd** daemon facility. Unless otherwise indicated, messages are logged at the **error** log level.

/etc/inetd.conf: Unusable configuration file

The Internet daemon is unable to access the configuration file **/etc/inetd.conf**. The error message preceding this one specifies the reason for the failure.

/etc/inetd.conf: line *number***:** *error*

There is an error on the specified line in **/etc/inetd.conf**. The line in the configuration file is skipped. This error does not stop the Internet daemon from reading the rest of the file and configuring itself accordingly.

Next step: Fix the line with the error and reconfigure the Internet daemon by executing the **inetd -c** command.

*system_call***:** *message*

system_call failed. See the corresponding manual entry for a description of *system_call*. The reason for the failure is explained in *message*.

Cannot configure inetd

None of the services/servers listed in the configuration file could be set up properly, due to configuration file errors.

Too many services (max *n***)**

The number of active services listed in the configuration file exceeds the "hard" limit that can be supported by the system (see *setrlimit*(2)).

Next step: Reduce the number of services listed in the configuration file, then reconfigure the Internet daemon by running the command **inetd -c**.

*file***: \ found before end of line** *line*

file can be either **inetd.conf** or **inetd.sec**. If a backslash is not immediately followed by an end of line, it is ignored and the information up to the end of line is accepted. In this case, the next line of the file is not appended to the end of the current line. Unless all the information required is present on a single line, configuration file error messages are also output. This message is logged at the **warning** log level.

*service/protocol***: Unknown service**

The call to the library routine **getservbyname** (see *getservent*(3N)) failed. The service is not listed in **/etc/services**.

Next step: Include that service in **/etc/services** or eliminate the entry for the service in **/etc/inetd.conf**.

service/protocol: **Server failing (looping), service terminated.**

When **inetd** tries to start 40 servers within 60 seconds for a datagram service, other than **bootp**, **rpc**, or **tftp**, it assumes that the server is failing to handle the connection. To avoid entering a potentially infinite loop, **inetd** issues this message, discards the packet requesting the socket connection, and refuses further connections for this service. After 10 minutes, **inetd** tries to reinstate the service, and once again accepts connections for the service.

service/protocol: **socket:** *message*
service/protocol: **listen:** *message*
service/protocol: **getsockname:** *message*

Any one of the three errors above makes the service unusable. For another host to communicate with the server host through this service, the Internet daemon needs to be reconfigured after any of these error messages.

service/protocol: **bind:** *message*

If this error occurs, the service is temporarily unusable. After 10 minutes, **inetd** tries again to make the service usable by binding to the Internet socket for the service.

service/protocol: **Access denied to** *remote_host* **(***address***)**

The remote host failed to pass the security test for the indicated service. This information can be useful when trying to determine if someone is repeatedly trying to access your system from a particular remote system (in other words, trying to break into your system). This message is logged at the **warning** log level.

service/protocol: **Connection from** *remote_host* **(***address***)**

When connection logging is enabled, this message indicates a successful connection attempt to the specified service. This message is logged at the **notice** log level.

service/protocol: **Added service, server** *executable*

Keeps track of the services added when reconfiguring the Internet daemon. This message is logged at the **info** log level.

service/protocol: **New** *list*

Lists the new user IDs, servers or executables used for the service when reconfiguring the Internet daemon. This message is logged at the **info** log level.

service/protocol: **Deleted service**

Keeps track of the services deleted when reconfiguring the Internet daemon. This message is logged at the **info** log level.

Security File (inetd.sec) Errors

The following errors, prefixed by **/var/adm/inetd.sec:**, are related to the security file **inetd.sec**:

Field contains other characters in addition to * for *service*

For example, field 2 of the Internet address **10.5*.8.7** is incorrect.

Missing low value in range for *service*

For example, field 2 of the Internet address **10.-5.8.7** is incorrect.

Missing high value in range for *service*

For example, field 2 of the Internet address **10.5-.8.7** is incorrect.

High value in range is lower than low value for *service*

For example, field 2 of the Internet address **10.5-3.8.7** is incorrect.

allow/deny field does not have a valid entry for *service*

The entry in the allow/deny field is not one of the keywords **`allow`** or **`deny`**. No security for this service is implemented by **`inetd`** since the line in the security file is ignored. This message is logged at the **warning** log level.

RPC Related Errors for NFS Users

These errors are specific to RPC-based servers:

`/etc/inetd.conf: line` *number***`: Missing program number`**
`/etc/inetd.conf: line` *number***`: Missing version number`**

Error on the specified line of **`/etc/inetd.conf`**. The program or version number for an RPC service is missing. This error does not stop the Internet daemon from reading the rest of the file and configuring itself accordingly. However, the service corresponding to the error message will not be configured correctly.

Next step: Fix the line with the error, then reconfigure the Internet daemon by executing the **`inetd -c`** command.

`/etc/inetd.conf: line` *number***`: Invalid program number`**

Error on the specified line of **`/etc/inetd.conf`**. The program number for an RPC service is not a number. This error does not stop the Internet daemon from reading the rest of the file and configuring itself accordingly. However, the service corresponding to the error message will not be correctly configured.

Next step: Fix the line with the error, then reconfigure the Internet daemon by executing the **`inetd -c`** command.

AUTHOR

`inetd` was developed by HP and the University of California, Berkeley.

NFS was developed by Sun Microsystems, Inc.

FILES

`/etc/inetd.conf`	List of Internet server processes.
`/var/adm/inetd.sec`	Optional security file.

SEE ALSO

umask(1), portmap(1M), syslogd(1M), getservent(3N), inetd.conf(4), inetd.sec(4), protocols(4), services(4), environ(5).

NAME

inetsvcs_sec - enable/disable secure internet services

SYNOPSIS

inetsvcs_sec [**enable**| **disable**| **status**]

DESCRIPTION

/usr/sbin/inetsvcs_sec is used to enable or disable secure internet services (SIS) by updating inetsvcs.conf(4) with the appropriate entry. SIS provide network authentication when used in conjunction with HP DCE security services, the HP Praesidium/Security Server, or other software products that provide a Kerberos V5 Network Authentication Services environment.

Options

inetsvcs_sec recognizes the following options:

enable The secure internet services are enabled. The services now provide network authentication through Kerberos V5.

disable The secure internet services are disabled. The services now follow the traditional behavior of prompting for passwords.

status This option displays the current authentication mechanism used (i.e., whether Kerberos authentication is enabled or not).

SEE ALSO

sis(5), inetsvcs.conf(4).

NAME

infocmp - compare or print out terminfo descriptions

SYNOPSIS

infocmp [**-d**] [**-c**] [**-n**] [**-I**] [**-L**] [**-C**] [**-r**] [**-u**] [**-s d**|**i**|**l**|**c**] [**-v**] [**-V**] [**-1**]
[**-w** *width*] [**-A** *directory*] [**-B** *directory*] [*termname*...]

DESCRIPTION

infocmp can be used to compare a binary **terminfo** entry with other terminfo entries, rewrite a **terminfo** description to take advantage of the **use=** terminfo field, or print out a **terminfo** description from the binary file (**term**) in a variety of formats. In all cases, the boolean fields will be printed first, followed by the numeric fields, followed by the string fields.

Default Options

If no options are specified and zero or one *termnames* are specified, the **-I** option will be assumed. If more than one *termname* is specified, the **-d** option will be assumed.

Comparison Options [-d] [-c] [-n]

infocmp compares the **terminfo** description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termnames*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: **F** for boolean variables, **-1** for integer variables, and **NULL** for string variables.

-d produces a list of each capability that is different between two entries. This option is useful to show the difference between two entries, created by different people, for the same or similar terminals.

-c produces a list of each capability that is common between two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the **-u** option is worth using.

-n produces a list of each capability that is in neither entry. If no *termnames* are given, the environment variable **TERM** will be used for both of the *termnames*. This can be used as a quick check to see if anything was left out of a description.

Source Listing Options [-I] [-L] [-C] [-r]

The **-I**, **-L**, and **-C** options will produce a source listing for each terminal named.

-I use the **terminf** names

-L use the long C variable name listed in <**term.h**>

-C use the **termcap** names

-r when using **-C**, put out all capabilities in **termcap** form

If no *termnames* are given, the environment variable **TERM** will be used for the terminal name.

The source produced by the **-C** option may be used directly as a **termcap** entry, but not all of the parameterized strings may be changed to th **termcap** format. **infocmp** will attempt to convert most of the parameterized information, but anything not converted will be plainly marked in the output and commented out. These should be edited by hand.

All padding information for strings will be collected together and placed at the beginning of the string where **termcap** expects it. Mandatory padding (padding information with a trailing '/') will become optional.

All **termcap** variables no longer supported by **terminfo**, but which are derivable from other **terminfo** variables, will be output. Not all **terminfo** capabilities will be translated; only those variables which were part of **termcap** will normally be output. Specifying the **-r** option will take off this restriction, allowing all capabilities to be output in **termcap** form.

Note that because padding is collected to the beginning of the capability, not all capabilities are output. Mandatory padding is not supported. Because **termcap** strings are not as flexible, it is not always possible to convert a **terminfo** string capability into an equivalent **termcap** format. A subsequent conversion of the **termcap** file back into **terminfo** format will not necessarily reproduce the original **terminfo** source.

Some common **terminfo** parameter sequences, their **termcap** equivalents, and some terminal types which commonly have such sequences, are:

terminfo	**termcap**	Representative Terminals
%p1%c	**%.**	adm
%p1%d	**%d**	hp, ANSI standard, vt100
%p1%'x'%+%c	**%+x**	concept
%i	**%i**	ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;	**%>xy**	concept
%p2 is printed before **%p1**	**%r**	hp

Use= Option [-u]

-u produces a **terminfo** source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals *termnames*. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with **use=** fields for the other terminals. In this manner, it is possible to retrofit generic terminfo entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times or by different people so that each description is a full description, using **infocmp** will show what can be done to change one description to be relative to the other.

A capability will get printed with an at-sign (@) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value gets printed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability gives a different value for the capability than that in the first *termname*.

The order of the other *termname* entries is significant. Since the terminfo compiler **tic** does a left-to-right scan of the capabilities, specifying two **use=** entries that contain differing entries for the same capabilities will produce different results depending on the order that the entries are given in. **infocmp** will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability after a **use=** entry that contains that capability will cause the second specification to be ignored. Using **infocmp** to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying extra **use=** fields that are superfluous. **infocmp** will flag any other *termname* **use=** fields that were not needed.

Other Options [-s d|i|l|c] [-v] [-V] [-1] [-w *width***]**

-s sorts the fields within each type according to the argument below:

- **d** leave fields in the order that they are stored in the **terminfo** database.
- **i** sort by **terminfo** name.
- **l** sort by the long C variable name.
- **c** sort by the **termcap** name.

If the **-s** option is not given, the fields printed out will be sorted alphabetically by the **terminfo** name within each type, except in the case of the **-C** or the **-L** options, which cause the sorting to be done by the **termcap** name or the long C variable name, respectively.

-v prints out tracing information on standard error as the program runs.

-V prints out the version of the program in use on standard error and exit.

-1 causes the fields to be printed out one to a line. Otherwise, the fields will be printed several to a line to a maximum width of 60 characters.

-w changes the output to *width* characters.

Changing Databases [-A *directory***] [-B** *directory***]**

The location of the compiled **terminfo** database is taken from the environment variable **TERMINFO**. If the variable is not defined, or the terminal is not found in that location, the system **terminfo** database, usually in **/usr/lib/terminfo**, will be used. The options **-A** and **-B** may be used to override this location. The **-A** option will set **TERMINFO** for the first *termname* and the **-B** option will set **TERMINFO** for the other *termnames*. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

FILES

`/usr/lib/terminfo/?/*`	Compiled terminal description database.

SEE ALSO

curses_intro(3X), captoinfo(1M), terminfo(4), tic(1M).

NAME

init - process control initialization

SYNOPSIS

/sbin/init [**0**|**1**|**2**|**3**|**4**|**5**|**6**|**S**|**s**|**Q**|**q**|**a**|**b**|**c**]

DESCRIPTION

The **init** daemon and command is a general process spawner. Its primary role is to create processes from a script stored in the file **/etc/inittab** (see *inittab*(4)). This file usually has **init** spawn a **getty** on each line where users can log in. It also controls autonomous processes required by any particular system.

At boot time, **init** is started as a system daemon.

While the system is running, a user-spawned **init** directs the actions of the boot **init**. It accepts a one-character argument and signals the boot **init** with the **kill()** system call to perform the appropriate action.

The arguments have the following effect:

0–**6**	Place the system in one of the run levels **0** through **6**.
a|**b**|**c**	Process the **inittab** entries that have the special "run level" **a**, **b**, or **c**, without changing the numeric run level.
Q|**q**	Re-examine the **inittab** entries without changing the run level.
S|**s**	Enter the single-user environment. When this level change occurs, the logical system console **/dev/syscon** is changed to the terminal from which the command was executed.

Boot **init** considers the system to be in a **run level** at any given time. A run level can be viewed as a software configuration of the system, where each configuration allows only a selected group of processes to exist. The processes spawned by boot **init** for each of these run levels are defined in the **inittab** file. Boot **init** can be in one of eight run levels, **0**–**6**, and **S** or **s**. The run level is changed by having a privileged user run the **init** command. This user-spawned **init** sends appropriate signals to the boot **init**.

Boot **init** is invoked inside the HP-UX system as the last step in the boot procedure. Boot **init** first performs any required machine-dependent initialization, such as setting the system context. Next, boot **init** looks for the **inittab** file to see if there is an entry of the type **initdefault** (see *inittab*(4)). If an **initdefault** entry is found, boot **init** uses the run level specified in that entry as the initial run level to enter. If this entry is not in **inittab**, or **inittab** is not found, boot **init** requests that the user enter a run level from the logical system console, **/dev/syscon**. If **S** or **s** is entered, boot **init** goes into the **single-user** level. This is the only run level that does not require the existence of a properly formatted **inittab** file. If **inittab** does not exist, then by default the only legal run level that boot **init** can enter is the single-user level.

In the single-user level, the logical system console terminal **/dev/syscon** is opened for reading and writing, and the command **/usr/bin/su**, **/usr/bin/sh**, or **/sbin/sh** is invoked immediately. To exit from the single-user run level, one of two options can be selected:

- If the shell is terminated with an end-of-file, boot **init** reprompts for a new run level.
- User **init** can signal boot **init** and force it to change the current system run level.

When attempting to boot the system, some processes spawned by boot **init** may send display messages to the system console (depending on the contents of **inittab**). If messages are expected but do not appear during booting, it may be caused by the logical system console (**/dev/syscon**) being linked to a device that is not the physical system console (**/dev/systty**). If this occurs, you can force boot **init** to relink **/dev/syscon** to **/dev/systty** by pressing the DEL (delete) key (ASCII 127) on the physical system console.

When boot **init** prompts for the new run level, you can only enter one of the digits **0** through **6** or the letter **S** or **s**. If you enter **S**, boot **init** operates as previously described in single-user mode with the additional result that **/dev/syscon** is linked to the user's terminal line, thus making it the logical system console. A message is generated on the physical system console, **/dev/systty**, identifying the new logical system console.

When boot **init** comes up initially, and whenever it switches out of single-user state to normal run states, it sets the states (see *ioctl*(2)) of the logical system console, **/dev/syscon**, to those modes saved in the file **/etc/ioctl.syscon**. This file is written by boot **init** whenever single-user mode is entered. If this file does not exist when boot **init** wants to read it, a warning is printed and default settings are assumed.

If **0** through **6** is entered, boot **init** enters the corresponding run level. Any other input is rejected and a new prompt is issued. If this is the first time boot **init** has entered a run level other than single-user, boot **init** first scans **inittab** for special entries of the type **boot** and **bootwait**. These entries are performed — provided that the run level entered matches that of the entry — before any normal processing of **inittab** takes place. In this way, any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. The **inittab** file is scanned to find all entries that are to be processed for that run level.

Run levels in HP-UX are defined as follows:

0	Shut down HP-UX.
S|**s**	Use for system administration (also known as "single-user state"). When booting into run level **S** at powerup, the only access to the the system is through a shell spawned at the system console as the root user. The only processes running on the system will be kernel daemons started directly by the HP-UX kernel, daemon processes started from entries of type **sysinit** in **/etc/inittab**, the shell on the system console, and any processes started by the system administrator. Administration operations that require the system to be in a quiescent state (such as the *fsck*(1M) operation to repair a file system) should be run in this state. Transitioning into run level **S** from a higher run level does not terminate other system activity and does not result in a "single-user state"; this operation should not be done.
1	Start a subset of essential system processes. This state can also be used to perform system administration tasks.
2	Start most system daemons and login processes. This state is often called the "multi-user state". Login processes either at local terminals or over the network are possible.
3	Export filesystems and start other system processes. In this state NFS filesystems are often exported, as may be required for an NFS server.
4	Activate graphical presentation managers and start other system processes.
5–**6**	These states are available for user-defined operations.

The default run level is usually run level **3** or **4,** depending on the system configuration.

When **init** transitions into a new run level **0**–**6**, the master sequencer script **rc** is invoked. **rc** in turn invokes each of the start or kill scripts for each installed subsystem for each intervening run level. When transitioning to a higher run level start scripts are invoked, and when transitioning to a lower run level kill scripts are invoked. See *rc*(1M).

In a multiuser environment, the **inittab** file is usually set up so that boot **init** creates a process for each terminal on the system.

For terminal processes, ultimately the shell terminates because of an end-of-file either typed explicitly or generated as the result of hanging up. When boot **init** receives a child death signal telling it that a process it spawned has died, it records the fact and the reason it died in **/etc/utmp** and **/var/adm/wtmp**, if they exist (see *who*(1)). A history of the processes spawned is kept in **/var/adm/wtmp**, if it exists.

To spawn each process in the **inittab** file, boot **init** reads each entry and, for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by the **inittab** file, boot **init** waits for one of its descendant processes to die, a powerfail signal, or until it is signaled by a user **init** to change the system's run level. When one of the above three conditions occurs, boot **init** re-examines the **inittab** file. New entries can be added to the **inittab** file at any time. However, boot **init** still waits for one of the above three conditions to occur. For an instantaneous response, use the **init Q** (or **init q**) command to wake up boot **init** to re-examine the **inittab** file without changing the run level.

If boot **init** receives a powerfail signal (**SIGPWR**) and is not in single-user mode, it scans **inittab** for special **powerfail** entries. These entries are invoked (if the run levels permit) before any other processing takes place by boot **init**. In this way, boot **init** can perform various cleanup and recording functions whenever the operating system experiences a power failure. Note, however, that although boot **init** receives **SIGPWR** immediately after a power failure, boot **init** cannot handle the signal until it resumes execution. Since execution order is based on scheduling priority, any eligible process with a higher priority executes before boot **init** can scan **inittab** and perform the specified functions.

When boot **init** is requested to change run levels via a user **init**, it sends the warning signal **SIGTERM** to all processes that are undefined in the target run level. Boot **init** waits 20 seconds before forcibly terminating these processes with the kill signal **SIGKILL**. Note that boot **init** assumes that all these processes (and their descendants) remain in the same process group that boot **init** originally created for them. If any process changes its process group affiliation with either **setpgrp()** or **setpgrp2()** (see *setsid*(2) and *setpgid*(2)), it will not receive these signals. (Common examples of such processes are the shells **csh** and **ksh** (see *csh*(1) and *ksh*(1).) Such processes need to be terminated separately.

A user **init** can be invoked only by users with appropriate privileges.

DIAGNOSTICS

If boot **init** finds that it is continuously respawning an entry from **inittab** more than 10 times in 2 minutes, it will assume that there is an error in the command string, generate an error message on the system console, and refuse to respawn this entry until either 5 minutes have elapsed or it receives a signal from a user **init**. This prevents boot **init** from using up system resources if there is a typographical error in the **inittab** file or a program is removed that is referenced in **inittab**.

WARNINGS

Boot **init** assumes that processes and descendants of processes spawned by boot **init** remain in the same process group that boot **init** originally created for them. When changing init states, special care should be taken with processes that change their process group affiliation, such as **csh** and **ksh**.

One particular scenario that often causes confusing behavior can occur when a child **csh** or **ksh** is started by a login shell. When boot **init** is asked to change to a run level that would cause the original login shell to be killed, the shell's descendant **csh** or **ksh** process does not receive a hangup signal since it has changed its process group affiliation and is no longer affiliated with the process group of the original shell. Boot **init** cannot kill this **csh** or **ksh** process (or any of its children).

If a **getty** process is later started on the same tty as this previous shell, the result may be two processes (the **getty** and the job control shell) competing for input on the tty.

To avoid problems such as this, always be sure to manually kill any job control shells that should not be running after changing init states. Also, always be sure that user **init** is invoked from the lowest level (login) shell when changing to an init state that may cause your login shell to be killed.

FILES

/dev/syscon
/dev/systty
/etc/inittab
/etc/ioctl.syscon
/etc/utmp
/var/adm/wtmp

SEE ALSO

csh(1), ksh(1), login(1), sh(1), who(1), getty(1M), rc(1M), ioctl(2), kill(2), setpgid(2), setsid(2), inittab(4), utmp(4).

STANDARDS CONFORMANCE

init: SVID2, SVID3

NAME

insf - install special (device) files

SYNOPSIS

/sbin/insf

/sbin/insf [**-C** *class* | **-d** *driver*] [**-D** *directory*] [**-e**] [**-H** *hw-path*] [**-I** *instance*]
[**-n** *npty*] [**-q**|**-v**] [**-s** *nstrpty*] [**-p** *first-optical-disk***:***last-optical-disk*]

DESCRIPTION

The **insf** command installs special files in the devices directory, normally **/dev**. If required, **insf** creates any subdirectories that are defined for the resulting special file.

If no options are specified, special files are created for all new devices in the system. New devices are those devices for which no special files have been previously created. A subset of the new devices can be selected with the **-C**, **-d**, and **-H** options.

With the **-e** option, **insf** reinstalls the special files for pseudo-drivers and existing devices. This is useful for restoring special files when one or more have been removed.

Normally, **insf** displays a message as the special files are installed for each driver. The **-q** (quiet) option suppresses the installation message. The **-v** (verbose) option displays the installation message and the name of each special file as it is created.

Options

insf recognizes the following options.

-C *class* Match devices that belong to a given device class, *class*. Device classes can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in the files in the directory **/usr/conf/master.d**. The special class **pseudo** includes all pseudo-drivers. This option cannot be used with **-d**.

-d *driver* Match devices that are controlled by the specified device driver, *driver*. Device drivers can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in the files in the directory **/usr/conf/master.d**. This option cannot be used with **-C**.

-D *directory* Override the default device installation directory **/dev** and install the special files in *directory* instead. *directory* must exist; otherwise, **insf** displays an error message and exits. See WARNINGS.

-e Reinstall the special files for pseudo-drivers and existing devices. This is useful for restoring special files if one or more have been removed.

-H *hw-path* Match devices at a given hardware path, *hw-path*. Hardware paths can be listed with the **ioscan** command (see *ioscan*(1M)). A hardware path specifies the addresses of the hardware components leading to a device. It consists of a string of numbers separated by periods (**.**), such as **52** (a card), **52.3** (a target address), and **52.3.0** (a device). If a hardware component is a bus converter, the following period, if any, is replaced by a slash (**/**) as in **2**, **2/3**, and **2/3.0**.

If the specified path contains fewer numbers than are necessary to reach a device, special files are made for all devices at addresses that extend the given path. If the specified path is **56**, then special files are made for the devices at addresses **56.0**, **56.1**, **56.2**, etc.

-I *instance* Match a device with the specified *instance* number. Instances can be listed with the **-f** option of the **ioscan** command (see *ioscan*(1M)).

This option is effective only if the **-e** option is specified or if an appropriate device class or driver is specified with a **-C** or **-d** option.

-n *npty* Install *npty* special files for each specified **ptym** and **ptys** driver. The **pty** driver specifies both the **ptym** and **ptys** drivers. *npty* is a decimal number.

This option is effective only if the **-e** option is specified or if an appropriate device class or driver is specified with a **-C** or **-d** option.

If this option is omitted, *npty* defaults to 60 for the **ptym** and **ptys** drivers.

-p *first-optical-disk***:***last-optical-disk*
Install the special files for those optical disks located in slots in the range *first-optical-disk* to *last-optical-disk*. The two variables can have values from the set **1a**, **1b**, ..., **32a**, **32b**. This option only applies to the **autox0** and **schgr** drivers. If it is omitted, the 64 special files for both sides of 32 optical disks (**1a** through **32b**) will be installed.

-q Quiet option. Normally, **insf** displays a message as each driver is processed. This option suppresses the driver message, but not error messages. See the **-v** option.

-s *nstrpty* Install *nstrpty* slave-side stream special files for the **pts** driver. *nstrpty* is a decimal number. This option only applies to the **pts** special file installation.

This option is effective only if the **-e** option is specified or if an appropriate device class or driver is specified with a **-C** or **-d** option.

If this option is omitted, *nstrpty* defaults to 60.

-v Verbose option. In addition to the normal processing message, display the name of each special file as it is created. See the **-q** option.

Naming Conventions

Many special files are named using the **c***card***t***target***d***device* naming convention. These variables have the following meaning wherever they are used.

card	The unique interface card identification number from **ioscan** (see *ioscan*(1M)). It is represented as a decimal number with a typical range of 0 to 255.
target	The device target number, for example the address on a HP-FL or SCSI bus. It is represented as a decimal number with a typical range of 0 to 15.
device	A address unit within a device, for example, the unit in a HP-FL device or the LUN in a SCSI device. It is represented as a decimal number with a typical range of 0 to 15.

Special Files

This subsection shows which special files are created and the permissions for each device driver.

The special file names are relative to the installation directory, normally **/dev**. This directory may be overridden with the **-D** option.

insf sets the file permissions and the owner and group IDs. They are shown here in a format similar to that of the **ll** command:

special-file *permissions owner group*

For example:

```
tty           rw-rw-rw- bin bin
```

Device Driver Special Files and Description

arp The following special file is installed:

```
arp           rw-rw-rw- root sys
```

asio0

For each card instance, the following special files are installed:

tty*card***p0** **rw--w--w- bin bin**
Direct connect

asyncdsk

The following special file is installed:

```
asyncdsk      rw-rw-rw- bin bin
```

audio

The following special files are installed. Note the underscore (_) before *card* in each special file name.

For *card* 0, the device files are linked to files without the trailing **_0** in their names.

audio_*card* **rw-rw-rw- bin bin**

	Default audio device
audioCtl_*card*	**rw-rw-rw- bin bin** Audio control device
audioBA_*card*	**rw-rw-rw- bin bin** All outputs, A-law format
audioBL_*card*	**rw-rw-rw- bin bin** All outputs, 16-bit linear format
audioBU_*card*	**rw-rw-rw- bin bin** All outputs, Mu-law format
audioEA_*card*	**rw-rw-rw- bin bin** External output, A-law format
audioEL_*card*	**rw-rw-rw- bin bin** External output, 16-bit linear format
audioEU_*card*	**rw-rw-rw- bin bin** External output, Mu-law format
audioIA_*card*	**rw-rw-rw- bin bin** Internal speaker output, A-law format
audioIL_*card*	**rw-rw-rw- bin bin** Internal speaker output, 16-bit linear format
audioIU_*card*	**rw-rw-rw- bin bin** Internal speaker output, Mu-law format
audioLA_*card*	**rw-rw-rw- bin bin** Line output, A-law format
audioLL_*card*	**rw-rw-rw- bin bin** Line output, 16 bit linear format
audioLU_*card*	**rw-rw-rw- bin bin** Line output, Mu-law format
audioNA_*card*	**rw-rw-rw- bin bin** No output, A-law format
audioNL_*card*	**rw-rw-rw- bin bin** No output, 16 bit linear format
audioNU_*card*	**rw-rw-rw- bin bin** No output, Mu-law format

autox0 schgr

Special file names for **autox0** and **schgr** use the format:

c*card***t***target***d** *device_surface*

surface: **1a** through **32b**, unless modified by the **-p** option. Note the underscore (_) between *device* and *surface*.

For each autochanger device, the following special files are installed:

ac/c*card***t***target***d***device_surface*	**rw-r----- bin sys** Block entry
rac/c*card***t***target***d***device_surface*	**rw-r----- bin sys** Character entry
rac/c*card***t***target***d***device*	**rw------- bin sys** Character entry

beep

The following special file is installed:

beep	**rw-rw-rw- bin bin**

CentIf
For each card instance, the following special file is installed.

c*card***t***target***d***device***_lp**	**rw-rw-rw- lp bin** Handshake mode 2, character entry

consp1
For each card instance, the following special files are installed:

tty*card***p0**	**rw--w--w- bin bin** Direct connect

cn The following special files are installed:

syscon	**rw--w--w- bin bin**
systty	**rw--w--w- bin bin**
console	**rw--w--w- root sys**
ttyconf	**rw------- root sys**

cs80 disc1 disc2 disc3 disc4 sdisk
For each disk device, the following special files are installed:

dsk/c*card***t***target***d***device*	**rw-r----- bin sys** Block entry
rdsk/c*card***t***target***d***device*	**rw-r----- bin sys** Character entry
	For **disc1** and **disc2** instances, the following additional special file is installed:
diag/rdsk/c*card***t***target***d***device*	**rw------- bin bin** Character entry
	For **cs80** and **disc1** instances, the following additional special files are installed:
ct/c*card***t***target***d***device*	**rw-r----- bin sys** Block entry
rct/c*card***t***target***d***device*	**rw-r----- bin sys** Character entry
	For **disc1** instances, the following additional special file is installed:
diag/rct/c*card***t***target***d***device*	**rw------- bin bin** Character entry
	For **disc3** instances, the following additional special files are installed:
floppy/c*card***t***target***d***device*	**rw-r----- bin sys** Block entry
rfloppy/c*card***t***target***d***device*	**rw-r----- bin sys** Character entry

devconfig
The following special file is installed:

config	**rw-r----- root sys**

diag0
The following special file is installed:

diag/diag0	**rw------- bin bin**

diag1
The following special file is installed:

diag/diag1	**rw------- bin bin**

i

diag2
The following special files are installed:

```
diag2          rw------- bin bin
diag/diag2     rw------- bin bin
```

diaghpib1
For each device, the following special files are installed:

diag/hpib/hp28650A/*instance* **rw------- bin bin**

disc1 disc2 disc3 disc4
See **cs80**.

dlpi
The following special files are installed:

```
dlpi           rw-rw-rw- root sys
dlpi0          rw-rw-rw- root sys
dlpi1          rw-rw-rw- root sys
dlpi2          rw-rw-rw- root sys
dlpi3          rw-rw-rw- root sys
dlpi4          rw-rw-rw- root sys
```

dmem
The following special file is installed:

```
dmem           rw------- bin bin
```

echo
The following special file is installed:

```
echo           rw-rw-rw- root sys
```

eisa_mux0 pci_mux0
For each instance of an EISA mux or PCI mux card, the following "Direct Connect" special files are created. The term "card" below refers to the instance number of the mux card.

tty*card***port_module***port*
rw--w--w- bin bin
letter: **a** to **p**, port module name
port: **1** to **16**, direct connect

mux*card* **rw------- bin bin**

diag/mux*card* **rw------- bin bin**

diag/mux*card***_1** **rw------- bin bin**

diag/mux*card***_2** **rw------- bin bin**

fddi
The following special file is installed:

lan*card* **rw-rw-rw- bin bin**

framebuf
For each graphics device, the following special files are installed.

crt*device_number* **rw-rw-rw- bin bin**

ocrt*device_number*
rw-rw-rw- bin bin

device_number is 0 indexed and is assigned in the order in which the devices appear in *ioscan*(1M) output.

If the console device is a graphics device, the files **crt** and **ocrt** are created as the console device. If the console is not a graphics device, **crt** and **ocrt** are identical to **crt0** and **ocrt0**.

i

hil For each device, the following special files are installed. Note the underscore (_) before *card* in each special file name.

For *card* 0, the device files are linked to files named **hil***addr* for the link addresses 1 to 7; **hilkbd** for the cooked keyboard device; and **rhil** for the **hil** controller device.

hil_*card***.***addr*	**rw-rw-rw- bin bin** *addr*: link addresses **1** to **7**
hilkbd_*card*	**rw-rw-rw- bin bin**
rhil_*card*	**rw-rw-rw- bin bin**

inet_clts
The following special file is installed:

inet_clts	**rw-rw-rw- root sys**

inet_cots
The following special file is installed:

inet_cots	**rw-rw-rw- root sys**

instr0
For each card instance, the following special files are installed:

hpib/c*card*	**rw-rw-rw- bin bin**
hpib/c*card***t***addr***d0**	**rw-rw-rw- bin bin** *addr*: **0** to **30**
diag/hpib/c*card*	**rw------- bin bin**

ip The following special file is installed:

ip	**rw-rw-rw- root sys**

kepd
The following special file is installed:

kepd	**rw-r--r-- root other**

klog
The following special file is installed:

klog	**rw------- bin bin**

lan0 lan1 lan2 lan3
For each card instance, the following special files are installed:

lan*card*	**rw-rw-rw- bin bin**
ether*card*	**rw-rw-rw- bin bin**
diag/lan*card*	**rw------- bin bin**

lantty0
For each card instance, the following special files are installed:

lantty*card*	**rw-rw-rw- bin bin** Normal access
diag/lantty*card*	**rw-rw-rw- bin bin** Exclusive access

lpr0 lpr1 lpr2 lpr3
For each card instance, the following special files are installed:

c*card***t***target***d***device***_lp**	**rw------- lp bin**
diag/c*card***t***target***d***device***_lp**	**rw------- bin bin**

mm The following special files are installed:

mem	**rw-r----- bin sys** Minor **0**

i

kmem	**rw-r----- bin sys** Minor **1**
null	**rw-rw-rw- bin bin** Minor **2**

mux0

For each instance of a 6-channel card, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0** to **5**, direct connect
mux*card*	**rw------- bin bin**
diag/mux*card*	**rw------- bin bin**

For each instance of a 16-channel card, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0** to **15**, direct connect
mux*card*	**rw------- bin bin**
diag/mux*card*	**rw------- bin bin**

mux2

For each instance of an 16-channel card, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0** to **15**, direct connect
mux*card*	**rw------- bin bin**
diag/mux*card*	**rw------- bin bin**

For each card instance of an 8-channel card, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0** to **7**, direct connect
mux*card*	**rw------- bin bin**
diag/mux*card*	**rw------- bin bin**

For each card instance of an 3-channel card, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0**, **1**, and **7**, direct connect
mux*card*	**rw------- bin bin**
diag/mux*card*	**rw------- bin bin**

mux4

For each card instance, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0** and **1**, direct connect

netqa

The following special file is installed:

netqa	**rw-rw-rw- root sys**

nuls

The following special file is installed:

nuls	**rw-rw-rw- root sys**

pci_mux0

The following "Direct Connect" special files are created. The term "card" below refers to the instance number of the mux card.

tty*card***port_module***port*
rw--w--w- bin bin

i

port_module: **a** to **p**, port module name
port: **1** to **16**, port number

mux*card*	**rw------- bin bin**
diag/mux*card*	**rw------- bin bin**
diag/mux*card***_1**	**rw------- bin bin**
diag/mux*card***_2**	**rw------- bin bin**

pflop sflop
For each card instance, the following special files are installed:

floppy/c*card***t***target***d***device*	**rw-r----- bin sys** Block entry
rfloppy/c*card***t***target***d***device*	**rw-r----- bin sys** Character entry

ps2 The following special files are installed:

ps2kbd	**rw-rw-rw- bin bin** Autosearch for first ps2 keyboard
ps2mouse	**rw-rw-rw- bin bin** Autosearch for first ps2 mouse
ps2_0	**rw-rw-rw- bin bin** ps2 port 0
ps2_1	**rw-rw-rw- bin bin** ps2 port 1

ptm The following special file is installed:

ptmx	**rw-rw-rw- root sys**

pts The following special files are installed:

pts/*number*	**rw-rw-rw- root sys** *number*: **0** to **59**

pty Specifying this driver tells **insf** to install the special files for both the master and slave pty drivers, **ptym** and **ptys**. The command **insf -d pty** is equivalent to the two commands **insf -d ptym** and **insf -d ptys**.

ptym
The following special files are installed:

ptym/clone	**rw-r--r-- root other**
ptym/pty*index number*	**rw-rw-rw- bin bin** *index*: **p** to **z**, **a** to **c**, **e** to **o**; *number*: **0** to **f** (hexadecimal) The first 48 special files **ptym/pty*** are linked to **pty***.
ptym/pty*index number*	**rw-rw-rw- bin bin** *index*: **p** to **z**, **a** to **c**, **e** to **o**; *number*: **00** to **99**
ptym/pty*index number*	**rw-rw-rw- bin bin** *index*: **p** to **z**, **a** to **c**, **e** to **o**; *number*: **000** to **999**

ptys
The following special files are installed:

pty/tty*index number*	**rw-rw-rw- bin bin** *index*: **p** to **z**, **a** to **c**, **e** to **o**; *number*: **0** to **f** (hexadecimal) The first 48 special files **pty/tty*** are linked to **tty***.
pty/tty*index number*	**rw-rw-rw- bin bin** *index*: **p** to **z**, **a** to **c**, **e** to **o**; *number*: **00** to **99**
pty/tty*index number*	**rw-rw-rw- bin bin**

index: **p** to **z**, **a** to **c**, **e** to **o**; *number*: **000** to **999**

rawip
The following special file is installed:

rawip	**rw-rw-rw- root sys**

root
The following special files are installed:

root	**rw-r----- bin sys**
rroot	**rw-r----- bin sys**

sad The following special file is installed:

sad	**rw-rw-rw- root sys**

sastty
For each card instance, the following special files are installed:

tty*card***p***port*	**rw--w--w- bin bin** *port*: **0** to **1**, direct connect

schgr
See **autox0**.

sdisk
See **cs80**.

sflop
See **pflop**.

stape tape1 tape2
For each driver instance, different special files are installed depending on the number of characters allowed in the target directory. There are two lists below, one for long file name directories and one for short file name directories (14 characters maximum). Short file names are used for files installed on an NFS file system.

Note that the first four special files in each list for tape driver instances 0-9 are also linked to **rmt/***instance***m**, **rmt/***instance***mb**, **rmt/***instance***mn**, and **rmt/***instance***mnb**, respectively.

For installation in a long file name directory:

rmt/c*card***t***target***d***device***BEST**	**rw-rw-rw- bin bin** AT&T-style, best available density, character entry
rmt/c*card***t***target***d***device***BESTb**	**rw-rw-rw- bin bin** Berkeley-style, best available density, character entry
rmt/c*card***t***target***d***device***BESTn**	**rw-rw-rw- bin bin** AT&T-style, no rewind, best available density, character entry
rmt/c*card***t***target***d***device***BESTnb**	**rw-rw-rw- bin bin** Berkeley-style, no rewind, best available density, character entry

For installation in a short file name directory:

rmt/c*card***t***target***d***device***f0**	**rw-rw-rw- bin bin** AT&T-style, best available density, character entry
rmt/c*card***t***target***d***device***f0b**	**rw-rw-rw- bin bin** Berkeley-style, best available density, character entry
rmt/c*card***t***target***d***device***f0n**	**rw-rw-rw- bin bin** AT&T-style, no rewind, best available density, character entry
rmt/c*card***t***target***d***device***f0nb**	**rw-rw-rw- bin bin** Berkeley-style, no rewind, best available density, character entry

For both long and short file name directories, the following additional files are created.

rmt/*driver_name***_config**	**rw-r--r-- bin bin** Tape configuration, character entry

diag/rmt/c*card***t***target***d***device* **rw------- bin bin**
For **tape1** and **tape2** only, diagnostic access, character entry

stcpmap
The following special file is installed:

stcpmap **rw-rw-rw- root sys**

strlog
The following special file is installed:

strlog **rw-rw-rw- root sys**

sy The following special file is installed:

tty **rw-rw-rw- bin bin**

tape1 tape2
See **stape**.

tcp The following special file is installed:

tcp **rw-rw-rw- root sys**

telm
The following special file is installed:

telnetm **rw-rw-rw- root sys**

tels
The following special files are installed:

pts/t*number* **rw-rw-rw- root sys**
number: **0** to **59**

tlclts
The following special file is installed:

tlclts **rw-rw-rw- root sys**

tlcots
The following special file is installed:

tlcots **rw-rw-rw- root sys**

tlcotsod
The following special file is installed:

tlcotsod **rw-rw-rw- root sys**

token2
The following special file is installed:

lan*card* **rw-rw-rw- bin bin**

udp The following special file is installed:

udp **rw-rw-rw- root sys**

unix_clts
The following special file is installed:

unix_clts **rw-rw-rw- root sys**

unix_cots
The following special file is installed:

unix_cots **rw-rw-rw- root sys**

RETURN VALUE

insf exits with one of the following values:

0 Successful completion, including warning diagnostics.
1 Failure.

DIAGNOSTICS

Most diagnostic messages from **insf** are self-explanatory. Listed below are some messages deserving further clarification.

Warnings

Device driver *name* **is not in the kernel**
Device class *name* **is not in the kernel**

The indicated device driver or device class is not present in the kernel. A device driver and/or device class can be added to the kernel using *config*(1M).

No instance number available for device class *name*

All of the instance numbers available for the device class are already assigned. Use the **rmsf** command to remove any unneeded devices from the system (see *rmsf*(1M)).

Don't know how to handle driver *name* **- no special files created for** *path*

insf does not know how to create special files for the specified device driver. Use **mknod** to create special files for the device (see *mknod*(1M)).

EXAMPLES

Install special files for all new devices belonging to the **tty** device class:

```
insf -C tty
```

Install special files to the new device added at hardware path **2/4.0.0**:

```
insf -H 2/4.0.0
```

WARNINGS

insf should only be run in single-user mode. It can change the mode, owner, or group of an existing special file, or unlink and recreate one; special files that are currently open may be left in an indeterminate state.

Many commands and subsystems assume their device files are in **/dev**, therefore the use of the **-D** option is discouraged.

AUTHOR

insf was developed by HP.

FILES

/dev/config	I/O system special file
/etc/ioconfig	I/O system configuration database

SEE ALSO

config(1M), ioscan(1M), lsdev(1M), lssf(1M), mknod(1M), mksf(1M), rmsf(1M).

NAME

install - install commands

SYNOPSIS

/usr/sbin/install [**-c** *dira*] [**-f** *dirb*] [**-i**] [**-n** *dirc*] [**-o**] [**-g** *group*] [**-s**] [**-u** *user*] *file* [*dirx* ...]

DESCRIPTION

install is a command most commonly used in "makefiles" (see *make*(1)) to install a *file* (updated target file) in a specific place within a file system. Each *file* is installed by copying it into the appropriate directory, thereby retaining the mode and owner of the original command. The program prints messages telling the user exactly what files it is replacing or creating and where they are going.

install is useful for installing new commands, or new versions of existing commands, in the standard directories (i.e. **/usr/bin**, **/usr/sbin**, etc.).

If no options or directories (*dirx*...) are given, **install** searches a set of default directories (**/usr/bin**, **/usr/sbin**, **/sbin**, and **/usr/lbin**, in that order) for a file with the same name as *file*. When the first occurrence is found, **install** issues a message saying that it is overwriting that file with *file* (the new version), and proceeds to do so. If the file is not found, the program states this and exits without further action.

If one or more directories (*dirx* ...) are specified after *file*, those directories are searched before the directories specified in the default list.

Options

Options are interpreted as follows:

-c *dira* Installs a new command (*file*) in the directory specified by *dira*, only if it is not found. If it is found, **install** issues a message saying that the file already exists, and exits without overwriting it. Can be used alone or with the **-s** option.

-f *dirb* Forces *file* to be installed in given directory, whether or not one already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to **755** and **bin**, respectively. If the file already exists, the mode and owner will be that of the already existing file. Can be used alone or with the **-o** or **-s** options.

-i Ignores default directory list, searching only through the given directories (*dirx* ...). Can be used alone or with any other options other than **-c** and **-f**.

-n *dirc* If *file* is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file will be set to **755** and **bin**, respectively. Can be used alone or with any other options other than **-c** and **-f**.

-o If *file* is found, this option saves the "found" file by copying it to **OLD***file* in the directory in which it was found. This option is useful when installing a normally busy text file such as **/usr/bin/sh** or **/usr/sbin/getty**, where the existing file cannot be removed. Can be used alone or with any other options other than **-c**.

-g *group* Causes *file* to be owned by group *group*. This option is available only to users who have appropriate privileges. Can be used alone or with any other option.

-u *user* Causes *file* to be owned by user *user*. This option is available only to users who have appropriate privileges. Can be used alone or with any other option.

-s Suppresses printing of messages other than error messages. Can be used alone or with any other options.

When no directories are specified (*dirx* ...), or when *file* cannot be placed in one of the directories specified, **install** checks for the existence of the file **/etc/syslist**. If **/etc/syslist** exists, it is used to determine the final destination of *file*. If **/etc/syslist** does not exist, the default directory list is further scanned to determine where *file* is to be located.

The file **/etc/syslist** contains a list of absolute pathnames, one per line. The pathname is the "official" destination (for example **/usr/bin/echo**) of the file as it appears on a file system. The file **/etc/syslist** serves as a master list for system command destinations. If there is no entry for *file* in the file **/etc/syslist** the default directory list is further scanned to determine where *file* is to be located.

Cross Generation

The environment variable **ROOT** is used to locate the locations file (in the form **$ROOT/etc/syslist**). This is necessary in cases where cross generation is being done on a production system. Furthermore, each pathname in **$ROOT/etc/syslist** is appended to **$ROOT** (for example, **$ROOT/usr/bin/echo**), and used as the destination for *file*. Also, the default directories are also appended to **$ROOT** so that the default directories are actually **$ROOT/usr/bin**, **$ROOT/usr/sbin**, **$ROOT/sbin**, and **$ROOT/usr/lbin**.

The file **/etc/syslist** (**$ROOT/etc/syslist**) does not exist on a distribution tape; it is created and used by local sites.

WARNINGS

install cannot create alias links for a command (for example, *vi*(1) is an alias link for *ex*(1)).

SEE ALSO

make(1), cpset(1M).

NAME

ioinit - test and maintain consistency between the kernel I/O data structures and /etc/ioconfig

SYNOPSIS

/sbin/ioinit -i [**-r**]

/sbin/ioinit -c

/sbin/ioinit -f *infile* [**-r**]

DESCRIPTION

The **ioinit** command is invoked by the **init** process when the system is booted, based on the **ioin** entry in **/etc/inittab**:

```
ioin::sysinit:/sbin/ioinitrc > /dev/console 2>&1
```

where **ioinitrc** is a script to invoke **ioinit** with the **-i** and **-r** options. Given the **-i** option, **ioinit** checks consistency between the kernel I/O data structures (initialized with **/stand/ioconfig**, which is accessible for NFS-diskless support when the system boots up) and information read from **/etc/ioconfig**. If these are consistent, **ioinit** invokes **insf** to install special files for all new devices. If the kernel is inconsistent with **/etc/ioconfig**, **ioinit** updates **/stand/ioconfig** from **/etc/ioconfig**, and, if the **-r** option is given, reboots the system.

If **/etc/ioconfig** is corrupted or missing when the system reboots, **ioinitrc** brings the system up in single-user mode. The user should then restore **/etc/ioconfig** from backup or invoke the **ioinit** with the **-c** option to recreate **/etc/ioconfig** from the kernel.

If the **-f** option is given, **ioinit** reassigns instance numbers to existing devices within a given class based on *infile*. Reassignment takes effect when the system reboots. If **ioinit** finds no errors associated with the reassignment, and the **-r** option is given, the system is rebooted. (See the WARNINGS section.)

If the **-c** option is given, **ioinit** recreates **/etc/ioconfig** from the existing kernel I/O data structures.

Options

ioinit recognizes the following options:

-i Invoke **insf** to install special files for new devices after checking consistency between the kernel and **/etc/ioconfig**.

-f *infile* Use the file *infile* to reassign instance numbers to devices within a specified class. *infile* may have multiple entries, each to appear on a separate line, each field in the entry separated by 1 or more blanks. Entries should conform to the following format:

```
h/w_path          class_name          instance_#
```

ioinit preprocesses the contents of *infile*, looking for invalid entries, and prints out explanatory messages. An entry is considered to be invalid if the specified hardware path or class name does not already exist in the system, or if the specified instance number already exists for the given class.

-r Reboot the system when it is required to correct the inconsistent state between the kernel and **/etc/ioconfig**, as used with the **-i** option. When used with the **-f** option, if there are no errors associated with the instance reassignment, **-r** reboots the system.

-c Recreate **/etc/ioconfig**, if the file is corrupted or missing and cannot be restored from backup. If **-c** is invoked, any previous binding of hardware path to device class and instance number is lost.

RETURN VALUE

0 No errors occurred, although warnings might be issued.

1 **ioinit** encountered an error.

DIAGNOSTICS

Most of the diagnostic messages from **ioinit** are self-explanatory. Listed below are some messages deserving further clarification. Errors cause **ioinit** to halt immediately.

Errors

`/etc/ioconfig is missing.`
`/etc/ioconfig is corrupted.`

Either restore **`/etc/ioconfig`** from backup and then reboot, or recreate **`/etc/ioconfig`** using **`ioinit -c`**.

`Permission to access /etc/ioconfig is denied.`

Change permissions to **`/etc/ioconfig`** to allow access by **`ioinit`**.

`exec of insf failed.`

`ioinit` completed successfully, but **`insf`** failed.

`Instance number is already in kernel.`

Instance number already exists for a given class. Use **`rmsf`** to remove the existing instance number, then retry.

`Hardware path is not in the kernel.`

The given hardware path is not in the kernel. Use **`ioscan -k`** to get the correct hardware path, then retry.

`Device class name is not in the kernel.`

The given class name is not in the kernel. Use **`ioscan -k`** to get the correct class name, then retry.

EXAMPLES

To reassign an instance number to a device and class (specified in *infile*) and reboot the system:

```
/sbin/ioinit -f infile -r
```

where **`infile`** contains the following:

```
56.52          scsi          2
```

`56.52` is the *h/w_path*, **`scsi`** is the *class_name*, and **`2`** is the *instance_#*.

WARNINGS

Running **`rmsf`** or **`insf`** overwrites the effect of reassignment by **`ioinit`** before the system is rebooted.

AUTHOR

`ioinit` was developed by HP.

FILES

`/stand/ioconfig`

`/etc/ioconfig`

SEE ALSO

init(1M), insf(1M), ioscan(1M), rmsf(1M), inittab(4), ioconfig(4).

NAME

ioscan - scan I/O system

SYNOPSIS

/usr/sbin/ioscan [**-k**|**-u**] [**-d** *driver*|**-C** *class*] [**-I** *instance*] [**-H** *hw_path*] [**-f**[**-n**]|**-F**[**-n**]] [*devfile*]

/usr/sbin/ioscan -M *driver* **-H** *hw_path* [**-I** *instance*]

DESCRIPTION

ioscan scans system hardware, usable I/O system devices, or kernel I/O system data structures as appropriate, and lists the results. For each hardware module on the system, **ioscan** displays by default the hardware path to the hardware module, the class of the hardware module, and a brief description.

By default, **ioscan** scans the system and lists all reportable hardware found. The types of hardware reported include processors, memory, interface cards and I/O devices. Scanning the hardware may cause drivers to be unbound and others bound in their place in order to match actual system hardware. Entities that cannot be scanned are not listed.

In the second form shown, **ioscan** forces the specified software driver into the kernel I/O system at the given hardware path and forces software driver to be bound. This can be used to make the system recognize a device that cannot be recognized automatically; for example, because it has not yet been connected to the system, does not support autoconfiguration, or because diagnostics need to be run on a faulty device.

i

Options

ioscan recognizes the following options:

-C *class*	Restrict the output listing to those devices belonging to the specified *class*. Cannot be used with **-d**.
-d *driver*	Restrict the output listing to those devices controlled by the specified *driver*. Cannot be used with **-C**.
-f	Generate a full listing, displaying the module's class, instance number, hardware path, driver, software state, hardware type, and a brief description.
-F	Produce a compact listing of fields (described below), separated by colons. This option overrides the **-f** option.
-H *hw_path*	Restrict the scan and output listing to those devices connected at the specified hardware path. The hardware path must be a bus path. Scanning below the bus level will not probe the hardware and may produce incorrect results. For example, specifying the path at the target level will always change the state of the device attached to it as NO_HW. When used with **-M**, this option specifies the full hardware path at which to bind the software modules.
-I *instance*	Restrict the scan and output listing to the specified instance, when used with either **-d** or **-C**. When used with **-M**, specifies the desired instance number for binding.
-k	Scan kernel I/O system data structures instead of the actual hardware and list the results. No binding or unbinding of drivers is performed. The **-d**, **-C**, **-I**, and **-H** options can be used to restrict listings. Cannot be used with **-u**. This option does not require superuser privileges.
-M *driver*	Specifies the software driver to bind at the hardware path given by the **-H** option. Must be used with the **-H** option.
-n	List device file names in the output. Only special files in the **/dev** directory and its subdirectories are listed.
-u	Scan and list usable I/O system devices instead of the actual hardware. Usable I/O devices are those having a driver in the kernel and an assigned instance number. The **-d**, **-C**, **-I**, and **-H** options can be used to restrict listings. The **-u** option cannot be used with **-k**.

The **-d** and **-C** options can be used to obtain listings of subsets of the I/O system, although the entire system is still scanned. Specifying **-d** or **-C** along with **-I**, or specifying **-H** or a *devfile* causes **ioscan** to restrict both the scan and the listing to the hardware subset indicated.

Fields

The **-F** option can be used to generate a compact listing of fields separated by colons (:), useful for producing custom listings with **awk**. Fields include the module's bus type, cdio, is_block, is_char, is_pseudo, block major number, character major number, minor number, class, driver, hardware path, identify bytes, instance number, module path, module name, software state, hardware type, a brief description, and card instance. If a field does not exist, consecutive colons hold the field's position. Fields are defined as follows:

class	A device category, defined in the files located in the directory **/usr/conf/master.d** and consistent with the listings output by **lsdev** (see *lsdev*(1M)). Examples are **disk**, **printer**, and **tape**.
instance	The instance number associated with the device or card. It is a unique number assigned to a card or device within a class. If no driver is available for the hardware component or an error occurs binding the driver, the kernel will not assign an instance number and a (**-1**), is listed.
hw path	A numerical string of hardware components, notated sequentially from the bus address to the device address. Typically, the initial number is appended by slash (**/**), to represent a bus converter (if required by your machine), and subsequent numbers are separated by periods (**.**). Each number represents the location of a hardware component on the path to the device.
driver	The name of the driver that controls the hardware component. If no driver is available to control the hardware component, a question mark (**?**) is displayed in the output.
software state	The result of software binding.

CLAIMED	software bound successfully
UNCLAIMED	no associated software found
DIFF_HW	software found does not match the associated software
NO_HW	the hardware at this address is no longer responding
ERROR	the hardware at this address is responding but is in an error state
SCAN	node locked, try again later

hardware type Entity identifier for the hardware component. It is one of the following strings:

UNKNOWN	There is no hardware associated or the type of hardware is unknown
PROCESSOR	Hardware component is a processor
MEMORY	Hardware component is memory
BUS_NEXUS	Hardware component is bus converter or bus adapter
INTERFACE	Hardware component is an interface card
DEVICE	Hardware component is a device

bus type	Bus type associated with the node.
cdio	The name associated with the Context-Dependent I/O module.
is_block	A boolean value indicating whether a device block major number exists. A **T** or **F** is generated in this field.
is_char	A boolean value indicating whether a device character major number exists. A **T** or **F** is generated in this field.
is_pseudo	A boolean value indicating a pseudo driver. A **T** or **F** is generated in this field.
block major	The device block major number. A **-1** indicates that a device block major number does not exist.
character major	The device character major number. A **-1** indicates that a device character major number does not exist.
minor	The device minor number.

identify bytes	The identify bytes returned from a module or device.
module path	The software components separated by periods (.).
module name	The module name of the software component controlling the node.
description	A description of the device.
card instance	The instance number of the hardware interface card.

RETURN VALUE

ioscan returns **0** upon normal completion and **1** if an error occurred.

EXAMPLES

Scan the system hardware and list all the devices belonging to the disk device class.

```
ioscan -C disk
```

Forcibly bind driver **tape1** at the hardware path **8.4.1**.

```
ioscan -M tape1 -H 8.4.1
```

AUTHOR

ioscan was developed by HP.

FILES

/dev/config
/dev/*

SEE ALSO

config(1M), lsdev(1M), ioconfig(4).

NAME

isl - initial system loader

DESCRIPTION

isl implements the operating system independent portion of the bootstrap process. It is loaded and executed after self-test and initialization have completed successfully.

The processor contains special purpose memory for maintaining critical configuration related parameters (e.g. Primary Boot, Alternate Boot, and Console Paths). Two forms of memory are supported: Stable Storage and Non-Volatile Memory (NVM).

Typically, when control is transferred to *isl*, an *autoboot* sequence takes place. An *autoboot* sequence allows a complete bootstrap operation to occur with no intervention from an operator. *isl* executes commands from the *autoexecute* file in a script-like fashion. *autoboot* is enabled by a flag in Stable Storage.

autosearch is a mechanism that automatically locates the boot and console devices. For further information, see *pdc*(1M).

During an *autoboot* sequence, *isl* displays its revision and the name of any utility it executes. However, if *autoboot* is disabled, after *isl* displays its revision, it then prompts for input from the console device. Acceptable input is any *isl* command name or the name of any utility available on the system. If a non-fatal error occurs or the executed utility returns, *isl* again prompts for input.

Commands

There are several commands available in *isl*. The following is a list with a short description. Parameters may be entered on the command line following the command name. They must be separated by spaces. *isl* prompts for any necessary parameters that are not entered on the command line.

? **help**	Help - List commands and available utilities
listf **ls**	List available utilities
autoboot	Enable or disable the *autoboot* sequence Parameter - on or off
autosearch	Enable or disable the *autosearch* sequence Parameter - on or off
primpath	Modify the Primary Boot Path Parameter - Primary Boot Path in decimal
altpath	Modify the Alternate Boot Path Parameter - Alternate Boot Path in decimal
conspath	Modify the Console Path Parameter - Console Path in decimal
lsautofl **listautofl**	List contents of the *autoexecute* file
display	Display the Primary Boot, Alternate Boot, and Console Paths
readnvm	Display the contents of one word of NVM in hexadecimal Parameter - NVM address in decimal or standard hexadecimal notation
readss	Display the contents of one word of Stable Storage in hexadecimal Parameter - Stable Storage address in decimal or standard hexadecimal notation

DIAGNOSTICS

isl displays diagnostic information through error messages written on the console and display codes on the LED display.

For the display codes, **CE0***x* are informative only. **CE1***x* and **CE2***x* indicate errors, some of which are fatal and cause the system to halt. Other errors merely cause *isl* to display a message.

Non-fatal errors during an *autoboot* sequence cause the *autoboot* sequence to be aborted and *isl* to prompt for input. After non-fatal errors during an interactive *isl* session, *isl* merely prompts for input.

Fatal errors cause the system to halt. The problem must be corrected and the system **RESET** to recover.

CE00	*isl* is executing.
CE01	*isl* is *autoboot*ing from the *autoexecute* file.
CE02	Cannot find an *autoexecute* file. *autoboot* aborted.
CE03	No console found, *isl* can only *autoboot*.
CE05	Directory of utilities is too big, *isl* reads only 2K bytes.
CE06	*autoexecute* file is inconsistent. *autoboot* aborted.
CE07	Utility file header inconsistent: SOM values invalid.
CE08	*autoexecute* file input string exceeds 2048 characters. *autoboot* aborted.
CE09	*isl* command or utility name exceeds 10 characters.
CE0F	*isl* has transferred control to the utility.
CE10	Internal inconsistency: Volume label - **FATAL**.
CE11	Internal inconsistency: Directory - **FATAL**.
CE12	Error reading *autoexecute* file.
CE13	Error reading from console - **FATAL**.
CE14	Error writing to console - **FATAL**.
CE15	Not an *isl* command or utility.
CE16	Utility file header inconsistent: Invalid System ID.
CE17	Error reading utility file header.
CE18	Utility file header inconsistent: Bad magic number.
CE19	Utility would overlay *isl* in memory.
CE1A	Utility requires more memory than is configured.
CE1B	Error reading utility into memory.
CE1C	Incorrect checksum: Reading utility into memory.
CE1D	Console needed - **FATAL**.
CE1E	Internal inconsistency: Boot device class - **FATAL**.
CE21	Destination memory address of utility is invalid.
CE22	Utility file header inconsistent: *pdc_cache* entry.
CE23	Internal inconsistency: *iodc_entry_init* - **FATAL**.
CE24	Internal inconsistency: *iodc_entry_init* - console - **FATAL**.
CE25	Internal inconsistency: *iodc_entry_init* - boot device - **FATAL**.
CE26	Utility file header inconsistent: Bad aux_id.
CE27	Bad utility file type.

SEE ALSO

boot(1M), pdc(1M).

NAME

itemap - load an ITE (Internal Terminal Emulator) keyboard mapping.

SYNOPSIS

itemap [*options*]

DESCRIPTION

The **itemap** command loads a keyboard mapping into the ITE (the graphics console driver), or displays ITE keyboard mappings. **itemap** is run by **/etc/bcheckrc** automatically. It is not usually explicitly invoked by the user.

Options

-d *name*
-d *keyboard_ID*
: Dump a keymap to standard output in hexadecimal notation.

-h Load the specified keymap into the kernel mapping table used for **HP-HIL** keyboards.

-i Interactively prompt for a **PS2 DIN** keyboard mapping. **itemap** scans the keymap database file for all mapping names beginning with a **PS2_DIN** prefix. Each of these names is displayed, and one must be selected.

-k *database_file_name*
: The name of the keymap database file to be used for input. The default is **/etc/X11/XHPKeymaps**.

-L Load the appropriate keymap. **itemap** scans the hardware for a keyboard, determines the language of that keyboard, and loads the keymap corresponding to that keyboard.

Because **itemap** cannot determine the language of **PS2 DIN** keyboards, use the **-i** option when using **-L** with **PS2 DIN** keyboards.

-l *name*
-l *keyboard_ID*
: Load a specified keyboard map. Once loaded, ITE uses the specified mapping.

When loading a keyboard mapping with the **-l** option, **itemap** matches the suffix of the name of the specified keyboard mapping with those found in **/etc/X11/XHPKeymaps** to determine the keyboard language. This information is used by the ITE to perform ISO 7-to-8 bit conversion. Keymap names added by users, via

/usr/contrib/bin/X11/keymap_ed

should use the same suffixes as those already used in **/etc/X11/XHPKeymaps.** For example, a French keyboard mapping can be named **New_French**, for consistency with existing **ITF_French** and **PS2_French** mappings. A mapping called **New_Stuff** would not match any suffix patterns found by itemap, and would result in incorrect ISO 7-to-8 bit conversion.

-p Load the specified keymap into the kernel mapping table used for **PS2 DIN** keyboards.

-v Perform actions verbosely.

-w *file_name* If a keymap for a **PS2 DIN** keyboard is loaded, write its name to *file_name*.

EXAMPLES

To automatically install the correct mapping for an **HP-HIL** keyboard:

```
itemap -L
```

To explicitly load the **ITF_French** mapping for an **HP-HIL** keyboard:

```
itemap -h -l ITF_French
```

To explicitly load the **PS2_DIN_French** mapping for a **PS2 DIN** keyboard:

```
itemap -p -l PS2_DIN_French
```

To interactively choose a **PS2 DIN** keyboard mapping:

```
itemap -Li
```

To generate a list of the available keyboard mappings:

```
/usr/contrib/bin/X11/keymap_ed -l
```

FILES

/usr/contrib/bin/X11/keymap_ed	Keymap database editor
/etc/X11/XHPKeymaps	System keymap database
/etc/kbdlang	Contains mapping name configured for **PS2 DIN** keyboards

SEE ALSO

ps2(7), termio(7), keymap_ed(1X111).

NAME

keyenvoy - talk to keyserver

SYNOPSIS

keyenvoy

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

keyenvoy is a setuid root process that is used by some RPC programs to intermediate between a user process and the keyserv process, *keyserv*(1M), which will not talk to anything but a root process.

This program cannot be run interactively.

AUTHOR

keyenvoy was developed by Sun Microsystems, Inc.

SEE ALSO

keyserv(1M).

NAME

keyserv - server for storing private encryption keys

SYNOPSIS

keyserv [**-d**] [**-D**] [**-n**]

DESCRIPTION

keyserv is a daemon that is used for storing the private encryption keys of each user logged into the system. These encryption keys are used for accessing secure network services such as NIS+.

Normally, root's key is read from the file **/etc/.rootkey** when the daemon is started. This is useful during power-fail reboots when no one is around to type a password.

Options

-d Disable the use of default keys for **nobody**.

-D Run in debugging mode and log all requests to **keyserv**.

-n Root's secret key is not read from **/etc/.rootkey**. Instead, **keyserv** prompts the user for the password to decrypt root's key stored in the **publickey** database and then stores the decrypted key in **/etc/.rootkey** for future use. This option is useful if the **/etc/.rootkey** file ever gets out of date or corrupted.

FILES

/etc/.rootkey

AUTHOR

keyserv was developed by Sun Microsystems, Inc.

SEE ALSO

keylogin(1), keylogout(1), publickey(4).

NAME

killall - kill all active processes

SYNOPSIS

/usr/sbin/killall [*signal*]

DESCRIPTION

killall is a procedure used by **/usr/sbin/shutdown** to kill all active processes not directly related to the shutdown procedure.

killall is chiefly used to terminate all processes with open files so that the mounted file systems are no longer busy and can be unmounted. **killall** sends the specified *signal* to all user processes in the system, with the following exceptions:

the **init** process;

all processes (including background processes) associated with the terminal from which **killall** was invoked;

any **ps -ef** process, if owned by **root**;

any **sed -e** process, if owned by **root**;

any **shutdown** process;

any **killall** process;

any **/sbin/rc** process.

killall obtains its process information from **ps**, and therefore may not be able to perfectly identify which processes to signal (see *ps*(1)).

If no *signal* is specified, a default of **9** (kill) is used.

killall is invoked automatically by **shutdown** The use of **shutdown** is recommended over using **killall** by itself (see *shutdown*(1M)).

FILES

/usr/sbin/shutdown

SEE ALSO

fuser(1M), kill(1), ps(1), shutdown(1M), signal(5).

STANDARDS CONFORMANCE

killall: SVID2, SVID3

NAME

/usr/sbin/killsm - kill the sendmail daemon

SYNOPSIS

killsm

DESCRIPTION

killsm reads the **/etc/mail/sendmail.pid** file to find the pid number of the currently running sendmail daemon, and then kills that daemon. The "**/sbin/init.d/sendmail stop**" command does the same thing.

HP recommends that system administrators use "**/sbin/init.d/sendmail start**" and "**/sbin/init.d/sendmail stop**" to start and stop sendmail; these startup scripts are used when the system is booting to start sendmail. Advanced system administrators can put **/usr/sbin** into their search path and just reference "**sendmail -bd -q30m**" to start sendmail, and **killsm** to stop it.

The previous **sendmail -bk** option of former releases is no longer supported.

SEE ALSO

sendmail(1M).

NAME

kmadmin - kernel module administration

SYNOPSIS

/usr/sbin/kmadmin -d *directory_name* | **-D**
/usr/sbin/kmadmin -k
/usr/sbin/kmadmin -L *module_name* ... | *pathname* ...
/usr/sbin/kmadmin -q *module_id* ...
/usr/sbin/kmadmin -Q *module_name* ...
/usr/sbin/kmadmin -s | **-S**
/usr/sbin/kmadmin -u *module_id* ...
/usr/sbin/kmadmin -U *module_name* ...

DESCRIPTION

kmadmin is the administrative command for static and loadable kernel modules. It performs the following functions:

- loads a kernel module into a running system
- unloads a kernel module from a running system
- displays the status of kernel module(s) currently loaded or registered
- modifies the search path for kernel modules

The loadable modules feature enables adding a module to a running system without rebooting the system or rebuilding the kernel. When the module is no longer needed, this feature also allows the module to be dynamically removed, thereby freeing system resources for other use.

Loadable modules are maintained in individual object files in the same manner as statically configured modules. Unlike static modules, loadable modules:

- are not linked to the kernel until they are needed
- must be configured into the system and registered with the running kernel using the **config** command, before they can be loaded
- must be configured in loadable form (requires writing additional module initialization or *wrapper* code)
- can be loaded and unloaded by using the **kmadmin** command
- can be loaded by the kernel itself (called an auto load)

Auto-load occurs when the kernel detects a particular loadable module is required to accomplish some task, but is not currently loaded. The kernel automatically loads the module.

Options

The **kmadmin** options have the following meanings:

-d *pathname*

Prepend the *pathname* to the current loadable modules search path, where *pathname* specifies directories that should be searched:

for all subsequent demand loads initiated by a **kmadmin** command with the option **-L** and a named *module_name*,

for all subsequent loads performed by the kernel's auto-load mechanism (see note below),

prior to searching any directories already prepended to the search path by a prior **kmadmin** command with the **-d** option, and

prior to searching the default search path **/stand/dlkm/mod.d** or **/stand/dlkm.***current.vmunix***/mod.d**.

pathname must specify an absolute pathname or a list of absolute pathnames delimited by colons. The directories identified by *pathname* do not have to exist on the system at the time the request to modify the search path using **kmadmin** is made. If these directories do not exist at the time a load takes place, the load operation ignores them.

All modifications to the search path made using this option take effect immediately and affect all subsequent loads (demand and auto-load) and all users on the system.

-D Reset the kernel modules search path to its default value. The default value can be one of two search paths depending upon the running kernel. When the running kernel is **/stand/vmunix**, the default value is **/stand/dlkm/mod.d**. When the running kernel is **/stand/current.vmunix**, the default value is **/stand/dlkm.** *current.vmunix* **/mod.d**. The reset takes effect immediately and affects all subsequent loads (demand and auto-load) and all users on the system.

-k Print a list of all statically configured modules.

-L *module_name*
Load the named module(s), using the current value of the search path to locate the module's object file on disk.

This option searches for a matching file in all directories specified in the search path. The default search *pathname* can be one of two values. The *pathname* is **/stand/dlkm.** *current.vmunix***/mod.d** when the running kernel is **/stand/***current.vmunix* or *pathname* is **/stand/dlkm/mod.d** when the running kernel is **/stand/vmunix**.

The load operation performs all tasks associated with link editing the module to the kernel and making the module accessible to the system. If the module depends on other kernel modules (as defined in **/usr/conf/master.d**), and these modules are not currently loaded, **kmadmin** will automatically load the dependent modules during the load operation.

When loading completes, an integer *module_id* prints on the standard output to identify the module(s) that was loaded.

-L *pathname*
Same as **-L** *module_name*, except the absolute pathname, *pathname*, is used to locate the kernel module's object file.

-U *module_name*
Unload the named module(s) *module_name*.

The unload operation performs all tasks associated with disconnecting the module from the kernel and releasing any memory acquired by the module. When unloading completes, a message is displayed to standard output notify the user that the module(s) that has been unloaded.

If the module(s) to be unloaded are currently in use, are dependents of a loadable module that is currently loaded, or are currently being loaded or unloaded, the unload request will fail.

-u *module_id*
Same as **-U** *module_name*, except that module(s) to be unloaded is identified by the integer value *module_id*. If *module_id* is 0 (zero), **kmadmin** attempts to unload all loaded modules.

-q *module_id*
Print the status of loaded or registered module(s) identified by the integer value *module_id*. Information returned by this option includes:

module name

module identifier (*module_id*)

the module's *pathname*

module status

module size

the module's virtual load address

the memory size of BSS

the base address of BSS

the module's reference count

the module's dependent count

the module's unload delay value

the module's descriptive name

the type of module

Depending on the type of module, information on the module's character major number, block major number and flags may also be printed.

-Q *module_name*
Same as **-q** *module_id*, except the module(s) for which status information is to be reported is specified by *module_name* rather than *module_id*.

-s Print an abbreviated status for all modules currently registered or loaded. This option returns a listing of module name, module id, status and type.

Example:

```
Name          ID      Status          Type
============================================
hello         1       UNLOADED        Misc
misato        2       UNLOADED        WSIO
stape         3       UNLOADED        WSIO
```

-S Print the full status for all modules currently loaded. This option returns status information of the form returned by the **-q** options.

DIAGNOSTICS

kmadmin fails in the following cases:

kmadmin: Incorrect usage

Command line input contained one or more syntax errors. See the **SYNOPSIS** section for the correct usage.

kmadmin: *module_id* **: Invalid argument**

Unable to load the module corresponding to *module_id* because the module does not exist.

kmadmin : Device busy

Unable to load a module because the module is currently in-use.

kmadmin : Non-numeric ID string: string

Unable to unload or obtain status for a module because the *module_id* string specified a non-numeric value.

kmadmin: modstat: Invalid argument

Unable to obtain status for module, *module_id*, because the module does not exist.

kmadmin: Module: *module_name***, not found**

Unable to obtain status for *module* because the module is currently not registered.

FILES

/stand/dlkm/mod.d/*	Default search path for kernel modules when **/stand/vmunix** is the running kernel.
/stand/dlkm.*current.vmunix***/mod.d/***	Default search path for kernel modules when **/stand/***current.vmunix* is the running kernel.

SEE ALSO

config(1M), kmmodreg(1M), kmtune(1M), modload(2), modpath(2), modstat(2), moduload(2), loadmods(4).

k

NAME

kminstall - add, delete, update a kernel module

SYNOPSIS

/usr/sbin/kminstall [**-a**|**-d**|**-u**] *module_name*

DESCRIPTION

kminstall will add (**-a**), delete (**-d**) or update (**-u**) a module on the system.

kminstall expects to find the module component files in the current directory. When components are installed or updated with **-a** or **-u** option, they are copied into subdirectories of the **/usr/conf** and **/stand** directories.

Options

The options for **kminstall** are:

-a Add the components for the named module, *module_name*.

To create the module's components, **kminstall** copies:

mod.o to **/usr/conf/km.d/***module_name***/mod.o**

master to **/usr/conf/master.d/***module_name*

system to **/stand/system.d/***module_name*

If **node**, **space.h**, and **Modstub.o** files are also present in the current directory, **kminstall** also copies:

node to **/usr/conf/node.d/***module_name*

space.h to **/usr/conf/km.d/**module_name**/space.h**

Modstub.o to **/usr/conf/km.d/**module_name**/Modstub.o**

kminstall expects a readable **mod.o**, **master**, and **system** file in the current directory. It creates the required directories if they do not exist. If *module_name* already exists on the system, **kminstall** prints a message and fails.

-d Remove the components for the named module, *module_name*.

To remove the module's components, **kminstall** first unloads and unregisters the module. Then **kminstall** deletes the following files:

/usr/conf/km.d/*module_name***/mod.o**

/usr/conf/master.d/*module_name*

/usr/conf/km.d/*module_name***/space.h** (if present)

/usr/conf/km.d/*module_name***/Modstub.o** (if present)

/usr/conf/node.d/*module_name* (if present)

/stand/system.d/*module_name*

/stand/dlkm/mod.d/*module_name*

/stand/dlkm/system.d/*module_name* (if present)

/stand/dlkm/node.d/*module_name* (if present)

kminstall also deletes the directory entries:

/usr/conf/km.d/*module_name*

/stand/dlkm/mod.d/*module_name*

If *module_name* is configured as a loadable module and its entry is in the **/etc/loadmods** file (see *loadmods*(4)), then **kminstall** prints a warning message and removes the module entry from **/etc/loadmods**.

If *module_name* is loaded, **kminstall** tries to unload the module. If the unload fails, it prints a message and exits with an error; otherwise, **kminstall** tries to unregister the module. If the unregistration fails, then **kminstall** prints a message and exits with an error.

-u Update the components for the named module, *module_name*.

To update the module's components, **kminstall** copies:

mod.o to **/usr/conf/km.d/***module_name***/mod.o**

master to **/usr/conf/master.d/***module_name*

updated **system** to **/stand/system.d/***module_name*

If **node**, **space.h**, and **Modstub.o** files are also present in the current directory, **kminstall** copies:

node to **/usr/conf/node.d/***module_name*

space.h to **/usr/conf/km.d/***module_name***/space.h**

Modstub.o to **/usr/conf/km.d/***module_name***/Modstub.o**

kminstall expects a readable **mod.o**, **master**, and **system** file in the current directory. If *module_name* already exists on the system, **kminstall** updates the module. When updating an existing module, the values of the tunable parameters and the **$LOADABLE** and **$CONFIGURATION** flags are taken from the running system and replace those in the new **system** file for the module.

If *module_name* does not exist on the system, then **kminstall** prints a warning and adds the module to the system.

kminstall creates the required directories if they do not exist.

k

RETURN VALUE

An exit value of zero indicates success. If an error occurs, **kminstall** exits with a non-zero value and reports an error message. Error messages are self-explanatory.

FILES

/usr/conf/master.d/*	Default input master kernel configuration tables
/stand/system.d/*	Default kernel module description files
/usr/conf/km.d/*module_name***/mod.o**	Module object file
/usr/conf/master.d/*module_name*	Module master file
/stand/system.d/*module_name*	Module description file
/usr/conf/km.d/*module_name***/space.h**	Module configuration file
/usr/conf/node.d/*module_name*	Module node file
/usr/conf/km.d/*module_name***/Modstub.o**	Module object file required by stub module
/stand/dlkm/mod.d/*module_name*	Loadable image of module

SEE ALSO

config(1M), loadmods(4), master(4).

NAME

kmmodreg - register or unregister loadable kernel modules with the running kernel

SYNOPSIS

/usr/sbin/kmmodreg [[**-M** *module_name*]...][**-r** *mod_register_root*] [**-c** *mod_reg_root*]

/usr/sbin/kmmodreg [[**-U** *module_name*]...][**-r** *mod_register_root*] [**-c** *mod_reg_root*]

DESCRIPTION

kmmodreg registers all of the loadable kernel modules listed in the **mod_register** file located under either **/stand/dlkm.***current_vmunix***/** when the running kernel is *current_vmunix*, or **/stand/dlkm** when the running kernel is **/stand/vmunix**. All loadable kernel modules need to be registered by **kmmodreg** before they can be automatically-loaded by the running kernel (i.e., upon module access by an application or user process), or demand-loaded by an administrator issuing the **kmadmin** command.

The **mod_register** file is generated whenever **config** is run to create a new kernel and contains the registration information for any (and all) configured loadable modules. When **config -M** is run to configure a loadable kernel module, the entries for the module are appended to the **mod_register**. The **mod_register** file is not expected to be edited manually. An individual module's registration information is also created by **config** and stored in the **mod_reg** file located under **/stand/dlkm/mod_bld.d** directory.

Options

kmmodreg takes the following options:

-r *mod_register_root*
Use to specify a directory other than **/stand/dlkm.***current_vmunix* or **/stand/dlkm/** as the location for the **mod_register** file that is used to register modules.

-c *mod_reg_root*
Use the individual module registration information under the *mod_reg_root* directory instead of **/stand/dlkm/mod_bld.d.**

-M *module_name* [*module_name*]
Register the specified loadable kernel module, and append an entry (or entries) for the module(s) to the **mod_register** file. This will effect registration of the specified module(s) at every system reboot.

-U *module_name*
Unregister the specified loadable kernel module, and remove an entry (or entries) for the module from the *mod_register* file, so it will not be registered every time the system is rebooted.

NOTES

The **kmmodreg** command is executed automatically at every system reboot. **kmupdate** also calls **kmmodreg**, with the **-M** option, when a loadable kernel module configuration is requested. **kmmodreg** can also be invoked as a user-level command to register all of the loadable kernel modules.

WARNINGS

The **mod_register** file format may change or be eliminated in the future.

FILES

/stand/dlkm	Default *mod_register_root* directory
/stand/dlkm.*current_vmunix***/mod_register**	Default *mod_register* file
/stand/dlkm/mod_bld.d/*module_name***/mod_reg**	Module registration information

Each **mod_register** file entry provides registration information about a single module. The information is contained in a single-line entry. All fields are positional and are separated by colons. The subfields are separated by commas. The entry is of the form:

module-name **:** *module-type* **:** *type-specific-data*

where:

- *module-name* identifies the module to which the entry belongs
- *module-type* contains an integer representing the module type
- *type-specific-data* includes additional information that depends on the type of the module

RETURN VALUE

An exit value of zero indicates successful completion of the command. If errors occur, **kmmodreg** reports error messages for each error and exits with the return value **1**. If the error is a failure to register a module, an error message is reported, but the command continues processing the remaining modules listed in the **mod_register** file. If no modules are processed, **kmmodreg** returns a value of **2**.

SEE ALSO

config(1M), kmadmin(1M), kmupdate(1M).

NAME

kmsystem - set, query configuration and loadable flags for a module

SYNOPSIS

/usr/sbin/kmsystem [**-S** *system_file*]

/usr/sbin/kmsystem [**-c** {**Y**|**y**|**N**|**n**}] [**-l** {**Y**|**y**|**N**|**n**}] [**-q**]
[**-S** *system_file*] *module_name*

DESCRIPTION

Without any option or with the **-S** option only, **kmsystem** prints the information on the **$LOADABLE** and **$CONFIGURATION** flags of all modules. The **-q** option may be used to print information about the specified module only. The **$CONFIGURATION** flag for *module_name* is set using the **-c** option, and the **$LOADABLE** flag is set with the **-l** flag. When *module_name* is specified on the command line, one or more of the **-c**, **-l**, or **-q** flags must also be specified.

Options

-c *value* Set the configuration status of *module_name*to *value*. *value* must be **Y** or **y** to configure the module, or **N** or **n** to not configure it.

If the system file for the module (**/stand/system.d/***module_name*) exists but does not contain the **$CONFIGURE** flag, then an error message is printed. Otherwise, the flag is set to *value*.

If the system file for the module does not exist, then the standard system file (see **-S** option) is searched. *module_name* is added or removed from that system file according to *value*.

k

-l *value* Set the **$LOADABLE** flag in the system file of *module_name* to *value*. *value* must be **Y** or **y** to make the module loadable, or **N** or **n** to specify that it should be statically linked. If the system file for the module does not exist, **kmsystem** exits with an error. If the system file exists, but the **$LOADABLE** flag is not present in the file, then the module is a static module, and **kmsys-tem** exits with an error.

-q Print the loadable and configuration flag information for *module_name*. If the loadable information does not apply, then a **-** is printed.

-S *system_file*
Specify the HP-UX system description file name. Users should specify the complete path to the file name; otherwise, **kmsystem** will search the current directory for the specified file. The default HP-UX system description file if the **-S** option is not specified is **/stand/system**. This option is for backward compatibility.

EXAMPLES

To display the configuration and loadable status of the stape module:

```
/usr/sbin/kmsystem -q stape
```

To specify that the stape module should be statically linked:

```
/usr/sbin/kmsystem -l N stape
```

NOTES

System administrators are encouraged to use **kmsystem** and **kmtune** instead of editing system description files manually. File format of system description files are subject to change, and **kmsystem** provides compatibility in the event of a format change.

RETURN VALUE

Upon successful completion, **kmsystem** returns with one a 0; otherwise it returns with a 1.

DIAGNOSTICS

Output for queries is sent to stdout. Error messages are sent to stderr. Messages from **kmsystem** are self explanatory.

FILES

/usr/conf/master.d/*	Master configuration tables for kernel and kernel modules

`/stand/system`	Default HP-UX system description file
`/stand/system.d/*`	Kernel module system description files

SEE ALSO

kmtune(1M), master(4).

NAME

kmtune - query, set, or reset system parameter

SYNOPSIS

/usr/sbin/kmtune [**-l**] [[**-q** *name*]...] [**-S** *system_file*]

/usr/sbin/kmtune [[**-s** *name* {**=**|**+**}*value*]...] [[**-r** *name*]...]
[**-S** *system_file*]

DESCRIPTION

kmtune is used to query, set, or reset system parameters. **kmtune** displays the value of all system parameters when used without any options or with the **-S** or **-l** option. **kmtune** reads the master files and the system description files of the kernel and kernel modules.

Options

The following options are recognized by **kmtune**:

-l Print a detail report. The **-l** option cannot be used with the **-r** or **-s** options.

-q *name*
Query the value of the specified system parameter.

-r *name*
Reset the value of a system parameter to the default.

-s *name*{**=**|**+**}*value*
Set the value to a system parameter. If the separator is an equal sign (**=**), the parameter is set to the value specified. If the separator is a plus sign (**+**), the parameter is incremented by the value specified. Negative values cannot be used with plus sign (**+**). The name {**=**|**+**}*value* format must not include spaces or tabs.

-S *system_file*
Specify the HP-UX system description file name. If not specified, **/stand/system** is used as the default.

If the **-q** query option is specified, **kmtune** displays the following format:

Brief report without **-l** option

```
Parameter          Value
===========================
name               value
```

Detailed report with **-l** option

```
Parameter:      name
Value:          value
Default:        default
Minimum:        minimum
Module:         module
```

If the **-l** option is specified without the **-q** query option, a detailed report on all the parameters is displayed. The information between the parameters is separated by blank lines.

If the parameter has no minimum value specified in master file, *minimum* will be displayed as '**-**'. If the parameter is not supplied by kernel modules, *module* will be displayed as '**-**'.

If the **-s** set option is specified with an equal (**=**) separator and the minimum value of the parameter is described in a master file, the value range is checked. If the minimum value or the specified value is a formula, the check is not made.

If the **-s** set option with a plus (**+**) separator is specified and the original value is non numeric, an error is reported.

NOTES

System administrators are encouraged to use **kmsystem** and **kmtune** instead of editing description files manually. File format of description files are subject to change, and **kmtune** is intended to provide compatibility in case of format change.

RETURN VALUE

Upon completion, `kmtune` returns with one of the following exit values:

0	Successful.
1	Requested parameter is not found, the value is out of range, or the type of value is formula.
2	Syntax error.
>2	Environmental error.

Results of query requests are sent to stdout. Error and warning messages are sent to stderr.

EXAMPLES

```
# kmtune -q shmseg
  Parameter            Value
  ================================
  shmseg               120
# kmtune -s shmseg=128
# kmtune -l -q shmseg
  Parameter:      shmseg
  Value:          128
  Default:        120
  Minimum:        -
  Module:         -
# kmtune -r shmseg
# kmtune -q shmseg
 Parameter        Value
 ================================
 shmseg           120
```

FILES

`/usr/conf/master.d/*`	Master configuration tables for kernel and kernel modules
`/stand/system`	Default HP-UX system description file
`/stand/system.d/*`	Kernel module system description files

SEE ALSO

kmsystem(1M), master(4).

NAME

kmupdate - update default kernel file and files associated with the kernel, or update specified kernel modules

SYNOPSIS

/usr/sbin/kmupdate [*kernel_file*]

/usr/sbin/kmupdate -M *module_name* [[**-M** *module_name*]...] [**-i** | **-a**]

DESCRIPTION

This command can be invoked to either update the kernel and the kernel modules associated with the kernel (i.e., **/stand/dlkm**, which is the kernel function set directory), or to update only the specified kernel modules.

Updating the Kernel and the Associated Kernel Function Set Directory

The first form of **kmupdate** is used to initiate the move of the specified *kernel_file* to the default kernel located at **/stand/vmunix** during the next system shutdown or startup. The directory associated with the specified *kernel_file*, the kernel function set directory, is also moved to **/stand/dlkm** at the next shutdown or startup. If *kernel_file* is not specified, **/stand/build/vmunix_test** is used as the *kernel_file* to use for the update.

kmupdate is useful in cases where the kernel is built either by **config** without its **-u** option, or by **mk_kernel** with its **-o** option (which specifies a kernel other than the default). In these cases the administrator should use **kmupdate** to update the kernel file and its associated kernel function set directory for the next shutdown or startup.

NOTE: Overwriting or replacing the kernel file and associated kernel function set directory using commands like **cp** or **mv** should be avoided.

Options for Updating Specified Loadable Kernel Modules

The second form of **kmupdate** supports the following options.

-M *module_name*
: Update specified *module_name* module. Without **-a** or **-i**, **kmupdate** will attempt to update *module_name* immediately. If *module_name* cannot be updated immediately, the module will be updated asynchronously, as described below.

-i
: When specified, **kmupdate** will only attempt an immediate update.

-a
: When specified, **kmupdate** will update asynchronously without attempting an immediate update.

Immediate Update of Specified Kernel Modules

kmupdate may be used for immediately updating the loadable image of a newly created kernel module, without a reboot. If the *module_name* is loaded, **kmupdate** tries to unload it and, if the **-i** option is specified and the module cannot be unloaded, **kmupdate** exits with an error. If the kernel module was either not loaded or successfully unloaded, **kmupdate** checks if it is registered, and if so, unregisters the module. If the kernel module cannot be unregistered, **kmupdate** exits with an error if **-i** is specified; otherwise the module will be updated asynchronously. If the unregistration succeeds, **kmupdate** overlays the existing loadable image of the module with the newly generated image. It then registers the module with the latest registry information and performs module type specific initialization, if required. If the module was loaded originally, **kmupdate** reloads the module before exiting.

Asynchronous Update of Specified Kernel Modules

If the **-a** option is specified, the module will be updated asynchronously without first attempting an immediate update. An asynchronous update occurs at shutdown. When the system shuts down, the module's loadable image is updated. The module is registered when the system is restarted.

RETURN VALUE

kmupdate returns 0 upon normal completion, and 1 if an error occurred.

DIAGNOSTICS

Messages that notify an update is successful are sent to stdout. Error messages are sent to stderr.

FILES

`/stand/vmunix`	Default kernel file
`/stand/dlkm`	Default kernel function set directory

SEE ALSO

mk_kernel(1M), config(1M).

NAME

lanadmin - local area network administration program

SYNOPSIS

/usr/sbin/lanadmin [**-e**] [**-t**]

/usr/sbin/lanadmin [**-a**] [**-A** *station_addr*] [**-b**] [**-B** on|off] [**-m**] [**-M** *mtu_size*] [**-R**] [**-s**] [**-S** *speed*] *PPA*

DESCRIPTION

The **lanadmin** program administers and tests the Local Area Network (LAN). For each interface card, it allows you to:

- Display and change the station address.
- Display and change the 802.5 Source Routing options (RIF).
- Display and change the maximum transmission unit (MTU).
- Display and change the speed setting.
- Clear the network statistics registers to zero.
- Display the interface statistics.
- Reset the interface card, thus executing its self-test.

For operations other than display, you must have superuser privileges.

lanadmin reads commands from standard input, writes prompts and error messages to standard error, and writes status information to standard output. When the program is run from a terminal, the interrupt key (usually **^C**) interrupts a currently executing command; the eof key (usually **^D**) terminates the program.

lanadmin operates in two modes: Menu Mode (see the first SYNOPSIS line) and Immediate Mode (see the second SYNOPSIS line). If at least one **-aAbBmMRsS** option is supplied, **lanadmin** executes in Immediate Mode. Otherwise, it executes in Menu Mode.

NOTE: **lanadmin** replaces the now obsolete **landiag** command beginning at 10.0.

Options and Arguments

lanadmin recognizes the following Immediate Mode options and arguments. At least one **-aAbBmMRsS** option and the *PPA* argument must be supplied.

PPA	The Physical Point of Attachment (PPA) number of the LAN interface. This argument is ignored if none of the **-aAbBmMRsS** options are used (Menu Mode). Any options specified after *PPA* are ignored. Appropriate values can be displayed with the **lanscan** command (see *lanscan*(1M)).
-a	Display the current station address of the interface corresponding to *PPA*.
-A *station_addr*	Set the new station address of the interface corresponding to *PPA*. The *station_addr* must be entered in hex format with a ’0x’ prefix. You must have superuser privileges. WARNING: To ensure the interface and the system work correctly, the interface MUST be brought down before setting the new station address. After the new station address is set, the interface should be brought up in order to be functional. See *ifconfig*(1M) for bringing down and bringing up the interface.
-b	Display the current 802.5 source routing option for the interface corresponding to *PPA*.
-B on|off	Turn the 802.5 source routing option “on” or “off” for the interface corresponding to *PPA*. The default value for HP devices is “on”. You must have superuser privileges.
-m	Display the current MTU size of the interface corresponding to *PPA*. You must have superuser privileges.
-M *mtu_size*	Set the new MTU size of the interface corresponding to *PPA*. The *mtu_size* value must be within the link specific range. You must have superuser privileges.
-R	Reset the MTU size of the interface corresponding to *PPA* to the default for that link type. You must have superuser privileges.

-s	Display the current link speed setting of the interface corresponding to *PPA*.
-S *speed*	Set the new link speed setting of the interface corresponding to *PPA*. You must have superuser privileges.

lanadmin recognizes the following Menu Mode options. They are ignored if they are given with an Immediate Mode option.

-e	Echo the input commands on the output device.
-t	Suppress the display of the command menu before each command prompt. This is equivalent to the Test Selection Mode **terse** command. The default is **verbose**.

Immediate Mode

In Immediate Mode, you can display the station address, source routing option, MTU size, and link speed of LAN interface *PPA*. For certain interfaces, if you have superuser privileges you can also modify the station address, source routing option, MTU size, and link speed. See "Options and Arguments" above.

Menu Mode

In Menu Mode, you can select an interface card, display statistics for the selected card, reset the card, and clear the statistics registers.

Menu Mode accepts either complete command words or unique abbreviations, and no distinction is made between uppercase and lowercase letters in commands. Multiple commands can be entered on one line if they are separated by spaces, tabs, or commas.

Test Selection Mode Menu

This menu is entered when Menu Mode is first selected. The available Test Selection Mode commands are:

lan	Select the LAN Interface Test Mode menu.
menu	Display the Test Selection Mode command menu.
quit	Terminate the **lanadmin** program.
terse	Suppress the display of command menus.
verbose	Restore the display of command menus.

LAN Interface Test Mode Menu

The following commands are available:

clear	Clear the LAN interface network statistics registers to zero. You must have superuser privileges.
display	Display the RFC 1213 MIB II statistics. Depending on the link, the type-specific MIB statistics may also be displayed. For instance, for Ethernet links, the RFC 1398 Ethernet-like statistics are displayed.
end	Return **lanadmin** to Test Selection Mode.
menu	Display the LAN Interface Test Mode command menu.
ppa	Prompt for a *PPA* that corresponds to a LAN interface card. It defaults to the first LAN interface encountered in an internal list. Appropriate values can be displayed with the **lanscan** command (see *lanscan*(1M)).
quit	Terminate the **lanadmin** program.
reset	Reset the local LAN interface card, causing it to execute its self-test. Local access to the network is interrupted during execution of **reset**. You must have superuser privileges.

WARNINGS

Changes made to an interface's station address or mtu interactively with the **lanadmin** command will not be preserved between system reboots. A user must modify the initialization configuration files for this feature, either manually editing configuration files or through the **SAM** interface.

AUTHOR

lanadmin was developed by HP.

SEE ALSO

netstat(1), lanscan(1M), linkloop(1M), ping(1M), lan(7).

DARPA Requests for Comments: RFC 1213, RFC 1398.

NAME
lanscan - display LAN device configuration and status

SYNOPSIS
lanscan [**-aimnpv**] [*system* [*core*]]

DESCRIPTION
lanscan displays the following information about each LAN device that has software support on the system:

- Hardware Path.
- Active Station Address (also known as Physical Address).
- Card Instance Number
- Hardware State.
- Network Interface "NamePPA". The Network Interface "Name" and the "PPA" (Physical Point of Attachment) number are concatenated together. A single hardware device may have multiple "NamePPA" identifiers, which indicates multiple encapsulation methods may be supported on the device. For Ethernet/IEEE 802.3 links, the "Name" **lan** is used to designate Ethernet encapsulation, and **snap** for IEEE 802.3 encapsulation. For other links (FDDI, Token Ring), only the **lan** encapsulation designation is used.
- Network Management ID.
- MAC Type.
- HP DLPI Supported. Indicates whether or not the lan device driver will work with HP's Common Data Link Provider Interface.
- DLPI Major Number.
- Extended Station Address for those interfaces which require more than 48 bits. This is displayed only when the **-v** option is selected.
- Encapsulation Methods that the Network Interface supports. This is displayed only when the **-v** option is selected.

The arguments *system* and *core* allow substitution for the default values **/stand/vmunix** and **/dev/kmem**.

Options
lanscan recognizes the following command-line options:

-a	Display station addresses only. No headings.
-i	Display interface names only. No headings.
-m	Display MAC types only. No headings.
-n	Display Network Managements IDs only. No headings.
-p	Display PPA numbers only. No headings.
-v	Verbose output. Two lines per interface. Includes displaying of extended station address and supported encapsulation methods.

WARNINGS
lanscan does not display information about LAN devices that do not have software support such as LAN interface cards that fail to bind properly at boot-up time.

AUTHOR
lanscan was developed by HP.

SEE ALSO
ifconfig(1M), ioscan(1M), lanadmin(1M), linkloop(1M), lan(7).

NAME

link, unlink - execute `link()` and `unlink()` system calls without error checking

SYNOPSIS

/usr/sbin/link *file1 file2*

/usr/sbin/unlink *file*

DESCRIPTION

The **link** and **unlink** commands perform their respective system calls (**link()** or **unlink()**) on their arguments, abandoning most error checking.

These commands can be executed only by users who have appropriate privileges.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **link** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

link and **unlink** return the following values:

0	Operation successful.
1	Input syntax error.
2	The **link()** or **unlink()** call failed.

WARNINGS

If a directory that contains files other than **.** and **..** is unlinked, the files become orphans, unless they are also linked by some other directory.

Not all file systems permit linking to directories.

SEE ALSO

ln(1), rm(1), link(2), unlink(2).

STANDARDS CONFORMANCE

link: SVID2, SVID3

unlink: SVID2, SVID3

NAME

linkloop - verify LAN connectivity with link-level loopback

SYNOPSIS

linkloop [**-i** *PPA*] [**-n** *count*] [**-r** *rif*] [**-s** *size*] [**-t** *timeout*] [**-v**] *linkaddr* ...

DESCRIPTION

The **linkloop** command uses IEEE 802.2 link-level test frames to check connectivity within a local area network (LAN).

linkaddr is the hardware station address of a remote node. Several addresses can be specified at one time.

linkloop tests the connectivity of the local node and the remote node specified by each hardware station address. The hardware station address of a remote node can be found by executing **lanscan** on the remote node. This hardware station address is usually represented as a hexadecimal string prefixed with **0x**. It can also be represented as a octal string prefixed with **0** or as a decimal string. The hardware station address must not be a multicast or broadcast address.

Options

linkloop recognizes the following options:

-i *PPA* Specify the *PPA* to use. If this option is omitted, **linkloop** uses the first *PPA* it encounters in an internal data structure.

-n *count* Set the number of frames to transmit. If *count* is **0**, **linkloop** transfers frames indefinitely until an interrupt signal (defined by the user shell) is received. The default value for *count* is **1**.

-r *rif* Specify the particular bridge route over which token ring packets should be delivered. *rif* is the *routing information field* used for token-ring networks. Its value is given as an even number of hexadecimal bytes separated by colons, up to a maximum of 16 bytes.

-s *size* Set the size in bytes of the data message to send. The maximum data size is dependent on the type of LAN link being used. The default value is the maximum data byte count that can be used for the particular link.

-t *timeout* Set the amount of time in seconds to wait for a reply from the remote node before aborting. If *timeout* is **0**, **linkloop** waits indefinitely for a reply. The default value is 2 seconds.

-v Set the verbose option. In addition to the regular summary of test results, this option displays more extensive error information. If there are header or length errors, appropriate messages are displayed. All verbose output is preceded by the number of replies accepted before an error occurred.

Connectivity Test Results

linkloop aborts upon receipt of an interrupt signal. If aborted, the current results are printed.

linkloop prints the result of the link-level connectivity test. If the test fails, it prints a summary of the test and indicates the type of error. The possible messages are:

address has bad format

An incorrect hardware station address was entered on the command line.

address is not individual

The station address entered on the command line is either a multicast or broadcast address.

frames sent

Total number of frames sent.

frames received correctly

Total number of frames received without errors.

frames with length error

Received frame length does not match transmitted frame length. If the verbose option is set, the length received is printed.

`frames with data error`

Received frame does not match transmitted frame.

`frames with header error`

Number of frames received containing unexpected frame header information. Either the source address does not match the remote address, the destination address does not match the local address, or the control field is not the TEST **frame control field**. These frames are ignored. **`linkloop`** continues to try to receive the reply frame until the **`read`** operation times out.

`reads that timed out`

Count of how many **`read`** operations timed out before the reply was received.

DIAGNOSTICS

`illegal count parameter`

The *count* specified in the **`-n`** option is a negative integer, or the number specified is too large for the local computer.

`illegal timeout parameter`

The *timeout* specified in the **`-t`** option is a negative integer, or the value specified multiplied by 1000 is too large for the local computer.

`illegal size parameter`

The *size* specified in the **`-s`** option is not in the range from 0 to the maximum link data size. Remember that the maximum link data size can vary in value for different LAN connection types. The current MTU can be obtained with the **`linkloop`** command.

`No valid interface associated with PPA`

The *PPA* specified in the **`-i`** option is not a valid PPA.

`Unable to open device file /dev/dlpi`

Device file **`/dev/dlpi`** does not exist.

`invalid rif parameter`

The *rif* value in the **`-r`** option is invalid.

`rif parameter too long`

The number of bytes in *rif* in the **`-r`** option exceeded 16, which is the maximum allowed.

`rif parameter length must be even`

The number of bytes in *rif* in the **`-r`** option is odd. The number of bytes must be even.

AUTHOR

`linkloop` was developed by HP.

SEE ALSO

lanadmin(1M), lanscan(1M), lan(7).

NAME

localedef - generate a locale environment

SYNOPSIS

localedef [**-cenvw**] [**-C** *compiler_options*] [**-L** *loader_options*] [**-m** *method_file*]
[**-f** *charmap_file*] [**-i** *locale_definition*] *locale_name*

DESCRIPTION

localedef sets up the language environment for the named locale. **localedef** reads a **locale definition** file (see *localedef*(4) for a detailed description) from standard input (default) or from *locale_definition* file, creates a locale file with the same name as specified for the *locale_name* parameter, and optionally installs this locale in the appropriate directory. Installation of public locales (those accessible to all users) requires appropriate privileges. Creation of locales (both private and public) requires access to the ANSI C compiler.

Options

localedef recognizes the following options:

-c Create permanent output even if warning messages have been generated.

-e Generate 64-bit locale in addition to the 32-bit locale. This is the default on a 64-bit operating system and is included to allow cross platform development.

-n (noinstall) Create the locale file in the current directory.

-v (verbose) Generate as many diagnostic messages as possible.

-w Generate additional warning messages for duplicate definitions and ellipses use in the **LC_COLLATE** category.

-f *charmap_file*
If **locale definition** file contains symbolic names (of the form **<***name***>**) use *charmap_file*. See *charmap*(4) for a description of the format of a *charmap_file*.

-i *locale_definition*
Use *locale_definition* file as input, instead of standard input (default).

-m *method_file*
Use the specified *method_file* to overwrite use of default methods in processing the **locale definition**.

-C *compiler_options*
Specify additional compiler options to be applied in compiling the locale. See *cc*(1) for a complete list of options. Use with care on a 64-bit operating system since the additional default option includes +DA2.0W.

-L *loader_options*
Specify additional loader options to be applied in linking the locale. See *ld*(1) for a complete list of options.

locale_name This argument is required, and identifies the name of the language following the naming convention of the **LANG** environment variable (see *environ*(5)):

language [*_territory*] [*.codeset*]

The following is a brief description of the components that make up a locale. For a complete description of the form and syntax of a **locale definition** file, see *localedef*(4). For a complete description of the form and effects of a charmap file, see *charmap*(4).

Six categories of data in the **locale_name** file are recognized by *setlocale*(3C), and make up a language definition:

LC_COLLATE	Information in this category affects behavior of regular-expressions and NLS string-collation functions.
LC_CTYPE	Information in this category affects behavior of character classification and conversion functions.
LC_MONETARY	Information in this category affects behavior of functions that handle monetary values.

LC_NUMERIC	Information in this category affects handling of the radix character in formatted-input/output and string-conversion functions.
LC_TIME	Information in this category affects behavior of time-conversion functions.
LC_MESSAGES	This category contains information affecting interpretation of yes/no responses.

A **locale definition** file also consists of six categories. The beginning of each category is identified by a **category tag** having the form **LC_***category* where *category* is one of the following: **CTYPE**, **COLLATE**, **MONETARY**, **NUMERIC**, **TIME**, or **MESSAGES**. The end of each category is identified by a tag consisting of the word **END** followed by a space and the category identifier; for example, **END LC_COLLATE**. Categories can appear in any order in the **locale definition** file. At least one category specifications is required. If a category is not specified, **setlocale()** sets up the default "C" locale for that category (see *setlocale*(3C) and *lang*(5)).

Each category is composed of one or more statements. Each statement begins with a keyword followed by one or more expressions. An expression is a set of well-formed metacharacters, strings, and constants. **localedef** also recognizes comments and separators.

More than one definition specified for each category constitutes a hard error (causes **localedef** to exit without generating a locale). Any category can be specified by the keyword **copy** followed by the name of a valid locale. This causes the information for the category to be identical to that in the named locale. Note that the **copy** keyword, if used for a category, must be the first and only keyword following the category tag.

A methods file is used to creat locales for user-specific character encoding schemes.

Operating System Requirements

For cross platform development and development on a 64-bit operating system several requirements must be observed. Both the 32-bit and 64-bit method libraries must exist. In the case of the 64-bit shared library it must be in the directory **pa20_64** under the location where the 32-bit library is located. When the **-e** option is specified, or when executing on a 64-bit operating system, the resulting locale is placed in the directory **pa20_64** under the current working directory unless the install option has been specified.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use when neither **LC_ALL** or the other category variables specify a locale.

LC_ALL determines locale to be used. It overrides any values specified by **LANG** or any other **LC_*** variables.

LC_COLLATE and **LC_CTYPE** have no effect on the processing of localedef, which behaves as if these two variables were set to the C locale.

LC_MESSAGES determines the language in which messages are displayed.

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

localedef returns the following values:

0	No errors occurred and the locale was successfully created.
1	Warnings occurred and the locale was successfully created.
2	The locale specification exceeded implementation limits or the coded character set used is not supported.
>3	Warnings or errors occurred, and no output was generated.

AUTHOR

localedef was developed by OSF and HP.

FILES

/usr/lib/nls/config
/usr/lib/nls/loc/src
/usr/lib/nls/loc/charmaps
/usr/lib/nls/loc/methods
/usr/lib/nls/loc/pa20_64/methods

`/usr/lib/nls/loc/locales/`*language*[*_territory*][`.`*codeset*]

SEE ALSO

locale(1), localedef(4), charmap(4), setlocale(3C), environ(5).

STANDARDS CONFORMANCE

`localedef`: XPG4, POSIX.2

NAME

lockd - network lock daemon

SYNOPSIS

/usr/sbin/rpc.lockd [**-l** *log_file*] [**-t** *timeout*] [**-g** *graceperiod*]

DESCRIPTION

lockd is an RPC server that processes NFS file locking requests from the local kernel or from another remote lock daemon. **lockd** forwards lock requests for remote data to the server site's lock daemon through the RPC/XDR package (see *rpc*(3C)). **lockd** then requests the status monitor daemon, **statd** for monitor service (see *statd*(1M)). The reply to the lock request is not sent to the kernel until the status daemon and the server site's lock daemon have replied.

If either the status monitor or server site's lock daemon is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for all NFS client-site **lockd**s to submit reclaim requests. Client-site *lockd*s are notified by the **statd** of the server recovery, and promptly resubmit previously granted lock requests. If a **lockd** fails to secure a previously granted lock at the server site, the **lockd** sends a **SIGLOST** to the process holding that lock.

Options

lockd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, host name, process ID and name of the function generating the error, and the error message.

-t *timeout* **lockd** uses *timeout* (seconds) as the interval instead of the default value (10 seconds) to retransmit a lock request to the remote server. Note that changing this value also changes the value for grace period duration.

-g *graceperiod* **lockd** uses $[1+(graceperiod/timeout)]\times timeout$ (seconds) as the grace period duration instead of the default value ($5\times timeout$ seconds). If both **-t** and **-g** are specified, the **-t** should appear first since the grace period duration is dependent on the value of timeout.

AUTHOR

lockd was developed by Sun Microsystems, Inc., and HP.

SEE ALSO

fcntl(2), lockf(2), signal(2), statd(1M).

NAME

logins - display system and user login data

SYNOPSIS

logins [**-admopstux**] [**-g** *groups*] [**-l** *logins*]

DESCRIPTION

logins displays data concerning system and user logins. The format and content of the output is controlled by command options and may include: system or user login, user ID number, **/etc/passwd** comment field value (e.g., user name, etc...), primary group name, primary group ID, supplementary group names, supplementary group IDs, home directory, login shell, user security level, user audit events, and password aging parameters. The default data is: login, user ID, primary group name, primary group ID, and **/etc/passwd** comment field value. Output is sort by user ID, with user logins following system logins. The default output consists of login, user ID, primary group, primary group ID and comment field formatted into columns.

The following options are available to this command:

-a Displays two account expiration fields. The fields show how long the account can be unused (in days) before it becomes inactive and the date the account will expire.

-d Display logins with duplicate UIDs.

-m Show multiple group membership data.

-o Display with alternate format of one line of colon separated fields.

-p Display logins with no passwords

-s Display all system logins

-t Sort output by login rather than UID.

-u Display all user logins.

-x Display extended information about selected users. This extended information includes home directory, login shell and password aging data, each on its own line. Password information consists of password status (PS for valid password, LK for locked and NP for no password) and, if a password is present, date of last change, required number of days between changes, and number of days allowed between changes. In the case of non-trusted systems, the date of last change will be the latest Thursday since the change.

-g *groups*
Display all users belonging to *groups*, sorted by login. A comma separated list specifies multiple groups.

-l *logins*
Display the requested *logins*. A comma separated list specifies multiple logins.

Multiple options may be used. Any login matching any of the criteria will be displayed. A login will be displayed only once, even if it meets multiple criteria.

EXAMPLES

logins	List all logins in default format.
logins -p -d	List all logins that have no password or have a duplicate UID in default format.
logins -s -o	List all system logins in the alternate format.

FILES

/etc/passwd	HP-UX password file.
/etc/group	HP-UX group file.

SEE ALSO

listusers(1), passwd(1), group(4), passwd(4).

STANDARDS COMPLIANCE

logins: SVID3

NAME

lpadmin - configure the LP spooling system

SYNOPSIS

/usr/sbin/lpadmin -p*printer* [*options*]

/usr/sbin/lpadmin -x*dest*

/usr/sbin/lpadmin -d[*dest*]

DESCRIPTION

lpadmin configures LP spooling systems to describe printers, classes and devices. It is used to add and remove destinations, change membership in classes, change devices for printers, change printer interface programs, and to change the system default destination. **lpadmin** cannot be used when the LP scheduler, *lpsched*(1M), is running, except where noted below.

Exactly one of the **-p**, **-x** or **-d** options must be present for every legal invocation of *lpadmin*.

-p*printer* Names a *printer* to which all of the *options* below refer. If *printer* does not exist, it will be created.

-x*dest* Removes destination *dest* from the LP system. If *dest* is a printer and is the only member of a class, the class is deleted, too. No other *options* are allowed with **-x**.

-d[*dest*] Makes existing destination *dest* the new system default destination. If *dest* is not supplied, there is no system default destination. This option can be used when *lpsched*(1M) is running. No other *options* are allowed with **-d**.

The following *options* are only useful with **-p** and can appear in any order. For ease of discussion, the printer is referred to below as printer *P*.

-c*class* Inserts printer *P* into the specified *class*. *class* is created if it does not already exist.

-e*printer* Copies an existing *printer*'s interface program to be the new interface program for printer *P*.

-g*priority* Sets the default priority for printer *P* associated with *lp*(1). If omitted, the default priority is set to 0.

-h Indicates that the device associated with printer *P* is hardwired. This *option* is assumed when creating a new printer unless the **-l** option is specified.

-i*interface* Establishes a new interface program for printer *P*. *interface* is the pathname of the new program.

-l Indicates that the device associated with printer *P* is a login terminal. The LP scheduler (see *lpsched*(1M)) disables all login terminals automatically each time it is started. Before re-enabling printer *P*, its current *device* should be established using *lpadmin*.

-m*model* Selects a model interface program for printer *P*. *model* is one of the model interface names supplied with the LP software (see Models below).

-r*class* Removes printer *P* from the specified *class*. If printer *P* is the last member of the *class*, the *class* is removed.

-v*device* Associates a new *device* with printer *P*. *device* is the pathname of a file that is writable by the LP administrator *lp*. Note that there is nothing to stop an administrator from associating the same *device* with more than one *printer*. If only the **-p** and **-v** *options* are supplied, **lpadmin** can be used while the scheduler is running.

The following *options* are only useful with **-p** and can appear in any order. They are provided with systems that provide remote spooling.

-ob3 Uses three-digit request numbers associated with the printer directory. This is for contact with BSD systems. The default is to not use three-digit request numbers.

-oci*remcancel* Specifies that the local command *remcancel* is used to cancel requests to remote printers. To ensure that the correct command is used, specify the full path name.

-ocm*remcancel*	Specifies that the local model *remcancel* is used to cancel requests to remote printers.
-orm*machine*	The name of the remote machine is *machine*.
-orp*printer*	The name of the printer to use on the remote machine is *printer*.
-orc	Restricts users to canceling only their own requests. Default is to not restrict the cancel command.
-osi*remstatus*	Specifies that the command *remstatus* is used to obtain the status of requests to remote printers. To ensure that the correct command is used, specify the full path name.
-osm*remstatus*	Specifies that the model *remstatus* is used to obtain the status of requests to remote printers.

Restrictions

When creating a new printer, the **-v** option and one of the **-e**, **-i** , or **-m** options must be specified. Only one of the **-e**, **-i** or **-m** options can be specified. The **-h** and **-l** key letters are mutually exclusive. Printer and class names must not exceed 14 characters and must consist entirely of the characters **A**-**Z**, **a**-**z**, **0**-**9** and _ (underscore).

Models

Model interface programs are supplied with the LP software. They are shell procedures, C programs, or other executable programs that interface between *lpsched*(1M) and devices. All printer models reside in directory **/usr/lib/lp/model** and can be used without modification with **lpadmin -m**. All cancel models reside in directory **/usr/lib/lp/cmodel** and can be used without modification with **lpadmin -ocm**. All status models reside in directory **/usr/lib/lp/smodel** and can be used without modification with **lpadmin -osm**. Models should have 644 permission if owned by **lp** and **bin**, or 664 permission if owned by **bin** and **bin**. Model file names must not exceed 14 characters. Alternatively, LP administrators can modify copies of models then use **lpadmin -m** to associate them with printers.

The LP model interface program does the actual printing on the device that is currently associated with the printer. The LP spooler sets standard input to **/dev/null** and standard output and standard error output to the device specified in the **-v** option of *lpadmin*. The interface program is then invoked for printer *P* from the directory **/etc/lp** as follows:

interface/*P id user title copies options file* . . .

where arguments are as follows:

id	request id returned by *lp*(1).
user	login name of the user who made the request.
title	optional title specified with the **-t** option of *lp*(1).
copies	number of copies to be printed.
options	blank-separated list of class-dependent or printer-dependent options specified with the **-o** option of *lp*(1). Options from a BSD system have the character sequence **BSD** attached to the beginning of the option (for example, **BSDl**).
file	full pathname of the file to be printed.

Given the command line arguments and the output directed to the device, interface programs can format their output in any way they choose.

When printing is completed, it is the responsibility of the interface program to exit with a code indicative of the success of the print job. Only return values of **0** indicating that the job completed successfully, or values of positive **1** through **127** indicating that some error was encountered that does not affect future print jobs should be used. Negative values and positive values greater than 127 are reserved for system use and should not be used by interface programs. *lpsched*(1M) notifies users by mail when there is an error in printing the request. If problems are detected that are likely to affect future print jobs, the interface program should disable the printer so that other pending print requests are not lost.

The cancel and status model interface programs perform the actual communication with the remote system to cancel requests or get the status of requests. See *rcancel*(1M) and *rlpstat*(1M) for command line arguments.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG.

If any internationalization variable contains an invalid setting, **lpadmin** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

EXAMPLES

Assuming an existing Hewlett-Packard HP 2934A line printer named **lp1**, it will use the **hp2934a** model interface through **/dev/lp** after the command:

```
/usr/sbin/lpadmin -plp1 -mhp2934a -v/dev/lp
```

Assuming a printer **lp** on a remote system **system2**, the command:

```
/usr/sbin/lpadmin -plp3 -v/dev/null -mrmodel -ocmrcmodel -osmrsmodel
-ob3 -ormsystem2 -orplp -v/dev/null
```

causes the spool system to use the local line printer **lp3** and the model **rmodel**. The spool system also uses the model **rcmodel** to cancel remote requests and **rsmodel** to get status from **system2**. In addition, the three-digit sequence numbers, the remote system name **system2** and the remote printer **lp** are used.

WARNINGS

When installing remote printers, use the option **-ocmrcmodel** instead of **-oci/usr/sbin/rcancel** to specify the method used to cancel remote requests. The option **-osmrsmodel** should be used instead of **-osi/usr/sbin/rlpstat** to specify the method used for displaying remote status.

l

*class*es must not include *remote* printers. HP-UX systems do not have the ability to distribute print jobs in this way. Printing to a class of printers on a remote system (**systemB** for example) must be accomplished by creating the class on the remote system, then identifying that class by using a command resembling the following (though you might have to change some of the specific values shown in the example):

```
lpadmin -plocal_name -ormsystemB -orpsystemB_class_name -v /dev/null
-mrmodel -ocmrcmodel -osmrsmodel
```

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

enable(1), lp(1), lpstat(1), nroff(1), accept(1M), lpana(1M), lpsched(1M), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

NAME

lpana - print LP spooler performance analysis information

SYNOPSIS

lpana [**-d** *dest*]

DESCRIPTION

lpana prints LP spooler performance information, which system administrators can use to optimize the configuration of the entire spooler system.

Options

lpana recognizes one option:

-d *dest*	Choose *dest* as the printer or the class of printers. If *dest* is a printer, the performance analysis information is printed on that specific printer. If *dest* is a class of printers, the performance analysis information is printed on the printers that are members of the class. By default, **lpana** prints the performance analysis information for all printers and/or classes.

lpana examines **/var/adm/lp/lpana.log** for the following items:

Wait AV	Average waiting time from when job is spooled until start of printing.
Wait SD	Standard Deviation for waiting time.
Print AV	Average printing time from start to end of job.
Print SD	Standard Deviation for printing time.
Bytes AV	Average of number of bytes printed per request.
Bytes SD	Standard Deviation for number of bytes.
Sum KB	Sum of bytes printed for all requests (in kilobytes).
Num of Requests	Total number of requests since logging started.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

WARNINGS

lpana performs its operation on the local system only.

AUTHOR

lpana was developed by HP.

FILES

/var/adm/lp/lpana.log

SEE ALSO

lp(1), lpstat(1), lpadmin(1M), lpsched(1M).

NAME

lpsched, lpshut, lpmove, lpfence - start/stop the LP request scheduler, move requests, and define the minimum priority for printing

SYNOPSIS

/usr/sbin/lpsched [**-v**] [**-a**]
/usr/sbin/lpshut
/usr/sbin/lpmove *requests dest*
/usr/sbin/lpmove *dest1 dest2*
/usr/sbin/lpfence *printer fence*

DESCRIPTION

lpsched Schedules requests taken by *lp*(1) for printing on line printers. *lpsched*(1M) is typically invoked in **/sbin/rc**. This creates a process which runs in the background until **lpshut** is executed. The activity of the process is recorded in **/var/adm/lp/log**.

lpsched recognizes the following options:

- **-v** Write a verbose record of the **lpsched** process on **/var/adm/lp/log**.
- **-a** Write *lpana(1M)* logging data on **/var/adm/lp/lpana.log**.

lpshut Shuts down the line printer scheduler. All printers that are printing at the time **lpshut** is invoked stop printing. Requests that were printing at the time a printer was shut down are reprinted in their entirety after **lpsched** is started again. All LP commands perform their functions even when **lpsched** is not running.

lpmove Moves requests that were queued by *lp*(1) between LP destinations. This command can be used only when **lpsched** is not running.

The first form of the command moves the named *requests* to the LP destination, *dest*. *requests* are request ids as returned by *lp*(1). The second form moves all requests for destination *dest1* to destination *dest2*. As a side effect, *lp*(1) rejects requests for *dest1*.

Note that **lpmove** never checks the acceptance status (see *accept*(1M)) for the new destination when moving requests.

lpfence Defines the minimum required *priority* for the spooled file to be printed. *fence* must be in between 0 (lowest fence) and 7 (highest fence). Each *printer* has its own *fence*, which is initialized to 0 when it is configured by the *lpadmin*(1M) command. **lpfence** is used only when **lpsched** is not running.

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of date and time strings.

LANG determines the language in which messages are displayed.

If LC_TIME is not specified in the environment or is set to the empty string, the value of LANG is used as a default for each unspecified or empty variable. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, *lpsched*, *lpmove*, and **lpshut** behave as if all internationalization variables are set to "C". See *environ*(5).

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

WARNINGS

Moving requests associated with remote printers can cause unpredictable results.

lpsched, *lpshut*, *lpmove*, and **lpfence** perform their operation on the local system only.

SEE ALSO

accept(1M), cancel(1), enable(1), lp(1), lpadmin(1M), lpana(1M), lpstat(1), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

NAME

lsdev - list device drivers in the system

SYNOPSIS

/usr/sbin/lsdev [**-h**] [**-d** *driver* | **-C** *class*] [**-b** *block_major*] [**-c** *char_major*] [**-e** *major*] [*major* ...]

DESCRIPTION

The **lsdev** command lists, one pair per line, the major device numbers and driver names of device drivers configured into the system and available for invocation via special files. A **−1** in either the block or character column means that a major number does not exist for that type.

If no arguments are specified, **lsdev** lists all drivers configured into the system.

If the **-h** option is specified, **lsdev** will not print a heading. This option may be useful when the output of **lsdev** will be used by another program.

The **-d**, **-C**, **-b**, **-c**, and **-e** options are used to select specific device drivers for output. If more than one option is specified, all drivers that match the criteria specified by those options will be listed. These search options are divided into two types: name search keys (the **-d** and **-C** options) and major number search keys (the **-b**, **-c**, and **-e** options). If both types of options are present, only entries that match both types are printed. The same type of option may appear more than once on the command line with each occurrence providing an ORing effect of that search type. The **-d** and **-C** options may not be specified at the same time.

The ability to process *major* arguments is provided for compatibility and functions like the **-e** option.

Options

-C *class*	List device drivers that match *class*.
-d *driver*	List device drivers with the name *driver*.
-b *block_major*	List device drivers with a block major number of *block_major*.
-c *char_major*	List device drivers with a character major number of *char_major*.
-e *major*	List device drivers with either a character major number or block major equal to *major*.

DIAGNOSTICS

Invalid combination of options

The **-d** and **-C** options may not be specified at the same time.

Invalid major number

A major number is malformed or out of range.

EXAMPLES

To output entries for all drivers in the **pseudo** class:

```
lsdev -C pseudo
```

To output entries that are in the class **disk** that have either a block or character major number of **0**:

```
lsdev -C disk -e 0
```

To get the character major number of **my_driver** into a shell environment variable:

```
C_MAJOR=$(lsdev -h -d my_driver | awk '{print $1}')
```

WARNINGS

Some device drivers available from the system may be intended for use by other drivers. Attempting to use them directly from a special file may produce unexpected results.

A driver may be listed even when the hardware requiring the driver is not present. Attempts to access a driver without the corresponding hardware will fail.

lsdev only lists drivers that are configured into the currently executing kernel. For a complete list of available drivers, please run **sam** (see *sam*(1M).

DEPENDENCIES

Since **lsdev** relies on the device driver information provided in a *driver*_**install** routine, **lsdev** may not list drivers installed by other means.

AUTHOR

lsdev was developed by HP.

SEE ALSO

sam(1M).

Section 7 entries related to specific device drivers.

Managing Systems and Workgroups manual.

NAME

lssf - list a special file

SYNOPSIS

/sbin/lssf *special_file* ...

DESCRIPTION

lssf lists information about a special file. For each *special_file* name, **lssf** determines the major number of the special file and whether it is block or character (using *stat*(2)). It then scans the system for the device that is associated with the special file. When the device is found, the minor number of the special file is decoded. A mnemonic description of the minor number is printed on standard output along with the hardware path (i.e., address) of the device. Mnemonics used to describe the fields are closely related to the options used with **mksf** (see *mksf*(1M)).

DIAGNOSTICS

Most diagnostic messages from **lssf** are self explanatory. Listed below are some messages deserving further clarification. Warnings allow **lssf** to continue.

Warnings

No such device in the system

There is no information about the device in the kernel. The special file is not usable. Use **rmsf** to remove the special file (see *rmsf*(1M)).

Character major <*major***> is not in the kernel**
Block major <*major***> is not in the kernel**

The major number associated with the special file is not in the kernel. Use **config** to add the appropriate driver to the kernel (see *config*(1M)).

Device driver <*name***> is not in the kernel**
Device class <*name***> is not in the kernel**

The indicated device driver or device class is not present in the kernel. An **open()** of a special file pointing to an unusable device fails. To make the device usable, the appropriate device driver and/or device class must be added to the **config** input file and a new kernel generated (see *config*(1M)). If the device is no longer needed, **rmsf** should be used to remove the special files and update **/etc/ioconfig**.

<*special_file***> is not a special file**

The file is not associated with an I/O device.

EXAMPLES

Suppose a special file is created with the command **mksf -d tape1 -H 8.6.1 -b 1600 -a rmt/c2t6d0m**. The command **lssf rmt/c2t6d0m** then produces:

```
tape1 instance 2 bpi 1600 att address 8.6.1 rmt/c2t6d0m
```

AUTHOR

lssf was developed by HP.

FILES

/dev/config	I/O system special file
/etc/ioconfig	I/O system configuration database

SEE ALSO

config(1M), insf(1M), mksf(1M), rmsf(1M).

NAME

lvchange - change LVM logical volume characteristics

SYNOPSIS

/usr/sbin/lvchange [**-a** *availability*] [**-A** *autobackup*] [**-c** *mirror_consistency*] [**-C** *contiguous*] [**-d** *schedule*] [**-M** *mirror_write_cache*] [**-p** *permission*] [**-r** *relocate*] [**-s** *strict*] [**-t** *IO_timeout*] *lv_path*

Remarks

Mirrored disk operations require the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

lvchange cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvchange** command changes certain characteristics of a logical volume. Other characteristics can be changed with the **lvextend** and **lvreduce** commands (see *lvextend*(1M) and *lvreduce*(1M)).

The command-line options specify the type and extent of change. Each current characteristic for a logical volume remains in effect until explicitly changed by the corresponding option. All options take effect immediately, except **-s**, which takes effect only when new extents are allocated by the **lvextend** command.

If a logical volume is striped, its scheduling policy is always parallel and its allocation policy is always strict and noncontiguous; these attributes cannot be changed with **lvchange**.

The **lvchange** command can also be used to change the timeout value for a logical volume. This can be useful to control how long an IO request will be retried (for a transient error, like a device timeout), before giving up and declaring a pending IO to be failed. The default behavior is for the system to continue to retry an IO for a transient error until the IO can complete. Thus, the IO will not be returned to the caller until the IO can complete. By setting a non-zero IO timeout value, this will set the maximum length of time that the system will retry an IO. If the IO cannot complete before the length of time specified by the IO timeout, then the IO will be returned to the caller with an error. The actual duration of the IO request may exceed the logical volume's maximum IO timeout value when the underlying physical volume(s) have timeouts which either exceed the logical volume's timeout value or are not an integer multiple of the logical volume's timeout value (see *pvchange*(1M) for details on how to change the IO timeout value on a physical volume).

Options and Arguments

The **-c**, **-d**, **-M**, and **-s** options are meaningful only if the optional HP MirrorDisk/UX software has been installed on the system.

lvchange recognizes the following options and arguments:

lv_path The block device path name of a logical volume.

-a *availability* Set logical volume availability. *availability* can have one of the following values:

- **y** Make a logical volume available. An open of the logical volume will succeed.
- **n** Make a logical volume temporarily unavailable. An open of the logical volume will fail. However, all current processes that have the logical volume open remain open.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.
- **n** Do not back up configuration changes this time.

-c *mirror_consistency* Set mirror consistency recovery. This option is effective only when **-M n** is specified or previously set. *mirror_consistency* can have one of the following values:

- **y** Set mirror consistency recovery on. LVM achieves mirror consistency during volume group activation by going through all logical extents and copying data from a nonstale copy to the other mirror copies.
- **n** Set mirror consistency recovery off. LVM does not perform mirror consistency recovery on this logical volume when the volume group is activated.

-C *contiguous* Set the contiguous allocation policy. *contiguous* can have one of the following values:

- **y** Set a contiguous allocation policy. Physical extents are allocated in ascending order without any gap between adjacent extents and all extents are contained in a single physical volume.
- **n** Do not set a contiguous allocation policy.

A nonempty logical volume that has a noncontiguous allocation policy cannot be changed to a contiguous allocation policy unless it happens to meet all the requirements of the contiguous allocation policy. See *lvcreate*(1M) for more information about the contiguous allocation policy.

-d *schedule* Set the scheduling policy when a logical extent with more than one mirror is written. (The scheduling policy of a striped logical volume is striped and cannot be changed.) *schedule* can have one of the following values:

- **p** Establish a parallel scheduling policy.
- **s** Establish a sequential scheduling policy. Use this value with care, because it leads to performance loss in most cases.

-M *mirror_write_cache* Set the Mirror Write Cache flag. This option is allowed only when the logical volume is not opened. *mirror_write_cache* can have one of the following values:

- **y** Set Mirror Write Cache on. Every write to a mirror copy is recorded in the Mirror Write Cache and written into the Mirror Consistency Record on the disk if a cache-miss occurs. This allows LVM to determine whether all mirror copies are identical, even across system crashes. When the volume group is activated, the Mirror Consistency Record is used to perform mirror consistency recovery.
- **n** Set Mirror Write Cache off. Mirror write does not incur an additional write to the Mirror Consistency Record on the disk.

-p *permission* Set the access permission. *permission* can have one of the following values:

- **w** Set the access permission to read-write.
- **r** Set the access permission to read-only.

-r *relocate* Set the bad block relocation policy. *relocate* can have one of the following values:

- **y** Allow bad block relocation. Upon a media failure (detection of a bad block of data on disk), LVM will mark the failed block in the Bad Block Directory, and attempt to relocate the block to a new location on disk. If relocation is successful then no error will be returned, and future I/O requests which contain the bad block will be directed to the new location. If relocation is unsuccessful, an I/O error will be returned, and subsequent I/O requests containing the bad block will again attempt relocation.
- **n** Prevent bad block relocation. Upon a media failure, LVM will mark the failed block as bad in the Bad Block Directory, but will NOT attempt to relocate the bad block to a new location on disk. Future I/O requests which contain the bad block will return with an I/O error.

No attempt will be made to access the bad block.

N Disable bad block relocation and the Bad Block Directory. Upon a media failure, LVM will NOT attempt to relocate the bad block. In addition it will NOT enter the block in the Bad Block Directory. LVM will have no record of the block being bad, and will attempt to access it on future I/O requests.

-s *strict* Set the strict allocation policy. Mirror copies of a logical extent can be allocated to share or not share the same physical volume or physical volume group. This option only makes sense when the physical volumes of the volume group that owns the specified logical volume reside on different physical disks. *strict* can have one of the following values:

y Set a strict allocation policy. Mirrors of a logical extent cannot share the same physical volume.

g Set a PVG-strict allocation policy. Mirrors of a logical extent cannot share the same physical volume group.

n Do not set a strict or a PVG-strict allocation policy. Mirrors of a logical extent can share the same physical volume.

When a logical volume is mirrored, the following changes are not allowed:

- From nonstrict to strict
- From nonstrict to PVG-strict
- From strict to PVG-strict

-t *IO_timeout* Set the *IO_timeout* for the logical volume to the number of seconds indicated. This value will be used to determine how long to wait for IO requests to complete before concluding that an IO request cannot be completed. An *IO_timeout* value of zero (0) causes the system to use the default value of "forever". NOTE: The actual duration of the request may exceed the specified *IO_timeout* value when the underlying physical volume(s) have timeouts which either exceed this *IO_timeout* value or are not integer multiples of this value.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Change the permission of a logical volume to read-only:

```
lvchange -p r /dev/vg01/lvol3
```

Change the allocation policy of a logical volume to nonstrict:

```
lvchange -s n /dev/vg01/lvol7
```

Turn the mirror write cache off on a logical volume:

```
lvchange -M n /dev/vg01/lvol1
```

Change the IO timeout value of a logical volume to 1 minute (60 seconds):

```
lvchange -t 60 /dev/vg01/lvol1
```

WARNINGS

For root, swap or dump logical volumes, the allocation policy is always contiguous. This attribute cannot be changed with **lvchange**.

DEPENDENCIES

The **-r** option is not available on HP-IB devices.

SEE ALSO

lvcreate(1M), lvdisplay(1M), lvextend(1M).

NAME

lvcreate - create logical volume in LVM volume group

SYNOPSIS

/usr/sbin/lvcreate [**-A** *autobackup*] [**-c** *mirror_consistency*] [**-C** *contiguous*] [**-d** *schedule*]
[**-i** *stripes* **-I** *stripe_size*] [**-l** *le_number* | **-L** *lv_size*] [**-m** *mirror_copies*]
[**-M** *mirror_write_cache*] [**-n** *lv_name*] [**-p** *permission*] [**-r** *relocate*] [**-s** *strict*] *vg_name*

Remarks

Mirrored disk operations require the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

lvcreate cannot be performed if the volume group is activated in shared mode.

Logical volumes that were created using the striped option are not supported in shared mode.

DESCRIPTION

The **lvcreate** command creates a new logical volume within the volume group specified by *vg_name*. Up to 255 logical volumes can be created in one volume group.

If you specify the **-n** *lv_name* option, a new logical volume is created with that name. Otherwise, a system-generated name of the form **lvol***N* is created, where *N* is the decimal equivalent of the two least significant bytes of the minor number of the new logical volume, in the range **1** to **255** (see *lvm*(7)). Two device files are created in *vg_name*: a block device file named *lv_name* or **lvol***N*, and a character (raw) device file named **r***lv_name* or **rlvol***N*.

If you omit the **-l** and **-L** options, the logical volume is created with zero length. This permits you to choose its physical volume location when you allocate logical extents with the **lvextend** command (see *lvextend*(1M)). If you specify **-l** or **-L**, the location is determined automatically.

The default settings provide the most commonly used characteristics. Use the options to tailor the logical volume to the requirements of the system. Once a logical volume is created, some of its characteristics can be changed with the **lvchange**, **lvextend**, and **lvreduce** commands (see *lvchange*(1M), *lvextend*(1M), and *lvreduce*(1M)).

Options and Arguments

The **-c**, **-d**, **-m**, **-M**, and **-s** options are only meaningful if the optional HP MirrorDisk/UX software has been installed on the system.

lvcreate recognizes the following options and arguments:

vg_name — The path name of a volume group.

-A *autobackup* — Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** — Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** — Do not back up configuration changes this time.

-c *mirror_consistency* — Set mirror consistency recovery. This option is effective only when **-M n** is specified. It is ignored for **-M y**. *mirror_consistency* can have one of the following values:

- **y** — Set mirror consistency recovery on. This is the default.

  LVM achieves mirror consistency during volume group activation by going through all logical extents and copying data from a nonstale copy to the other mirror copies.

- **n** — Set mirror consistency recovery off. LVM does not perform mirror consistency recovery on this logical volume when the volume group is activated.

-C *contiguous*
: Set the contiguous allocation policy. A contiguous logical volume has three characteristics:

- Physical extents are allocated in ascending order,
- No gap is allowed between physical extents within a mirror copy,
- Physical extents of any mirror copy all reside on a single physical volume.

Use the strict (**-s**) and contiguous (**-C**) options together to form various combined allocation policies on a logical volume. For example, **-s y -C y** defines a logical volume such that each mirror copy is contiguous, yet mirror copies of a logical extent cannot share the same physical volume.

contiguous can have one of the following values:

y Set a contiguous allocation policy.

n Do not set a contiguous allocation policy. This is the default.

-d *schedule*
: Set the scheduling policy when a logical extent with more than one mirror is written. (The scheduling policy of a striped logical volume is striped and cannot be changed.) *schedule* can have one of the following values:

p Establish a parallel scheduling policy. This is the default.

s Establish a sequential scheduling policy. Use this value with care, because it leads to performance loss in most cases.

-i *stripes*
: Set the number of disks to stripe across. *stripes* must be in the range **2** to the number of disks in the current volume group. **-i** and **-I** must be specified together.

-I *stripe_size*
: Set the size in kilobytes of the stripe. *stripe_size* can have the value **4**, **8**, **16**, **32**, or **64**. **-i** and **-I** must be specified together.

-l *le_number*
: Allocate space to the logical volume, specified in logical extents. *le_number* is a decimal value in the range **1** to **65535** (the implementation limit). The default is described above.

Either **-l** or **-L** can be specified, but not both.

-L *lv_size*
: Allocate space to the logical volume, specified in megabytes. *lv_size* is a decimal value in the range **1** to **4096** (4 gigabytes). *lv_size* is rounded up to the nearest multiple of the logical extent size, equivalent to the physical extent size defined for the volume group by the **vgcreate** command (see *vgcreate*(1M)). The default is described above.

Either the **-l** or the **-L** option can be specified, but not both.

-m *mirror_copies*
: Set the number of mirror copies allocated for each logical extent. A mirror copy contains the same data as the original. *mirror_copies* can have the value **1** or **2**. The default value is **0** (no mirror copies).

-M *mirror_write_cache*
: Set the Mirror Write Cache flag. *mirror_write_cache* can have one of the following values:

y Set Mirror Write Cache on. This is the default.

Every write to a mirror copy is recorded in the Mirror Write Cache. The Mirror Consistency Record in the Volume Group Reserved Area on the disk is updated whenever there is a write to a logical track group that is not already recorded in the cache. This allows LVM to determine whether all the mirror copies are identical, even across system crashes. When the volume group is activated, the Mirror Consistency Record is used to perform mirror consistency recovery.

n Set Mirror Write Cache to off. Mirror write does not incur an additional write to the Mirror Consistency Record.

-n *lv_name*
: Set the name of the new logical volume to *lv_name*, where *lv_name* is a simple file name, not a path name. The default is described above.

-p *permission* — Set the access permission. *permission* can have one of the following values:

- **w** Set the access permission to read-write. This is the default.
- **r** Set the access permission to read-only.

-r *relocate* — Set the bad block relocation policy. *relocate* can have one of the following values:

- **y** Allow bad block relocation. Upon a media failure (detection of a bad block of data on disk), LVM will mark the failed block in the Bad Block Directory, and attempt to relocate the block to a new location on disk. If relocation is successful then no error will be returned, and future I/O requests which contain the bad block will be directed to the new location. If relocation is unsuccessful, an I/O error will be returned, and subsequent I/O requests containing the bad block will again attempt relocation. This is the default.
- **n** Prevent bad block relocation. Upon a media failure, LVM will mark the failed block as bad in the Bad Block Directory, but will NOT attempt to relocate the bad block to a new location on disk. Future I/O requests which contain the bad block will return with an I/O error. No attempt will be made to access the bad block.
- **N** Disable bad block relocation and the Bad Block Directory. Upon a media failure, LVM will NOT attempt to relocate the bad block. In addition it will NOT enter the block in the Bad Block Directory. LVM will have no record of the block being bad, and will attempt to access it on future I/O requests.

-s *strict* — Set the strict allocation policy. Mirror copies of a logical extent can be allocated to share or not share the same physical volume or physical volume group. *strict* can have one of the following values:

- **y** Set a strict allocation policy. Mirrors of a logical extent cannot share the same physical volume. This is the default.
- **g** Set a PVG-strict allocation policy. Mirrors of a logical extent cannot share the same physical volume group. A PVG-strict allocation policy cannot be set on a logical volume in a volume group that does not have a physical volume group defined.
- **n** Do not set a strict or PVG-strict allocation policy. Mirrors of a logical extent can share the same physical volume.

Striped logical volumes are only allocated using the *strict* or *PVG-strict* allocation policies. The number of extents for a striped logical volume is always a multiple of the number of disks the logical volume is striped across. A logical volume striped across **n** disks, is allocated in sets of **n** extents, and each extent of a given set is allocated on a different physical volumes in the volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Create a logical volume in volume group **/dev/vg02**:

```
lvcreate /dev/vg02
```

Create a logical volume in volume group **/dev/vg03** with nonstrict allocation policy:

```
lvcreate -s n /dev/vg03
```

Create a logical volume of size 100 MB in volume group **/dev/vg03**:

```
lvcreate -L 100 /dev/vg03
```

Create a logical volume of size 90 MB striped across 3 disks with a stripe size of 64 KB:

```
lvcreate -L 90 -i 3 -I 64 /dev/vg03
```

WARNINGS

The **-m** and **-r** options cannot be used with HP-IB devices.

The root, swap, and dump logical volumes (see *lvlnboot*(1M)) must be created with contiguous allocation policy.

SEE ALSO

lvchange(1M), lvextend(1M), lvreduce(1M), pvchange(1M).

NAME

lvdisplay - display information about LVM logical volumes

SYNOPSIS

/usr/sbin/lvdisplay [**-k**] [**-v**] *lv_path* ...

Remarks

Mirrored disk information requires the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

DESCRIPTION

The **lvdisplay** command displays the characteristics and status of each logical volume specified by *lv_path*.

Options and Arguments

lvdisplay recognizes the following options and arguments:

lv_path	The block device path name of a logical volume, for example, **/dev/vg00/lvol1**.
-v	For each logical volume, display the physical volume distribution, and the mapping of the logical extents onto the physical extents of the physical volumes.
-k	This option displays the same information as the **-v** option, except in the column where **PV Name** is displayed, the **pvkey** (Physical Volume Number in VG) will be displayed instead. Use this option with the **-v** option.

Display Without –v Option

If you omit the **-v** option, **lvdisplay** displays the following information for each logical volume:

--- Logical volumes ---

LV Name The block device path name of the logical volume.

VG Name The path name of the volume group.

LV Permission Access permission: **read-only** or **read/write**.

LV Status State of the logical volume:

available/stale	Available but contains physical extents that are not current.
available/syncd	Available and synchronized.
available	Available but the stale or synchronized state cannot be confidently determined because both Mirror Write Cache and Mirror Consistency Recovery are turned off.
unavailable	Not available for use.

Mirror copies Number of physical extents beyond the original allocated for each logical extent; i.e., the number of mirrors: 0, 1, or 2.

Consistency Recovery
Mode of mirror consistency recovery which determines how LVM performs mirror consistency recovery during volume group activation:

MWC	Recover mirror consistency by using the Mirror Write Cache and Mirror Consistency Record. Implies that Mirror Write Cache is on.
NOMWC	Recover mirror consistency by going through all logical extents and copying data from a non-stale copy to the other mirror copies. Implies that Mirror Write Cache is off.
NONE	No mirror consistency recovery during volume group activation on this logical volume. Implies that Mirror Write Cache is off.

Schedule Striped, sequential or parallel scheduling policy. Striped policy is by default parallel scheduling for mirrored I/O.

`LV Size (Mbytes)`	Size of the logical volume in megabytes (MB).
`Current LE`	Number of logical extents currently in the logical volume.
`Allocated PE`	Number of physical extents allocated to the logical volume.
`Stripes`	The number of stripes. If this field is 0, then the logical volume is not striped.
`Stripe Size (Kbytes)`	The size of each stripe in kilobytes (KB).
`Bad block`	Bad block relocation policy.
`Allocation`	Current allocation state, displayed as one of:

```
non-strict      non-strict/contiguous
strict          strict/contiguous
PVG-strict      PVG-strict/contiguous
```

`contiguous`	Physical extents are allocated in an ascending order without any gap between adjacent extents. All physical extents of a given mirror are contained in a single physical volume.
`non-strict`	Physical extents that belong to the same logical extent can be allocated on the same physical volume or physical volume group.
`PVG-strict`	Mirror copies for a logical extent are not allocated on the same physical volume group.
`strict`	Mirror copies for a logical extent are not allocated on the same physical volume.

`IO Timeout (Seconds)`
The IO timeout used by LVM for all IO to this logical volume. A value of default, indicates that the system will use the value of "forever". (Note: the actual duration of a request may exceed this timeout value when the underlying physical volume(s) have timeouts which either exceed this value or are not integer multiples thereof.)

1

Display With −v Option

If you specify the **`-v`** option, **`lvdisplay`** also lists the distribution of each logical volume across the physical volumes of the volume group and the mapping of each logical extent of the logical volume on the physical extents of the physical volume.

`--- Distribution of logical volume ---`

The distribution of logical volume *lv_path* across the physical volumes of the volume group, displayed in the following columns:

`PV Name`	The block device path name of the physical volume where the logical extents are allocated.
`PVNUM`	The Physical Volume Number in VG (if **`-k`** option is specified).
`LE on PV`	Number of logical extents allocated on the physical volume.
`PE on PV`	Number of physical extents allocated on the physical volume.

`--- Logical extents ---`

The mapping of logical extents onto physical extents, displayed in the following columns:

`LE`	Logical extent number.
`PV1`	The block device path name of the physical volume that corresponds to the location of the first physical extent of the logical extent.
`PE1`	First physical extent number allocated to the logical extent.
`Status 1`	Status of the first physical extent: **`stale`** or **`current`**.

The following columns are displayed for one or two mirror copies:

PV2	The block device path name of the physical volume that corresponds to the location of the second physical extent (first copy) of the logical extent.
PE2	Second physical extent number allocated to the logical extent.
Status 2	Status of the second physical extent: **stale** or **current**.

The following columns are displayed for two mirror copies:

PV3	The block device path name of the physical volume that corresponds to the location of the third physical extent (second copy) of the logical extent.
PE3	Third physical extent number allocated to the logical extent.
Status 3	Status of the third physical extent: **stale** or **current**.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Display information about a logical volume:

```
lvdisplay /dev/vg01/lvol3
```

Display all the available information about a logical volume, including the characteristics, status and distribution map:

```
lvdisplay -v /dev/vg01/lvol3
```

Display all the available information about a logical volume, but display **pvkey** instead of **PV Name** in the status and distribution map.

```
lvdisplay -v -k /dev/vg01/lvol3
```

SEE ALSO

lvchange(1M), lvcreate(1M), lvextend(1M), lvreduce(1M), pvdisplay(1M), vgdisplay(1M).

NAME

lvextend - increase space, increase mirrors for LVM logical volume

SYNOPSIS

/usr/sbin/lvextend [**-A** *autobackup*] {**-l** *le_number* | **-L** *lv_size* | **-m** *mirror_copies*} *lv_path* [*pv_path* ... | *pvg_name* ...]

Remarks

Mirrored disk operations require the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

lvextend cannot be performed if the volume group is activated in shared mode.

Existing logical volumes that were created using the striped option are not supported in shared mode.

DESCRIPTION

The **lvextend** command can increase a logical volume's allocated extents, or increase its number of mirrored copies.

Other logical volume characteristics can be modified with the **lvchange** and **lvreduce** commands (see *lvchange*(1M) and *lvreduce*(1M)).

To limit the allocation to specific physical volumes, specify the physical volume names as *pv_path* arguments or specify the physical volume group names as *pvg_name* arguments. Otherwise, all of the physical volumes in a volume group are available for allocating new physical extents. LVM always ensures that physical extent allocation can satisfy the current allocation policy or policies. If a physical volume is not suitable for use with a certain allocation policy, it is not used during physical extent allocation, even it is specified in a *pv_path* argument or indirectly in a *pvg_name* argument.

LVM striped logical volumes are always allocated using a strict allocation policy. Consequently, striped logical volumes may only be extended by a number extents that is a multiple of disks the logical volume is striped across. For example, for a logical volume striped across 3 disks, the logical volume will be extended in increments of 3 extents, with each of the 3 extents allocated on a different disk in the volume group.

1

The *pvg_name* argument is allowed only if one of the allocation policies of the logical volume is PVG-strict.

Options and Arguments

The **-m** option is only meaningful if the optional HP MirrorDisk/UX software has been installed on the system.

lvextend recognizes the following options and arguments:

lv_path The block device path name of a logical volume.

pv_path The block device path name of a physical volume.

pvg_name The name of a physical volume group (see *lvmpvg*(4)).

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes this time.

-l *le_number* Increase the space allocated to the logical volume, specified in logical extents. *le_number* is a decimal value greater than the current number of logical extents, in the range **1** to **65535** (the implementation limit).

One, and only one, **-l**, **-L**, or **-m** option must be supplied.

-L *lv_size* Increase the space allocated to the logical volume, specified in megabytes. *lv_size* is a decimal value greater than the current logical volume size, in the range **1** to **4096** (4 gigabytes). *lv_size* is rounded up to the nearest multiple of the logical extent size, equivalent to the physical extent size defined for the volume group by the **vgcreate** command (see *vgcreate*(1M)).

One, and only one, **-l**, **-L**, or **-m** option must be specified.

-m *mirror_copies* Set the number of mirror copies allocated for each logical extent. A mirror copy contains the same data as the original. *mirror_copies* can have the value **1** or **2**. It must be greater than the current value.

Data in the new copies is synchronized. The synchronization process can be time consuming, depending on hardware characteristics and the amount of data.

One, and only one, **-l**, **-L**, or **-m** option must be specified.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Increase the number of the logical extents of a logical volume to 100:

```
lvextend -l 100 /dev/vg01/lvol3
```

Increase the logical volume size to 400 MB:

```
lvextend -L 400 /dev/vg01/lvol4
```

Allocate two mirrors (that is, two copies of the original) for each logical extent of a logical volume:

```
lvextend -m 2 /dev/vg01/lvol5
```

Mirror a logical volume onto a particular physical volume.

```
lvextend -m 1 /dev/vg00/lvol3 /dev/dsk/c0t3d0
```

Increase the size of a file system existing on a logical volume.

First, increase the size of the logical volume.

```
lvextend -L 400 /dev/vg06/lvol3
```

Unmount the file system.

```
umount /dev/vg06/lvol3
```

Extend the file system to occupy the entire (larger) logical volume.

```
extendfs /dev/vg06/rlvol3
```

Remount the file system.

```
mount /dev/vg06/lvol3 /mnt
```

WARNINGS

The **-m** option cannot be used on HP-IB devices.

SEE ALSO

lvchange(1M), lvcreate(1M), lvdisplay(1M), lvreduce(1M), pvchange(1M), pvdisplay(1M).

NAME

lvlnboot - prepare LVM logical volume to be root, boot, primary swap, or dump volume

SYNOPSIS

/usr/sbin/lvlnboot [[**-A** *autobackup*] { **-b** *boot_lv* | **-d** *dump_lv* | **-r** *root_lv* | **-R** | **-s** *swap_lv* }] [**-v**] [*vg_name*]

/usr/sbin/lvlnboot [**-c**]

Remarks

lvlnboot cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvlnboot** command updates all physical volumes in the volume group so that the logical volume becomes the root, boot, primary swap, or a dump volume when the system is next booted on the volume group. If a nonexistent logical volume is specified, this command fails. If a different logical volume is already linked to the root or primary swap, the command fails.

This command should be run in recovery mode (**-R**) whenever the configuration of the root volume group is affected by one of the following commands: **lvextend**, **lvmerge**, **lvreduce**, **lvsplit**, **pvmove**, **lvremove**, **vgextend**, or **vgreduce** (see *lvextend*(1M), *lvmerge*(1M), *lvreduce*(1M), *lvsplit*(1M), *pvmove*(1M), *lvremove*(1M), *vgextend*(1M), and *vgreduce*(1M)). Starting with HP-UX Release 10.0, this is done automatically.

Options and Arguments

lvlnboot recognizes the following options and arguments:

vg_name The path name of a volume group.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes this time.

-b *boot_lv* Define *boot_lv* to be the boot volume the next time the system is booted on the volume group. *boot_lv* must be the first logical volume on the physical volume. *boot_lv* must be contiguous, and must not allow bad block relocation.

boot_lv is used to locate the boot file system during the boot process. The boot file system has the kernel which is read by the boot loader *hpux*(1M).

-d *dump_lv* Define *dump_lv* to be one of the dump volumes the next time the system is booted on the volume group. *dump_lv* must be a contiguous logical volume and cannot have Bad Block Relocation enabled.

The command updates the Boot Data Reserved Area of each bootable physical volume in the volume group (see *pvcreate*(1M)).

The combined size of all the dump volumes should be at least 2048 bytes larger than the total memory of the system. The additional 2 KB is used to safeguard against a dump to the bottom of the disk.

Multiple dump devices can be configured, but each *dump_lv* must be entered with a separate **lvlnboot** command line.

-r *root_lv* Define *root_lv* to be the root volume the next time the system is booted on this volume group. *root_lv* must be a contiguous logical volume and cannot have bad block relocation enabled.

If *root_lv* is the first logical volume on the physical volume, then it is configured as the combined root-boot volume. Otherwise, *root_lv* is configured as the separate root volume in which case a separate boot volume needs to be configured using the **lvlnboot -b** option.

Either the separate root or the separate boot volume can be configured first.

The command updates the Boot Data Reserved Area of each bootable physical volume (see *pvcreate*(1M)) to enable the volume group to be used to locate the root file system. *root_lv* is also used as the root volume during a maintenance-mode boot (see *hpux*(1M)).

The physical volumes containing *root_lv* must have been created using the **pvcreate -B** option (see *pvcreate*(1M)), indicating that that physical volume is to be used as a bootable physical volume. Also, the **mkboot** command (see *mkboot*(1M)) must have been run on the physical volume to create the LIF area at the top of the physical volume (see *lif*(4)).

-R Recover any missing links to all of the logical volumes specified in the Boot Data Reserved Area and update the Boot Data Reserved Area of each bootable physical volume in the volume group (see *pvcreate*(1M)).

-s *swap_lv* Define *swap_lv* to be the primary swap volume the next time the system is booted on the volume group. *swap_lv* must be a contiguous logical volume, and a root logical volume must have been previously defined with this command.

The command updates the Boot Data Reserved Area of each bootable physical volume in the volume group (see *pvcreate*(1M)). Any existing swap area previously defined must be removed via *lvrmboot*(1M).

-c During normal boots (vs. maintenance-mode boots, see *hpux*(1M)), this command is automatically executed by **/sbin/ioinitrc** (see *inittab*(4)).

Since this command is performed during boot, it does not need to be performed manually unless **/stand/rootconf** is missing in a separate root/boot configuration (or alternatively, performing a normal reboot will recreate this file).

This command updates the **/stand/rootconf** file with the location of the root volume in the currently booted volume group.

The **/stand/rootconf** file is used during maintenance-mode boots to locate the root volume for volume groups with separate boot and root volumes.

During maintenance-mode boots, since the root volume group is not activated, **lvlnboot -c** does not update **/stand/rootconf**. For separate root/boot configurations, maintenance-mode boot will fail if **/stand/rootconf** does not already exist with the correct location of the root volume. See *WARNINGS*.

When a new volume group with separate boot and root volumes is created, the first boot must be a normal boot (versus. a maintenance-mode boot), so that **/stand/rootconf** gets created.

This option does not allow updating **/stand/rootconf** for any volume group other than the one that is booted.

-v Print verbose messages. With no other arguments present, print information on root, boot, swap, and dump logical volumes. If a combined root-boot volume is configured, no information for the boot volume is displayed.

l

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

The following examples show configuration of a combined root-boot volume.

Create a root volume group, **vglvmroot**, containing root, swap, and dump logical volumes. Assume that an appropriate directory called **/dev/vglvmroot** and a corresponding **group** file already exist (see *lvm*(7)).

First, initialize the disk, say **/dev/dsk/c0t0d0**, so that it can be used as an LVM boot disk.

```
pvcreate -B /dev/rdsk/c0t0d0
```

Place the LIF information on the disk using the **mkboot** command.

```
mkboot /dev/rdsk/c0t0d0
```

Create the volume group **vglvmroot**.

```
vgcreate /dev/vglvmroot /dev/dsk/c0t0d0
```

Create a logical volume that is suitable for use as the root volume. This logical volume has to be the first in the volume group and should be a contiguous volume with bad block relocation turned off.

```
lvcreate -n root -L 120 -C y -r n /dev/vglvmroot
```

Create a logical volume that will be used as primary swap. This volume should be contiguous.

```
lvcreate -n swap -L 64 -C y /dev/vglvmroot
```

Create a logical volume that will be used as the dump volume. This volume should be contiguous.

```
lvcreate -n dump -L 64 -C y /dev/vglvmroot
```

Specify that the logical volume, **root**, will be used as the root volume.

```
lvlnboot -r /dev/vglvmroot/root
```

Specify that the logical volume, **swap**, will be used as the primary swap.

```
lvlnboot -s /dev/vglvmroot/swap
```

Specify that the logical volume, **dump**, will be used as the dump volume.

```
lvlnboot -d /dev/vglvmroot/dump
```

Display the results of the previous operations.

```
lvlnboot -v /dev/vglvmroot
```

The following examples show configuration of separate root and boot volumes.

Create a root volume group, **vglvmroot**, containing root, boot, swap, and dump logical volumes. Assume that an appropriate directory called **/dev/vglvmroot** and a corresponding **group** file already exist (see *lvm*(7)).

First, initialize the disk, say **/dev/dsk/c0t0d0**, so that it can be used as an LVM boot disk.

```
pvcreate -B /dev/rdsk/c0t0d0
```

Place the LIF information on the disk using the **mkboot** command.

```
mkboot /dev/rdsk/c0t0d0
```

Create the volume group **vglvmroot**.

```
vgcreate /dev/vglvmroot /dev/dsk/c0t0d0
```

Create a logical volume that is suitable for use as the boot volume. This logical volume has to be the first in the volume group and should be a contiguous volume with bad block relocation turned off.

```
lvcreate -n boot -L 24 -C y -r n /dev/vglvmroot
```

Create a logical volume that is suitable for use as the root volume. This logical volume should be a contiguous volume with bad block relocation turned off.

```
lvcreate -n root -L 64 -C y -r n /dev/vglvmroot
```

Create a logical volume that will be used as primary swap. This volume should be contiguous.

```
lvcreate -n swap -L 64 -C y /dev/vglvmroot
```

Create a logical volume that will be used as the dump volume. This volume should be contiguous.

```
lvcreate -n dump -L 64 -C y /dev/vglvmroot
```

Specify that the logical volume, **root**, will be used as the root volume.

```
lvlnboot -r /dev/vglvmroot/root
```

Specify that the logical volume, **boot**, will be used as the boot volume.

```
lvlnboot -b /dev/vglvmroot/boot
```

Specify that the logical volume, **swap**, will be used as the primary swap.

```
lvlnboot -s /dev/vglvmroot/swap
```

Specify that the logical volume, **dump**, will be used as the dump volume.

```
lvlnboot -d /dev/vglvmroot/dump
```

Display the results of the previous operations.

```
lvlnboot -v /dev/vglvmroot
```

The following example shows configuration of multiple dump volumes.

Specify that logical volumes **/dev/vg00/swap1**, **/dev/vg00/dump2**, and **/dev/vg00/dump3** should be used as the dump logical volumes and that **/dev/vg00/swap1** should also be used as primary swap. Assume that the volume group and the logical volumes have been created and the logical volumes are contiguous.

```
lvlnboot -s /dev/vg00/swap1
lvlnboot -d /dev/vg00/swap1
lvlnboot -d /dev/vg00/dump2
lvlnboot -d /dev/vg00/dump3
```

WARNINGS

Dump Volume Warnings

A dump logical volume, or a swap logical volume used as a dump volume, must lie within the first 2 GB (< 2 GB) of the physical volume. The **lvlnboot** command will not allow a dump logical volume to be configured that crosses the 2 GB boundary, but it will allow such a swap logical volume to be configured.

l

For a system with high-density memory boards installed, **lvlnboot** will be able to support dump logical volumes up to 4 GB of the physical volume.

If the swap device is used as a dump volume by specifying the **dump** default in the system file (see *config*(1M)), care should be taken to ensure that the swap logical volume does not exceed the 2 GB boundary (or 4 GB for the system as mentioned above).

Separate Root/Boot Warnings

Whenever *mkboot*(1M) is used to restore the LIF area of a damaged root physical volume, the **-b** *boot_lv* option of **lvlnboot** must be performed afterwards to record the boot volume information inside the new LIF (see *lif*(4)). Subsequent **lvlnboot** commands such as **lvlnboot -R** are dependent on the *boot_lv* information inside the LIF.

If the **-v** option does not locate the boot volume *boot_lv*, and the **-r** *root_lv* has not yet been performed, then performing the **-r** *root_lv* option will enable the boot volume to be located. The **lvlnboot** command derives the location of boot volume from the location of the root volume.

Separate Root/Boot Maintenance-Mode Warnings

When creating additional root volumes with separate root/boot, a normal boot must be performed on each new root volume so that */stand/rootconf*, which is required for maintenance-mode boots (see *hpux*(1M)), gets created for each new root volume.

Mirrored *root_lv* volumes should start at the same offset on each physical volume so that the location stored in */stand/rootconf* works for maintenance-mode boots off of any mirror.

FILES

/stand/rootconf	Contains the location of the root volume. Used during maintenance-mode boots (see *hpux*(1M)) to locate the root volume for volume groups with separate boot and root volumes.

SEE ALSO

lvcreate(1M), lvrmboot(1M), mkboot(1M), pvcreate(1M), vgcreate(1M), inittab(4), lif(4), lvm(7).

(Requires Optional HP MirrorDisk/UX Software)

NAME

lvmerge - merge two LVM logical volumes into one logical volume

SYNOPSIS

/usr/sbin/lvmerge [**-A** *autobackup*] *dest_lv_path src_lv_path*

Remarks

This command requires the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

lvmerge cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvmerge** command merges two logical volumes of the same size. The number of mirrored copies of the *dest_lv_path* is increased by the number of copies in the *src_lv_path*.

Data previously contained in the *dest_lv_path* is resynchronized using the data in the *src_lv_path*. All new data on the *dest_lv_path* is destroyed.

Whenever a mirrored logical volume is split into two logical volumes, a bit map is stored that keeps track of all writes to either logical volume in the split pair. When the two logical volumes are subsequently merged using **lvmerge**, the bit map is used to decide which areas of the logical volumes need to be resynchronized. This bit map continues to exist until the merge is completed, or one of the logical volumes is extended or reduced, or the system is rebooted.

If there is no bit map available, the entire logical volume is resynchronized.

The normal usage for this command is to merge previously mirrored logical volumes that have been split using the **lvsplit** command (see *lvsplit*(1M). However, the two logical volumes are not required to have been the result of a previous **lvsplit** operation.

Options and Arguments

lvmerge recognizes the following options and arguments:

dest_lv_path	The block device path name of a logical volume.
src_lv_path	The block device path name of a logical volume.
-A *autobackup*	Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

y	Automatically back up configuration changes made to the logical volume. This is the default. After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.
n	Do not back up configuration changes this time.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Merge **/dev/vg00/lvol1b** with **/dev/vg00/lvol1**: Data in **/dev/vg00/lvol1b** will be overridden by **/dev/vg00/lvol1**.

```
lvmerge /dev/vg00/lvol1b /dev/vg00/lvol1
```

WARNINGS

If no bit map is found, all data on *dest_lv_path* is lost after the merge.

lvmerge does not check to guarantee that the allocation policy of *src_lv_path* is preserved after the merge.

SEE ALSO

lvcreate(1M), lvextend(1M), lvsplit(1M).

NAME

lvmmigrate - prepare root file system for migration from partitions to LVM logical volumes

SYNOPSIS

/usr/sbin/lvmmigrate [**-d** *disk_special_file*] [**-e** *file_system* ...] [**-f**] [**-i** *file_system* ...] [**-n**] [**-v**]

DESCRIPTION

The **lvmmigrate** command records the configuration information of the current system in the LIF volume of the boot section for use with a subsequent cold-install process. If there is no LIF volume on the disk, **lvmmigrate** creates it using *lifinit*(1), then records the information in a LIF file named **CUSTOM**. A copy of the LIF file is saved as **/tmp/LVMMIGRATE.CFG**. The information is also written to file **/tmp/LVMMIGRATE** for reviewing. The install process looks for the LIF file **CUSTOM**, and if it exists, uses the information found as the configuration defaults for the root volume group and the root file systems. After the install process has completed, a copy of the **CUSTOM** final configuration can be found on the newly created system in the file **/usr/lib/sw/hpux.install/config.local**.

All file system entries in the **/etc/mnttab** and **/etc/fstab** files are read. **lvmmigrate** also searches for unmounted file systems and possible character data sections in unused disk areas. The file systems appropriate for the root volume group are marked for migration. The default file systems are: **/**, **/home**, **/opt**, **/tmp**, **/usr**, **/var**, and any file system with a mount path beginning with: **/home/**, **/opt/**, **/tmp/**, **/usr/**, **/var/**.

lvmmigrate displays the following information on the standard output: disks and file system names that are marked for migration, disk areas and file systems to be backed up by the user, and instructions for reinstallation.

After executing **lvmmigrate**, the user *must* back up the file systems and any raw device section having useful data to tape. The system is then reinstalled on logical volumes using the configuration information recorded by **lvmmigrate**.

l

Options

lvmmigrate recognizes the following options:

-d *disk_special_file*	Use the specified root disk for reinstallation. Without this option, the current root disk (where root file system **/** is currently located) is assumed and the configuration is recorded in the boot section.
-e *file_system* ...	Exclude each specified default file system from the root volume group. Note that the **/** file system cannot be excluded.
-f	Force the recording of configuration information. Information is recorded in a LIF file named **CUSTOM** in the boot section. Without this option, if there is a file system or LVM record in the boot section, no write is done and a warning message is displayed.
-i *file_system* ...	Include each specified file system in the root volume group, along with the default file systems.
-n	Perform a "no write" operation for preview purposes. Migration information is displayed on the terminal screen, but is not recorded in the boot section of the disk. The **CUSTOM** LIF file is not written, but the files **/tmp/LVMMIGRATE** and **/tmp/LVMMIGRATE.CFG** are still created.
-v	Display all disks, file systems, and possible raw sections present in the system.

EXAMPLES

Prepare a system for migration to root logical volumes. Create a file in the LIF area that the cold-install can use to read default configuration information. Specify verbose mode. Create files **/tmp/LVMMIGRATE** and **/tmp/LVMMIGRATE.CFG**:

```
lvmmigrate -v
```

Display a detailed list of the disks, file systems, and possible raw data sections present in the current system.

```
lvmmigrate -v -n
```

Include file system **/mnt** in the root volume group for migration and exclude file system **/usr/source**. Write configuration information in the boot section of disk **/dev/dsk/c1t0d0**:

```
lvmmigrate -d /dev/dsk/c1t0d0 -i /mnt -e /usr/source
```

WARNINGS

Use of the **-f** option results in overwriting the contents of the boot section. Before using the **-f** option be sure to back up all data on the boot section of the disk specified with the **-d** option.

If there is no LIF volume, **lvmmigrate** uses **lifinit** to create it (see *lifinit*(1)). If file **CUSTOM** already exists in the LIF volume, **lvmmigrate** rewrites it.

Caution: All data on disks being used for reinstallation must be backed up to a *separate device* because the install process overwrites data on all disks used in the new root volume group.

SEE ALSO

lifinit(1).

NAME

lvreduce - decrease space allocation or the number of mirror copies of logical volumes

SYNOPSIS

/usr/sbin/lvreduce [**-A** *autobackup*] [**-f**] **-l** *le_number lv_path*

/usr/sbin/lvreduce [**-A** *autobackup*] [**-f**] **-L** *lv_size lv_path*

/usr/sbin/lvreduce [**-A** *autobackup*] **-m** *mirror_copies lv_path* [*pv_path* ...]

/usr/sbin/lvreduce [**-A** *autobackup*] **-k** *pvkey* **-m** *mirror_copies lv_path* [*pv_path* ...]

Remarks

Mirrored disk operations require the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

lvreduce cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvreduce** command reduces the number of logical extents allocated to a logical volume specified by *lv_path*. The excess physical extents in the logical volume and any mirror copies are deallocated.

Alternatively, it reduces the number of mirror copies in the logical volume. The physical extents that comprise the deleted mirror copy or copies are deallocated. If *pv_path* ... is specified, the mirror or mirrors to be removed will be deallocated from those specific physical volumes.

lvreduce asks for confirmation before deallocating logical extents if the **-f** option is omitted.

Options and Arguments

The **-m** option and *pv_path* argument are only meaningful if the optional HP MirrorDisk/UX software has been installed on the system.

lvreduce recognizes the following options and arguments:

lv_path The block device path name of a logical volume.

pv_path The block device path name of a physical volume.

-A *autobackup* Set automatic backup for invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes.

-f Force reduction of the number of logical extents without first requesting confirmation.

This option can be dangerous when there is a file system on the *lv_path* that is larger than the size that the logical volume is being reduced to. If the file system is unmounted, the **-f** option forces the reduction of the logical volume without reducing the file system. The file system becomes corrupt and is not mountable. If the file system is mounted, **lvreduce** fails, preventing a mounted file system from becoming corrupted.

-l *le_number* Decrease the space allocated to the logical volume, specified in logical extents. *le_number* is a decimal value smaller than the current number of logical extents, in the range **1** to **65535** (the implementation limit).

One, and only one, **-l**, **-L**, or **-m** option must be supplied.

-L *lv_size* Decrease the space allocated to the logical volume, specified in megabytes. *lv_size* is a decimal value smaller than the current logical volume size, in the range **1** to **4096** (4 gigabytes). *lv_size* is rounded up to the nearest multiple of the logical extent size, equivalent to the physical extent size defined for the volume group by the **vgcreate** command (see *vgcreate*(1M)).

One, and only one, **-l**, **-L**, or **-m** option must be specified.

-m *mirror_copies* Reduce the number of mirror copies allocated for each logical extent. A mirror copy contains the same data as the original. *mirror_copies* can have the value **0** or **1**. It must be smaller than the current value.

If optional *pv_path* arguments are specified, the mirror copies are deallocated from the specified physical volumes.

One, and only one, **-l**, **-L**, or **-m** option must be specified.

-k *pvkey* This option should be used only in the special instance when you want to reduce a mirrored logical volume on a physical volume that is missing or has failed. This option will remove a mirrored logical volume from the given *pvkey* (the Physical Volume Number in the volume group). In order to obtain the *pvkey*, use the **-k** option in the **lvdisplay** command. See *lvdisplay*(1M) for details.

Use this option with the **-m** option.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Decrease the number of the logical extents of a logical volume to one hundred:

```
lvreduce -l 100 /dev/vg01/lvol3
```

Reduce to one mirror (that is, an original and one copy) for each logical extent of a logical volume:

```
lvreduce -m 1 /dev/vg01/lvol5
```

Remove mirror copies of logical extents of a logical volume from the physical volume **/dev/dsk/c1t0d0**:

```
lvreduce -m 0 /dev/vg01/lvol4 /dev/dsk/c1t0d0
```

Remove a logical volume from a one-way mirrored set on the third physical volume in a volume group.

```
lvreduce -m 0 -k 3 /dev/vg01/lvol1
```

WARNINGS

LVM does not store any information about which physical extents within a logical volume contain useful data; therefore, reducing the space allocated to a logical volume without doing a prior backup of the data could lead to the loss of useful data. The **lvreduce** command on a logical volume containing a file system of greater length than the size being reduced to will cause data corruption.

To reduce a logical volume being used for swap, that swap area must not be currently in use.

SEE ALSO

lvcreate(1M), lvdisplay(1M), lvextend(1M), pvchange(1M), pvdisplay(1M).

NAME

lvremove - remove one or more logical volumes from LVM volume group

SYNOPSIS

/usr/sbin/lvremove [**-A** *autobackup*] [**-f**] *lv_path* ...

Remarks

lvremove cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvremove** command removes each logical volume specified by *lv_path* ....

Logical volumes must be closed before they can be removed. For example, if the logical volume contains a file system, unmount the file system before removing it.

Options and Arguments

lvremove recognizes the following options and arguments:

lv_path The block device path name of a logical volume.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes this time.

-f Specify that no user confirmation is required.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Remove a logical volume without requiring user confirmation:

```
lvremove -f /dev/vg01/lvol5
```

WARNINGS

This command destroys all data in the specified logical volumes.

SEE ALSO

lvchange(1M), umount(1M).

NAME

lvrmboot - remove LVM logical volume link to root, primary swap, or dump volume

SYNOPSIS

/usr/sbin/lvrmboot [**-A** *autobackup*] [**-d** *dump_lv*] [**-r**] [**-s**] [**-v**] *vg_name*

Remarks

lvrmboot cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvrmboot** command updates all physical volumes contained in the volume group *vg_name* such that the logical volume is removed as a root, primary swap, or dump volume when the system is next booted on the volume group.

Options and Arguments

lvrmboot recognizes the following options and arguments:

vg_name The path name of the volume group.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes this time.

-d *dump_lv* Remove the definition of *dump_lv* as one of the dump volumes. Update the Boot Data Reserved Area.

-r Remove the definitions of all of the root, primary swap, and all dump volumes from the given volume group. Update the Boot Data Reserved Area.

-s Remove the definition of the primary swap volume from the given volume group. Update the Boot Data Reserved Area.

-v Print verbose messages.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Specify that the logical volume **/dev/vg00/lvol3** should be removed as one of the dump logical volumes:

```
lvrmboot -v -d lvol3 /dev/vg00
```

Specify that volume group **/dev/vg00** should no longer be a root volume group. Primary swap and dump are also removed.

```
lvrmboot -r /dev/vg00
```

SEE ALSO

lvlnboot(1M).

(Requires Optional HP MirrorDisk/UX Software)

NAME

lvsplit - split mirrored LVM logical volume into two logical volumes

SYNOPSIS

/usr/sbin/lvsplit [**-A** *autobackup*] [**-s** *suffix*] [**-g** *PhysicalVolumeGroup] lv_path* ...

Remarks

This command requires the installation of the optional HP MirrorDisk/UX software (not included in the standard HP-UX operating system) before it can be used.

lvsplit cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **lvsplit** command splits a single- or double-mirrored logical volume, *lv_path*, into two logical volumes. A second logical volume is created containing one copy of the data. The original logical volume is appropriately reset as unmirrored or single-mirrored.

If the **-s** option is specified, the new logical volume name has the form *lv_pathsuffix*. If **-s** is not specified, *suffix* defaults to **b**, as in *lv_path***b**.

If more than one *lv_path* is specified on the command line, **lvsplit** ensures that all logical volumes are brought offline together in one system call, ensuring predictable results among the logical volumes. Up to 127 logical volumes can be specified on the command line. All logical volumes must belong to the same volume group, and there must be enough unused logical volumes remaining in the volume group to hold the newly split logical volumes. A volume group can contain up to 255 logical volumes.

If *PhysicalVolumeGroup* is specified, the offline logical volumes are created using the mirror copies on the physical volumes contained in the specified physical volume group.

Whenever a mirrored logical volume is split into two logical volumes, a bit map is stored that keeps track of all writes to either logical volume in the split pair. When the two logical volumes are subsequently merged using **lvmerge**, the bit map is used to decide which areas of the logical volumes need to be resynchronized (see *lvmerge*(1M)). This bit map remains in existence until the merge is completed, until one of the logical volumes is extended, reduced, or split again, or until the system is rebooted.

The new logical volume must be checked with the **fsck** command before it is mounted (see *fsck*(1M)). **lvsplit** flushes the file system to a consistent state except for pipes and unlinked but open files.

To rejoin two split copies of a logical volume, use the **lvmerge** command (see *lvmerge*(1M)).

Options and Arguments

lvsplit recognizes the following options and arguments:

lv_path The block device path name of a logical volume. Up to 127 logical volumes in the same volume group can be specified at one time.

-A *autobackup* Set automatic backup for invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes this time.

-g *PhysicalVolumeGroup*
The offline logical volumes will be created using the mirror copies on the physical volumes in the specified *PhysicalVolumeGroup.*

-s *suffix* Specify the suffix to use to identify the new logical volume. The new logical volume name has the form *lv_pathsuffix*. If **-s** is omitted, *suffix* defaults to **b**, as in *lv_path***b**.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Split the mirrored logical volume **/dev/vg00/lvol1** into two copies. Call the new logical volume **/dev/vg00/lvol1backup**:

```
lvsplit -s backup /dev/vg00/lvol1
```

Split the mirrored logical volume **/dev/vg00/lvol1** into two copies. The offline logical volume will be created using the mirror copy on the physical volumes contain in the physical volume group **pvg1**.

```
lvsplit -g pvg1 /dev/vg00/lvol1
```

Split an online logical volume which is currently mounted on **/usr** so that a backup can take place:

```
lvsplit /dev/vg00/lvol1
fsck /dev/vg00/lvol1b
mount /dev/vg00/lvol1b /usr.backup
```

Perform a backup operation, then:

```
umount /usr.backup
lvmerge /dev/vg00/lvol1b /dev/vg00/lvol1
```

Split two logical volumes at the same time:

```
lvsplit /dev/vg01/database1 /dev/vg01/database2
```

Perform operation on split logical volumes, then rejoin them:

```
lvmerge /dev/vg01/database1b /dev/vg01/database1
lvmerge /dev/vg01/database2b /dev/vg01/database1
```

WARNINGS

After a two-way mirrored logical volume has been split once, it cannot be split again without merging the logical volumes using the **lvmerge** command (see *lvmerge*(1M)).

SEE ALSO

lvcreate(1M), lvextend(1M), lvmerge(1M).

(Requires Optional HP MirrorDisk/UX Software)

NAME

lvsync - synchronize stale mirrors in LVM logical volumes

SYNOPSIS

/usr/sbin/lvsync *lv_path* ...

Remarks

This command requires the installation of the optional HP MirrorDisk/UX software (not included in the standard HP-UX operating system) before it can be used.

DESCRIPTION

The **lvsync** command synchronizes the physical extents of each logical volume specified by *lv_path*. Synchronization occurs only on physical extents that are stale mirrors of the original logical extent. The synchronization process can be time consuming, depending on the hardware characteristics and the amount of data.

Arguments

lvsync recognizes the following argument:

lv_path	The block device path name of a mirrored logical volume.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Synchronize the mirrors on a logical volume:

```
lvsync /dev/vg01/lvol5
```

SEE ALSO

lvdisplay(1M), vgsync(1M).

NAME

lwpstat - show Fibre Channel Light Weight Protocol network status

SYNOPSIS

lwpstat [**-s**]

DESCRIPTION

lwpstat displays Light Weight Protocol statistics for Fibre Channel network interfaces. When **lwpstat** is used without any options, the state of all active FC_LWP sockets are shown.

Options

The following options and parameters are recognized by **lwpstat**:

-s	Show summary statistics for datagram and stream based protocols.

AUTHOR

lwpstat was developed by the HP.

NAME

makedbm - make a Network Information System database

SYNOPSIS

/usr/sbin/makedbm [**-b**] [**-l**] [**-s**] [**-i** *nis_input_file*] [**-o** *nis_output_name*]
[**-d** *nis_domain_name*] [**-m** *nis_master_name*] *infile outfile*

/usr/sbin/makedbm -u *database_name*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

makedbm generates databases (maps) for the Network Information System (NIS) from *infile*. A database created by **makedbm** consists of two files: *outfile*.*pag* and *outfile*.*dir*. A **makedbm** database contains records called **dbm records** composed of key-value pairs.

Each line of *infile* is converted to a single dbm record; all characters up to the first tab or space form the key, and the remainder of the line is the value. If a value read from *infile* ends with ****, the value for that record is continued onto the next line. The NIS clients must interpret the **#** character (which means that **makedbm** does not treat the **#** as if it precedes a comment). If *infile* is a hyphen (**-**), **makedbm** reads standard input.

makedbm always generates a special dbm record with the key **YP_LAST_MODIFIED**, whose value is the time of last modification of *infile* (or the current time, if *infile* is **-**). This value is also known as the order number of a map, and **yppoll** prints it for a specified NIS map (see *yppoll*(1M)).

Another special dbm record created by **makedbm** has the key **YP_MASTER_NAME**. Its value is usually the host name retrieved by **gethostname()**; however, the **-m** option can be used to specify a different value (see *gethostname*(2)).

If the **-b** option is used, another special dbm record with the **YP_INTERDOMAIN** key is created. When this key exists in the NIS *host.by** maps and the NIS host name resolution fails, the ypserv process will query the Internet domain name server, *named*(1M), to provide the host name resolution. Before using the **-b** option, it is recommended that the name services switch, *switch*(4), be set to allow NIS host name resolution first. (Note that, since the ypserv process only checks *hosts.byname* and *hosts.byaddr* for the existence of the **YP_INTERDOMAIN** key, using the **-b** option on any other NIS map will have no effect. Also, the **-b** option should be used on both the *hosts.byname* and *hosts.byaddr* maps, not one exclusively.)

If the **-s** option is used, another special dbm record created is the **YP_SECURE** key. If this key exists in an NIS map, ypserv will only allow privileged processes (applications that can create reserved ports) to access the data within the map.

Options

makedbm recognizes the following options and command-line arguments.

-b Create a special dbm record with the key **YP_INTERDOMAIN**. This key, which is in the *hosts.byname* and *hosts.byaddr* maps, allows the ypserv process to query the Internet domain name server, (see *named*(1M)).

-l Convert the keys of the given map to lowercase. This command option allows host name matches to work independent of character-case distinctions.

-s Accept connections from secure NIS networks only.

-i Create a special dbm record with the key **YP_INPUT_FILE** and the value **nis_input_file**. If the -s option is used, another special dbm record created is the **YP_SECURE** key. If this key exists in an NIS map, ypserv will only allow privileged processes to access the data within the map. (i.e. applications that can create reserved ports.)

-o Create a special dbm record with the key **YP_OUTPUT_NAME** and the value *nis_output_name*.

-d Create a special dbm record with the key **YP_DOMAIN_NAME** and the value *nis_domain_name*.

-m Replace the value of the special dbm record whose key is **YP_MASTER_NAME** with *nis_master_name*.

-u Undo the *database_name* (i.e., write the contents of *database_name* to the standard output), one dbm record per line. A single space separates each key from its value.

EXAMPLES

Shell scripts can be written to convert ASCII files such as **/etc/netgroup** to the key-value form used by **makedbm**. For example,

```
#!/usr/bin/sh
/usr/bin/awk 'BEGIN { FS = ":" } { print $1, $0 }' \
     /etc/netgroup | \
makedbm - netgroup
```

converts the file **/etc/netgroup** to a form that is read by **makedbm** to make the NIS map **netgroup**. The keys in the database are *netgroup*(4) names, and the values are the remainders of the lines in the **/etc/netgroup** file.

AUTHOR

makedbm was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypinit(1M), ypmake(1M), yppoll(1M), gethostname(2), netgroup(4), ypfiles(4).

NAME

makemap - creates database maps for sendmail

SYNOPSIS

makemap [**-N**] [**-d**] [**-f**] [**-o**] [**-r**] [**-v**] *maptype mapname*

DESCRIPTION

makemap creates the database maps used by the keyed map lookups in *sendmail*(1M). It reads input from the standard input and outputs them to the indicated *mapname*.

makemap handles up to three different database formats, selected using the *maptype* parameter. They may be

dbm	DBM format maps. (.pag,.dir)
btree	B-Tree format maps. (.db)
hash	Hash format maps. (.db)

In all cases, **makemap** reads lines from the standard input consisting of two words separated by white space. The first is the database key, the second is the value. The value may contain **%***n* strings to indicated parameter substitution. Literal parentheses should be doubled (**%%**). Blank lines and lines beginning with pound sign (**#**) are ignored.

Flags

-N Include the null byte that terminates strings in the map. This must match the **-N** flag in the **sendmail.cf K** line.

-d Allow duplicate keys in the map. This is only allowed on B-Tree format maps. If two identical keys are read, they will both be inserted into the map.

-f Normally all upper case letters in the key are folded to lower case. This flag disables that behaviour. This is intended to mesh with the **-f** flag in the **K** line in **sendmail.cf**. The value is never case folded.

-o Append to an old file. This allows you to augment an existing file.

-r Allow replacement of existing keys. Normally **makemap** complains if you repeat a key, and does not do the insert.

-v Verbosely print what it is doing.

SEE ALSO

sendmail(1M).

HISTORY

The **makemap** command appeared in 4.4BSD. The manual page originally came from **sendmail** 8.7.

NAME

map-mbone - Multicast Router Connection Mapper

SYNOPSIS

/usr/sbin/map-mbone [**-d** *debuglevel*] [**-f**] [**-g**] [**-n**] [**-r** *retries*] [**-t** *timeout*] [*multicast-router*]

DESCRIPTION

map-mbone requests the multicast router connection information from the *multicast-router*, and prints the information to the standard out. **map-mbone** sends out the *ASK_NEIGHBORS* igmp message to the multicast-router. When the multicast-router receives the request, it sends back its configuration information. *multicast-router* can be either an ip address or a system name.

If the *multicast-router* is not specified, *flood* mode is on by default and the igmp request message is sent to all the multicast router on the local network. With *flood* mode on, when **map-mbone** finds new neighbor routers from the replies, it will send the same igmp request to the new neighbor routers. This activity continues until no new neighbor routers are reported in the replies.

The command line options are:

-d*debuglevel*	Sets the level for printing out the debug message. The default is 0, which prints only error and warning messages. Debug level three prints most the messages.
-r*retries*	Sets the retry times to poll the routing daemon for information. The default is 1.
-t*timeout*	It specifies the timeout value in seconds for waiting the reply. The default value is 2 seconds.
-f	Sets the *flood* mode on. It is the default value when no *multicast-router* is given on the command line input.
-g	Generates output in GRaphEd format.
-n	Disable DNS lookup for the multicast router names.

The output contains the interface configuration information of the requested router(s). The format for each interface output is:

```
interface_addr -> neighbor_addr (neighbor_name) [metrics/thresh/flags]
```

If there are multiple neighbor routers on one interface, they will all be reported. The *neighbor_name* will not be printed if the **-n** option is specified on the command line.

The possible values for **flags** are:

tunnel	Neighbors are reached via tunnel.
srcrt	The tunnel uses IP source routing.
down	The interface is down.
disabled	The interface is administratively disabled for multicast routing.
querier	The local router is the querier of the subnet.

The format of the GRaphEd output is:

```
interface_addr_in_integer {$ NP low_byte_addr high_byte_addr} node_name
[ neighbor_addr_in_integer metrics/threshold/flags ]
```

If there is no neighbor router on an interface, then a ***** will be put next to the node_name. If there are multiple neighbor routers on one interface, all of them will be reported. The possible values for **flags** are:

E	The neighbor is reached via tunnel.
P	The neighbor is on the same network/subnet.
D	The interface is down.

Please see *mrouted*(1M) for **metrics** and **thresh**.

EXAMPLES

Querying **camden.cup.hp.com** for the multicast router connection information.

```
map-mbone hpntclt.cup.hp.com
```

```
127.0.0.1 (localhost) [version 3.3]:
  193.2.1.39 -> 0.0.0.0 (all-zeros-broadcast) [1/1/disabled]
  15.13.106.144 -> 15.255.176.33 (matmos.hpl.hp.com) [10/1/tunnel]
  15.13.106.144 -> 15.17.20.7 (hpspddc.vid.hp.com) [10/1/tunnel/down]
```

Querying **hpntcbs.cup.hp.com** for multicast router connectivity with *-g* option:

```
map-mbone -g hpntcbs.cup.hp.com
```

```
GRAPH "Multicast Router Connectivity: Wed Feb  1 17:34:59 1995"=UNDIRECTED
  252537488 {$ NP 1440 1060 $} "hpntc1t.cup.hp.com*"
    ;
  252538974 {$ NP 940 1120 $} "hpntcbs.cup.hp.com"
    252537488 "10/1E"
    252539807 "1/1P"
    ;
  252539807 {$ NP 1590 1150 $} "hpntc1h.cup.hp.com*"
    ;
```

Note

map-mbone must be run as root.

AUTHOR

map-mbone was developed by Pavel Curtis.

SEE ALSO

mrouted (1M), mrinfo(1M).

NAME

mc - media changer manipulation utility

SYNOPSIS

mc [**-p** *device*] [**-a** *num*] [**-q**] [**-c** *<src_element_type><dest_element_type>*]

mc [**-p** *device*] [**-l 0**| **1**] [**-e** *element_type*] [**-r** *element_type*]

mc [**-p** *device*] **-s** *<element_type><num>* **-d** *<element_type><num>*

mc [**-h** |**-?**]

DESCRIPTION

The **mc** utility provides users with a command-line interface to send media manipulation commands to an autoloader or media changer device. It takes "element types" as arguments to most of the options. The valid element types (*element_types*) are:

D	Specifies a Data Transfer (DT) element.
I	Specifies an Import/Export (IE) element.
M	Specifies a Medium Transport (MT) element.
S	Specifies a Storage (ST) element.

An example of a Data Transfer element is the embedded tape drive of the autoloader. An example of an Import/Export element is the slot(s) by which an item of the media maybe inserted or removed from the autoloader. An example of a Medium Transport element is the robotic picker portion of the autoloader. An example of a Storage element is the magazine slot of the autoloader.

Please see examples below for usage.

Options

mc recognizes the following options and arguments:

-a *num* Prints the SCSI bus address of the drive slot specified by *num*.

-c *<src_element_type><dest_element_type>*
Determines whether a move from source to destination is valid. Uses device capabilities mode page and will return TRUE or FALSE. There should be no spaces in the the source and destination element type values. For example, **-c DS** specifies a Data Transfer element as the source and a Storage element as the destination.

-e *element_type*
Prints out the number of elements of element type. See element types above. Multiple types can be specified. For example, **-e IDSM** specifies all the valid element types.

-h|**-?** Prints out usage description.

-l 0|**1** Allow (**0**) or prevent (**1**) media removal.

-p *device* Specifies the pass-through device file to the library device.

-q Prints out Vendor ID, Product ID and Product Rev standard inquiry information.

-r *element_type*
Prints out the status (FULL/EMPTY/NONE) of element slots of element type(s). See element types above. Multiple types can be specified. For example, **-r IDSM** specifies all the valid element types.

-s *<element_type><num>*
Specifies the element type and slot number (*<num>*) for the move medium source. There should be no space between the element type and the slot number. For example, **-sS1** specifies a Storage element in slot number 1. This option cannot be specified more than twice per invocation.

-d *<element_type><num>*
Specifies the element type and slot number for the move medium destination. There should be no space between the element type and the slot number. For example, **-dD3** specifies a Data Transfer element in slot number 3. This option cannot be specified more than twice per invocation.

RETURN VALUE

`mc` returns 0 upon successful completion and -1 otherwise.

DIAGNOSTICS

`ERROR: 0x5 Illegal Request: 0x3b0d Medium Destination element full`

The above error message could be a result of the following command **`mc -s S2 -d D1`** that was used to move a media to an embedded drive that is already full.

`ERROR: /dev/scsi/3: No such file or directory`

If the default SCSI pass-through device file does not exist and no other device file is specified, then the above error message will be printed.

EXAMPLES

Using a DDS-2 autoloader with a six cartridges magazine as an example:

To see the status of the autoloader's Data Transfer and Storage element types.

```
mc -r DS
```

The following shows an example output from the above command. The output indicates that there is an item of media in slot 2 (ST_slot_2), an item of media in the embedded drive (DT_slot_1), and all the other slots are empty.

```
DT_slot_1 FULL
ST_slot_1 EMPTY
ST_slot_2 FULL
ST_slot_3 EMPTY
ST_slot_4 EMPTY
ST_slot_5 EMPTY
ST_slot_6 EMPTY
```

To move media from an embedded drive to slot 5 and then move media from slot 2 to an embedded drive.

```
mc -s D1 -d S5 -s S2 -d D1
```

To check if a move from a Data Transfer element to a Storage element is possible.

```
mc -c DS
```

The following example output indicates that moves from Data Transfer element types to Storage element types are valid.

```
DT->ST: TRUE
```

WARNINGS

Note for all DDS autoloaders. After the **mc** command has been used for the first time, the autoloader will enter into random mode. Once in random mode, all front panel button features are disabled except for the Eject Button. To go back to stacker mode, the magazine must be ejected and then reinserted.

DEPENDENCIES

The **mc** command supports the following autoloaders and libraries:

C1553A	HP DDS-2 Autoloader
C1557A	HP DDS-3 Autoloader
C1194	HP DLT Library
STK 9710	StorageTek DLT/4890 Library
ATL 4/52	ATL DLT Library

A SCSI pass-through driver must be configured and the device file created before this command can be used to manipulate the autoloader.

Series 700

The **ctl** pass-through driver must be configured. See *scsi_ctl*(7).

Series 800

The **spt** pass-through driver must be configured. See *scsi_pt*(7).

AUTHOR

`mc` was developed by Hewlett-Packard.

FILES

`/dev/scsi/3`   Default pass-through device file.

SEE ALSO

scsi(7), scsi_ctl(7), scsi_pt(7).

NAME

mk_kernel - build a bootable HP-UX kernel and/or kernel modules

SYNOPSIS

usr/sbin/mk_kernel [**-o** *pathname*] [**-s** *system_file*] [**-S**] [**-v**]

/usr/sbin/mk_kernel -M *module_name* [[**-M** *module_name*]...] [**-v**]

DESCRIPTION

mk_kernel builds an executable file which can be used as a bootable kernel and kernel modules if any are configured. If the build succeeds, the newly built kernel is called **vmunix_test**, and the kernel function set directory (where the function set directory is the directory structure containing the set of modules that correspond to the kernel) is called **dlkm.vmunix_test**. The file and directory are placed in the build directory, as defined below.

The build directory is the target directory where **mk_kernel** places files and directories. In addition to the kernel and kernel modules, files such as **conf.c**, **conf.o**, and **tune.h** are also placed in the build directory.

If the path used to designate the system file is **/stand/system**, the build directory is **/stand/build**. If another path is used to designate the system file, the build directory is the current working directory. System files for the kernel modules are expected to be found in **/stand/system.d**. Libraries for the kernel are expected to be found in **/usr/conf/lib**. The master file used is the composite of files found under **/usr/conf/master.d**.

If the **-o** option is not specified, the kernel file and kernel function set directory remain in the working directory. If **-o /stand/vmunix** is specified, the target kernel file and kernel function set directory are not overwritten. The new kernel file and the kernel function set directory are moved to the default path as the system shuts down or starts up. The previous versions of the file and directory are renamed to **/stand/vmunix.prev** and **/stand/dlkm.vmunix.prev**. Until the system reboots, the new kernel file and the directory must be kept as **vmunix_test** and **dlkm.vmunix_test**, respectively.

If the **-o** option is specified with other than **/stand/vmunix**, the kernel file and kernel function set directory is created or updated immediately. In case the administrator needs to place these targets to the system default path, the **kmupdate** command must be used to trigger the replacement. Manually replacing the default kernel (**/stand/vmunix**) or any file under the kernel function set directory (**/stand/dlkm**) must be avoided.

mk_kernel exits with no action if the environment variable **SW_INITIAL_INSTALL** has the value of 1. **SW_INITIAL_INSTALL** is exported by SD with that value only when the system is undergoing its initial software system installation.

Options

mk_kernel recognizes the following options.

-M *module_name*
: Specify the module to configure. No kernel image will be generated. For details see *config*(1M).

-o *pathname*
: Specify the target file path. The created kernel file, **vmunix_test**, is moved from the build directory to the path specified by the option argument. The associated kernel function set directory, **dlkm.vmunix_test**, is moved to the same destination directory.

  If the default kernel, **/stand/vmunix,** is specified or the **-o** option is not specified, the created kernel file does not replace **/stand/vmunix** and remains as **vmunix_test**.

  The kernel file and associated kernel function set directory are automatically moved to **/stand/vmunix** and **/stand/dlkm** during either shutdown or startup.

-s *system_file*
: Specify the kernel template file. If this option is not specified, the system file **/stand/system** is used.

-S
: Specify that all configured kernel modules are to be statically linked into the kernel. For details see *config*(1M).

-v
: Verbose mode.

RETURN VALUE

mk_kernel returns 0 upon normal completion, and 1 if an error occurred.

DIAGNOSTICS

Messages and warnings are sent to **stdout**. Messages from **config** and other commands are displayed when invoked from **mk_kernel**. Errors cause **mk_kernel** to halt immediately; warnings allow the program to continue.

EXAMPLES

mk_kernel -o /stand/vmunix

Uses the file **/stand/system** to build a new kernel and kernel module(s). The new kernel file is placed in **/stand/build/vmunix_test** upon success. Kernel function set directory is placed in **/stand/build/dlkm.vmunix_test**. These files are moved automatically to **/stand/vmunix** and **/stand/dlkm** during shutdown or startup. The current set is saved as **/stand/vmunix.prev** and **/stand/dlkm.vmunix.prev**.

mk_kernel -s /mnt/altsys/stand/system.new

Uses the file **/mnt/altsys/stand/system.new** to build a new kernel and kernel module(s). The new kernel is named **vmunix_test** in the present working directory. The kernel function set directory, **dlkm.vmunix_test**, is placed in the current working directory.

mk_kernel -s /stand/system -o /tmp/new_kernel

Uses the file **/stand/system** to build a new kernel and kernel module(s). The new kernel file is placed in **/tmp/new_kernel**. The kernel function set directory is in **/tmp/dlkm.new_kernel.** If the administrator wants to use this kernel as the default kernel, the **kmupdate** command can be used.

WARNINGS

System administrators are expected to treat the kernel and dlkm, *kernel_name*, as a set. Do not manually copy the kernel or manually update the current kernel file with its associated kernel function set directory. To update the default kernel, always use the **kmupdate** command.

Kernel modules are separate objects to be independently configured into the system without requiring a reboot. To accomplish this, the kernel relies on several files under the kernel function set directory.

- kernel file: *kernel_name* or **/stand/vmunix**
- kernel function set directory: **dlkm.***kernel_name* or **/stand/dlkm**

The kernel function set directory contains kernel modules, a module database file, and a kernel symbol table file. These files and directories are expected to be found in a directory whose name matches the booted kernel. If the kernel function set directory is not found, the dynamically loadable kernel module feature is disabled.

FILES

/stand/vmunix	Default kernel
/stand/dlkm	Default kernel function set directory
/stand/system	Default system file
/stand/build/vmunix_test	Kernel built by **mk_kernel**
/stand/build/dlkm.vmunix_test	Kernel function set directory build by **mk_kernel**
/stand/vmunix.prev	Saved kernel
/stand/dlkm.vmunix.prev	Saved kernel function set directory

SEE ALSO

config(1M), kmupdate(1M).

NAME

mkboot, rmboot - install, update or remove boot programs from disk

SYNOPSIS

/usr/sbin/mkboot [**-b** *boot_file_path*] [**-c** [**-u**] | **-f** | **-h** | **-u**] [**-i** *included_lif_file*] [**-p** *preserved_lif_file*] [**-l** | **-H** | **-W**] [**-v**] *device*

/usr/sbin/mkboot [**-a** *auto_file_string*] [**-v**] *device*

/usr/sbin/rmboot *device*

DESCRIPTION

mkboot is used to install or update boot programs on the specified device file.

The position on *device* at which boot programs are installed depends on the disk layout of the device. **mkboot** examines *device* to discover the current layout and uses this as the default. If the disk is uninitialized, the default is LVM layout. The default can be overriden by the **-l, -H** or **-W** options.

Boot programs are stored in the boot area in Logical Interchange Format (LIF), which is similar to a file system. For a device to be bootable, the LIF volume on that device must contain at least the **ISL** (the initial system loader) and **HPUX** (the HP-UX bootstrap utility) LIF files. If, in addition, the device is an LVM physical volume, the **LABEL** file must be present (see *lvlnboot*(1M)).

Options

mkboot recognizes the following options:

-a *auto_file_string*
: If the **-a** option is specified, mkboot creates an autoexecute file **AUTO** on *device,* if none exists. **mkboot** deposits *auto_file_string* in that file. If this string contains spaces, it must be quoted so that it is a single parameter.

-b *boot_file_path*
: If this option is given, boot programs in the pathname specified by *boot_file_path* are installed on the given device.

-c
: If this option is specified, **mkboot** checks if the available space on *device* is sufficient for the boot programs. If the **-i** option is also specified, **mkboot** checks if each *included_lif_file* is present in the boot programs. If the **-p** option is specified, it checks if each *preserved_lif_file* is present on the *device.* If all these checks succeed, **mkboot** exits with a status code of 0. If any of these checks fail, **mkboot** exits with a status code of 1. If the verbose option is also selected, a message is also displayed on the standard output.

-f
: This option forces the information contained in the boot programs to be placed on the specified *device* without regard to the current swapping status. Its intended use is to allow the boot area to grow without having to boot the system twice (see **-h** option).

: This option should only be used when the system is in the single user state.

: This could be a dangerous operation because swap space that is already allocated and possibly in use will be overwritten by the new boot program information. A message is also displayed to the standard output stating that the operator should immediately reboot the system to avoid system corruption and to reflect new information on the running system.

: A safer method for reapportioning space is to use the **-h** option.

: This option is valid only if *device* has the Whole Disk layout.

-h
: Specifying this option shrinks the available space allocated to swap in the LIF header by the amount required to allow the installation of the new boot programs specified by *boot_file_path*.

: After the LIF header has been modified, reboot the system to reflect the new swap space on the running system. At this point, the new boot programs can be installed and the system rebooted again to reflect the new boot programs on the running system. This is the safe method for accomplishing the capability of the **-f** option.

: This option is valid only if *device* has the Whole Disk layout.

-H	If this option is specified, **mkboot** treats *device* to be a Hard Partition layout disk. This option cannot be used along with the **-l** and **-W** options.
-i *included_lif_file*	If the **-i** option is specified one or more times, **mkboot** copies each *included_lif_file* and ignores any other LIF files in the boot programs. The sole exceptions to this rule are the files **ISL** and **HPUX**, which are copied without regard to the **-i** options. If *included_lif_file* is also specified with the **-p** option, the **-i** option is ignored. If the **-i** option is used with **LABEL** as its argument and the file **LABEL** does not exist in the boot programs, and *device* is an LVM layout disk or the **-l** option is used, **mkboot** creates a minimal **LABEL** file on *device* which will permit the system to boot on *device,* possibly without swap or dump.
-l	If this option is used, **mkboot** treats *device* as an LVM layout disk, regardless of whether or not it is currently set up as one. This option cannot be used along with the **-H** and **-W** options.
-p *preserved_lif_file*	If the **-p** option is specified one or more times, **mkboot** keeps each specified *preserved_lif_file* intact on *device.* If *preserved_lif_file* also appears as an argument to the **-i** option, that **-i** option is ignored. This option is typically used with the autoexecute file **AUTO** and with the LVM and SwitchOver/UX file **LABEL.** If **LABEL** is specified as an argument to the **-p** option and **LABEL** does not exist on the *device,* and if the layout is LVM, **mkboot** creates a minimal **LABEL** file. In general, if *preserved_lif_file* is not on the *device,* **mkboot** fails. An exception to this condition is if the *preserved_lif_file* is **LABEL** and the layout is not LVM, in which case the **LABEL** file is ignored.
-u	If **-u** is specified, **mkboot** uses the information contained in the LIF header to identify the location of the swap area, boot area, and raw I/O so that installation of the boot programs does not violate any user data. Normally, the LIF header information is overwritten on each invocation of mkboot. This option is typically used with the **-W** option, to modify boot programs on a disk that is actively supporting swap and/or raw I/O.
-v	If this option is specified, **mkboot** displays its actions, including the amount of swap space available on the specified device.
-W	If this option is specified, **mkboot** treats *device* as a disk having the Whole Disk layout. This option cannot be used along with the **-l** and **-H** options.
device	Install the boot programs on the given device special file. The specified *device* can identify either a character-special or block-special device. However, **mkboot** requires that both the block and character device special files be present. **mkboot** attempts to determine whether *device* is character or block special by examining the specified path name. For this reason, the complete path name must be supplied. If **mkboot** is unable to determine the corresponding device file, a message is written to the display, and **mkboot** exits.

m

rmboot removes the boot programs from the boot area.

EXAMPLES

Install default boot programs on the specified disk, treating it as an LVM disk:

```
mkboot -l /dev/dsk/c0t5d0
```

Use the existing layout, and install only SYSLIB and ODE files and preserve the EST file on the disk:

```
mkboot -i SYSLIB -i ODE -p EST /dev/rdsk/c0t5d0
```

Install only the SYSLIB file and retain the ODE file on the disk. Use the Whole Disk layout. Use the file **/tmp/bootlf** to get the boot programs rather than the default. (The **-i ODE** option will be ignored):

```
mkboot -b /tmp/bootlf -i SYSLIB -i ODE -p ODE -W /dev/rdsk/c0t5d0
```

WARNINGS

If *device* has a Whole Disk layout, a file system must reside on the device being modified.

When executing from a recovery system, the **`mkboot`** command (if used) must be invoked with the **`-f`** option; otherwise it will not be able to replace the boot area on your disk.

If *device* is, or is intended to become an LVM physical volume, *device* must specify the whole disk.

If *device* is, or is intended to become a Hard Partitioned disk, *device* must specify section 6.

DEPENDENCIES

`mkboot` and **`rmboot`** fail if file system type on *device* is not HFS.

LVM and Hard Partition Layouts

The **`-f`**, **`-h`** and **`-u`** options are not supported.

AUTHOR

`mkboot` and **`rmboot`** were developed by HP.

FILES

`/usr/lib/uxbootlf`	file containing default boot programs
`ISL`	initial system loader
`HPUX`	HP-UX bootstrap and installation utility
`AUTO`	defines default/automatic boot behavior (see *hpux*(1M))
`LABEL`	used by SwitchOver/UX and LVM
`RDB`	diagnostics tool
`IOMAP`	diagnostics tool

SEE ALSO

boot(1M), hpux(1M), isl(1M), lif(4), lvlnboot(1M), mkfs(1M), newfs(1M).

NAME

mkfs (generic) - construct a file system

SYNOPSIS

/usr/sbin/mkfs [**-F** *FStype*] [**-o** *specific_options*] [**-V**] *special* [*operands*]

/usr/sbin/mkfs [**-F** *FStype*] [**-m**] [**-V**] *special*

DESCRIPTION

The **mkfs** command creates a file system by writing on the special file *special*. *operands* are listed on file system specific manual pages (see "SEE ALSO").

Options

mkfs recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-m Display the command line that was used to create the file system. The file system must already exist. This option provides a means of determining the parameters used to construct the file system.

-o *specific_options*
Specify options specific to the file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for an *FStype*-specific module of the command. See the file system specific manual entries for a description of the *specific_options* that are supported, if any.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the specified options and arguments with other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Execute the **mkfs** command to create a 32MB HFS file system on **/dev/dsk/c1t2d0**:

```
mkfs -F hfs /dev/dsk/c1t2d0 32768
```

Execute the **mkfs** command on an HFS file system, **/dev/dsk/c1t2d0**, to recreate the command that was used to create the file system on **/dev/dsk/c1t2d0**:

```
mkfs -F hfs -m /dev/dsk/c1t2d0
```

AUTHOR

mkfs was developed by HP and the University of California, Berkeley.

FILES

/etc/default/fs	Specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

chmod(1), bdf(1M), df(1M), fsadm(1M), fsck(1M), fstyp(1M), mkfs_hfs(1M), mkfs_vxfs(1M), newfs(1M), fstab(4), group(4), passwd(4), fs_wrapper(5).

STANDARDS CONFORMANCE

mkfs: SVID3

NAME

mkfs (hfs) - construct an HFS file system

SYNOPSIS

/usr/sbin/mkfs [**-F hfs**] [**-d**] [**-L**|**-S**] [**-V**] [**-o** *specific_options*] *special* [*size* [*nsect ntrack blksize fragsize ncpg minfree rps nbpi*]]

/usr/sbin/mkfs [**-d**] [**-F hfs**] [**-L**|**-S**] [**-V**] [**-o** *specific_options*] *special* [*proto* [*nsect ntrack blksize fragsize ncpg minfree rps nbpi*]]

/usr/sbin/mkfs [**-F hfs**] [**-m**] [**-V**] *special*

Remarks

HFS file systems are normally created with the **newfs** command (see *newfs_hfs*(1M)).

DESCRIPTION

The **mkfs** command constructs an HFS file system by writing on the special file *special*. The **mkfs** command builds the file system with a root directory and a **lost+found** directory (see *fsck_hfs*(1M)). The **FS_CLEAN** magic number for the file system is stored in the superblock.

The **mkfs** command creates the file system with a rotational delay value of zero (see *tunefs*(1M)).

Options

mkfs recognizes the following options:

-F hfs Specify the HFS file system type.

-d This option allows the **mkfs** command to make the new file system in an ordinary file. In this case, *special* is the name of an existing file in which to create the file system. When this option is used, the size of the new file system cannot be defaulted. It must either be specified on the command line following *special*, or if a prototype file is being used, it must be the second token in the prototype file as usual.

-L|**-S** There are two types of HFS file systems, distinguished mainly by directory formats that place different limits on the length of file names.

If **-L** is specified, build a long-file-name file system that allows directory entries (file names) to be up to **MAXNAMLEN** (255) bytes long.

If **-S** is specified, build a short-file-name file system that allows directory entries (file names) to be up to **DIRSIZ** (14) bytes long.

If neither **-L** nor **-S** is specified, build a file system of the same type as the root file system.

-m Display the command line that was used to create the file system. The file system must already exist. This option provides a means to determine the parameters used to construct the file system.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-o *specific_options*
Specify a list of comma separated suboptions and/or keyword/attribute pairs from the list below.

largefiles|nolargefiles
Controls the *largefile featurebit* for the file system. The default is **nolargefiles**. This means the bit is not set, and files created on the file system will be limited to less than 2 gigabytes in size. If **largefiles** is specified, the bit is set and the maximum size for files created on the file system is not limited to 2 gigabytes (see *mount_hfs*(1M) and *fsadm_hfs*(1M)).

Arguments

mkfs recognizes the following arguments:

special The file name of a special file.

One of the following arguments can be included after *special*:

m

size The number of **DEV_BSIZE** blocks in the file system. **DEV_BSIZE** is defined in **<sys/param.h>**. The default value is the size of the entire disk or disk section minus any swap or boot space requested.

The size of HFS file systems are limited by **UFS_MAXDEVBLK** (defined in **<sys/fs.h>**) to 256GB-1 or 268,435,455 blocks.

proto The name of a file that can be opened. The **mkfs** command assumes it is a prototype file and takes its directions from that file. See "Prototype File Structure" below.

The following optional arguments allow fine-tune control over file system parameters:

nsect The number of sectors per track on the disk. The default value is 32 sectors per track.

ntrack The number of tracks per cylinder on the disk. The default value is 16 tracks per cylinder.

blksize The primary block size for files on the file system. Valid values are: 4096, 8192, 16384, 32768, and 65536. The default value is 8192 bytes.

fragsize The fragment size for files on the file system. *fragsize* represents the smallest amount of disk space to be allocated to a file. It must be a power of two no smaller than **DEV_BSIZE** and no smaller than one-eighth of the file system block size. The default value is 1024 bytes.

ncpg The number of disk cylinders per cylinder group. This number must be in the range 1 to 32. The default value is 16 cylinders per group.

minfree The minimum percentage of free disk space allowed. The default value is 10 percent.

Once the file system capacity reaches this threshold, only users with appropriate privileges can allocate disk blocks.

rps The number of disk revolutions per second. The default value is 60 revolutions per second.

nbpi The density of inodes in the file system specified as the number of bytes per inode. The default value is 6144 bytes per inode.

This number should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used; if more inodes are desired, a smaller number should be used.

Note: The number of inodes that will be created in each cylinder group of a file system is approximately the size of the cylinder group divided by the number of bytes per inode, up to a limit of 2048 inodes per cylinder group. If the size of the cylinder group is large enough to reach this limit, the default number of bytes per inode will be increased.

Prototype File Structure

A prototype file describes the initial file structure of a new file system. The file contains tokens separated by spaces or newline characters. It cannot contain comments.

The first token is the name of a file to be copied onto block zero as the bootstrap program (usually **/etc/BOOT**). If the file name is **""**, no bootstrap code is placed on the device. The second token is a number specifying the number of **DEV_BSIZE** blocks in the file system.

The next three tokens specify the mode, user ID, and group ID of the root directory of the new file system, followed by the initial contents of the root directory in the format described for a directory file below, and terminated with a **$** token.

A file specification consists of four tokens giving the name, mode, user ID, and group ID, and an initial contents field. The syntax of the initial contents field depends on the mode.

A name token is a file name that is valid for the file system. The root directory does not have a name token.

A mode token is a 6-character string. The first character specifies the type of the file. It can be one of the following characters:

- Regular file

b	Block special file
c	Character special file
d	Directory
l	Symbolic link
L	Hard link

The second character of a mode token is either **u** or **-** to specify set-user-ID mode or not. The third character of a mode token is either **g** or **-** to specify the set-group-ID mode or not. The rest of a mode token is a three-digit octal number giving the *owner*, *group*, and *other* read, write, and execute permissions (see *chmod*(1)).

The user-ID and group-ID tokens define the owner of the file. These values can be specified numerically or with symbolic names that appear in the current password and group databases.

Regular file. The initial contents field is the path name of an existing file in the current file system whose contents and size are copied to the new file.

Block or character special file. The initial contents field is two numeric tokens that specify the major and minor device numbers.

Directory file. The initial contents field is a list of file specifications for the entries in the directory. The list is terminated with a **$** token. Directories can be nested. For each directory, the **mkfs** command automatically makes the **.** and **..** entries.

Symbolic link. The initial contents field is a path name that is used as the path to which the symbolic link should point.

Hard link. The initial contents field is a path name that is used as the name of a file within the new file system to which the entry should be linked. The mode, user-ID and group-ID tokens of this entry are ignored; they are taken from the target of the link. The target of the link must be listed before the entry specifying the link. Hard links to directories are not permitted.

With the exception of the permissions field of the mode token (which is always an octal number), all numeric fields can be specified in hexadecimal (using a leading **0x**), octal (using a leading **0**), or decimal.

Here is a sample prototype specification. The indentation clarifies the directory recursion.

```
/etc/BOOT
12288
d--555 bin  bin
sbin    d--755 bin  bin
        init     ---555 bin  bin /sbin/init
        savecore ---555 bin  bin /sbin/savecore
        $
dev     d--555 bin  bin
        b0       b--640 root sys 0 0x0e0000
        c0       c--640 root sys 4 0x0e0000
        $
etc     d--755 bin  bin
        init     l--777 bin  bin /sbin/init
        passwd   ---444 bin  bin /etc/passwd
        group    ---444 bin  bin /etc/group
        $
usr     d--755 bin  bin
        bin     d--755 bin  bin
                sh       ---555 bin  bin  /usr/bin/sh
                rsh      L--555 bin  bin  /usr/bin/sh
                su       -u-555 root bin  /usr/bin/su
                mailq    l--777 bin  bin  /usr/sbin/sendmail
                $
        sbin    d--755 bin  bin
                sendmail -ug555 root  mail /usr/sbin/sendmail
                $
        $
$
```

m

Access Control Lists

Every file with one or more optional ACL entries consumes an extra (continuation) inode. If you anticipate significant use of ACLs on a new file system, you can allocate more inodes by reducing the value of *nbpi* appropriately. The small default value typically causes allocation of many more inodes than are actually necessary, even with ACLs. To evaluate your need for extra inodes, run the **bdf -i** command on existing file systems. For more information on access control lists, see *acl*(5).

EXAMPLES

Execute the **mkfs** command to create a 32MB HFS file system on the non-LVM disk **/dev/dsk/c1t2d0**:

```
mkfs -F hfs /dev/dsk/c1t2d0 32768
```

Display the command that was used to construct the file system on **/dev/dsk/c1t2d0**:

```
mkfs -F hfs -m /dev/dsk/c1t2d0
```

Create an HFS file system within a logical volume **/dev/vg01/my_lvol** of a size equal to the size of **my_lvol**:

```
mkfs -F hfs /dev/vg01/my_lvol
```

WARNINGS

The old **-F** option, from prior releases of *mkfs*(1M), is no longer supported.

mkfs_hfs(1M) cannot be executed specifying creation of a file system on a whole disk if that disk was previously used as an LVM disk. If you wish to do this, use *mediainit*(1) to reinitialize the disk first.

The **-o largefile** option should be used with care, since older applications will not react correctly when confronted with large files.

AUTHOR

mkfs was developed by HP and the University of California, Berkeley.

FILES

/var/adm/sbtab	List of locations of the superblocks for the created file system. The **mkfs** command appends entries to this file.

SEE ALSO

chmod(1), bdf(1M), df(1M), fsadm_hfs(1M), fsck(1M), fsck_hfs(1M), fsclean(1M), mkfs(1M), mount_hfs(1M), newfs(1M), newfs_hfs(1M), dir(4), fs(4), fstab(4), group(4), passwd(4), symlink(4), acl(5).

STANDARDS CONFORMANCE

mkfs: SVID3

NAME

mkfs (vxfs) - construct a VxFS file system

SYNOPSIS

/usr/sbin/mkfs [**-F vxfs**] [**-V**] **-m** *special*

/usr/sbin/mkfs [**-F vxfs**] [**-V**]
[**-o** [**N**] [**X**] [**ninode=***n*] [**nau=***n*] [**bsize=***n*] [**logsize=***n*] [**ausize=***n*] [**aufirst=***n*] [**aupad=***n*] [**version=***n*] [**inosize=***n*] [**largefiles**|**nolargefiles**]] *special size*

DESCRIPTION

The **mkfs** command creates a VxFS file system by writing on the *special* device file. *special* must be the first argument after the options are given. The file system is created based on the *options* and *size* specified on the command line. The *size* specifies the number of sectors in the file system. By default, size is specified in units of DEV_BSIZE sectors. However, the letter **k**, **m**, or **g** can be appended to the number to indicate that the value is in kilobytes, megabytes, or gigabytes, respectively. The **mkfs** command builds a file system with a root directory and a **lost+found** directory.

Options

mkfs recognizes the following options:

-F vxfs Specify the VxFS file system type.

-m Display the command line which was used to create the file system. The file system must already exist. This option provides a means of determining the command used in constructing the file system.

-o *specific_options*
Specify options specific to the VxFS file system type. *specific_options* is a comma separated list of suboptions and/or keyword/attribute pairs intended for the VxFS-specific module of the command.

The following *specific_options* are valid on a VxFS file system:

N Do not write the file system to the *special* file. This option gives all the information needed to create a file system but does not create it.

X Create a file system in a file. This is used for debugging purposes only.

version=*n*
n is the VxFS disk layout version number. *n* can be 2 or 3 to indicate the Version 2 or Version 3 disk layout. Version 2 supports dynamic inode allocation. Version 3 adds support for large files and large *UID*s. The default is the Version 3.

inosize=*n*
n is the on-disk inode structure size for files on the file system. The only allowed value is 256 bytes.

bsize=*n*
n is the block size for files on the file system and represents the smallest amount of disk space that will be allocated to a file. *n* must be a power of 2 selected from the range 1024 to 8192. The default is 1024.

ninode=*n*
n is the maximum number of inodes in the file system. The actual maximum number of inodes is *n* rounded up to an appropriate boundary. For a Version 2 or 3 disk layout this is the maximum number of inodes, The number 0 and the string "**unlimited**" are interpreted to mean that the number of inodes is unlimited. The default is "unlimited" for a Version 2 or 3 disk layout.

nau=*n*
n is the number of allocation units on the file system. If *nau* is specified, then *ausize* is determined by evenly dividing the sectors among the allocation units. By default, the number of allocation units will be set based on the value of *ausize*. This option is ignored for a Version 3 disk layout.

ausize=*n*
n is the size, in blocks of size *bsize*, of an allocation unit. This is an alternate way of specifying the number of allocation units. This option may not be used in

conjunction with the *nau* option. With this option, the last allocation unit on the file system may be shorter than the others. If the last allocation unit on the file system is not long enough to contain an entire allocation unit header, the resulting size of the file system will be to the end of the last complete allocation unit. This parameter may not exceed 262144 blocks.

The algorithm used to choose the default value is rather complicated, but is intended to balance the number of allocation units (4 to 16 is a good range), the size of the allocation units (at least 32768 blocks), and other factors. For a Version 3 disk layout the allocation unit size is fixed at 32768 blocks, and this option is ignored.

aufirst=*n*
n is the starting block number, in blocks of size *bsize*, of the first allocation unit. This option allows the allocation units to be aligned to a particular boundary, such as a cylinder boundary. For a Version 3 file system, **aufirst** is always 0, and this option is ignored.

aupad=*n*
n is the size, in blocks of size *bsize*, to leave between the end of the inode list and the first data block in each allocation unit. This option allows the data blocks of an allocation unit to be aligned to a particular boundary, such as a cylinder boundary. For a Version 3 file system, **aupad** is always 0, and this option is ignored.

logsize=*n*
n is the number of blocks to allocate for an activity logging area. *n* must be in the range 32 blocks to 16384 Kbytes. Although *logsize* is specified in blocks, the maximum value is 16384 Kbytes. This means that for a *bsize* of 1024, 2048, 4096, or 8192 bytes the maximum value of *logsize* is 16384, 8192, 4096, or 2048 blocks, respectively. To avoid wasting space, the default *logsize* is 1024 blocks for a file system 8 megabytes or larger, 128 blocks for a file system 2 megabytes or larger but less than 8 megabytes, and 32 blocks for a file system less than 2 megabytes.

largefiles|nolargefiles
Controls the *largefile compatibility bit* for the file system. By default the bit is not set, and files created on the file system will be limited to less than 2 gigabytes in size. If **largefiles** is specified, the bit is set and the maximum file size for files created on the file system is not limited to 2 gigabytes (see *mount_vxfs*(1M) and *fsadm_vxfs*(1M)). This option is only valid for a Version 3 disk layout. The default is *nolargefiles*, although the default may change in the future.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

m

EXAMPLES

Execute the **mkfs** command to create a VxFS file system on **/dev/rdsk/c0t6d0**:

```
mkfs -F vxfs /dev/rdsk/c0t6d0 1024
```

Execute the **mkfs** command on a VxFS file system, **/dev/rdsk/c0t6d0**, to determine the command that was used to create the file system on **/dev/rdsk/c0t6d0**:

```
mkfs -F vxfs -m /dev/rdsk/c0t6d0
```

WARNINGS

mkfs_vxfs(1M) cannot be executed on a device that belonged to a logical volume group, unless the device is initialized by *mediainit*(1).

The **-o largefiles** option should be used with care, since older applications will not react correctly when confronted with large files.

RETURN VALUE

Upon successful completion, the **mkfs** command returns a value of 0. The return value is 1 if a syntax error occurs. Other errors return a value of 32.

FILES

`/etc/fstab`	Default list of file systems to check.

SEE ALSO

chmod(1), df(1M), bdf(1M), fsadm_vxfs(1M), fsck(1M), mount_vxfs(1M), newfs(1M), chown(2), group(4), passwd(4), mkfs(1M).

STANDARDS CONFORMANCE

`mkfs` : SVID3

NAME

mklost+found - make a lost+found directory for *fsck*(1M)

SYNOPSIS

`/usr/sbin/mklost+found`

DESCRIPTION

The **`mklost+found`** command creates a directory named **`lost+found`** in the current directory. It also creates several empty files which are then removed to provide empty slots for the **`fsck`** command (see *fsck*(1M)).

For an HFS file system, the **`mklost+found`** command is not normally needed since the **`mkfs`** command automatically creates the **`lost+found`** directory when a new file system is created (see *mkfs*(1M)).

AUTHOR

`mklost+found` was developed by the University of California, Berkeley.

SEE ALSO

fsck(1M), mkfs(1M).

NAME

mknod - create special files

SYNOPSIS

/sbin/mknod *name* **c** *major minor*

/sbin/mknod *name* **b** *major minor*

/sbin/mknod *name* **p**

DESCRIPTION

The **mknod** command creates the following types of files:

- Character device special file (first SYNOPSIS form),
- Block device special file (second SYNOPSIS form),
- FIFO file, sometimes called a named pipe (third SYNOPSIS form).

name is the path name of the file to be created. The newly created file has a default mode that is readable and writable by all users (0666), but the mode is modified by the current setting of the user's file mode creation mask (see *umask*(1)).

Character and Block Special Files

Character device special files are used for devices that can transfer single bytes at a time, such as nine-track magnetic tape drives, printers, plotters, disk drives operating in "raw" mode, and terminals. To create a character special file, use the **c** argument.

Block device special files are used for devices that usually transfer a block of data at a time, such as disk drives. To create a block device special file, use the **b** argument.

The remaining arguments specify the device that will be accessible through the new special file:

major	The major number specifies the major device type (for example, the device driver number).
minor	The minor number specifies the device location, which is typically, but not always, the unit, drive, HP-IB bus address and/or line number.

The *major* and *minor* values can each be specified in hexadecimal, octal, or decimal, using C language conventions (decimal: no leading zero; octal: leading zero; hexadecimal: leading **0x**).

The assignment of major and minor device numbers is specific to each HP-UX system. Refer to the System Administrator manuals supplied with your system for details.

Only users who have appropriate privileges can use **mknod** to create a character or block device special file.

FIFO files

To create a FIFO (named pipe or buffer) file, use the **p** argument. You can also use the **mkfifo** command for this purpose (see *mkfifo*(1)). All users can use **mknod** to create FIFO files.

WARNINGS

Access Control Lists

Optional ACL entries can be added to special files and FIFOs with the **chacl** command (see *chacl*(1)). However, system programs are likely to silently change or eliminate the optional ACL entries for these files.

SEE ALSO

chacl(1), mkdir(1), mkfifo(1), umask(1), lsdev(1M), sam(1M), mknod(2), acl(5), mknod(5).

HP-UX System Administrator manuals.

STANDARDS CONFORMANCE

mknod: SVID2, SVID3, XPG2

NAME

mkpdf - create a Product Description File from a prototype PDF

SYNOPSIS

mkpdf [**-c** *comment_string*] [**-n**] [**-r** *alternate_root*] *prototype_PDF new_PDF*

DESCRIPTION

The **mkpdf** program reads a prototype PDF and generates a new PDF (see *pdf*(4)) that reflects the current status of the file system files defined by path names in the prototype file.

If *pathname* is a directory, the *size*, *version*, *checksum*, and *linked_to* target fields are forced to be empty. If the file is a device, the *version*, *checksum*, and *linked_to* fields are forced to be empty and the *size* field contains the major and minor device numbers.

If a path name in *prototype_PDF* is prefaced with a question mark (**?**), the file is assumed to be an optional file. This file is processed in the same manner as all other files except that, if the file does not exist, values provided in the prototype are reproduced, and the **?**, is passed through to *new_PDF*. If a path name is not preceded with **?**, and the file does not exist on the file system, an error is reported and no entry is added to *new_PDF*.

If a dash (**-**) is used for *prototype_PDF* or *new_PDF*, **mkpdf** assumes that standard input and/or standard output, respectively, is being used for the appropriate value.

Comments in *prototype_PDF* are supported as follows: Lines beginning with the percent character (**%**) are generally passed through, in order, to *new_PDF*, except that any "**% Product Description File**" and "**% total size is ...**" lines are removed to prevent duplication of these automatically generated lines in *new_PDF* when *prototype_PDF* is a PDF. Lines beginning with a pound character (**#**), and lines containing only the newline character (**\n**) are not passed through to *new_PDF*. Note that blank space preceding these special characters is not allowed and will generally result in error messages about files not found.

A size summary is produced as a comment at the end of the PDF.

m

Options

-c *comment_string*	Insert a string that contains a comment about the product for which this PDF is being generated. This is used as a second comment line of the PDF. See *pdf*(4) for a description of the first comment line. If this option is not specified, no second comment line is produced.
-n	Record numerical representation of user ID from **/etc/passwd** and group ID from **/etc/group** for each file instead of the usual text representation.
-r *alternate_root*	Prefix the string *alternate_root* to each path name in the prototype (after removing the optional **?**) to form a modified path name to be used to gather attributes for the entry. Default is an empty string.

EXAMPLES

Given a file **Proto** with contents:

```
/usr/bin/basename
/usr/bin/cat
/usr/bin/ccat
/usr/bin/dirname
/usr/bin/grep
/usr/bin/ls
/usr/bin/ll:::::::/usr/bin/ls
/usr/bin/su
```

the command:

```
mkpdf -c "fileset TEST, Release 1.0" Proto -
```

produces the PDF shown in the EXAMPLE section of *pdf*(4).

The following example creates a totally new PDF for the fileset **ALBA_CORE**. The *pathname* and *linked_to* are taken from the prototype PDF. All other fields are generated from the file system.

```
mkpdf /tmp/ALBA_CORE /system/ALBA_CORE/new.pdf
```

The next example shows how to create a completely new PDF from just a list of files. The PDF for the files under the **/PRODUCT** directory is created by executing the **find** command (see *find*(1)) on all the files in the directory structure under **/PRODUCT**. A **/** is edited onto the beginning of each path name to make it absolute. The path names are then piped to **mkpdf**. The **-r** option specifies that a root of **/PRODUCT** should be prefixed to each path name while the directory is being searched. A **-** in the *prototype_PDF* position specifies that **stdin** is being used for the prototype PDF file. The resulting PDF does not contain the **/PRODUCT** prefix. Note that, with only a list of path names, the *linked_to* field of linked files will not conform to the convention explained in *pdf*(4).

```
cd /PRODUCT
find * -print | sed -e 's:^:/:' |
mkpdf -r /PRODUCT - PDF
```

RETURN VALUE

Upon completion, **mkpdf** returns one of the following values:

0 Successful completion.

1 Nonoptional files in the prototype file were not found.

2 **mkpdf** encountered other problems.

DIAGNOSTICS

filename: **no such file or directory**

A nonoptional file was not found on the file system and will not appear in the new PDF.

WARNINGS

Sizes reported do not reflect blocks allocated to directories.

Use of PDFs is discouraged since this functionality is obsolete and is being replaced with Software Distributor (see *sd*(4)).

AUTHOR

mkpdf was developed by HP.

SEE ALSO

pdfck(1M), pdfdiff(1M), pdf(4).

NAME

mksf - make a special (device) file

SYNOPSIS

/sbin/mksf [**-C** *class* | **-d** *driver*] [**-D** *directory*] [**-H** *hw-path*] [**-I** *instance*] [**-q**|**-v**] [*driver-options*] [*special-file*]

/sbin/mksf [**-C** *class* | **-d** *driver*] [**-D** *directory*] [**-H** *hw-path*] **-m** *minor* [**-q**|**-v**] [**-r**] *special-file*

DESCRIPTION

The **mksf** command makes a special file in the devices directory, normally **/dev**, for an existing device, a device that has already been assigned an instance number by the system. The device is specified by supplying some combination of the **-C**, **-d**, **-H**, and **-I** options. If the options specified match a unique device in the system, **mksf** creates a special file for that device; otherwise, **mksf** prints an error message and exits. If required, **mksf** creates any subdirectories relative to the device installation directory that are defined for the resulting special file.

For most drivers, **mksf** has a set of built-in driver options, *driver-options*, and special-file naming conventions. By supplying some subset of the driver options, as in the first form above, the user can create a special file with a particular set of characteristics. If a *special-file* name is specified, **mksf** creates the special file with that special file name; otherwise, the default naming convention for the driver is used.

In the second form, the *minor* number and *special-file* name are explicitly specified. This form is used to make a special file for a driver without using the built-in driver options in **mksf**. The **-r** option specifies that **mksf** should make a character (raw) device file instead of the default block device file for drivers that support both.

m

Options

mksf recognizes the following options:

-C *class* Match a device that belongs to a given device class, *class*. Device classes can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in the files in the directory **/usr/conf/master.d**. This option is not valid for pseudo devices. This option cannot be used with **-d**.

-d *driver* Match a device that is controlled by the specified device driver, *driver*. Device drivers can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in the files in the directory **/usr/conf/master.d**. This option cannot be used with **-C**.

-D *directory* Override the default device installation directory **/dev** and install the special files in *directory* instead. *directory* must exist; otherwise, **mksf** displays an error message and exits. See WARNINGS.

-H *hw-path* Match a device at a given hardware path, *hw-path*. Hardware paths can be listed with the **ioscan** command (see *ioscan*(1M)). A hardware path specifies the addresses of the hardware components leading to a device. It consists of a string of numbers separated by periods (**.**), such as **52** (a card), **52.3** (a target address), and **52.3.0** (a device). If a hardware component is a bus converter, the following period, if any, is replaced by a slash (**/**) as in **2**, **2/3**, and **2/3.0**. This option is not valid for pseudo devices.

-I *instance* Match a device with the specified *instance* number. Instances can be listed with the **-f** option of the **ioscan** command (see *ioscan*(1M)). This option is not valid for pseudo devices.

-m *minor* Create the special file with the specified minor number *minor*. The format of *minor* is the same as that given in *mknod*(1M) and *mknod*(5).

-q Quiet option. Normally, **mksf** displays a message as each driver is processed. This option suppresses the driver message, but not error messages. See the **-v** option.

-r Create a character (raw) special file instead of a block special file.

-v Verbose option. In addition to the normal processing message, display the name of each special file as it is created. See the **-q** option.

Naming Conventions

Many special files are named using the **c***card***t***target***d***device* naming convention. These variables have the following meaning wherever they are used.

card	The unique interface card identification number from **ioscan** (see *ioscan*(1M)). It is represented as a decimal number with a typical range of 0 to 255.
target	The device target number, for example the address on a HP-FL or SCSI bus. It is represented as a decimal number with a typical range of 0 to 15.
device	A address unit within a device, for example, the unit in a HP-FL device or the LUN in a SCSI device. It is represented as a decimal number with a typical range of 0 to 15.

Special Files

The driver-specific options (*driver-options*) and default special file names (*special-file*) are listed below.

asio0 sastty

-a *access-mode*
Port access mode (0-2). The default access mode is 0 (Direct connect). The *access-mode* meanings are:

access-mode	**Port Operation**
0	Direct connect
1	Dial out modem
2	Dial in modem

-c CCITT.

-f Hardware flow control (RTS/CTS).

-i Modem dialer. Cannot be used with **-l.**

-l Line printer. Cannot be used with **-i.**

-p *port* Multiplexer port number (0 for **asio0**; 0–1 for **sastty**). The default port number is 0.

-r *fifo-trigger* *fifo-trigger* should have a value between 0 and 3. The following table shows the corresponding FIFO trigger level for a given *fifo-trigger* value.

fifo-trigger	**Receive FIFO Trigger Level**
0	1
1	4
2	8
3	14

-t Transparent mode (normally used by diagnostics).

-x *xmit-limit* *xmit-limit* should have a value between 0 and 3. The following table shows the corresponding transmit limit for a given *xmit-limit* value.

xmit-limit	**Transmit Limit**
0	1
1	4
2	8
3	12

special-file The default special file name depends on the *access-mode* and whether the **-i** and **-l** options are used.

access-mode	**-i**	**-l**	**Special File Name**
—	no	yes	**c***card***p0_lp**
2	no	no	**ttyd***card***p0**
1	no	no	**cul***card***p0**
0	yes	no	**cua***card***p0**
0	no	no	**tty***card***p0**

audio

-f *format* Audio format (0-3). The *format* meanings are:

format	**Audio Format**	**File Name Modifier** *format-mod*
0	No change in audio format	
1	8-bit Mu-law	**U**
2	8-bit A-law	**A**
3	16-bit linear	**L**

-o *output-dest*

Output destination (0-4). The *output-dest* should have a value between 0 and 4. The following table shows the corresponding output destinations for a given *output-dest* value.

output-dest	**Output Destinations**	**File Name Modifier** *output-mod*
0	All outputs	**B**
1	Headphone	**E**
2	Internal Speaker	**I**
3	No output	**N**
4	Line output	**L**

-r Raw, control access. This option cannot be used with either the **-f** or **-o** options.

special-file The default special file name depends on the options specified.

Options	**Special File Name**
-r	**audioCtl**_*card*
-f 0	**audio**_*card*
all others	**audio***output-modformat-mod_card*

The optional *output-mod* and *format-mod* values are given in the tables above. Note the underscore (_) before *card* in each special file name. Also note that for *card* **0**, each file will be linked to a simpler name without the trailing *_card*.

autox0 schgr

Note that **-i** cannot be used with either **-r** or **-p**.

-i Ioctl; create picker control special file.

-p *optical-disk*[**:***last-optical-disk*]

The optical disk number (starts with 1). If the optional **:***last-optical-disk* is given then special files for the range of disks specified will be created.

-r Raw; create character, not block, special file.

special-file A special file cannot be given if a range of optical disks is given with the **-p** option. If one is given for the single disk case, the name will have an **a** appended to the end for the A-side device and a **b** appended to the end for the B-side device. The default special file name depends on whether the **-r** option is used.

-r	**Special File Name**
yes	**rac/c***card***t***target***d***device_optical-disk***a** **rac/c***card***t***target***d***device_optical-disk***b**
no	**ac/c***card***t***target***d***device_optical-disk***a** **ac/c***card***t***target***d***device_optical-disk***b**

Note the underscore (_) between *device* and *optical-disk*.

CentIf

-h *handshake-mode*
Handshake mode. Valid values range from 1 to 6:

handshake-mode	**Handshake operation**
1	Automatic NACK/BUSY handshaking
2	Automatic BUSY only handshaking
3	Bidirectional read/write
4	Stream mode (NSTROBE only, no handshaking)
5	Automatic NACK/BUSY with pulsed NSTROBE
6	Automatic BUSY with pulsed NSTROBE

special-file The default special file name is **c***card***t0d0_lp** for *handshake-mode* **2** and **c***card***t0d0h***handshake-mode***_lp** for all others.

consp1

-r *fifo-trigger* *fifo-trigger* should have a value between 0 and 3. The following table shows the corresponding FIFO trigger level for a given *fifo-trigger* value.

fifo-trigger	**Receive FIFO Trigger Level**
0	1
1	4
2	8
3	14

-t Transparent mode (normally used by diagnostics).

-x *xmit-limit* *xmit-limit* should have a value between 0 and 3. The following table shows the corresponding transmit limit for a given *xmit-limit* value.

xmit-limit	**Transmit Limit**
0	1
1	4
2	8
3	12

special-file The default special file name is as follows:

Special File Name
tty*card***p0**

disc1

-c This option must be present if the unit is a cartridge tape.

-r Raw; create character, not block, special file.

-s *section* The section number.

-t Transparent mode (normally used by diagnostics).

-u *unit* The CS/80 unit number (for example, unit 0 for disk, unit 1 for tape).

special-file The default special file name depends on whether the **-c**, **-r**, and **-s** options are used:

`-c`	`-r`	`-s`	**Special File Name**
yes	yes	invalid	**`rct/c`***card***`t`***target***`d`***device*
no	yes	no	**`rdsk/c`***card***`t`***target***`d`***device*
no	yes	yes	**`rdsk/c`***card***`t`***target***`d`***device***`s`***section*
yes	no	invalid	**`ct/c`***card***`t`***target***`d`***device*
no	no	no	**`dsk/c`***card***`t`***target***`d`***device*
no	no	yes	**`dsk/c`***card***`t`***target***`d`***device***`s`***section*

`disc2`

`-r`	Raw; create character, not block, special file.
`-s` *section*	The section number.
`-t`	Transparent mode (normally used by diagnostics).
`-u` *unit*	The cs80 unit number (typically 0).
special-file	The default special file name depends on whether the **`-r`** and **`-s`** options are used:

`-r`	`-s`	**Special File Name**
yes	no	**`rdsk/c`***card***`t`***target***`d`***device*
yes	yes	**`rdsk/c`***card***`t`***target***`d`***device***`s`***section*
no	no	**`dsk/c`***card***`t`***target***`d`***device*
no	yes	**`dsk/c`***card***`t`***target***`d`***device***`s`***section*

`disc3`

`-f`	Floppy.
`-r`	Raw; create character, not block, special file.
`-s` *section*	The section number.
special-file	The default special file name depends on whether the **`-r`** and **`-s`** options are used:

`-r`	`-s`	**Special File Name**
yes	no	**`rdsk/c`***card***`t`***target***`d`***device* and **`rfloppy/c`***card***`t`***target***`d`***device*
yes	yes	**`rdsk/c`***card***`t`***target***`d`***device***`s`***section*
no	no	**`dsk/c`***card***`t`***target***`d`***device* and **`floppy/c`***card***`t`***target***`d`***device*
no	yes	**`dsk/c`***card***`t`***target***`d`***device***`s`***section*

`disc4 sdisc`

`-r`	Raw; create character, not block, special file.
`-s` *section*	The section number.
special-file	The default special file name depends on whether the **`-r`** and **`-s`** options are used:

`-r`	`-s`	**Special File Name**
yes	no	**`rdsk/c`***card***`t`***target***`d`***device*
yes	yes	**`rdsk/c`***card***`t`***target***`d`***device***`s`***section*
no	no	**`dsk/c`***card***`t`***target***`d`***device*
no	yes	**`dsk/c`***card***`t`***target***`d`***device***`s`***section*

`instr0`

`-a` *address*	The HP-IB instrument address (0-30). Cannot be used with the **`-t`** option.
`-t`	Transparent mode (normally used by diagnostics). Cannot be used with the **`-a`** option.
special-file	The default special file name depends on the arguments **`-a`** and **`-t`**:

m

-a	**-t**	**Special File Name**
no	no	**hpib/c***card*
no	yes	**diag/hpib/c***card*
yes	no	**hpib/c***card***t***target***d***address*

hil

Note that only one of **-a**, **-k**, or **-r** is allowed.

-a *address*	The link address (1-7).
-k	Cooked keyboard.
-n	The hil controller device.
special-file	The default special file name depends on the **-a**, **-k**, and **-r** options:

Option	**Special File Name**
-a	**hil_***card***.***address*
-k	**hilkbd_***card*
-r	**rhil_***card*

Note the underscore (_) before *card*. Also note that for *card* **0**, each file will be linked to a simpler name without *_card*, either **hil***address*, **hilkbd**, or **rhil**.

lan0 lan1 lan2 lan3

Note that only one of **-e** or **-i** is allowed.

-e	Ethernet protocol.
-i	IEEE 802.3 protocol.
-t	Transparent mode (normally used by diagnostics).
special-file	The default special file name depends on the **-e**, **-i**, and **-t** options:

Option	**-t**	**Special File Name**
-e	no	**ether***card*
-e	yes	**diag/ether***card*
-i	no	**lan***card*
-i	yes	**diag/lan***card*

lantty0

-e	Exclusive access.
special-file	The default special file name depends on whether the **-e** option is used:

-e	**Special File Name**
no	**lantty***card*
yes	**diag/lantty***card*

lpr0 lpr1 lpr2 lpr3

-c	Capital letters. Convert all output to uppercase.
-e	Eject page after paper-out recovery.
-n	No form-feed.
-o	Old paper-out behavior (abort job).
-r	Raw.
-t	Transparent mode (normally used by diagnostics).
-w	No wait. Don't retry errors on open.
special-file	The default special file name depends on whether the **-r** option is used:

-r	**Special File Name**
no	**c***card***t***target***d***device*_**lp**
yes	**c***card***t***target***d***device*_**rlp**

mux0 mux2 mux4 eisa_mux0 pci_mux0

-a *access-mode*
Port access mode (0-2). The default access mode is 0 (Direct connect). The *access-mode* meanings are:

access-mode	**Port Operation**
0	Direct connect
1	Dial out modem
2	Dial in modem

-c CCITT.

-f Hardware flow control (RTS/CTS).

-i Modem dialer. Cannot be used with **-l.**

-l Line printer. Cannot be used with **-i.**

-p *port* Multiplexer port number (0–15 for **mux0** and **mux2**; 0–1 for **mux4**; a1 - a16, b1 - b16, c1 - c16 & etc for the **eisa_mux0** or **pci_mux0**). Some MUX cards controlled by a particular driver have fewer than the maximum supported ports.

-t Transparent mode (normally used by diagnostics).

special-file The default special file name depends on the *access-mode* and whether the **-i** and **-l** options are used. The term "card" below refers to the Instance number of the mux card.

access-mode	**-i**	**-l**	**Special File Name**
—	no	yes	**c***card***p***port*_**lp**
2	no	no	**ttyd***card***p***port*
1	no	no	**cul***card***p***port*
0	yes	no	**cua***card***p***port*
0	no	no	**tty***card***p***port*

pflop sflop

-r Raw; create character, not block, special file.

special-file The default special file name depends on whether the **-r** option is used:

-r	**Special File Name**
no	**floppy/c***card***t***target***d***device*
yes	**rfloppy/c***card***t***target***d***device*

ps2

Note that only one of **-a**, or **-p** is allowed.

-a *auto_device*
Autosearch device. An *auto_device* value of 0 means first mouse; a value of 1 means first keyboard.

-p *port* PS2 port number.

special-file The default special file name depends on the **-a**, and **-p** options:

Option	**Special File Name**
-a 0	**ps2mouse**
-a 1	**ps2kbd**
-p	**ps2**_*port*

Note the underscore (_) before *port*.

sastty See **asio0**.

scc1

m

-a *access-mode*
Port access mode (0–2). The default access mode is 0. The *access-mode* meanings are:

access-mode	**Port Operation**
0	Direct connect
1	Dial out modem
2	Dial in modem

-b Port B.

-c CCITT.

-i Modem dialer. Cannot be used with **-l.**

-l Line printer. Cannot be used with **-i**.

special-file The default special file name depends on the *access-mode* and whether the **-i** and **-l** options are used.

access-mode	**-i**	**-l**	**Special File Name**
—	no	yes	**c***card***p***port*_**lp**
2	no	no	**ttyd***card***p***port*
1	no	no	**cul***card***p***port*
0	yes	no	**cua***card***p***port*
0	no	no	**tty***card***p***port*

schgr See **autox0**.

sdisk See **disc4**.

sflop See **pflop**.

stape

-a AT&T-style rewind/close.

-b *bpi* Bits per inch or tape density. The recognized values for *bpi* are:
BEST, **D1600**, **D3480**, **D3480C**, **D6250**, **D6250C**, **D800**, **D8MM_8200**, **D8MM_8200C**, **D8MM_8500**, **D8MM_8500C**, **DDS1**, **DDS1C**, **DDS2**, **DDS2C**, **NOMOD**, **QIC_1000**, **QIC_11**, **QIC_120**, **QIC_1350**, **QIC_150**, **QIC_2100**, **QIC_24**, **QIC_2GB**, **QIC_525**, **QIC_5GB**,
or a decimal number density code.

-c [*code*] Compression with optional compression code. The optional decimal code is used to select a particular compression algorithm on drives that support more than one compression algorithm. This option must be specified at the end of an option string. See *mt*(7) for more details.

-e Exhaustive mode. This option allows the driver to experiment with multiple configuration values in an attempt to access the media. The default behavior is to use only the configuration specified.

-n No rewind on close.

-p Partition one.

-s [*block-size*] Fixed block size mode. If a numeric *block-size* is given, it is used for a fixed block size. If the **-s** option is used alone, a device-specific default fixed block size is used. This option must be specified at the end of an option string.

-u UC Berkeley-style rewind/close.

-w Wait (disable immediate reporting).

-x *index* Use the *index* value to access the tape device driver property table entry. Recognized values for *index* are decimal values in the range 0 to 30.

special-file Put all tape special files in the **/dev/rmt** directory. This is required for proper maintenance of the Tape Property Table (see *mt*(7)). Device files located outside the **/dev/rmt** directory may not provide consistent behavior across system reboots. The default special file names are dependent on the tape drive being accessed and the options specified. All default special files begin with **rmt/c***card***t***target***d***device*. See *mt*(7) for a complete description of the default special file naming scheme for tapes.

tape1 tape2

-a	AT&T-style rewind/close.
-b *bpi*	Bits per inch or tape density. The recognized values for *bpi* are: **BEST**, **D1600**, **D3480**, **D3480C**, **D6250**, **D6250C**, **D800**, **D8MM_8200**, **D8MM_8200C**, **D8MM_8500**, **D8MM_8500C**, **DDS1**, **DDS1C**, **DDS2**, **DDS2C**, **NOMOD**, **QIC_1000**, **QIC_11**, **QIC_120**, **QIC_1350**, **QIC_150**, **QIC_2100**, **QIC_24**, **QIC_2GB**, **QIC_525**, **QIC_5GB**, **DLT_42500_24**, **DLT_42500_56**, **DLT_62500_64**, **DLT_81633_64**, **DLT_62500_64C**, **DLT_81633_64C**, or a decimal number density code.
-c [*code*]	Compression with optional compression code. The optional decimal code is used to select a particular compression algorithm on drives that support more than one compression algorithm. This option must be specified at the end of an option string. See *mt*(7) for more details.
-n	No rewind on close.
-o	Console messages disabled.
-t	Transparent mode, normally used by diagnostics.
-u	UC Berkeley-style rewind/close.
-w	Wait (disable immediate reporting).
-x *index*	Use the index value to access the tape device driver property table entry. The recognized values for *index* are decimal values in the range 0 to 30.
-z	RTE compatible close.
special-file	Put all tape special files in the **/dev/rmt** directory. This is required for proper maintenance of the Tape Property Table (see *mt*(7)). Device files located outside the **/dev/rmt** directory may not provide consistent behavior across system reboots. The default special file names are dependent on the tape drive being accessed and the options specified. All default special files begin with **rmt/c***card***t***target***d***device*. See *mt*(7) for a complete description of the default special file naming scheme for tapes.

m

RETURN VALUE

mksf exits with one of the following values:

0	Successful completion.
1	Failure. An error occurred.

DIAGNOSTICS

Most of the diagnostic messages from **mksf** are self-explanatory. Listed below are some messages deserving further clarification. Errors cause **mksf** to abort immediately.

Errors

Ambiguous device specification

Matched more than one device in the system. Use some combination of the **-d**, **-C**, **-H**, and **-I** options to specify a unique device.

No such device in the system

No device in the system matched the options specified. Use **ioscan** to list the devices in the system (see *ioscan*(1M)).

Device driver *name* **is not in the kernel**
Device class *name* **is not in the kernel**

The indicated device driver or device class is not present in the kernel. Add the appropriate device driver and/or device class to the **config** input file and generate a new kernel (see *config*(1M)).

Device has no instance number

The specified device has not been assigned an instance number. Use **ioscan** to assign an *instance* to the device.

`Directory` *directory* **`doesn’t exist`**

The *directory* argument of the **`-D`** option doesn't exist. Use **`mkdir`** to create the directory (see *mkdir*(1)).

EXAMPLES

Make a special file named **`/dev/printer`** for the line printer device associated with instance number 2.

```
mksf -C printer -I 2 /dev/printer
```

Make a special file, using the default naming convention, for the tape device at hardware path 8.4.1. The driver-specific options specify 1600 bits per inch and no rewind on close.

```
mksf -C tape -H 8.4.1 -b D1600 -n
```

WARNINGS

Many commands and subsystems assume their device files are in **`/dev`**; therefore, the use of the **`-D`** option is discouraged.

AUTHOR

`mksf` was developed by HP.

FILES

`/dev/config`	I/O system special file
`/etc/mtconfig`	Tape driver property table database

SEE ALSO

mkdir(1), config(1M), insf(1M), ioscan(1M), lsdev(1M), mknod(1M), rmsf(1M), mknod(2), ioconfig(4), mknod(5), mt(7).

NAME

mount (generic), umount (generic) - mount and unmount file systems

SYNOPSIS

/usr/sbin/mount [**-l**] [**-p**|**-v**]

/usr/sbin/mount -a [**-F** *FStype*] [**-eQ**]

/usr/sbin/mount [**-F** *FStype*] [**-eQrV**] [**-o** *specific_options*] {*special*|*directory*}

/usr/sbin/mount [**-F** *FStype*] [**-eQrV**] [**-o** *specific_options*] *special directory*

/usr/sbin/umount [**-v**] [**-V**] {*special*|*directory*}

/usr/sbin/umount -a [**-F** *FStype*] [**-v**]

DESCRIPTION

The **mount** command mounts file systems. Only a superuser can mount file systems. Other users can use **mount** to list mounted file systems.

The **mount** command attaches *special*, a removable file system, to *directory*, a directory on the file tree. *directory*, which must already exist, will become the name of the root of the newly mounted file system. *special* and *directory* must be given as absolute path names. If either *special* or *directory* is omitted, **mount** attempts to determine the missing value from an entry in the **/etc/fstab** file. **mount** can be invoked on any removable file system, except **/**.

If **mount** is invoked without any arguments, it lists all of the mounted file systems from the file system mount table, **/etc/mnttab**.

The **umount** command unmounts mounted file systems. Only a superuser can unmount file systems.

Options (mount)

The **mount** command recognizes the following options:

-a Attempt to mount all file systems described in **/etc/fstab**. All optional fields in **/etc/fstab** must be included and supported. If the **-F** option is specified, all file systems in **/etc/fstab** with that *FStype* are mounted. If **noauto** is specified in an entry's option list, this entry is skipped. File systems are not necessarily mounted in the order listed in **/etc/fstab**.

-e Verbose mode. Write a message to the standard output indicating which file system is being mounted.

-F *FStype* Specify *FStype*, the file system type on which to operate. See *fstyp*(1M). If this option is not included on the command line, then it is determined from either **/etc/fstab**, by matching *special* with an entry in that file, or from file system statistics of *special*, obtained by **statfsdev()** (see *statfsdev*(3C)).

-l Limit actions to local file systems only.

-o *specific_options*
Specify options specific to each file system type. *specific_options* is a list of comma separated suboptions and/or keyword/attribute pairs intended for a *FStype*-specific version of the command. See the *FStype*-specific manual entries for a description of the *specific_options* supported, if any.

-p Report the list of mounted file systems in the **/etc/fstab** format.

-Q Prevent the display of error messages that result from an attempt to mount already mounted file systems.

-r Mount the specified file system as read-only. Physically write-protected file systems must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

-v Report the regular output with file system type and flags; however, the *directory* and *special* fields are reversed.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Options (umount)

The **umount** command recognizes the following options:

-a	Attempt to unmount all file systems described in **/etc/mnttab**. All optional fields in **/etc/mnttab** must be included and supported. If *FStype* is specified, all file systems in **/etc/mnttab** with that *FStype* are unmounted. File systems are not necessarily unmounted in the order listed in **/etc/mnttab**.
-F *FStype*	Specify *FStype*, the file system type on which to operate. If this option is not included on the command line, then it is determined from **/etc/mnttab** by matching *special* with an entry in that file. If no match is found, the command fails.
-v	Verbose mode. Write a message to standard output indicating which file system is being unmounted.
-V	Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

List the file systems currently mounted:

```
mount
```

Mount the HFS file system **/dev/dsk/c1t2d0** at directory **/home**:

```
mount -F hfs /dev/dsk/c1t2d0 /home
```

Unmount the same file system:

```
umount /dev/dsk/c1t2d0
```

AUTHOR

mount was developed by HP, AT&T, the University of California, Berkeley, and Sun Microsystems.

m

FILES

/etc/fstab	Static information about the systems
/etc/mnttab	Mounted file system table

SEE ALSO

fsadm(1M), mount_*FStype*(1M), umount_*FStype*(1M), setmnt(1M), mount(2), fstab(4), mnttab(4), fs_wrapper(5), quota(5).

STANDARDS CONFORMANCE

mount: SVID3

umount: SVID3

NAME

mount(cdfs), umount(cdfs) - mount and unmount an CDFS file systems

SYNOPSIS

/usr/sbin/mount [**-l**] [**-p**|**-v**]

/usr/sbin/mount -a [**-F cdfs**] [**-eQ**]

/usr/sbin/mount [**-F cdfs**] [**-eQrV**] [**-o** *specific_options*] {*special*|*directory*}

/usr/sbin/mount [**-F cdfs**] [**-eQrV**] [**-o** *specific_options*] *special directory*

/usr/sbin/umount -a [**-F cdfs**] [**-v**]

/usr/sbin/umount [**-v**] [**-V**] {*special*|*directory*}

DESCRIPTION

The **mount** command mounts file systems. Only a superuser can mount file systems. Other users can use **mount** to list mounted file systems.

The **mount** command attaches *special*, a removable file system, to *directory*, a directory on the file tree. *directory*, which must already exist, will become the name of the root of the newly mounted file system. *special* and *directory* must be given as absolute path names. If either *special* or *directory* is omitted, **mount** attempts to determine the missing value from an entry in the **/etc/fstab** file. **mount** can be invoked on any removable file system, except **/**.

If **mount** is invoked without any arguments, it lists all of the mounted file systems from the file system mount table, **/etc/mnttab**.

The **umount** command unmounts mounted file systems. Only a superuser can unmount file systems.

m

Options (mount)

mount recognizes the following options:

-a Attempt to mount all file systems described in **/etc/fstab**. All optional fields in **/etc/fstab** must be included and supported. If **-F cdfs** is specified, all CDFS file systems in **/etc/fstab** are mounted. If **noauto** is specified in an entry's option list, this entry is skipped. File systems are not necessarily mounted in the order listed in **/etc/fstab**.

-e Verbose mode. Write a message to standard output indicating which file system is being mounted.

-F cdfs Specify the CDFS file system type (see *fstyp*(1M)).

-l Limit actions to local file systems only.

-o *specific_options*
Specify options specific to the CDFS file system type. *specific_options* is a list of comma separated suboptions and/or keyword/attribute pairs intended for the CDFS specific module of the command.

The following *specific_options* are valid on CDFS file systems.

cdcase	Suppress the display of version numbers. Show and match file names as lower case.
defaults	Use all default options. When given, this must be the only option specified.
ro	Mount read-only (default).
suid	Allow set-user-ID execution (default).
nosuid	Do not allow set-user-ID execution.

-p Report the list of mounted file systems in the **/etc/fstab** format.

-Q Prevent the display of error messages resulting from an attempt to mount already mounted file systems.

-r Mount the specified file system as read-only. This option is equivalent to the **-o ro** *specific_option*. For CDFS file systems this is a default option.

-v	Report the regular output with file system type and flags; however, *directory* and *special* fields are reversed.
-V	Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Options (umount)

umount recognizes the following options:

-a	Attempt to unmount all file systems described in **/etc/mnttab**. All optional fields in **/etc/mnttab** must be included and supported. If **-F cdfs** is specified, all CDFS file systems in **/etc/mnttab** are unmounted. File systems are not necessarily unmounted in the order listed in **/etc/mnttab**.
-F cdfs	Specify the CDFS file system type (see *fstyp*(1M)).
-v	Verbose mode. Write a message to standard output indicating which file system is being unmounted.
-V	Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

DIAGNOSTICS

umount complains if the special file is not mounted or if it is busy. The file system is busy if it contains an open file or some logged-in user's working directory.

EXAMPLES

Mount a local CDFS disk:

```
mount -F cdfs /dev/dsk/c0t0d4 /cdrom
```

Unmount a local CDFS disk:

```
umount /dev/dsk/c0t0d4
```

WARNINGS

Some degree of validation is done on the file system, however, it is generally unwise to mount file systems that are defective, corrupt, or of unknown origin.

NOTES

Additional CD-ROM formats are supported using PFS (Portable File System) utilities. See *pfs*(4) for more details.

AUTHOR

mount was developed by HP, AT&T, the University of California, Berkeley, and Sun Microsystems.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table

SEE ALSO

fsclean(1M), mount(1M), quotaon(1M), mount(2), fstab(4), mnttab(4), pfs(4), fs_wrapper(5), quota(5).

STANDARDS CONFORMANCE

mount: SVID3

umount: SVID3

NAME

mount(hfs), umount(hfs) - mount and unmount an HFS file systems

SYNOPSIS

/usr/sbin/mount [**-l**] [**-p**|**-v**]

/usr/sbin/mount -a [**-F hfs**] [**-eQ**] [**-f**]

/usr/sbin/mount [**-F hfs**] [**-eQrV**] [**-f**] [**-o** *specific_options*] {*special*|*directory*}

/usr/sbin/mount [**-F hfs**] [**-eQrV**] [**-f**] [**-o** *specific_options*] *special directory*

/usr/sbin/umount -a [**-F hfs**] [**-v**]

/usr/sbin/umount [**-v**] [**-V**] {*special*|*directory*}

DESCRIPTION

The **mount** command mounts file systems. Only a superuser can mount file systems. Other users can use **mount** to list mounted file systems.

The **mount** command attaches *special*, a removable file system, to *directory*, a directory on the file tree. *directory*, which must already exist, will become the name of the root of the newly mounted file system. *special* and *directory* must be given as absolute path names. If either *special* or *directory* is omitted, **mount** attempts to determine the missing value from an entry in the **/etc/fstab** file. **mount** can be invoked on any removable file system, except **/**.

If **mount** is invoked without any arguments, it lists all of the mounted file systems from the file system mount table, **/etc/mnttab**.

The **umount** command unmounts mounted file systems. Only a superuser can unmount file systems.

Options (mount)

mount recognizes the following options:

-a Attempt to mount all file systems described in **/etc/fstab**. All optional fields in **/etc/fstab** must be included and supported. If **-F hfs** is specified, all HFS file systems in **/etc/fstab** are mounted. If **noauto** is specified in an entry's option list, this entry is skipped. File systems are not necessarily mounted in the order listed in **/etc/fstab**.

-e Verbose mode. Write a message to standard output indicating which file system is being mounted.

-f Force the file system to be mounted, even if the file system clean flag indicates that the file system should have **fsck** run on it before mounting (see *fsck*(1M)). This option is valid only on HFS file systems.

-F hfs Specify the HFS file system type (see *fstyp*(1M)).

-l Limit actions to local file systems only.

-o *specific_options*

Specify options specific to the HFS file system type. *specific_options* is a list of comma separated suboptions and/or keyword/attribute pairs intended for the HFS specific module of the command.

The following *specific_options* are valid on HFS file systems.

defaults	Use all default options. When given, this must be the only option specified.
rw	Mount read-write (default).
ro	Mount read-only.
suid	Allow set-user-ID execution (default).
nosuid	Do not allow set-user-ID execution.
behind	Enable, where possible, asynchronous writes to disk. This is the default on 700 systems.

delayed Enable delayed or buffered writes to disk. This is the default on 800 systems.

fs_async Enable relaxed posting of file system metadata.

no_fs_async
Enable rigorous posting of file system metadata. This is the default.

largefiles Attempt to enable the creation of files greater than 2 gigabytes in size. File systems have to be created or configured to enable large files (see *mkfs_hfs(1M)* and *fsadm_hfs(1M))*.

nolargefiles
Attempt to disable the creation of files greater than 2 gigabytes in size. File systems have to be created or configured to disable large files. (see *mkfs_hfs(1M)* and *fsadm_hfs(1M))*.

quota Enable disk quotas (valid only for **rw** file systems).

noquota Disable disk quotas (default).

Mounting with the **quota** option also enables quotas for the file system, unlike some other systems, which require the additional invocation of the **quotaon** command after the file system has been mounted (see *quotaon*(1M)). Running **quotaon** does no harm, but it is not necessary.

-p Report the list of mounted file systems in the **/etc/fstab** format.

-Q Prevent the display of error messages resulting from an attempt to mount already mounted file systems.

-r Mount the specified file system as read-only. This option is equivalent to the **-o ro** *specific_option*. Physically write-protected file systems must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

-v Report the regular output with file system type and flags; however, *directory* and *special* fields are reversed.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Options (umount)

umount recognizes the following options:

-a Attempt to unmount all file systems described in **/etc/mnttab**. All optional fields in **/etc/mnttab** must be included and supported. If **-F hfs** is specified, all HFS file systems in **/etc/mnttab** are unmounted. File systems are not necessarily unmounted in the order listed in **/etc/mnttab**.

-F hfs Specify the HFS file system type (see *fstyp*(1M)).

-v Verbose mode. Write a message to standard output indicating which file system is being unmounted.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

DIAGNOSTICS

umount complains if the special file is not mounted or if it is busy. The file system is busy if it contains an open file or some logged-in user's working directory.

EXAMPLES

Mount a local HFS disk:

m

```
mount -F hfs /dev/dsk/c0t0d4 /usr
```

Unmount a local HFS disk:

```
umount /dev/dsk/c0t0d4
```

WARNINGS

Some degree of validation is done on the file system, however, it is generally unwise to mount file systems that are defective, corrupt, or of unknown origin.

AUTHOR

mount was developed by HP, AT&T, the University of California, Berkeley, and Sun Microsystems.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table

SEE ALSO

fsclean(1M), mount(1M), mkfs_hfs(1M), fsadm_hfs(1M), quotaon(1M), mount(2), fstab(4), mnttab(4), fs_wrapper(5), quota(5).

STANDARDS CONFORMANCE

mount: SVID3

umount: SVID3

NAME

mount(lofs), umount(lofs) - mount and unmount an LOFS file system

SYNOPSIS

/usr/sbin/mount [**-p**|**-v**]

/usr/sbin/mount -a [**-F lofs**] [**-eQ**]

/usr/sbin/mount [**-F lofs**] [**-eQrV**] [**-o** *specific_options*] {*special_directory*|*directory*}

/usr/sbin/mount [**-F lofs**] [**-eQrV**] [**-o** *specific_options*] *special_directory directory*

/usr/sbin/umount [**-v**] [**-V**] {*special_directory*|*directory*}

/usr/sbin/umount -a [**-F lofs**] [**-v**]

DESCRIPTION

The **mount** command mounts LOFS file systems. Only superuser can mount LOFS file systems. Other users can use **mount** to list mounted file systems.

mount, attaches *special_directory*, a directory from one of the mounted file systems, to *directory*, an another directory in one of the mounted file systems. This enables new file systems to be created, which provide access to existing directories or file systems using alternate path names. Both *special_directory* and *directory* should already exist. *directory* will become the root of the newly mounted LOFS file system, containing the file system hierarchy under *special_directory*. *special_directory* and *directory* must be specified as absolute path names. If either *special_directory* or *directory* is omitted, **mount** attempts to determine the missing value from an entry in the **/etc/fstab** file. **mount** can be invoked on any removable file system, except **/**.

If **mount** is invoked without any arguments, it lists all the mounted file systems from the file system mount table, **/etc/mnttab**.

The **umount** command unmounts mounted file systems. Only a superuser can unmount file systems.

m

Options (mount)

mount recognizes the following options:

-a Attempt to mount all file systems described in **/etc/fstab**. All optional fields in **/etc/fstab** must be included and supported. If **-F lofs** is specified, all LOFS file systems in **/etc/fstab** are mounted. If **noauto** is specified in an entry's option list, this entry is skipped. File systems are not necessarily mounted in the order listed in **/etc/fstab**.

-e Verbose mode. Write a message to standard output indicating which file system is being mounted.

-F lofs Specify the LOFS file system type (see *fstyp*(1M)).

-l Limit actions to local file systems only. LOFS is a local file system.

-o *specific_options*
Specify options specific to the LOFS file system type. *specific_options* is a list of comma separated suboptions and/or keyword/attribute pairs intended for the LOFS specific module of the command.

The following *specific_options* are valid on an LOFS file system:

defaults Use all default options. When used, this must be the only option specified.

ro Read-only (see *WARNINGS* below).

-p Report the list of mounted file systems in the **/etc/fstab** format.

-Q Prevent display of error messages resulting from an attempt to mount already mounted file systems.

-r Mount the specified file system as read-only (see *WARNINGS* below).

-v Report the output in a new style. The new style has the file system type and flags displayed in addition to the old output. The *directory* and *special_directory* fields are reversed.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Options (umount)

The **umount** command recognizes the following options:

-a	Attempt to unmount all file systems described in **/etc/mnttab**. All optional fields in **/etc/mnttab** must be included and supported. If **-F lofs** file system type is specified, all the LOFS file systems in **/etc/mnttab** are unmounted. File systems are not necessarily unmounted in the order listed in **/etc/mnttab**.
-F lofs	Specify the LOFS file system type (see *fstyp*(1M)).
-v	Verbose mode. Write a message to standard output indicating which file system is being unmounted.
-V	Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Mount an LOFS file system:

```
mount /usr /tmp/usr
```

Mount another LOFS file system:

```
mount -F lofs /usr/sbin /tmp/sbin
```

WARNINGS

LOFS file systems provide the user with numerous applications; however, they may be potentially confusing. LOFS file systems should generally be created by an experienced user.

For LOFS file systems which are mounted read-only, if the underlying file system is mounted writable, certain write operations on the LOFS will succeed. Thus LOFS should not be relied upon to provide a strictly write-only alternative image of a read-write file system.

AUTHOR

mount was developed by HP, AT&T, the University of California, Berkeley, and Sun Microsystems.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table

SEE ALSO

mount(1M), mount(2), fstab(4), mnttab(4).

STANDARDS CONFORMANCE

mount: SVID3

NAME

mount(nfs), umount(nfs) - mount and unmount an NFS file systems

SYNOPSIS

/usr/sbin/mount [**-l**] [**-p**|**-v**]

/usr/sbin/mount -a [**-F nfs**] [**-eQ**]

/usr/sbin/mount [**-F nfs**] [**-eQrV**] [**-o** *specific_options*] {*host:path* | *directory*}

/usr/sbin/mount [**-F nfs**] [**-eQrV**] [**-o** *specific_options*] *host:path directory*

/usr/sbin/umount -a [**-F nfs**] [**-h** *host*] [**-v**]

/usr/sbin/umount [**-v**] [**-V**] {*host:path* | *directory*}

DESCRIPTION

The **mount** command mounts file systems. Only a superuser can mount file systems. Other users can use **mount** to list mounted file systems.

The **mount** command attaches *host:path* to *directory*. *host* is a remote system, *path* is a directory on this remote system and *directory* is a directory on the local file tree. *directory* must already exist, be given as an absolute path name and will become the name of the root of the newly mounted file system. If either *host:path* or *directory* is omitted, **mount** attempts to determine the missing value from an entry in the **/etc/fstab** file. **mount** can be invoked on any removable file system, except **/**.

If **mount** is invoked without any arguments, it lists all of the mounted file systems from the file system mount table, **/etc/mnttab**. The **umount** command unmounts mounted file systems. Only a superuser can unmount file systems.

OPTIONS

-r Mount the specified file system read-only.

-o *specific_options*
Set file system specific options according to a comma-separated list chosen from words below.

rw | **ro**	*resource* is mounted read-write or read-only. The default is **rw**.
suid | **nosuid**	Setuid execution allowed or disallowed. The default is **suid**.
remount	If a file system is mounted read-only, remounts the file system read-write.
bg | **fg**	If the first attempt fails, retry in the background, or, in the foreground. The default is **fg**.
quota	Enables **quota**(1M) to check whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system. The default is **quota**.
noquota	Prevent **quota**(1M) from checking whether the user exceeded the quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.
retry=*n*	The number of times to retry the mount operation. The default is **1**.
vers=*<NFS version number>*	By default, the version of NFS protocol used between the client and the server is the highest one available on both systems. If the NFS server does not support NFS Version 3, then the NFS mount will use NFS Version 2 .
port=*n*	Set server UDP port number to *n* (the default is the port customarily used for NFS servers).
grpid	By default, the GID associated with a newly created file will obey the System V semantics; that is, the GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis by setting the set-GID bit of the parent directory; in this case, the GID of a newly created file is set to the GID of the parent directory (see **open**(2) and **mkdir**(2)). Files created on file systems that are mounted with the **grpid** option will obey BSD semantics independent of whether the set-GID bit of the parent directory is set; that is, the GID is unconditionally inherited from that of the parent directory.

rsize=*n*	Set the read buffer size to *n* bytes. The default value is set by kernel.
wsize=*n*	Set the write buffer size to *n* bytes. The default value is set by kernel.
timeo=*n*	Set the NFS timeout to *n* tenths of a second. The default value is set by kernel.
retrans=*n*	Set the number of NFS retransmissions to *n*. The default value is **5**.
soft | **hard**	Return an error if the server does not respond, or continue the retry request until the server responds. The default value is **hard**.
intr | **nointr**	Allow (do not allow) keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system. The default is **intr**.
noac	Suppress attribute caching.
nocto	Suppress fresh attributes when opening a file.
devs | **nodevs**	Allow (do not allow) access to local devices. The default is **devs**.
acdirmax=*n*	Hold cached attributes for no more than *n* seconds after directory update. The default value is **60**.
acdirmin=*n*	Hold cached attributes for at least *n* seconds after directory update. The default value is **30**.
acregmax=*n*	Hold cached attributes for no more than *n* seconds after file modification. The default value is **60**.
acregmin=*n*	Hold cached attributes for at least *n* seconds after file modification. The default value is **3**.
actimeo=*n*	Set *min* and *max* times for regular files and directories to *n* seconds. **actimeo** has no default; it sets **acregmin**, **acregmax**, **acdirmin**, and **acdirmax** to the value specified.

m

-O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error **device** busy**.**

Options (umount)

umount recognizes the following options:

-a	Attempt to unmount all file systems described in **/etc/mnttab**. All optional fields in **/etc/mnttab** must be included and supported. If **-F nfs** option is specified, all NFS file systems in **/etc/mnttab** are unmounted. File systems are not necessarily unmounted in the order listed in **/etc/mnttab**.
-F nfs	Specify the NFS file system type (see *fstyp*(1M)).
-h *host*	Unmount only those file systems listed in **/etc/mnttab** that are remote-mounted from *host*.
-v	Verbose mode. Write a message to standard output indicating which file system is being unmounted.
-V	Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

NFS File Systems

Background vs. Foreground

File systems mounted with the **bg** option indicate that **mount** is to retry in the background if the server's mount daemon (**mountd**(1M)) does not respond. **mount** retries the request up to the count specified in the **retry=***n* option. Once the file system is mounted, each NFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=***n* option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Hard vs. Soft

File systems that are mounted read-write or that contain executable files should always be mounted with the **hard** option. Applications using **soft** mounted file systems may incur unexpected I/O errors.

To improve NFS read performance, files and file attributes are cached. File modification times get updated whenever a write occurs. However, file access times may be temporarily out-of-date until the cache gets refreshed. The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=***n* sets flush time to *n* seconds for both regular files and directories.

EXAMPLES

To mount an NFS file system:

```
example# mount serv:/usr/src /usr/src
```

To mount an NFS file system readonly with no suid privileges:

```
example# mount -r -o nosuid serv:/usr/src /usr/src
```

To mount an NFS file system over Version 3:

```
example# mount -o vers=3 serv:/usr/src /usr/src
```

FILES

/etc/mnttab	table of mounted file systems.
/etc/fstab	list of default parameters for each file system.

SEE ALSO

fsclean(1M), mount(1M), quotaon(1M), mount(2), fstab(4), mnttab(4), fs_wrapper(5), quota(5).

STANDARDS COMPLIANCE

mount: SVID3

umount: SVID3

NAME

mount, umount (vxfs) - mount and unmount a VxFS file system

SYNOPSIS

/usr/sbin/mount [**-l**] [**-v**|**-p**]

/usr/sbin/mount [**-F vxfs**] [**-eQ**] **-a**

/usr/sbin/mount [**-F vxfs**] [**-eQrV**] [**-o** *specific_options*] {*special* | *mount_point*}

/usr/sbin/mount [**-F vxfs**] [**-eQrV**] [**-o** *specific_options*] *special mount_point*

/usr/sbin/umount [**-V**] [**-v**] {*special* | *directory*}

/usr/sbin/umount [**-F vxfs**] [**-v**] **-a**

DESCRIPTION

The **mount** command attaches *special*, a removable file system, to *directory*, a directory on the file tree. *directory*, which must already exist, will become the name of the root of the newly mounted file system. **mount** can be invoked on any removable file system, except /. If **mount** is invoked with no arguments it lists all the mounted file systems from the mounted file system table, **/etc/mnttab**. *special* and *directory* must be given as absolute path names.

The **umount** command unmounts mounted file systems.

Only the superuser can **mount** and **umount** file systems. Other users can use **mount** to list mounted file systems.

Options

mount recognizes the following options:

-a Attempt to mount all file systems described in **/etc/fstab.** All optional fields in **/etc/fstab** must be included and supported. If **-F vxfs** is specified, all VxFS file systems in **/etc/fstab** are mounted. If **noauto** is specified in an entry's option list, this entry is skipped. File systems are not necessarily mounted in the order listed in **/etc/fstab**.

-e Verbose mode. Write a message to the standard output indicating which file system is being mounted.

-F vxfs Specifies the file system type (**vxfs**).

-l Limit actions to local file systems only.

-o *specific_options*

Specifies options specific to the VxFS file system type. *specific_options* is a list of comma separated suboptions and/or keyword/attribute pairs intended for the VxFS-specific module of the command.

The following *specific_options* are valid on a VxFS file system:

rw Read-write (default).

ro Read-only.

suid Set-user-ID execution allowed (default).

nosuid Set-user-ID execution not allowed.

quota Disk quotas enabled (valid only for **rw** type file systems). VxFS maintains quota information in a private area of the file system. If the file system is mounted with quotas enabled, and the file system was previously mounted with quotas disabled and was modified, then the quota information is rebuilt. This may take awhile.

remount Changes the mount options for a mounted file system, such as logging and caching policies or whether the file system can be written to.

log|delaylog|tmplog|nolog

Controls intent logging. File system integrity across system failure requires that logging be enabled. The default is **log**. In **log** mode, file system structural changes are logged to disk before the system call returns to the application. If the system crashes, *fsck_vxfs*(1M) will complete logged operations that have not completed.

In **delaylog** mode, some system calls return before the intent log is written. This improves the performance of the system, but some changes are not guaranteed until a short time later when the intent log is written. This mode approximates traditional UNIX system guarantees for correctness in case of system failures.

In **tmplog** mode, the intent log is almost always delayed. This improves performance, but recent changes may disappear if the system crashes. This mode is only recommended for temporary file systems.

In **nolog** mode, the intent log is disabled. The other three logging modes provide fast file system recovery; **nolog** does not provide fast file system recovery. With **nolog** mode, a full structural check must be performed after a crash; this may result in loss of substantial portions of the file system, depending upon activity at the time of the crash. Usually, a **nolog** file system should be rebuilt with *mkfs_vxfs*(1M) after a crash. The **nolog** mode should only be used for memory resident or very temporary file systems.

blkclear Ensure that all data extents are cleared before being allocated to a file (requires synchronous zeroing, on disk, of certain newly allocated extents). This prevents uninitialized data from appearing in a file being written at the time of a system crash.

snapof=*filesystem*
Mount the file system as a snapshot of *filesystem*, where *filesystem* is either the directory on which a VxFS file system is mounted, or is the block special file containing a mounted VxFS file system. An explicit **-F vxfs** option is required to mount a snapshot file system.

snapsize=*blocks*
Used in conjunction with **snapof**. *blocks* is the size in sectors of the snapshot file system being mounted. This option is required only when the device driver is incapable of determining the size of *special*, and will default to the entire device if not specified.

mincache=direct|dsync|closesync|tmpcache
This option is used to alter the caching behavior of the file system.

The **direct** value causes any writes without the **O_SYNC** flag and all reads to be handled as if the **VX_DIRECT** caching advisory was set instead.

The **dsync** value causes any writes without either the **O_SYNC** flag or the **VX_DIRECT** caching advisory to be handled as if the **VX_DSYNC** caching advisory has been set.

The **closesync**, **dsync** and **direct** values all cause the equivalent of an *fsync*(2) to be run when the file is closed. See *vxfsio*(7) for an explanation of **VX_DIRECT** and **VX_DSYNC**.

The **tmpcache** value disables delayed extending writes, trading off integrity for performance. When this option is chosen, VxFS does not zero out new extents allocated as files are sequentially written. Uninitialized data may appear in files being written at the time of a system crash.

convosync=direct|dsync|closesync|delay
This option is used to alter the caching behavior of the file system for **O_SYNC** I/O operations.

The **direct** value causes any reads or writes with the **O_SYNC** flag to be handled as if the **VX_DIRECT** caching advisory was set instead.

The **dsync** value causes any writes with the **O_SYNC** flag to be handled as if the **VX_DSYNC** caching advisory was set instead.

The **closesync** value causes **O_SYNC** writes to be delayed rather than to take effect immediately. The **closesync**, **dsync**, and **direct** values all cause the equivalent of an *fsync*(2) to be run when any file is accessed with the **O_SYNC** flag is closed.

The **delay** value causes **O_SYNC** writes to be delayed rather than to take effect immediately. Choosing this option causes VxFS to change all **O_SYNC** writes into delayed writes. No special action is performed when closing a file. This option effectively cancels any data integrity guarantees normally provided by opening a file with **O_SYNC**.

datainlog|nodatainlog
Normally, the VxFS file system performs small **O_SYNC** write requests and NFS write requests by logging both the data and the time change to the inode (**datainlog**). If the **nodatainlog** option is used, the logging of synchronous write data is disabled; such writes will write the data into the file and update the inode synchronously before returning to the user.

largefiles|nolargefiles
If one of these options is specified, the file system mount will fail if the *largefile compatibility bit* for the file system does not match the option specified. If **nolargefiles** is specified and the mount succeeds, then the file system does not contain any files whose size is 2 gigabytes or larger, and such files cannot be created. If **largefiles** is specified and the mount succeeds, then the file system may contain files whose size is 2 gigabytes or larger, and large files can be created. The default is to mount the file system according to the *largefile compatibility bit* (see *fsadm_vxfs*(1M) and *mkfs_vxfs*(1M).)

-p Report the list of mounted file systems in the **/etc/fstab** format.

-Q Prevent display of error messages, resulting from an attempt to mount already mounted file systems.

-r Mount the specified file system as read-only. Physically write-protected file systems must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

-v Reports the regular output with file system type and flags, however, *directory* and *special* fields are reversed.

-V Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

umount recognizes the following options:

-a Attempt to unmount all file systems described in **/etc/mnttab**. All optional fields in **/etc/mnttab** must be included and supported. If **-F vxfs** is specified, all VxFS file systems in **/etc/mnttab** are unmounted. File systems are not necessarily unmounted in the order listed in **/etc/mnttab**.

-F vxfs Specifies the file system type (**vxfs**).

-v Verbose mode. Write a message to the standard output indicating which file system is being unmounted.

-V Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

List the file systems currently mounted:

```
mount
```

Mount a VxFS file system **/dev/dsk/c1t2d0** at directory **/home**

```
mount -F vxfs /dev/dsk/c1t2d0 /home
```

Unmount the same file system:

```
umount /dev/dsk/c1t2d0
```

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table

SEE ALSO

fsadm_vxfs(1M), fsck_vxfs(1M), mkfs_vxfs(1M), mount(1M), mount(2), fsync(2), fstab(4), mnttab(4), quota(5), vxfsio(7).

STANDARDS CONFORMANCE

mount: SVID3

umount: SVID3

NAME

mountall, umountall - mount and unmount multiple file systems

SYNOPSIS

/sbin/mountall [**-F** *FStype*] [**-l**|**-r**] [*file_system_table* | **-**]
/sbin/mountall [**-l**|**-r**] [**-m**]
/sbin/mountall [**-n**]
/sbin/umountall [**-F** *FStype*] [**-k**] [**-l**|**-r**]

DESCRIPTION

mountall is used to mount file systems according to *file_system_table*. By default, **/etc/fstab** is the *file_system_table*. If a dash (**-**) is specified, **mountall** reads *file_system_table* from the standard input; the standard input must be in the same format as the **/etc/fstab**.

Before each file system is mounted, a check is done using **fsck** (see *fsck*(1M)) to ensure that the file system is mountable. If the file system is not mountable, it is repaired by **fsck** before the mount is attempted.

umountall causes all mounted file systems except the non-removable file systems such as **root** to be unmounted.

Options

mountall and **umountall** recognize the following options:

-F *FStype*	Specify the file system type (*FStype*) to be mounted or unmounted.
-l	Specify action on local file systems only.
-r	Specify action on remote file systems only.
-k	Send a **SIGKILL** signal to processes that have files opened.
-m	Attempt to mount all the unmounted file systems. This option will not perform the file system consistency check and repair.
-n	Perform the file system consistency check and repair on all unmounted file system. This option will not mount the file systems.

m

DIAGNOSTICS

Error and warning messages may originate from **fsck**, **mount**, **fuser**, or **umount**. See *fsck*(1M), *mount*(1M), or *fuser*(1M) to interpret the error and warning messages.

EXAMPLES

Mount all unmounted file systems listed in **/etc/fstab**:

```
mountall
```

Mount all local file systems listed in **/etc/fstab**:

```
mountall -l
```

Mount all remote file systems listed in **/etc/fstab**:

```
mountall -r
```

Mount all local hfs file systems:

```
mountall -F hfs -l
```

Unmount all NFS file systems and kill any processes that have files opened in the file system:

```
umountall -F nfs -k
```

WARNINGS

umountall, especially with the **-k** option, should be used with extreme caution, because it can cause severe damage.

The **-n** option may not be available in future releases.

mountall may not be effective with some cases of LOFS file systems.

FILES

`/etc/fstab`	Static information about the file systems
`/etc/mnttab`	Mounted file system table

SEE ALSO

fsck(1M), mount(1M), fuser(1M), mnttab(4), fstab(4), signal(2)

NAME

mountd - NFS mount request server

SYNOPSIS

/usr/sbin/rpc.mountd [**-l** *log_file*] [**-t** *n*] [**-p**|**-e**|**-n**]

DESCRIPTION

mountd is an RPC server that answers file system mount requests. It reads file **/etc/xtab** (described in *exports*(4)) to determine which directories are available to which machines. It also provides information on what file systems are mounted by which clients. This information can be printed using the **showmount** command (see *showmount*(1M)).

rpc.mountd can be started at boot time by setting the variable **NFS_SERVER** to 1 in the file **/etc/rc.config.d/nfsconf**. It can also be started through **/etc/inetd.conf** (see *inetd*(1M)), provided that the **START_MOUNTD** variable is set to 0 in **/etc/rc.config.d/nfsconf**.

Options

mountd recognizes the following options:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

The information logged to the file includes the date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file since enough information is included to uniquely identify each error.

-p Run from unreserved ports. This option restores the old default behavior on HP-UX. The default has been changed for the mount daemon to run from reserved ports unless this option is set.

-e Exit after serving each RPC request. When this option is used, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services. This option only supports UDP requests.

-n Exit only if:

- **portmap** dies (see *portmap*(1M)),
- another **rpc.mountd** registers with **portmap**, or
- **rpc.mountd** becomes unregistered with **portmap**.

This option is more efficient because a new process is not launched for each RPC request. This option is the default.

-t*n* Specify tracing level *n* , where *n* can have one of the following values:

1 Errors only (default)
2 Errors, mount requests and mount failures

WARNINGS

The default behavior of the mount daemon is to run from reserved ports. If the daemon needs to be run from unreserved ports, use the -p option.

If a client crashes, executing **showmount** on the server will show that the client still has a file system mounted; i.e., the client's entry is not removed from **/etc/rmtab** until the client reboots and executes **umount -a** (see *showmount*(1M)).

Also, if a client mounts the same remote directory twice, only one entry appears in **/etc/rmtab**. Doing a **umount** of one of these directories removes the single entry and **showmount** no longer indicates that the remote directory is mounted.

AUTHOR

mountd was developed by Sun Microsystems, Inc.

FILES

/etc/rmtab List of all hosts having file systems mounted from this machine

SEE ALSO

inetd(1M), mount(1M), portmap(1M), showmount(1M), exports(4), inetd.conf(4), inetd.sec(4), rmtab(4), services(4).

NAME

mrinfo - Multicast Routing Configuration Information Tool

SYNOPSIS

/usr/sbin/mrinfo [**-d** *debuglevel*] [**-r** *retries*] [**-t** *timeout*] [*multicast-router*]

DESCRIPTION

mrinfo requests the configuration information from the *multicast-ourter*, and prints the information to the standard out. *multicast-router* can be either an IP address or a system name. **mrinfo** sends out the *ASK_NEIGHBORS* igmp message to the specified *multicast-router*, when the router receives the request, it sends back its configuration information. If the *multicast-router* is not specified, the request is sent the local router.

The the configuration information for each interface is printed in the following format:

```
interface_addr -> neighbor_addr (neighbor_name) [metrics/thresh/flags]
```

If there are multiple neighbor routers on one interface, they will all be reported on the output. The possible values for *flag* are:

tunnel	Neighbors are reached via tunnel.
srcrt	The tunnel uses IP source routing.
down	The interface is down.
disabled	The interface is administratively disabled for multicast routing.
querier	The local router is the querier of the subnet.

Please see *mrouted*(1M) for **metrics** and **thresh**.

The command line options are:

-d*debuglevel*	Sets the level for printing out the debug message. The default is 0, only error and warning messages will be printed. Debug level three prints most the messages.
-r*retries*	Sets the retry times to pull the routing daemon for information. The default is 3.
-t*timeout*	Specifies the timeout value in seconds for waiting the reply. The default value is 4.

EXAMPLE

The following is an example of quering the multicasting configuration from the local routing daemon.

```
mrinfo

127.0.0.1 (localhost) [version 3.3]:
  15.13.106.144 -> 15.13.106.145 (hpntcbs.cup.hp.com) [10/1/querier]
  193.2.1.39 -> 0.0.0.0 (all-zeros-broadcast) [1/1/disabled]
  15.13.106.144 -> 15.255.176.33 (matmos.hpl.hp.com) [10/1/tunnel]
  15.13.106.144 -> 15.17.20.7 (hpspddc.vid.hp.com) [10/1/tunnel/down]
```

Note

mrinfo must be run as root.

AUTHOR

mrinfo was developed by Van Jacobson.

SEE ALSO

mrouted(1M), map-mbone(1M).

NAME

mrouted - IP multicast routing daemon

SYNOPSIS

/usr/sbin/mrouted [**-p**] [**-c** *config_file*] [**-d** *debug_level*]

DESCRIPTION

The **mrouted** command is an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP), an earlier version of which is specified in RFC-1075. It maintains topological knowledge via a distance-vector routing protocol (like RIP, described in RFC-1058), upon which it implements a multicast datagram-forwarding algorithm called Reverse Path Multicasting.

mrouted forwards a multicast datagram along a shortest (reverse) path tree rooted at the subnet on which the datagram originates. The multicast delivery tree may be thought of as a broadcast delivery tree that has been pruned back so that it does not extend beyond those subnetworks that have members of the destination group. Hence, datagrams are not forwarded along those branches which have no listeners of the multicast group. The IP time-to-live of a multicast datagram can be used to limit the range of multicast datagrams.

In order to support multicasting among subnets that are separated by (unicast) routers that do not support IP multicasting, **mrouted** includes support for "tunnels", which are virtual point-to-point links between pairs of **mrouted**s located anywhere in an internet. IP multicast packets are encapsulated for transmission through tunnels, so that they look like normal unicast datagrams to intervening routers and subnets. The encapsulation is added on entry to a tunnel and stripped off on exit from a tunnel. By default, the packets are encapsulated using the IP-in-IP protocol (IP protocol number 4).

The tunnelling mechanism allows **mrouted** to establish a virtual internet for the purpose of multicasting only, which is independent of the physical internet and which may span multiple Autonomous Systems.

mrouted handles multicast routing only; there may or may not be unicast routing software running on the same machine as **mrouted**. With the use of tunnels, it is not necessary for **mrouted** to have access to more than one physical subnet in order to perform multicast forwarding.

Invocation

If the **-d** option is not specified or if the debug level is specified as 0, **mrouted** detaches from the invoking terminal. Otherwise, it remains attached to the invoking terminal and responsive to signals from that terminal. If **-d** is specified with no argument, the debug level defaults to 2. Regardless of the debug level, **mrouted** always writes warning and error messages to the system log demon. Non-zero debug levels have the following effects:

level 1	all **syslog** messages are also printed to **stderr**.
level 2	all level 1 messages plus notifications of "significant" events are printed to **stderr**.
level 3	all level 2 messages plus notifications of all packet arrivals and departures are printed to **stderr**.

Upon startup, **mrouted** writes its *pid* to the file **/var/tmp/mrouted.pid**.

Configuration

mrouted automatically configures itself to forward on all multicast-capable interfaces (i.e., interfaces that have the IFF_MULTICAST flag set, excluding the loopback "interface"). **mrouted** finds other **mrouted**s directly reachable via those interfaces. To override the default configuration or to add tunnel links to other **mrouted**s, configuration commands may be placed in **/etc/mrouted.conf** (or an alternative file, specified by the **-c** option). There are four types of configuration commands:

```
phyint <local-addr>   [disable]   [metric <m>]
                     [threshold <t>] [rate_limit <b>]
                       [boundary (<boundary-name> | <scoped-addr>/<mask-len>)]
                       [altnet <network>/<mask-len>]

tunnel <local-addr> <remote-addr> [metric <m>]
                     [threshold <t>] [rate_limit <b>]
                       [boundary (<boundary-name> | <scoped-addr>/<mask-len>)]

cache_lifetime <ct>

pruning <off/on>

name <boundary-name> <scoped-addr>/<mask-len>
```

The file format is free-form; white space (including newlines) is not significant. The *boundary* and *altnet* options may be specified as many times as necessary.

The **phyint** command can be used to disable multicast routing on the physical interface identified by local IP address <local-addr>, or to associate a non-default metric or threshold with the specified physical interface. The local IP address <local-addr> may be replaced by the interface name (such as **lan0**). If **phyint** is attached to multiple IP subnets, describe each additional subnet with the *altnet* option. **phyint** commands must precede **tunnel** commands.

The **tunnel** command can be used to establish a tunnel link between local IP address <local-addr> and remote IP address <remote-addr>, and to associate a non-default metric or threshold with that tunnel. The local IP address <local-addr> may be replaced by the interface name (such as **lan0**). The remote IP address <remote-addr> may be replaced by a host name, if and only if the host name has a single IP address associated with it. The tunnel must be set up in the **mrouted.conf** files of both routers before it can be used.

m

cache_lifetime is a value that determines the amount of time that a cached multicast route stays in kernel before timing out. The value of this entry should lie between 300 (5 min) and 86400 (1 day). It defaults to 300.

The **pruning** command is provided for **mrouted** to act as a non-pruning router. It is also possible to start **mrouted** in a non-pruning mode using the **-p** option on the command line. It is expected that a router would be configured in this manner for test purposes only. The default mode is pruning enabled.

You may assign names to boundaries to make configuration easier with the **name** command. The *boundary* option on **phyint** or **tunnel** commands can accept either a name or a boundary.

The *metric* option is the "cost" associated with sending a datagram on the given interface or tunnel; it may be used to influence the choice of routes. The metric defaults to 1. Metrics should be kept as small as possible because **mrouted** cannot route along paths with a sum of metrics greater than 31.

The threshold is the minimum IP time-to-live required for a multicast datagram to be forwarded to the given interface or tunnel. It is used to control the scope of multicast datagrams. (The TTL of forwarded packets is only compared to the threshold; it is not decremented by the threshold. Every multicast router decrements the TTL by 1.) The default threshold is 1.

In general, all **mrouted**s connected to a particular subnet or tunnel should use the same metric and threshold for that subnet or tunnel.

The *rate_limit* option allows the network administrator to specify a certain bandwidth in Kbits/second which would be allocated to multicast traffic. It defaults to 500Kbps on tunnels and 0 (unlimited) on physical interfaces.

The *boundary option* allows an interface to be configured as an administrative boundary for the specified scoped address. Packets belonging to this address will not be forwarded on a scoped interface. The boundary option accepts either a name or a boundary spec.

mrouted will not initiate execution if it has fewer than two enabled **vifs** (virtual interface), where a **vif** is either a physical multicast-capable interface or a tunnel. It will log a warning if all of its **vifs** are tunnels; such an **mrouted** configuration would be better replaced by more direct tunnels.

Example Configuration

This is an example configuration for a multicast router at a large school.

```
#
# mrouted.conf example
#
# Name our boundaries to make it easier
name LOCAL 239.255.0.0/16
name EE 239.254.0.0/16
#
# lan1 is our gateway to compsci, don't forward our
# local groups to them
phyint lan1 boundary EE
#
# lan2 is our interface on the classroom net, it has four
# different length subnets on it.
# note that you can use either an ip address or an
# interface name
phyint 172.16.12.38 boundary EE altnet 172.16.15.0/26
      altnet 172.16.15.128/26 altnet 172.16.48.0/24
#
# atm0 is our ATM interface, which doesn't properly
# support multicasting.
phyint atm0 disable
#
# This is an internal tunnel to another EE subnet
# Remove the default tunnel rate limit, since this
# tunnel is over ethernets
tunnel 192.168.5.4 192.168.55.101 metric 1 threshold 1
      rate_limit 0
#
# This is our tunnel to the outside world.
# Careful with those boundaries, Eugene.
tunnel 192.168.5.4 10.11.12.13 metric 1 threshold 32
      boundary LOCAL boundary EE
```

Signals

mrouted responds to the following signals:

HUP	restarts **mrouted**. The configuration file is reread every time this signal is evoked.
INT	terminates execution gracefully (i.e., by sending good-bye messages to all neighboring routers).
TERM	same as INT
USR1	dumps the internal routing tables to **/usr/tmp/mrouted.dump**.
USR2	dumps the internal cache tables to **/usr/tmp/mrouted.cache**.
QUIT	dumps the internal routing tables to **stderr** (only if **mrouted** was invoked with a non-zero debug level).

For convenience in sending signals, **mrouted** writes its *pid* to **/var/tmp/mrouted.pid** upon startup.

EXAMPLES

The routing tables look like this:

```
    Virtual Interface Table
     Vif  Local-Address                    Metric  Thresh  Flags
      0   36.2.0.8      subnet: 36.2          1       1    querier
                        groups: 224.0.2.1
                                224.0.0.4
                       pkts in: 3456
                      pkts out: 2322323

      1   36.11.0.1     subnet: 36.11         1       1    querier
                        groups: 224.0.2.1
                                224.0.1.0
                                224.0.0.4
                       pkts in: 345
                      pkts out: 3456

      2   36.2.0.8      tunnel: 36.8.0.77     3       1
                         peers: 36.8.0.77 (2.2)
                    boundaries: 239.0.1
                              : 239.1.2
                       pkts in: 34545433
                      pkts out: 234342

      3   36.2.0.8        tunnel: 36.6.8.23           3       16

    Multicast Routing Table (1136 entries)
     Origin-Subnet   From-Gateway    Metric Tmr In-Vif  Out-Vifs
     36.2                                 1   45    0    1* 2  3*
     36.8            36.8.0.77            4   15    2    0* 1* 3*
     36.11                                1   20    1    0* 2  3*
     .
     .
     .
```

In this example, there are four **vifs** connecting to two subnets and two tunnels. The **vif** 3 tunnel is not in use (no peer address). The **vif** 0 and **vif** 1 subnets have some groups present; tunnels never have any groups. This instance of **mrouted** is the one responsible for sending periodic group membership queries on the **vif** 0 and **vif** 1 subnets, as indicated by the "querier" flags. The list of boundaries indicate the scoped addresses on that interface. A count of the number of incoming and outgoing packets is also shown at each interface.

Associated with each subnet from which a multicast datagram can originate is the address of the previous hop router (unless the subnet is directly connected), the metric of the path back to the origin, the amount of time since an update was received for this subnet, the incoming **vif** for multicasts from that origin, and a list of outgoing **vifs**. The asterisk (*****) indicates that the outgoing **vif** is connected to a leaf of the broadcast tree rooted at the origin, and a multicast datagram from that origin will be forwarded on that outgoing **vif** only if there are members of the destination group on that leaf.

The **mrouted** command also maintains a copy of the kernel forwarding cache table. Entries are created and deleted by **mrouted**.

The cache tables look like this:

```
    Multicast Routing Cache Table (147 entries)
     Origin             Mcast-group     CTmr  Age Ptmr IVif Forwvifs
     13.2.116/22        224.2.127.255     3m   2m    -  0    1
    >13.2.116.19
    >13.2.116.196
     138.96.48/21       224.2.127.255     5m   2m    -  0    1
    >138.96.48.108
     128.9.160/20       224.2.127.255     3m   2m    -  0    1
    >128.9.160.45
     198.106.194/24     224.2.135.190     9m  28s   9m  0P
    >198.106.194.22
```

Each entry is characterized by the origin subnet number, mask, and the destination multicast group. The **CTmr** field indicates the lifetime of the entry. The entry is deleted from the cache table when the timer decrements to zero. The **Age** field is the time since this cache entry was originally created. Since cache entries get refreshed if traffic is flowing, routing entries can grow very old. The **Ptmr** field is simply a dash if no prune was sent upstream, or the amount of time until the upstream prune will time out. The **Ivif** field indicates the incoming **vif** for multicast packets from that origin. Each router also maintains a record of the number of prunes received from neighboring routers for a particular source and group. If there are no members of a multicast group on any downward link of the multicast tree for a subnet, a prune message is sent to the upstream router. They are indicated by a **P** after the **vif** number. The **Forwvifs** field shows the interfaces along which datagrams belonging to the source-group are forwarded. A **p** indicates that no datagrams are being forwarded along that interface. An unlisted interface is a leaf subnet with no members of the particular group on that subnet. A **b** on an interface indicates that it is a boundary interface; that is, traffic will not be forwarded on the scoped address on that interface. An additional line with a **>** as the first character is printed for each source on the subnet. Note that there can be many sources in one subnet.

FILES

```
/etc/mrouted.conf
/var/run/mrouted.pid
/var/tmp/mrouted.dump
/var/tmp/mrouted.cache
```

SEE ALSO

mrinfo(1M), map-mbone(1M).

DVMRP is described, along with other multicast routing algorithms, in the paper "Multicast Routing in Internetworks and Extended LANs" by S. Deering, in the *Proceedings of the ACM SIGCOMM '88 Conference*.

AUTHORS

Steve Deering, Ajit Thyagarajan, Bill Fenner.

NAME

/usr/sbin/mtail - Tails the mail log file.

SYNOPSIS

`mtail` [*n*]

DESCRIPTION

`mtail` displays the last part of the mail log, typically **`/var/adm/syslog/mail.log`**. By default, it displays the last 20 lines of this log.

Options

n	Display last *n* lines of **`/var/adm/syslog/mail.log`** instead of just 20.

SEE ALSO

sendmail(1M).

NAME

mvdir - move a directory

SYNOPSIS

/usr/sbin/mvdir *dir newdir*

DESCRIPTION

mvdir moves one directory tree into another existing directory (within the same file system), or renames a directory without moving it.

dir must be an existing directory.

If *newdir* does not exist but the directory that would contain it does, *dir* is moved and/or renamed to *newdir*. Otherwise, *newdir* must be an existing directory not already containing an entry with the same name as the last pathname component of *dir*. In this case, *dir* is moved and becomes a subdirectory of *newdir*. The last pathname component of *dir* is used as the name for the moved directory.

mvdir refuses to move *dir* if the path specified by *newdir* would be a descendent directory of the path specified by *dir*. Such cases are not allowed because cyclic sub-trees would be created as in the case, for example, of **mvdir x/y x/y/z/t** which is prohibited.

mvdir does not allow directory **.** to be moved.

Only users who have appropriate privileges can use **mvdir**.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

mvdir was developed by OSF and HP.

SEE ALSO

cp(1), mkdir(1), mv(1).

STANDARDS CONFORMANCE

mvdir: SVID2, SVID3

NAME

named-xfer - ancillary agent for inbound zone transfers

SYNOPSIS

named-xfer -z *zone_to_transfer* **-f** *db_file* **-s** *serial_no* [**-d** *debuglevel*] [**-l** *debug_log_file*] [**-t** *trace_file*] [**-p** *port#*] [**-S**] *nameserver*...

DESCRIPTION

named-xfer is an ancillary program executed by *named*(1M) to perform an inbound zone transfer. It is rarely executed directly, and then generally by system administrators trying to debug a zone transfer problem. See RFC's 1033, 1034, and 1035 for more information on the Internet name-domain system.

Options are:

-z specifies the name of the zone to be transferred.

-f specifies the name of the file into which the zone should be dumped when it is received from the primary server.

-s specifies the serial number of the current copy of the zone selected for transfer. If the SOA resource record received from the specified remote nameserver(s) does not have a serial number higher than one specified with this option, the transfer will be aborted.

-d Print debugging information. A number after the **d** determines the level of messages printed.

-l Specifies a log file for debugging messages. The default file uses the prefix **xfer.ddt.** and is located in the **/var/tmp** directory. Note this option only applies if **-d** is also specified.

-t Specifies a trace file which will contain a protocol trace of the zone transfer. This is probably only of interest when debugging the name server itself.

-p Use a different port number. The default is the standard port number as returned by *getservbyname*(3) for service "domain".

-S Perform a restricted transfer of only the SOA, NS and glue A records for the zone. The SOA will not be loaded by **named**, but will be used to determine when to verify the NS records. See the **stubs** directive in *named*(1M) for more information.

n

Additional arguments are taken as name server addresses in so-called "dotted-quad" syntax only; no host name are allowed here. At least one address must be specified. Any additional addresses will be tried in order if the first one fails to transfer to us successfully.

RETURN VALUE

0 Indicates that the zone was up-to-date and no transfer was needed.

1 Indicates a successful transfer.

2 Indicates that the host(s) **named-xfer** queried cannot be reached or that an error occurred and **named-xfer** did not log a corresponding error message.

3 Indicates that an error occurred and **named-xfer** logged an error message.

AUTHOR

named-xfer was developed by the University of California, Berkeley.

SEE ALSO

named(1M), resolver(3N), resolver(4), hostname(5).

RFC 882, RFC 883, RFC 973, RFC 974, RFC 1033, RFC 1034, RFC 1035, RFC 1123.

NAME

named - Internet domain name server

SYNOPSIS

named [**-d** *debuglevel*] [**-p** *port_number*] [[**-b**] *bootfile*] [**-q**] [**-r**]

DESCRIPTION

named is the Internet domain name server. See RFC1034 and RFC1035 for more information on the Domain Name System. Without any arguments, **named** reads the default boot file **/etc/named.boot**, reads any initial data, and listens for queries.

Options are:

- **-d** Print debugging information. A number after the **d** determines the level of messages printed.
- **-p** Use a different port number.
- **-b** Use a boot file other than **/etc/named.boot**.
- **-q** Trace all incoming queries. The boot file directive **options query-log** can also be used to enable this functionality.
- **-r** Turns recursion off in the server. Answers can only come from local (primary or secondary) zones. This can be used on root servers. The boot file directive **options no-recursion** can also be used to enable this functionality.

Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server gets its initial data. If multiple boot files are specified, only the last is used. Lines in the boot file cannot be continued on subsequent lines. The following is a small example:

```
;
;     boot file for name server
;
directory   /usr/local/domain

; type      domain                   host/file       backup file

cache       .                        db.cache
primary     berkeley.edu             db.berkeley
primary     32.128.in-addr.arpa      db.128.32
secondary   cc.berkeley.edu          128.32.137.8    db.cc
secondary   6.32.128.in-addr.arpa    128.32.137.8    db.128.32.6
primary     0.0.127.in-addr.arpa     db.127.0.0
forwarders  10.0.0.78 10.2.0.78
limit       max-xfers 10
options     no-recursion
sortlist    10.0.0.0 26.0.0.0
```

Comments in the boot file start with a **;** and end at the end of the line. Comments can start anywhere on the line.

The **directory** line causes the server to change its working directory to the directory specified. This can be important for the correct processing of $INCLUDE files (described later) in primary servers' master files. There can be more than one **directory** line in the boot file if master files are in separate directories.

Files referenced in the boot file contain data in the master file format described in RFC1035.

A server can access information from servers in other domains given a list of root name servers and their addresses. The **cache** line specifies that data in **db.cache** is to be placed in the backup cache. Its use is to prime the server with the locations of root domain servers. This information is used to find the current root servers and their addresses. The current root server information is placed in the operating cache. Data for the root nameservers in the backup cache are never discarded. There can be more than one **cache** file specified.

The first **primary** line states that the master file **db.berkeley** contains authoritative data for the **berkeley.edu** zone. A server authoritative for a zone has the most accurate information for the zone. All domain names are relative to the origin, in this case, **berkeley.edu** (see below for a more detailed description). The second **primary** line states that the file **db.128.32** contains authoritative data for the domain **32.128.in-addr.arpa**. This domain is used to translate addresses in network **128.32** to hostnames. The third **primary** line states that the file **db.127.0.0** contains authoritative data for the domain **0.0.127.in-addr.arpa**. The domain is used to translate **127.0.0.1** to the name used by the loopback interface. Each master file should begin with an SOA record for the zone (see below).

The first **secondary** line specifies that all authoritative data in the **cc.berkeley.edu** zone is to be transferred from the name server at Internet address 128.32.137.8 and will be saved in the backup file **db.cc**. Up to 10 addresses can be listed on this line. If a transfer fails, it will try the next address in the list. The secondary copy is also authoritative for the specified domain. The first non-Internet address on this line will be taken as a filename in which to backup the transferred zone. The name server will load the zone from this backup file (if it exists) when it boots, providing a complete copy, even if the master servers are unreachable. Whenever a new copy of the domain is received by automatic zone transfer from one of the master servers, this file is updated. If no file name is given, a temporary file will be used and will be deleted after each successful zone transfer. This is not recommended because it causes a needless waste of bandwidth.

A **stub** line (not shown) is similar to a **secondary**. Stub zones are intended to ensure that a primary for a zone always has the correct NS records for children of that zone. If the primary is not a secondary for a child zone, it should be configured with stub zones for all its children. Stub zones provide a mechanism to allow NS records for a zone to be specified in only one place.

This feature is experimental as of release 4.9.3 of BIND.

```
primary   csiro.au        db.csiro
stub      dms.csiro.au    130.155.16.1    db.dms-stub
stub      dap.csiro.au    130.155.98.1    db.dap-stub
```

The **forwarders** line specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the boot file specifies one or more forwarders, then the server will send all queries for data not in the cache or in its authoritative data to the forwarders first. Each forwarder will be asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from a forwarder, the server will continue as it would have without the forwarders line unless it is in **forward-only** (slave) mode. The forwarding facility is useful to cause a large sitewide cache to be generated on a master, and to reduce traffic over links to outside servers.

n

The **sortlist** line can be used to indicate networks that are preferred over other, unlisted networks. Address sorting only happens when the query is from a host on the same network as the server. The best address is placed first in the response. The address preference is local network addresses, then addresses on the sort list, then other addresses.

The **xfrnets** directive (not shown) can be used to implement primitive access control. If this directive is given, then your name server will only answer zone transfer requests from hosts which are on networks listed in your **xfrnets** directives.

The **include** directive (not shown) can be used to process the contents of some other file as though they appeared in place of the **include** directive. This may be useful if you have many zones or have logical groupings of zones maintained by different people. The directive takes one argument, that being the name of the file whose contents are to be included. No quotation marks are necessary around the file name.

The **bogusns** directive (not shown) tells the server that no queries are to be sent to the specified name server addresses (specified as dotted quads, not as domain names). This may be useful if you know that a popular server has bad data in a zone or cache and you wish to avoid contamination while the problem is being fixed.

The **limit** directive can be used to change the servers' internal limits.

The **transfers-in** argument is the number of **named-xfer** subprocesses which the server will spawn at any one time. The **transfers-per-ns** argument is the maximum number of zone transfers to be simultaneously initiated to any given remote name server.

The **options** directive introduces a boolean specifier that changes the server behaviour. More than one option can be specified in a single directive. The options are as follows:

no-recursion
: which will cause the server to answer with a referral rather than actual data whenever it receives a query for a name it is not authoritative for. This must not be enabled on a server that is listed in any host's **resolv.conf** file.

query-log
: which causes all queries to be logged to the **syslog** file. This produces a lot of data, so some consideration should be given before enabling this for any period of time.

forward-only
which causes the server to query only its forwarders. This option is normally used on a machine that wishes to run a server but for physical or administrative reasons cannot be given access to the Internet. Enabling this option is equivalent to the use of the "slave" line used in previous versions of **named**.

fake-iquery
which tells the server to send back a useless and bogus reply to inverse queries rather than responding with an error.

no-round-robin
which disables the default round-robin cycling of returned IP addresses for multi-homed hosts.

The **max-fetch** directive (not shown) is allowed for backward compatibility; its meaning is identical to **limit transfers-in**.

The **slave** directive (not shown) is allowed for backward compatibility; its meaning is identical to **options forward-only**.

The master file consists of control information and a list of resource records for objects in the zone of the forms:

$INCLUDE *filename opt_domain*
$ORIGIN *domain*
domain opt_ttl opt_class type resource_record_data

where *domain* is **.** for root domain, **@** for the current origin (where current origin is the domain from the boot file or the origin from an **$ORIGIN** line), or a standard domain name. If *domain* is a standard domain name that does not end with **.**, the current origin is appended to the domain. Domain names ending with **.** are unmodified. The *opt_domain* field is used to define an origin for the data in an included file. It is equivalent to placing a **$ORIGIN** statement before the first line of the included file. The field is optional. Neither the *opt_domain* field nor **$ORIGIN** statements in the included file modify the current origin for this file. The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero, meaning the minimum value specified in the SOA record for the zone. The *opt_class* field is the object address type; currently only two types are supported: **IN** and, to a limited extent, **HS**. **IN** is for objects connected to the DARPA Internet and **HS** is for objects in the Hesiod class. The *type* field contains one of the following tokens; the data expected in the *resource_record_data* field is in parentheses:

n

A	A host address (dotted quad)
AFSDB	DCE or AFS server information
CNAME	Canonical name for an alias (domain)
HINFO	host information (*cpu_type OS_type*)
MX	a mail exchanger (domain)
NS	an authoritative name server (domain)
PTR	a domain name pointer (domain)
PX	Pointer to X.400/RFC822 mapping information
SOA	marks the start of a zone of authority (domain of originating host, domain address of maintainer, a serial number and the following parameters in seconds: refresh, retry, expire and minimum TTL)
TXT	text data (string)
WKS	a well known service description (IP address followed by a list of services)

Not all of the data types are used during normal operation. The following data types are for experimental use and are subject to change: AFSDB, PX.

Resource records normally end at the end of a line, but can be continued across lines between opening and closing parentheses. Comments are introduced by semicolons and continue to the end of the line.

Each master zone file should begin with an SOA record for the zone. An example SOA record is as follows:

```
@     IN     SOA     ucbvax.berkeley.edu. root.ucbvax.berkeley.edu. (
                            89         ; Serial
                            10800      ; Refresh every 3 hours
                            3600       ; Retry every hour
                            604800     ; Expire after a week
                            86400 )    ; Minimum ttl of 1 day
```

The SOA lists a serial number, which should be increased each time the master file is changed. Secondary servers check the serial number at intervals specified by the refresh time in seconds; if the serial number increases, a zone transfer is done to load the new data. If a master server cannot be contacted when a refresh is due, the retry time specifies the interval at which refreshes should be attempted until successful. If a master server cannot be contacted within the interval given by the expire time, all data from the zone is discarded by secondary servers. The minimum value is the time-to-live used by records in the file with no explicit time-to-live value.

NOTES

The boot file directives **domain** and **suffixes** have been obsoleted by a more useful resolver-based implementation of suffixing for partially qualified domain names.

The following signals have the specified effect when sent to the server process using the *kill*(1) command:

SIGHUP	Causes server to read the boot file and reload database.
SIGINT	Dumps current data base and cache to **/var/tmp/named_dump.db**.
SIGIOT	Dumps statistics data into **/var/tmp/named.stats**. Statistics data is appended to the file.
SIGUSR1	Turns on debugging; each **SIGUSR1** increments debug level.
SIGUSR2	Turns off debugging completely.
SIGWINCH	Toggles the logging of all incoming queries via **syslog()**

sig_named(1M) can also be used for sending signals to the server process.

DIAGNOSTICS

Any errors encountered by **named** in the boot file, master files, or in normal operation are logged with syslog and in the debug file, **/var/tmp/named.run**, if debugging is on.

AUTHOR

named was developed by the University of California, Berkeley.

FILES

/etc/named.boot	name server configuration boot file
/var/run/named.pid	process ID
/var/tmp/named.run	debug output
/var/tmp/named_dump.db	dump of the name server database
/var/tmp/named.stats	nameserver statistics data

SEE ALSO

kill(1), hosts_to_named(1M), named-xfer(1M), sig_named(1M), signal(2), gethostent(3N), resolver(3N), resolver(4), hostname(5),

RFC 882, RFC 883, RFC 973, RFC 974, RFC 1032, RFC 1033, RFC 1034, RFC 1035, RFC 1123.

NAME

ncheck (generic) - generate a list of path names from inode numbers

SYNOPSIS

/usr/sbin/ncheck [**-F** *FStype*] [**-V**] [**-o** *specific_options*] [*special* ...]

DESCRIPTION

ncheck, when invoked without arguments, generates a list of path names corresponding to the inode numbers of all files contained on the file systems listed in **/etc/fstab.** If *special* is specified, ncheck reports on the *special* only. Path names generated by **ncheck** are relative to the given *special*.

Options

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching each *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-o *specific_options*
Specify options specific to each file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for a specific *FStype*-specific module of the command. See the file-system-specific manual pages for a description of the *specific_options* supported, if any.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Execute the **ncheck** command on all *special* in **/etc/fstab:**

```
ncheck
```

Execute the **ncheck** command on HFS file system **/dev/dsk/c1d2s0:**

```
ncheck -F hfs /dev/dsk/c1d2s0
```

Display a completed command line without executing the command:

```
ncheck -V /dev/dsk/c1d2s0
```

FILES

/etc/default/fs	Specifies the default system type.
/etc/fstab	Static information about the file systems.

AUTHOR

ncheck was developed by AT&T and HP.

SEE ALSO

fstab(4), fstyp(1M), fs_wrapper(5), ncheck_*FStype*(1M).

STANDARDS CONFORMANCE

ncheck: SVID2, SVID3

NAME

ncheck (hfs) - generate a list of path names from inode numbers for a HFS file system

SYNOPSIS

/usr/sbin/ncheck [**-F hfs**] [**-V**] [**-S sector_ranges**] [**-i** *inode-numbers*]
[**-a**] [**-s**] [*special* ...]

DESCRIPTION

ncheck, when invoked without arguments, generates a list of path names corresponding to the inode numbers of all files contained on the HFS file systems listed in **/etc/fstab**. If *special* is specified, ncheck reports on the *special* only. Path names generated by **ncheck** are relative to the given *special*. Names of directory files are followed by **/**.

Options

-a Allow printing of the names **.** and **..**, which are ordinarily suppressed.

-F hfs Specify the HFS file system type.

-i *inode-numbers*
Report only on files whose inode numbers are specified on the command line, in *inode-numbers*. *inode-numbers* is a comma separated list of inode numbers.

-s Report only on special files and regular files with set-user-ID mode. The **-s** option is intended to discover concealed violations of security policy.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-S *sector_ranges*
Report only on files using sector numbers specified on the command line in *sector_ranges*. *sector_ranges* is a comma separated list of sector ranges. A sector range is a starting sector number and an ending sector number separated by a dash, or just a sector number. The sector numbers should be in DEV_BSIZE units. If no pathname contains the sector number it will be reported as free or containing file system structure. Sectors beyond the end of the file system will be reported as illegal.

Access Control Lists

Continuation inodes (that is, inodes containing additional access control list information) are quietly skipped since they do not correspond to any path name.

EXAMPLES

Execute the **ncheck** command on all *special* in **/etc/fstab:**

```
ncheck
```

Execute the **ncheck** command on HFS file system **/dev/dsk/c1d2s0**:

```
ncheck -F hfs /dev/dsk/c1d2s0
```

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

When the file system structure is improper, **??** denotes the “parent” of a parentless file and a path-name beginning with **...** denotes a loop.

AUTHOR

ncheck was developed by AT&T and HP.

FILES

/etc/default/fs	Specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

acl(5), fsck(1M), fstab(4), fs_wrapper(5), ncheck(1M), sort(1).

STANDARDS CONFORMANCE

ncheck: SVID2, SVID3

NAME

ncheck (vxfs) - generate pathnames from inode numbers for a VxFS file system

SYNOPSIS

/usr/sbin/ncheck [**-F vxfs**] [**-V**] [**-i** *ilist*] [**-a**] [**-s**] [**-S** *sector_list*] [**-o** *specific_options*] *special* ...

DESCRIPTION

The **ncheck** program generates a pathname-*vs*-inode-number list of files for the specified VxFS file system.

Some options accept a *range* as a value. A range consists of a single number, or two numbers separated by a "**-**", indicating an inclusive range of values. If "**-**" is specified and the first number is omitted, 0 is assumed. If the second number is omitted, the end of the file system is assumed.

Names of directory files are followed by "**/.**".

Options

-F vxfs Specifies the file-system type (**vxfs**).

-V Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-i *ilist* Limits the report to the files on the ilist that follows. The *ilist* must be separated by commas without spaces.

-a Allow printing of the names "**.**" and "**..**" (dot and dotdot), which are ordinarily suppressed.

-s Report only on special files and regular files with set-user-ID mode. This option may be used to detect violations of security policy.

-S *sector_list* Report on files containing or referencing the specified sector(s). Output consists of the fileset name, fileset index, inode number, and pathname of file or file type if a structural inode or attribute inode. Sectors not allocated to any file or file system structure are reported as **<free>**. Sectors not part of the file system are reported as **<unused>**. Unused or irrelevant fields are printed as "**-**".

sector_list consists of one or more *ranges* of sector numbers, separated by commas without intervening spaces. Multiple **-S** options accumulate.

-o *specific_options*

Specifies options specific to the VxFS file-system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for the VxFS-specific module of the command.

The available options are

m Print mode information (used in conjunction with **-i** option).

b=*block*

Print pathname containing file system block number *block*.

sector=*sector_range*

Report on all inodes containing or referencing the sector(s) in *sector_range*. The output includes the inode number, fileset index of the inode, sector(s) contained and the pathname or inode type. Inodes searched include structural inodes and attribute inodes, so a pathname is only generated when the sector is contained by a file. If the sector is not contained in any file, the inode type is printed as "**<free>**". Multiple **-o sector=** options accumulate.

block=*block_range*

Print information on all inodes containing or referencing block numbers in the range specified. The output format is the same as that for **-o sector=**, but the units used are file-system blocks rather than sectors.

surface[**=***sector_range*]

Perform a surface analysis. If a *sector_range* is specified perform a surface analysis only for that range. All the sector are read and if the read of a sector fails, its sector number is printed. If any bad sectors are found, **ncheck** treats the list of bad sector as input to the **-o sector=**# option and produces a list of containing or referencing inodes.

EXAMPLES

Report on all inodes or file system structures containing or referencing sector 20 through 35 (inclusive) in the file system **/dev/vg01/rlvol1**:

```
ncheck -F vxfs -S 20-35 /dev/vg01/rlvol1
```

Same as above but report on all inodes or file system structures referencing any sector in the file system **/dev/vg01/rlvol1**:

```
ncheck -F vxfs -S - /dev/vg01/rlvol1
```

DIAGNOSTICS

When the file-system structure is improper, "**???**" denotes the "parent" of a parentless file, a pathname beginning with "**...**" denotes a loop, and a pathname beginning with "*******" denotes a directory entry whose "**..**" (dotdot) entry is not in accord with the directory in which it was found.

FILES

/etc/fstab	Static information about the file systems.

SEE ALSO

sort(1), fsck(1M), fsck_vxfs(1M), ncheck(1M). inode numbers for VxFS file system

NAME

ndd - network tuning

SYNOPSIS

ndd -get *network_device parameter*

ndd -set *network_device parameter value*

ndd -h sup[**ported**]

ndd -h unsup[**ported**]

ndd -h [*parameter*]

ndd -c

DESCRIPTION

The **ndd** command allows the examination and modification of several tunable parameters that affect networking operation and behavior. It accepts arguments on the command line or may be run interactively. The **-h** option displays all the supported and unsupported tunable parameters that **ndd** provides. Valid *network_device* names are: **/dev/arp**, **/dev/ip**, **/dev/rawip**, **/dev/tcp**, and **/dev/udp**. Set *parameter* to **?** to get a list of parameters for a particular *network_device*.

ndd -get Get the value of the *parameter* for *network_device* and print the *value* to standard output. Returned numbers are always displayed as decimal strings.

ndd -set Set *parameter* for *network_device* to *value*.

All times are specified in milliseconds, e.g. 240000 for 4 minutes. Unless stated otherwise, numbers are assumed to be in decimal. Use "0x" prefix to specify hexadecimal values.

In general, all tunable parameters are global, i.e., they affect all instances of the network module. Some settings take effect immediately, while others are used to initialize data for an instance and will only affect newly opened streams.

ndd -h supported

Display all the supported tunable parameters. This set of parameters are supported by HP and detailed descriptions of these tunable parameters are available through the **-h** *parameter* command.

ndd -h unsupported

Display all the unsupported tunable parameters. This set of parameters are not supported by HP and modification of these tunable parameters are not suggested nor recommended. Setting any unsupported tunable parameters on your system may result in adverse effects to your networking operations.

ndd -h When *parameter* is specified, a detail description of the *parameter*, along with its minimum, maximum, and default value are displayed. If no parameter is specified, it displays all supported and unsupported tunable parameters.

ndd -c Read input from the configuration file **/etc/rc.config.d/nddconf** and set the tunable parameters. A user may specify tunable parameters in the nddconf configuration file, and these parameters will be set automatically each time the system boots.

DIAGNOSTICS

When the command fails, an error message is printed to the standard error and the command terminates with an exit value of one.

WARNINGS

Care must be used when setting parameters for a network_device. Setting a tunable parameter to an inappropriate value can result in adverse affects to your networking operations.

EXAMPLES

To get help information on all supported tunable parameters:

```
ndd -h supported
```

To get a detail description of the tunable parameter, **ip_forwarding**:

```
ndd -h ip_forwarding
```

To get a list of all TCP related parameters:

```
ndd -get /dev/tcp ?
```

To get the current value of the tunable parameter, **ip_forwarding**:

```
ndd -get /dev/ip ip_forwarding
```

To set the value of the default TTL parameter for UDP to 128:

```
ndd -set /dev/udp udp_def_ttl 128
```

FILES

/etc/rc.config.d/nddconf	Contains tunable parameters that will be set automatically each time the system boots.

AUTHOR

ndd was developed by HP.

NAME

netfmt - format tracing and logging binary files

SYNOPSIS

/usr/sbin/netfmt -s [**-t** *records*] [[**-f**] *file_name*]

/usr/sbin/netfmt -p [**-c** *config_file*]

/usr/sbin/netfmt [**-c** *config_file*] [**-F**] [**-t** *records*] [**-v**] [**-l**] [**-n**]
[**-N** | [**-1** [**-L**] [**-T**]]] [[**-f**] *file_name*]

DESCRIPTION

netfmt is used to format binary trace and log data gathered from the network tracing and logging facility (see *nettl*(1M)). The binary trace and log information can be read from a file or from standard input (if standard input is a tty device, an informative message is given and **netfmt** quits). Formatted data is written to standard output.

Formatting options are specified in an optional filter configuration file. Message inclusion and format can be controlled by the filter configuration file. If no configuration commands are specified, all messages are fully formatted. A description of the filter configuration file follows the option descriptions.

Options

netfmt recognizes the following command-line options and arguments:

-s Display a summary of the input file. The summary includes the total number of messages, the starting and ending timestamps, the types of messages, and information about the system that the data was collected on. The contents of the input file are not formatted; only a summary is reported.

-t *records* Specifies the number of records from the tail end of the input file to format. This allows the user to bypass extraneous information at the beginning of the file, and get to the most recent information quickly. The maximum number of *records* that can be specified is 1000. If omitted, all records are formatted. The **-t** option is not allowed when the input file is a FIFO (pipe).

n

-f *file_name* Specifies the input file containing the binary log or trace data. *file_name* may not be the name of a tty device. Other options may impose additional restrictions on the type of the input file allowed. If omitted, data is read from standard input.

-p Parse input: this switch allows the user to perform a syntax check on the *config_file* specified by the **-c** parameter. All other parameters are ignored. If the syntax is correct, **netfmt** terminates with no output or warnings.

-c *config_file* Specifies the file containing formatter filter configuration commands. Syntax for the commands is given below. When **-c** is omitted the file **$HOME/.netfmtrc** is read for both logging and tracing filter configuration commands if it exists.

-F Follow the input file. Instead of closing the input file when end of file is encountered, **netfmt** keeps it open and continues to read from it as new data arrives. This is especially useful for watching events occur in real time while troubleshooting a problem. Another use would be for recording events to a console or hard-copy device for auditing. (Note that console logging is controlled by the configuration files **/etc/nettlgen.conf** and **/var/adm/conslog.opts**; see *nettlgen.conf*(4).) The **-F** option is not allowed when the input file is redirected.

The following options are not supported by all subsystems. If a subsystem does not support an option, that option is ignored during formatting of data from that subsystem. Consult the product documentation of the subsystem for information regarding the support of these options.

-v Enables output of verbose information. This includes additional cause and action text with formatted output. This information describes the possible cause of the message and any actions that may be required by the subsystem.

After the contents of the input file have been formatted a summary of the file is displayed. When this option is used with the **-t** option, only a summary of the last *records* is reported. No summary is produced when this option is used in conjunction with the **-F** option or if formatting is interrupted.

`-l`	(*ell*) Turn off inverse video highlighting of certain traced fields. Use this flag when sending formatted trace data to a line printer. By default, certain fields in the trace file are highlighted in inverse video when viewing the formatted trace format at a terminal that supports highlighting.
`-n`	Shows port numbers and network addresses(such as IP and x121) as numbers (normally, `netfmt` interprets numbers and attempts to display them symbolically).
`-N`	Enables "nice" formatting where Ethernet/IEEE802.3, SLIP, IP, ICMP, IGMP, TCP, UDP, and RPC packets are displayed symbolically. All remaining user data is formatted in hexadecimal and ASCII.
`-1`	*(one)* Attempts to tersely format each traced packet on a single line. If `-L` and/or `-T` options are used, the output lines will be more than 80 characters long.
`-T`	Places a time stamp on terse tracing output. Used with the `-1` (*minus one*) option.
`-L`	Prefixes local link address information to terse tracing output. Used with the `-1` (*minus one*) option.

Filter Configuration File

Note: Filter configuration file syntax converges the syntax used with the obsolete `nettrfmt` network trace formatter and `netlogfmt` network log formatter commands with new `netfmt` syntax for controlling formatter options. The first section below describes the general use and syntax of the filter configuration file. Specific options for subsystem Naming and Filtering are listed in the *Subsystem Filtering* section below.

The filter configuration file allows specification of two types of information:

- Specify options in order to control how the input data is to be formatted. These options determine what the output looks like and allow a user to select the best format to suit their needs.
- Specify filters in order to precisely tailor what input data is to be discarded and what is to be formatted. **Global filters** control all subsystems; **subsystem filters** pertain only to specific subsystems.

A filter is compared against values in the input data. If the data matches a filter, the data is formatted; otherwise, the input data is discarded. A filter can also specify *NOT* by using `!` before the filter value in the configuration file. If the input data matches a *NOT* filter, it is discarded. A filter can also be a "wildcard" (matching any value) by specifying an asterisk `*` before the filter value in the configuration file. "Wild card" filters pass all values of the input data. Specifying `!*` as the filter means *NOT ALL*.

n

Filter Configuration File Syntax

- The formatter ignores white space, such as spaces or tabs. However, newlines (end of line characters) are important, as they terminate comments and filter specifications.
- The formatter is not case sensitive. For example `error` and `ERROR` are treated as equivalent.
- To place comments in the file, begin each comment line with a `#` character. The formatter ignores all remaining characters on that line. There are no inline comments allowed.
- An exclamation point (`!`) in front of an argument indicates *NOT*. This operator is not supported for timestamp, log instance, and ID filtering.
- The asterisk (`*`), when used as an argument, indicates *ALL*. Since the default for all formatting options is *ALL*, it is unnecessary to use the asterisk alone. It can be used along with the exclamation point, (`!*`) to indicate *NOT ALL*. This operator is not available for timestamp, log instance, and ID filtering.

Global Filtering:

Global filtering commands start with the word `formatter`, followed by the keywords `verbosity`, `mode`, `option`, or `filter`.

`formatter verbosity` *value*,
value should be either of

`high`	Enables output of netfmt internal debugging information to standard error. Same as the `-v` option.

low	No internal debugging information is to be displayed.

formatter mode value,
value should be one of

raw	Dumps out the messages in hex format.
nice	Enables "nice" formatting. Same as **-N** option.
terse	Attempts to tersely format each traced packet on a single line. Same as **-1** (*minus one*) option.
normal	Normal formatting.

formatter option [**!**] *value,*
value should be

suppress	Normally repeated lines in hex output are condensed into a single line and a message stating that redundant lines have been skipped is displayed. Specifying **!suppress** will print all redundant data. This is useful when the formatted output is used as input into other commands.
highlight	Normally the formatter will highlight certain fields in its trace output in inverse video. Specifying **!highlight** will turn this feature off. Same as the **-l** (*minus ell*) option.

formatter filter *type* [**!**] *value* | *****
Six *types* of filtering are provided:

class	log classes
kind	trace kinds
id	connection, process, path, and user
log instance	specific thread of events
subsystem	subsystem names
time	specify ranges of time(s)

The following combinations are recognized:

formatter filter class *value* [*subsystem*]
value indicates the log class. This option allows the user to select one or more classes to be formatted. Initially all log classes are formatted. Only one class is allowed per line. Classes in multiple lines are logically "OR"ed. The optional *subsystem* name sets the class filter only for the specified subsystem. The log classes are:

INFORMATIVE	Describes routine operations and current system values.
WARNING	Indicates abnormal events possibly caused by subsystem problems.
ERROR	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
DISASTER	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

formatter filter Connection_ID *value*
formatter filter Device_ID *value*
formatter filter Path_ID *value*
formatter filter Process_ID *value*
formatter filter User_ID *value*
value specifies the ID number of the messages to format. Last-entered value has precedence over any previous ones. See the record header in the formatted output to determine which ID numbers to filter on. The **!** operator is *not* allowed in *value*.

formatter filter kind *value* [*subsystem*]
value can either be an established trace kind or a mask. A mask is a hexadecimal representation of a (set of) trace kind(s). Masks in multiple lines are logically "OR"ed. The optional *subsystem* name sets the kind filter only for the specified subsystem. Trace kinds and their corresponding masks are:

Name	Mask	Name	Mask
hdrin	0x80000000	**state**	0x04000000
hdrout	0x40000000	**error**	0x02000000
pduin	0x20000000	**logging**	0x01000000
pduout	0x10000000	**loopback**	0x00800000
proc	0x08000000		

hdrin	Inbound Protocol Header.
hdrout	Outbound Protocol Header.
pduin	Inbound Protocol Data Unit (including header and data).
pduout	Outbound Protocol Data Unit (including header and data).
proc	Procedure entry and exit.
state	Protocol or connection states.
error	Invalid events or condition.
logging	Special kind of trace that contains a log message.
loopback	Packets whose source and destination system is the same.

formatter filter log_instance *value*

value specifies the log instance number of the messages to filter. Selecting a log instance allows the user to see the messages from a single thread of network events. Only one log instance is allowed per filter configuration file. The log instance can not be negated with the **!** operator.

formatter filter subsystem *value*

value specifies the subsystem name. Available subsystem names can be listed by using the command:

```
nettlconf -status
```

Only one subsystem name is allowed per line; multiple lines "OR" the request. To eliminate a given subsystem name, use the **!** operator, which formats all subsystems except those excluded by the list of negated subsystems. To include all subsystems (the default), use the ***** operator. To eliminate all subsystems, use the **!*** operator.

formatter filter time_from *value*
formatter filter time_through *value*

time_from indicates the inclusive starting time. **time_through** indicates the inclusive ending time. *value* consists of *time_of_day* and optionally *day_of_year*, (usually separated by one or more blanks for readability).

time_of_day specifies the time on the 24-hour clock in hours, minutes, seconds and decimal parts of a second (resolution is to the nearest microsecond). Hours, minutes and seconds are required; fractional seconds are optional. *time_of_day* format is *hh*:*mm*:*ss*.*dddddd*.

day_of_year specifies the day of the year in the form month/day/year in the format: *mm*/*dd*/*yy*. Specify month and day numerically, using one or two digits. For example, January can be specified as **1** or **01**; the third day of the month as **3** or **03**. Specify the year by its last two digits. For example, specify 1993 as **93**. *day_of_year* is an optional field; the current date is used as a default.

The **time_from** specification includes *only* those records starting from the resolution of time given. For example, if the *time_of_day* for **time_from** is specified as 10:08:00, all times before that, from 10:07:59.999999 and earlier, are excluded from the formatted output. Records with times of 10:08:00.000000 and later are included in the formatted output. Similarly, the **time_through** specification includes *only* up to the resolution of time given. For example, if the *time_of_day* for **time_through** is specified as 10:08:00, all records with times after that, from 10:08:00.000001 onward, are excluded from the formatted output.

Subsystem Filtering

Note: Global filtering described above takes precedence over individual subsystem tracing and logging filtering described below.

Subsystem filters are provided to allow filtering of data for individual subsystems or groups of subsystems. Their behavior varies among individual subsystems. Subsystem filters are valid only when the corresponding subsystems have been installed and configured on the system. See the subsystem documentation for a description of supported subsystem filters and their behavior.

Subsystem filtering commands start with the name of the subsystem followed by the subsystem filter keywords. However, to provide convenience and backwards compatibility, several other filter keywords are provided for the group of LAN subsystems: **NAME** and **FILTER**. Currently, four types of subsystem filters are provided: LAN, X25, STREAMS, and OTS. The collection of LAN subsystems use the subsystem filters identified by the **FILTER** and **NAME** keywords and the collection of OTS subsystems use the subsystem filters with the **OTS** keyword. The collection of X25 subsystems start their filter commands with the X25 subsystem names.

LAN Naming and Filtering

LAN naming can be used to symbolically represent numbers with more recognizable labels.

name *nodename value*

nodename is a character string to be displayed in place of all occurrences of *value*. *value* is a (IEEE802.3/Ethernet) hardware address consisting of 6 bytes specified in hexadecimal (without leading "0x"), optionally separated by **-**. **netfmt** substitutes all occurrences of *value* with *nodename* in the formatted output. The mapping is disabled when the **-n** option is used. This option applies to tracing output only.

LAN filtering is used to selectively format packets from the input file. There are numerous filter types, each associated with a particular protocol layer:

Filter Layer	Filter Type	Description
Layer 1	**dest**	hardware destination address
	source	hardware source address
	interface	software network interface
Layer 2	**ssap**	IEEE802.2 source sap
	dsap	IEEE802.2 destination sap
	type	Ethernet type
Layer 3	**ip_saddr**	IP source address
	ip_daddr	IP destination address
	ip_proto	IP protocol number
Layer 4	**tcp_sport**	TCP source port
	tcp_dport	TCP destination port
	udp_sport	UDP source port
	udp_dport	UDP destination port
	connection	a level 4 (TCP, UDP) connection
Layer 5	**rpcprogram**	RPC program
	rpcprocedure	RPC procedure
	rpcdirection	RPC call or reply

n

Filtering occurs at each of the five layers. If a packet matches any filter within a layer, it is passed up to the next layer. The packet must pass every layer to pass through the entire filter. Filtering starts with Layer 1 and ends with Layer 5. If no filter is specified for a particular layer, that layer is "open" and all packets pass through. For a packet to make it through a filter layer which has a filter specified, it must match the filter. Filters at each layer are logically "OR"ed. Filters between layers are logically "AND"ed.

LAN trace and log filters use the following format:

filter *type* [**!**] *value* | *****

filter is the keyword identifying the filter as a LAN subsystem filter.

The following filters are available for LAN tracing.

filter connection *value*

value takes the form:

*local_addr***:***port remote_addr***:***port*

where *local_addr* and *remote_addr* can be a hostname or a 4-byte Internet address specified in decimal dot notation (see *inet*(3N) for more information on Internet addresses and decimal dot notations). *port* can be a service name or an *integer*. *integer* represents a port and can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or base-10 integers (0 through 65 535).

filter dest *value*
filter source *value*
value is a hardware address consisting of 6 bytes specified in hexadecimal (without leading **0x**), optionally separated by **-**.

filter dsap *value*
filter ssap *value*
value is a hexadecimal integer of the form: **0x***digit*; an octal integer of the form: **0***digits*; or a base-ten integer, 0 through 255.

filter interface *value*
value identifies a network interface and takes the form: **lan***n* for LAN interface, or **lo***n* for loopback interface, where *n* is the logical unit number, as in **lan0**.

filter ip_daddr *value*
filter ip_saddr *value*
value is a hostname or a 4-byte Internet address specified in decimal dot notation (see *inet*(3N) for more information on Internet addresses and decimal dot notations).

filter ip_proto *value*
value is a hexadecimal integer of the form: **0x***digit*; an octal integer of the form: **0***digits*; or a base-ten integer, 0 through 255 (see *protocols*(4) for more information on protocol numbers).

filter tcp_dport *value*
filter tcp_sport *value*
filter udp_dport *value*
filter udp_sport *value*
value is a port number designated as a 2-byte integer value or a service name. The integer value can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or a base-10 integer (0 through 65 535).

filter rpcprogram *value*
value is a RPC program name or an integer RPC program number (see *rpc*(4) for more information on RPC program names). The integer value can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or a base-10 integer (0 through 65 535).

filter rpcprocedure *value*
value is an integer RPC procedure number. The integer value can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or a base-10 integer (0 through 65 535).

filter rpcdirection *value*
value can be either **call** or **reply**.

filter type *value*
value is a hexadecimal integer of the form: **0x***digits*; an octal integer of the form: **0***digits*; or a base-ten integer (0 through 65 535).

LAN log filtering command has the following form:

filter subsystem *value*
value takes the form:

subsys_name **event** *event_list*

where *subsys_name* is a subsystem name obtained using the **nettlconf -status** command or one of the following abbreviations:

axin	**bufs**	**caselib**	**caserouter**
ip	**ipc**	**lan**	**loopback**
netisr	**nfs**	**nft**	**ni**
nsdiag	**nse**	**probe**	**pxp**
rlbdaemon	**sockregd**	**strlog**	**tcp**
timod	**tirdwr**	**udp**	

event_list takes the form:

event_spec [**,** *event_spec* **. . .**]

n

where *event_spec* takes one of the three forms:

[!] *integer* **[!]***range* **[!]****

integer is an integer in hexadecimal (leading **0x**), octal (leading **0**), or decimal, which specifies a log event for the subsystem indicated.

range takes the form *integer***-***integer*, and indicates an inclusive set of events.

X25 Naming and Filtering

The X25 product provides capabilities to assign symbolic names to important numbers and to filter log events and trace messages. See *x25log*(1M) and *x25trace*(1M) for more information about X25 naming and filtering.

OTS Filtering

The OTS subsystem filter allows filtering of the message ID numbers that are typically found in the data portion of an OTS subsystem's log or trace record. The OTS subsystem filter is effective for any subsystem that is a member of the OTS subsystem group.

OTS trace filtering configuration commands have the following form in *config_file*:

OTS [*subsystem*] **msgid** [**!**] *message_ID* | *

Keywords and arguments are interpreted as follows:

OTS Identifies the filter as an OTS subsystem filter.

subsystem One of the following group of OTS subsystems:

OTS	**ACSE_PRES**	**NETWORK**
TRANSPORT	**SESSION**	

Note: The absence of *subsystem* implies that the filter applies to all OTS subsystems.

message_ID is the value of the message ID to filter. A message ID is used by OTS subsystems to identify similar types of information. It can be recognized as a 4 digit number contained in brackets (**[ ]**) at the beginning of an OTS subsystem's trace or log record. Initially all *message_ID*s are enabled for formatting. To format records with specific *message_ID*s, turn off all message IDs using the **!*** operator, then selectively enable the desired message IDs. Only one *message_ID* is allowed on each line. Multiple lines are "OR"ed together.

STREAMS Filtering

The STREAMS subsystem filter allows filtering on some fields of the messages logged by STREAMS modules and drivers. See *strlog*(7) for more information.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported in data. Single-byte character codesets are supported in filenames.

DEPENDENCIES

netfmt only recognizes subsystems and filters from products which have been installed and configured.

WARNINGS

The syntax that was used for the obsolete LAN trace and log options has been mixed with the syntax for the **netfmt** command such that any old options files can be used without any changes. The combination of syntax introduces some redundancy and possible confusion. The global filtering options have the string **formatter filter** as the first two fields, while the LAN filtering options merely have the string **filter** as the first field. It is expected that the older LAN filtering options may change to become more congruent with the global filtering syntax in future releases.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are executed. These commands will not operate if the file becomes corrupted (see *nettl*(1M) and *netfmt*(1M)).

DIAGNOSTICS

Messages describe illegal use of **netfmt** command and unexpected EOF encountered.

EXAMPLES

The first group of examples show how to use command line options.

1. Format the last 50 records in file **/var/adm/nettl.LOG00** (the default log file):

```
netfmt -t 50 -f /var/adm/nettl.LOG00
```

2. Use the follow option to send *all* log messages to the console (normally, only **DISASTER**-class log messages are sent to the console in console form):

```
netfmt -f  /var/adm/nettl.LOG00  -F  > /dev/console
```

3. Monitor all log messages in a **hpterm** window:

```
hpterm -e /usr/sbin/netfmt -F -f /var/adm/nettl.LOG00
```

4. Read file **/var/adm/trace.TRC1** for binary data and use **conf.file** as the filter configuration file:

```
netfmt -c conf.file -f /var/adm/trace.TRC1
```

The remaining examples show how to specify entries in the filter configuration file used with the **-c** option.

1. Tell **netfmt** to format only **INFORMATIVE**-class log messages coming from the **NS_LS_IP** subsystem between 10:31:53 and 10:41:00 on 23 November 1993.

```
formatter   filter   time_from    10:31:53   11/23/93
formatter   filter   time_through 10:41:00   11/23/93
formatter   filter   class        !*
formatter   filter   class        INFORMATIVE
formatter   filter   subsystem    !*
formatter   filter   subsystem    NS_LS_IP
```

2. Map hardware address to name(LAN):

```
name        node1       08-00-09-00-0e-ca
name        node3       02-60-8c-01-33-58
```

3. Format only packets from either of the above hardware addresses:

```
filter      source      08-00-09-00-0e-ca
filter      source      02-60-8c-01-33-58
```

4. Format all packets transmitted from the local node, **local**, to the remote node, **192.6.1.3**, which reference local TCP service ports **login** or **shell**, or remote UDP port **777**:

```
filter      ip_saddr    local
filter      ip_daddr    192.6.1.3
filter      tcp_sport   login
filter      tcp_sport   shell
filter      udp_dport   777
```

5. Format a TCP connection from local node **node2** to **192.6.1.3** which uses **node2** service port **ftp** and remote port **1198**.

```
filter      connection  node2:ftp   192.6.1.3:1198
```

6. Format all packets except those that use interface **lan0**:

```
filter      interface   ! lan0
```

7. Format all logged events for subsystem **ip**. No other events are formatted. (By default, all events are formatted):

```
filter      subsystem   ip   event      *
```

8. Format only event **5003** for subsystem **ip**. Format all events except **3000** for subsystem **tcp**. No other events are formatted.

```
filter      subsystem   ip    event  5003
filter      subsystem   tcp   event  *,!3000
```

9. Format only events **5003**, **5004**, **5005**, and **5006** for subsystem **ip**. Format all events except events **3000**, **3002**, and **3003** for subsystem **tcp**. No other events are formatted:

```
filter      subsystem      ip    event  5003-5006
filter      subsystem      tcp   event  *,!3000,!3002-3003
```

10. Format only those records containing message IDs **9973** and **9974** for subsystem **session** and those not containing message ID **9974** for subsystem **transport**. All records from other subsystems are formatted:

```
ots session    msgid          !*
ots session    msgid          9973
ots session    msgid          9974
ots transport  msgid          !9974
```

11. Combine LAN and general filtering options into one configuration file. Format 15 minutes of pduin and pduout data starting at 3:00 PM on 2 April 1990 for data from **lan0** interface.

```
formatter     filter       kind           0x30000000
formatter     filter       time_from      15:00:00 04/02/90
formatter     filter       time_through   15:15:00 04/02/90
filter        interface    !*
filter        interface    lan0
```

FILES

/etc/nettlgen.conf	default subsystem configuration file
/var/adm/conslog.opts	default console logging options filter file
$HOME/.netfmtrc	default filter configuration file if the **-c config_file** option is not used on the command line.

SEE ALSO

nettl(1M), nettlconf(1M), nettlgen.conf(4), strlog(7).

AUTHOR

netfmt was developed by HP.

NAME

nettl - control network tracing and logging

SYNOPSIS

/usr/sbin/nettl -start

/usr/sbin/nettl -stop

/usr/sbin/nettl -firmlog 0|1|2 -card *dev_name* ...

/usr/sbin/nettl -log *class* ... **-entity** *subsystem* ...

/usr/sbin/nettl -status [**log** |**trace** |**all**]

/usr/sbin/nettl -traceon *kind* ... **-entity** *subsystem* ... [**-card** *dev_name* ...]
[**-file** *tracename*] [**-m** *bytes*] [**-size** *portsize*] [**-tracemax** *maxsize*]

/usr/sbin/nettl -traceoff -entity *subsystem* ...

DESCRIPTION

The **nettl** command is a tool used to capture network events or packets. Logging is a means of capturing network activities such as state changes, errors, and connection establishment. Tracing is used to capture or take a snapshot of inbound and outbound packets going through the network, as well as loopback or header information. A subsystem is a particular network module that can be acted upon, such as **ns_ls_driver**, or **X25L2**. **nettl** is used to control the network tracing and logging facility.

Except for the **-status** option, **nettl** can be used only by users who have an effective user ID of 0.

Options

nettl recognizes the following options, which can be used only in the combinations indicated in SYNOPSIS. Some option and argument keywords can be abbreviated as described below. All keywords are case-insensitive.

-start (Abbr.: **-st**)
Used alone without other options.

Initialize the tracing and logging facility, start up default logging, and optionally start up console logging. Logging is enabled for *all* subsystems as determined by the **/etc/nettlgen.conf** file. Log messages are sent to a log file whose name is determined by adding the suffix **.LOG00** to the log file name specified in the **/etc/nettlgen.conf** configuration file. Console logging is started if console logging has been configured in the **/etc/nettlgen.conf** file. See *nettlconf*(1M) and *nettlgen.conf*(4) for an explanation of the configuration file. If the log file (with suffix) already exists, it is opened in *append* mode; that is, new data is added to the file. The default name is **/var/adm/nettl** (thus logging starts to file **/var/adm/nettl.LOG00**). See "Data File Management" below for more information on how the log file is handled.

A **nettl -start** command is performed during system startup if the **NETTL** variable in the **/etc/rc.config.d/nettl** file has a value of **1**.

Note: It is strongly recommended that the tracing and logging facility be turned on before any networking is started and remain on as long as networking is being used. Otherwise, information about disasters will be lost. To minimize the impact on the system, all subsystems can be set with the **-log** option to capture only **disaster**-class log messages.

-stop (Abbr.: **-sp**)
Used alone without other options.

Terminate the trace/log facility. Once this command is issued, the trace/log facility is no longer able to accept the corresponding trace/log calls from the network subsystems.

Note: See note for the **-start** option.

-card *dev_name* ...
(Abbr.: **-c**)
This option is required by the X.25 subsystems; it is optional for other subsystems. Some subsystems do not support this option.

Limit the trace information gathered to only the data that comes from the specified network interface card. More than one *dev_name* can be specified at a time in order to trace multiple network interfaces.

dev_name specifies a device which corresponds to a network interface card that has been installed and configured. It can be either an integer representing the network interface, or the device file name of the network interface. Some subsystems do not support both types of *dev_name*. For example, the X25 subsystems require that *dev_name* be a device file name. The product documentation for the subsystems should explain if the **-card** option is applicable and how to choose an appropriate *dev_name*.

If *dev_name* is not an integer it is assumed to be a device file name. The path prefix **/dev/** will be attached in front of *dev_name* if it is not an absolute path name to form the device file name, **/dev/***dev_name*. *dev_name* must refer to a valid network device file.

-entity all
-entity *subsystem* ...

(Abbr.: **-e**)

Limit the action of **-log**, **-traceoff**, or **-traceon** to the specified protocol layers or software modules specified by *subsystem*.

The number and names of *subsystem*s on each system are dependent on the products that have been installed. Use the command **nettlconf -status** to obtain a full listing of supported subsystems and the products that own them.

Examples of OSI subsystems:

acse_pres	**ftam_init**	**mms**
asn1	**ftam_resp**	**network**
cm	**ftam_vfs**	**ots**
em	**ftp_ftam_gw**	**transport**
ftam_ftp_gw	**hps**	**ula_utils**

Examples of LAN subsystems:

ns_ls_driver	**ns_ls_loopback**	**ns_ls_ni**
ns_ls_icmp	**ns_ls_netisr**	**ns_ls_tcp**
ns_ls_igmp	**ns_ls_nfs**	**ns_ls_udp**
ns_ls_ip	**ns_ls_nft**	**ns_ls_x25**

Two X.25-specific subsystems are used for tracing only:

X25L2 **X25L3**

-file *tracename*

(Abbr.: **-f**)
Used with the first **-traceon** option only.

The first time the **-traceon** keyword is used, it initializes tracing, creating a file *tracename***.TRC0** which receives the binary tracing data. If a trace file of the name *tracename***.TRC0** already exists the binary trace data is appended to the end of the file.

To start a fresh trace file, first turn off tracing then turn it back on again using a different *tracename*. See "Data File Management" below for more information on file naming.

If **-file** is omitted, *binary* trace output goes to standard output. If standard output is a terminal device, an error message is issued and no tracing is generated.

-firmlog 0|1|2

(Abbr.: **-fm**)
Requires the **-card** option.
Series 800 and X.25 only.

Set the X.25/800 interface card logging mask to level 0, 1, or 2. The default level is 0. The X.25/800 interface logs a standard set of messages. A level of 1 specifies cautionary messages as well as the default messages. A level of 2 specifies information messages in addition to the cautionary and default messages. This option is recognized only by the **ns_ls_x25** subsystem.

-log *class* ... (Abbr.: **-l**)
Requires the **-entity** option.

Control the class of log messages that are enabled for the subsystems specified by the **-entity** option.

class specifies the logging class. Available classes are:

Full	Abbr.	Mask
informative	**i**	1
warning	**w**	2
error	**e**	4
disaster	**d**	8

informative	Describes routine operations and current system values.
warning	Indicates abnormal events possibly caused by subsystem problems.
error	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
disaster	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

Classes can be specified as keywords or as a single numeric mask depicting which classes to log. The mask is formed by adding the individual masks of the log classes. If you choose to indicate several classes at once, be sure to separate each log class with a space.

disaster logging is always on. The default logging classes for each subsystem is configured into the configuration file, **/etc/nettlgen.conf**. When the tracing/logging facility is started, the information in the configuration file is read and subsystems are enabled for logging with the specified classes. To change the log class, use the "**nettl -log** *class* **-entity** *subsystem*" command with a new log class value. If desired, the command can be run for different log classes and different entities.

n

-m *bytes* Specify the number of bytes (*bytes*) of each trace record to trace. This option allows the user to specify the number of bytes to be captured in the trace packet. The user may prefer not to capture an entire PDU trace, such as when the user is only interested in the header.

The maximum value for *bytes* is 2000. By default, the entire packet is traced. A value of 0 will also cause the entire packet to be traced. This option currently applies only to kernel subsystems.

-size *portsize*
(Abbr.: **-s**)
Used with first **-traceon** option only.

Set the size in kilobytes (KB) of the trace buffer used to hold trace messages until they are written to the file. The default size for this buffer is 68 KB. The possible range for *portsize* is 1 to 1024. Setting this value too low increases the possibility of dropped trace messages from fast subsystems.

-status log
-status trace
-status [**all**]
(Abbr.: **-ss**)
Used alone without other options.

Report the tracing and logging facility status. The facility must be operational, that is, **nettl -start** has been completed. The additional options define the type of trace or log information that is to be displayed. The default value is **all**.

log	Log status information
trace	Trace status information
all	Trace and log status information

-tracemax *maxsize*
(Abbr.: **-tm**)
Used with first **-traceon** option only.

Tracing uses a circular file method such that when one file fills up, a second is used. Two trace files can exist on a system at any given time. See "Data File Management" below for more information on file behavior.

maxsize specifies the maximum size in kilobytes (KB) of both trace files combined. The default value for the combined file sizes is 1000 KB. The possible range for *maxsize* is 100 to 99999.

-traceoff (Abbr.: **-tf**)
Requires the **-entity** option.

Disable tracing of *subsystem*s specified by the **-entity** option. If **all** is specified as an argument to the **-entity** option, all tracing is disabled. The trace file remains, and can be formatted by using the **netfmt** command to view the trace messages it contains (see *netfmt*(1M)).

-traceon all
-traceon *kind* ...
(Abbr.: **-tn**)
Requires the **-entity** option. The **-card** option is required for X.25 subsystems. Other options are not required.

Start tracing on the specified subsystems. The tracing and logging facility must have been initialized by **nettl -start** for this command to have any effect. The default trace file is standard output; it can be overridden by the **-file** option. If standard output is a terminal device, then an informative message is displayed and no trace data is produced.

When tracing is enabled, every operation through the subsystems is recorded if the *kind* mask is matched.

kind defines the trace masks used by the tracing facility before recording a message. If **-traceon all** is specified, all trace masks are enabled. *kind* can be entered as one or several of the following keywords or masks:

keyword	mask	keyword	mask
hdrin	**0x80000000**	**state**	**0x04000000**
hdrout	**0x40000000**	**error**	**0x02000000**
pduin	**0x20000000**	**logging**	**0x01000000**
pduout	**0x10000000**	**loopback**	**0x00800000**
proc	**0x08000000**		

hdrin	Inbound Protocol Header.
hdrout	Outbound Protocol Header.
pduin	Inbound Protocol Data Unit (including header and data).
pduout	Outbound Protocol Data Unit (including header and data).
proc	Procedure entry and exit.
state	Protocol or connection states.
error	Invalid events or condition.
logging	Special kind of trace that contains a log message.
loopback	Packets whose source and destination system is the same.

For multiple *kind*s, the masks can be specified separately or combined into a single number. For example, to enable both **pduin** and **pduout** (to trace all packets coming into and out of the node) use either **pduin pduout** or **0x10000000 0x20000000** or the combination **0x30000000**.

Not all subsystems support all trace *kind*s. *No* error is returned if a given subsystem does not support a particular trace *kind*.

n

If a **-traceon** is issued on a subsystem that is already being traced, the tracing mask and optional values are changed to those specified by the new command, but the new **-file**, **-size**, and **-tracemax** options are ignored and a message is issued.

If **-entity all** is specified, all recognized subsystems are traced except X.25-specific subsystems. To turn on tracing for X.25, use the command

```
nettl -traceon kind -e x.25_subsys -card dev_name
```

where the value of *x.25_subsys* is **X25L2** or **X25L3**.

Data File Management

Data files created by the tracing and logging facility require special handling by the facility that the user must be aware of. When files are created, they have the suffix **.LOG00** or **.TRC0** appended to them, depending on whether they are log or trace files, respectively. This scheme is used to keep the files distinct for cases where the user specifies the same name in both places. Also, the files implement a type of circular buffer, with new data always going into the file appended with **.LOG00** or **.TRC0**. When a file is full, it is renamed to the next higher number in its sequence; that is, *logname***.LOG01** or *tracename***.TRC1** and a new file named *logname***.LOG00** or *tracename***.TRC0** is created. Currently, only two generations of files are possible; thus, only two log files and two trace files can appear on the system simultaneously: *logname***.LOG00**, *logname***.LOG01**, *tracename***.TRC0**, and *tracename***.TRC1**.

Note: The file name prefix (*logname* or *tracename*) specified by the user must not exceed eight characters so that the file name plus suffix does not exceed fourteen characters. Longer names are truncated. To see the actual name of the trace or log file, use the **nettl -status all** command.

Console Logging

Console logging is used to display significant log events on the system console. The values in the **/etc/nettlgen.conf** file determine if console logging is to be started and the entries in the **/var/adm/conslog.opts** file determine what log messages will be reported to the console. The **nettlconf** command can be used to configure and maintain the information in the **/etc/nettlgen.conf** file (see *nettlconf*(1M)). If changes are made to these files, **nettl** must be stopped and restarted for the new information to take effect.

All log messages written to the console as a result of this configuration information are in a special short form. If more information is desired on the console, the **netfmt** formatter can be used to direct output to the console device. This may be most useful in an X windows environment.

Console logging may be disabled if conservation of system resources is valued more than notification of log events.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported in data; single-byte character code sets are supported in file names.

EXAMPLES

1. Initialize the tracing/logging facility:

   ```
   nettl -start
   ```

   (See note for the **-start** option.)

2. Display the status of the tracing/logging facility.

   ```
   nettl -status all
   ```

3. Change log class to **error** and **warning** for all the subsystems. **disaster** logging is always on for all subsystems.

   ```
   nettl -log e w -e all
   ```

4. Turn on inbound and outbound PDU tracing for the **transport** and **session** (OTS/9000) subsystems and send binary trace messages to file **/var/adm/trace.TRC0**.

   ```
   nettl -traceon pduin pduout -entity transport session \
         -file /var/adm/trace
   ```

5. Turn on outbound PDU tracing for X.25 level two, and subsystem **ns_ls_ip**. Trace messages go to the trace file set up in the previous example. This example also uses the abbreviated options. Tracing

for X.25 requires a **-card** option to indicate which X.25 card to trace.

```
nettl -tn pduout -e X25L2 ns_ls_ip -c x25_0
```

6. Determine status of tracing from the previous two examples.

```
nettl -status trace
```

The output should resemble the following:

```
Tracing Information:
Trace Filename:                  /var/adm/trace.TRC*
Trace file size(Kbytes): 1000
User's ID:               0       Buffer Size:           32768
Messages Dropped:        0       Messages Queued:       0

Subsystem Name:     Trace Mask:                 Card:
TRANSPORT           0x30000000
SESSION             0x30000000
NS_LS_IP            0x10000000
X25L2               0x10000000                  x25_0
```

7. Stop tracing for all subsystems.

```
nettl -traceoff -e all
```

8. Enable **pduin** and **pduout** tracing for **ns_ls_driver** (LAN driver) subsystem. Binary trace data goes to file **/var/adm/LAN.TRC0**.

   The **-file** option of this command is only valid the first time tracing is called. The trace file is not automatically reset with the **-file** option. To change the trace output file, stop tracing and start up again. This example assumes that the **-traceon** option is being used for the first time.

```
nettl -tn pduin pduout -e ns_ls_driver -file /var/adm/LAN
```

9. Terminate the tracing and logging facility.

```
nettl -stop
```

(See note for the **-start** option.)

n

WARNINGS

Although the **nettl** command allows the specification of all log classes and all trace kinds for all subsystems, many subsystems do not support all log classes and all trace kinds. No error or warning will be issued if a subsystem does not support a log class or trace kind. Refer to the product documentation of the subsystem for information on supported log classes and trace kinds.

Tracing to a file that resides on a NFS file system can impact system performance and result in loss of trace data. It is recommended that NFS file systems not be used to contain tracing output files.

Tracing to a file may not be able to keep up with a busy system, especially when extensive tracing information is being gathered. If some data loss is encountered, the trace buffer size can be increased. Be selective about the number of subsystems being traced, as well as the kinds of trace data being captured.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are run (see *nettl*(1M) and *netfmt*(1M)). If the file becomes corrupted, these commands will no longer be operational.

FILES

/dev/netlog	Kernel log pseudo-device file.
/dev/nettrace	Kernel trace pseudo-device file.
/etc/nettlgen.conf	Tracing and logging subsystem configuration file.
/etc/rc.config.d/nettl	Contains variables which control the behavior of **nettl** during system startup.
/var/adm/conslog.opts	Default console logging options filter file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.LOG00	Default log file as specified in **/etc/nettlgen.conf**.

AUTHOR

`nettl` was developed by HP.

SEE ALSO

netfmt(1M), nettlconf(1M), nettlgen.conf(4).

NAME

nettladm - network tracing and logging administration manager

SYNOPSIS

/opt/nettladm/bin/nettladm [**-t**|**-l**] [**-c** *filter_file*]

DESCRIPTION

The **nettladm** command is a tool used to administer network tracing and logging. It provides an interactive user interface to the nettl, netfmt, and nettlconf commands. The interface runs in either text terminal mode or in a Motif graphical environment. To run **nettladm** using Motif windows set the **DISPLAY** environment variable to match the system name (e.g., **DISPLAY=***system***:0.0**) prior to using the command.

The **nettladm** command starts a menu-driven program that makes it easy to perform network tracing and logging tasks with only limited specialized knowledge of HP-UX. **nettladm** is a self-guided tool, and context-sensitive help is available at any point by pressing the f1 function key.

Options

nettladm recognizes the following options:

-l	Shortcut to enter the “Logging Subsystems” (logging) area. This is the default.
-t	Shortcut to enter the “Tracing Subsystems” (tracing) area.
-c *filter_file*	Use the contents of *filter_file* as the default set of subsystem formatting criteria when creating reports within the “Create Report” area. The defaults can be overridden through the interface screens. Global filters (those beginning with the word **FORMATTER**) and comments are ignored. See *netfmt*(1M) for the description and syntax of *filter_file*.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported in data; single-byte character code sets are supported in file names.

n

WARNINGS

Changes to logging and tracing levels and states are not preserved across system reboots or stops and restarts from outside of the **nettladm** command. Permanent changes must be made to the **/etc/nettlgen.conf** file using the **nettlconf** command. Note that changes to console logging and all logging startup parameters are preserved.

Although the **nettladm** command allows the specification of all log classes and all trace kinds for all subsystems, many subsystems do not support all log classes and all trace kinds. No error or warning will be issued if a subsystem does not support a log class or trace kind. Refer to the product documentation of the subsystem for information on supported log classes and trace kinds.

The **nettladm** command reads the **/etc/nettlgen.conf** and **/var/adm/conslog.opts** files each time it is run (see *nettlgen.conf*(4)). If the files become corrupted, this command will no longer be operational.

DEPENDENCIES

nettladm runs in an X Windows environment as well as on the following kinds of terminals or terminal emulators:

- HP-compatible terminal with programmable function keys and on-screen display of function key labels.
- VT-100

FILES

/etc/nettlgen.conf	Tracing and logging subsystem configuration file.
/var/adm/conslog.opts	Default console logging options filter file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.LOG00	Default log file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.TRC0	Default trace file.

`/opt/nettladm/lib/X11/app-defaults/Nettladm`
X11 application defaults file.

AUTHOR

`nettladm` was developed by HP.

SEE ALSO

nettl(1M), netfmt(1M), nettlconf(1M), nettlgen.conf(4).

NAME

nettlconf - configure network tracing and logging command subsystem database

SYNOPSIS

/usr/sbin/nettlconf -status

/usr/sbin/nettlconf -L [**-console** *conlog*] [**-portsize** *logportsize*] [**-space** *maxlogspace*] [**-filename** *logfilename*] [**-option** *logoptfile*]

/usr/sbin/nettlconf [**-S**] **-id** *ssid* **-name** *ssname* [**-class** *logclass*] [**-kernel|-st[reams]**] **-lib** *sslib* **-msg** *ssmsgcat* [**-fmtfn** *fmtfunc*] [**-optfn** *optfunc*] **-group** *ssgrpname*

/usr/sbin/nettlconf -delete *ssid*

DESCRIPTION

nettlconf maintains the database file **/etc/nettlgen.conf** which contains information required by the **nettl** and **netfmt** commands (see *nettl*(1M) and *netfmt*(1M)). This database contains system logging information along with a description of each subsystem that uses the network tracing and logging facilities.

nettlconf can be used to update the system logging parameters or to add, update, and delete subsystem descriptions. If a subsystem already exists with the same *ssid*, the values given are substituted for those in the database; otherwise a new entry is created.

System administrators may use the **nettlconf** command to customize the system logging parameters stored in the database such as console logging behavior, the system log file name, the maximum system log file size, and the amount of memory required by the tracing and logging facilities.

nettlconf is also called during system startup to change the database to reflect the values of any relevant environment variables in the **/etc/rc.config.d/nettl** file.

Products use the **nettlconf** command during product installation to configure subsystems into the tracing and logging facility. The installation will execute the **nettlconf** command for each subsystem it installs in order to provide the information necessary for the subsystem to use the tracing and logging facilities.

Only users with appropriate privileges can invoke **nettlconf** to modify the configuration file.

Options

The following option can be used to view the system logging parameters and all subsystem descriptions from the **nettlgen.conf** database.

-status (abbrev: **-s**) display the contents of the database.

The following options can be used to update the system logging parameters.

-L This indicates that subsequent options apply to updating logging information. Changes to logging information will not take effect until **nettl** has been stopped and restarted. This is a required field.

-console *conlog* (abbrev: **-c**) *conlog* is set to 1 if console logging is to be enabled when **nettl** is started, 0 if not. (Console logging is used to report interesting events on the system console.) This is an optional field.

NOTE: during system startup *conlog* will be changed to match the value of the *NETTL_CONSOLE* variable in the **/etc/rc.config.d/nettl** file.

-portsize *logportsize*
(abbrev: **-p**) *logportsize* determines the number of outstanding messages possible in the log queue. The value is in multiples of 1024 bytes. Valid range is 1 through 64. The default is 8. This is an optional field.

-space *maxlogspace*
(abbrev: **-s**) *maxlogspace* is the maximum logging file space to be allowed. This is the combined size of the 2 ping-ponged log files. Specify the size in multiples of 1024 bytes. Valid range is 1 through 10240. Default is 1000. This is an optional field.

-filename *logfilename*
(abbrev: **-f**) *logfilename* is the path and file name to be used as the system log file,

without the ping-pong extension (.LOGx). The default system log file is **/var/adm/nettl**. This is an optional field.

-option *logoptfile*
(abbrev: **-o**) *logoptfile* is the path and file name to be used as the console log options file. The information in this file will be used to select logged events that will be reported to the system console. The default console logging options file is **/var/adm/conslog.opts**. This is an optional field.

The following options are used to add or update a subsystem description to the database.

-S Indicates that subsequent options apply to adding or updating a subsystem entry. This is an optional field.

-id *ssid* (abbrev: **-i**) *ssid* (subsystem ID number) is used as the key field in the **nettlgen.conf** database. It uniquely identifies a subsystem to the tracing and logging facility. This is a required field.

-name *ssname* (abbrev: **-n**) *ssname* is the subsystem-name mnemonic. This string is used to identify the subsystem on the **nettl** command line and also in the subsystem header displayed by the formatter (see *nettl*(1M) and *netfmt*(1M)). This is a required field.

-class *logclass* (abbrev: **-c**) *logclass* is the default log class mask assigned to the subsystem at start-up of the tracing/logging facility. For multiple classes, the masks must be combined into a single decimal number. For example, to initially log **DISASTER** and **ERROR** events use **12** as the *logclass*. Default is an empty field in **nettlgen.conf**. **nettl** substitutes **12** (disaster and error) for an empty class field. This is an optional field.

Class	Abbreviation
informative	1
warning	2
error	4
disaster	8

-kernel (abbrev: **-k**) flags the given subsystem as a kernel subsystem. **nettl** uses this information to control certain tracing and logging properties of the subsystem. A subsystem is defaulted to non-kernel unless this option is used. This is an optional field.

-streams (abbrev: **-st**) flags the given subsystem as a streams based kernel subsystem. **nettl** uses this information to control certain tracing and logging properties of the subsystem. A subsystem is defaulted to non-kernel unless this option is used. This is an optional field.

-lib *sslib* (abbrev: **-l**) *sslib* is the name of the shared library where the subsystem formatter resides. This should be an absolute path name unless the library resides in **/usr/lib**. Multiple subsystems can reference the same library. This is a required field.

-msg *ssmsgcat* (abbrev: **-m**) *ssmsgcat* is the name of the subsystem formatter message catalog. If the pathname and **.cat** filename extension are excluded, */usr/lib/nls/%L/%N.cat* is used to locate *ssmsgcat*. Otherwise, *ssmsgcat* must be formatted similarly to the **NLSPATH** environment variable (see *environ*(5)). Multiple subsystems can refer to the same message catalog. This is a required field.

-fmtfn *fmtfunc* (abbrev: **-f**) *fmtfunc* specifies the function to call when formatting data from the given subsystem. Multiple subsystems can reference the same formatting function. Default is to form the function name from the subsystem ID as follows:

subsys__N_**_format**

where *N* is the subsystem *ID* number. If a null function is needed for this subsystem, specify

```
-f NULL
```

This is an optional field.

-optfn *optfunc* (abbrev: **-o**) *optfunc* specifies the function used to process options in the **netfmt** filter configuration file (see *netfmt*(1M)). Multiple subsystems can reference the same options processing function. The default is an empty field in **nettlgen.conf**.

netfmt assumes a *NULL* function for an empty *optfunc* field. This is an optional field.

-group *ssgrpname*
(abbrev: **-g**) *ssgrpname* is a group name associated with the subsystem. It is typically the product name of the subsystem. Several subsystems can be grouped together so that a common banner is printed in the formatted header. This is a required field.

The following option is used to remove a subsystem description from the database.

-delete *ssid* (abbrev: **-d**) Deletes all information associated with the *ssid* (subsystem ID) from the database.

WARNINGS

The **nettlconf** utility is intended primarily for use by HP subsystems to configure themselves into the tracing and logging facility at installation time. System administrators may wish to use this command to alter the default logging class each subsystem starts up with, but no other information about the subsystem should be changed.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are executed. If the file becomes corrupted these commands cannot function.

Some changes to the **/etc/nettlgen.conf** file do not take effect until **nettl** and **netfmt** are stopped and restarted.

AUTHOR

nettlconf was developed by HP.

FILES

/etc/nettlgen.conf	subsystem configuration file maintained by **nettlconf**
/etc/rc.config.d/nettl	configuration file controlling **nettl** during system startup

SEE ALSO

netfmt(1M), nettl(1M), nettlgen.conf(4), environ(5).

NAME

newaliases - rebuilds the database for the mail aliases file

SYNOPSIS

newaliases [**-v**]

DESCRIPTION

newaliases rebuilds the random access database for the mail aliases file **/etc/aliases**. It must be run each time this file is changed in order for the change to take effect.

newaliases is identical to **sendmail -bi**.

RETURN VALUE

The **newaliases** utility exits 0 on success, and >0 if an error occurs.

FILES

/etc/aliases	The mail aliases file.

SEE ALSO

aliases(5), sendmail(1M).

HISTORY

The **newaliases** command appeared in 4.0BSD. The manual page originally came from **sendmail** 8.7.

NAME

newarray - configure a disk array

SYNOPSIS

newarray [**-N***Config_Name* | -r*RAID_Level*] [**Options**] *device_file*

DESCRIPTION

newarray, a front-end program for the utility **cfl** (see *cfl*(1M)), facilitates the configuration of Hewlett-Packard SCSI disk arrays. It is the recommended utility for all array configuration. Array configuration maps a set of one or more physical disk mechanisms in an array to a set of one or more logical disks, addressable by HP-UX. Logical disks are addressed through device files. Each logical disk in an array (also known as a LUN, for Logical UNit), has its own device file. A logical disk can consist of a single physical disk, a portion of a single physical disk, multiple physical disks, or portions of multiple physical disks. For additional information about possible array configurations, see the array configuration table contained in the file **/etc/hpC2400/arraytab**, and *arraytab*(4).

Supported configurations for the array device are pre-defined in the array configuration table, located in file **/etc/hpC2400/arraytab**.

newarray can configure a complete set of logical partitions for an array in one operation. Due to the inter-dependency of logical partitions, this is the recommended method for configuration. A single logical partition can be added to an array configuration using an entry from the array configuration table by using the **-L** option.

device_file is a character device file that specifies the I/O address, and driver to use when configuring the disk array. The way that this file is used by **newarray** is system dependent. See dependencies below. Logical partitions in an array are independently addressable by using the appropriate device file to address the logical unit assigned to a partition.

Prior to configuring the array (except with the **-L** option), all currently configured logical partitions are removed from the configuration.

To simplify array configuration **newarray** obtains much of the necessary information directly from the array device, and its attached disk mechanisms. The array model number, and the number of available physical disks available, is determined by querying the device. This information is used to locate the appropriate configuration entry in the array configuration table. Optional parameters can be used to override the default, and inquiry values.

The preferred configuration method is to use the **-N** option to specify a configuration by name. The name determines which configuration **newarray** uses from the array configuration table. Configuration parameters are obtained from the named configuration entry. Parameters of the chosen configuration can be overridden using options to **newarray**, or by creating and using a custom configuration entry in the array configuration table. See the WARNINGS section of this manpage.

Because the array controller type, and disk mechanism types are used in addition to the configuration name to select an entry from the array configuration table the configuration name does not have to be unique within the array configuration table. However, the combination of configuration name, array controller type, and disk mechanism types must be unique within the array configuration table. During configuration, the array controller type, and disk mechanism types are obtained by querying the devices.

The **-r** option specifies an operating mode, rather than specifying a configuration by name. The **-d** option, which specifies the size of a disk group, is often used with the **-r** option. If **-d** is not used, **newarray** selects the configuration in the array configuration table that most closely matches the disks in the array.

When the configuration parameters have been determined, **newarray** calls **cfl.**

If the **-V** option is used, **newarray** prints its actions, and the parameters it passes to **cfl** to configure the array (see *cfl*(1M)).

Array Configuration

newarray obtains its configuration values from the array configuration table. If not specified there, default values are provided by **cfl** (see *cfl*(1M)). Configuration values can be overridden by **newarray** options.

Options

-L *unit addr* Configures a single LUN from the specified configuration. The **-L** option is useful for adding disks to an array without changing the existing configuration. Because the order in

n

which LUN's are configured determines the physical mapping on the disks within the array, be very careful when using the **-L** option.

-N *config_name*
The name of the configuration to be used, as specified in the configuration file **/etc/hpC2400/arraytab**. See *arraytab(4)*.

-V Display the parameters of array configuration, and the utility commands issued as part of the configuration process.

-b *block_size* The size in bytes of the LUN block. Must be an integral number of the physical disk mechanism sector size. Currently supported values are 512, 1024, 2048, and 4096.

-c *capacity* The size in blocks of the LUN. A value of 0 defaults to the largest capacity available. If the LUN type is set to sub-LUN, the capacity is the available capacity of the composite drive group or 2 GByte if the 2 GByte flag is set, which ever is smaller. See **-f** option.

-d *group_size* Physical drive group will contain this number of disks in the logical partition configuration.

-f *flags* Configuration flags. There are 16 flags, represented by a 16 bit hexadecimal number. Currently only four of the flags are defined. The flag definitions and their default value are:

Bit 0	*off*	Not used.
Bit 1	*on*	Disable auto reconstruction. When set (on), disables the automatic detection, and initiation of failed disk data reconstruction.
Bit 2	*off*	Not used.
Bit 3	*off*	Not used.
Bit 4	*on*	When set (on), enables AEN (automatic event notification) polling.
Bit 5	*on*	When set (on), enables read parity verification.
Bit 6	*on*	When set (on), enables write with parity verification.
Bit 7	*off*	Not used.
Bit 8	*off*	Mode Sense default pages. Bit 8 and Bit 9 concurrently set is reserved.
Bit 9	*off*	Mode Sense current pages. Bit 8 and Bit 9 concurrently set is reserved.
Bit 10-15	*off*	Not used.

-g *group_name*
Use physical drive group configuration with label GroupName (in array configuration table) for this LUN configuration.

-i *seg0_size* The size in bytes of the first segment LUN. This allows this area to be set to a size different than the remainder of the disk, an area typically used as the boot block for some systems. This must be a integral number of the block-size. If there are no special requirements, this parameter should be set to 0.

-k *recon_size* Reconstruction size. The **-k** option specifies (in LUN blocks) the amount of data to be reconstructed in a single operation during reconstruction of a redundant drive configuration. Larger values provide more efficient (faster) reconstruction, but hold off the servicing of I/O requests. Smaller values allow quicker servicing of I/O requests, but with less efficient (slower) reconstruction.

-l *recon_freq* Reconstruction frequency. The **-l** option specifies (in tenths of a second) the time period between reconstruction of disk segments in a redundant drive configuration. Small time periods cause the array to consume most of its time reconstructing data, but allow the reconstruction to complete more quickly. Large time periods allocate more time to I/O processing, but require longer reconstruction times.

-r *raid_level* The RAID (redundancy level) to apply to the disks in the array. Valid entries for *raid_level* are RAID_0, RAID_1, RAID_3, and RAID_5. Some RAID levels require specific physical drive configurations. See also the **-g** option.

-s *seg_size* The number of bytes of a contiguous segment of the logical address space residing on a single physical disk. This affects how many physical disks are involved in a single I/O request. If I/O requests are mostly random, single-block requests, set this value to the integral number of the LUN block size that minimizes the number of disks necessary to service most

I/O requests. A larger size will allocate more time to I/O processing.

-t *LUN_type* LUNs can be configured as regular LUNs (**reg**), or sub-LUNs (**sub**). A regular LUN utilizes all the available capacity of a disk group, or limits the LUN configuration to 2 GBytes if the 2 GByte limiter is set. If a regular LUN configuration is used, the **-c** option is ignored. A sub-LUN allows logical partitioning of the disk group capacity into a maximum of eight LUNs. Valid values for *LUN_type* are "**reg**" and "**sub**".

Custom Configurations

You can create array configurations that might be better suited to a particular application by using **newarray**'s command line parameters to override default values, or by creating special entries in the array configuration table in the file **/etc/hpC2400/arraytab**. Before you do, see cautionary notes in the WARNINGS section of this manpage.

RETURN VALUES

newarray will return the following values:

0 Successful completion.
-1 Command failed (an error occurred).

ERRORS

```
newarray: device busy
```

To ensure that **newarray** does not modify a disk array that is being used by another process, **newarray** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

EXAMPLES:

The following examples use configurations contained in **/etc/hpC2400/arraytab**.

Raid Level Specification

To configure an HP C2425D with 5 internal disks to a five drive RAID level 0 configuration (on Series 700 computer):

```
newarray -rRAID_0 /dev/rdsk/c2t3d0
```

To configure an HP C2425D with 5 internal disks to a one drive RAID level 0 configuration (on Series 700):

```
newarray -rRAID_0 -d1 /dev/rdsk/c2t3d0
```

Name Specification

To configure an HP C2430D with five disks connected on SCSI channel 3 (on a Series 800) using the configuration "Raid_3_5d" in **/etc/hpC2400/arraytab**:

```
newarray -NRaid_3_5d /dev/rdsk/c2t3d0
```

WARNINGS

We strongly recommend that you use the array configurations that are specified, and delivered by Hewlett-Packard, in the file **/etc/hpC2400/arraytab.** These configurations have been tested and certified for proper use on Hewlett-Packard computer systems. Custom configurations cannot be warranted for proper operation.

Configuring a disk array causes the loss of user data on the array.

When using the **-L** option, physical media is assigned to the logical unit in the order in which the logical units are configured. Existing logical unit configurations are NOT removed prior to configuration with this option. The use of this option is not recommended at this time.

DEPENDENCIES

File System Considerations

The disk array maps the address space of one or more physical disk mechanisms onto logical "disk" partitions. The parameters defined in the configuration, together with the data access patterns of the user's application, determine the operating characteristics of the logical disk. Some configurations create multiple logical partitions, that share a set of physical disks. I/O traffic to each of the logical partitions affects performance, due to the common physical disk resources. The file system or application using the "logical" disk may require or assume certain characteristics. For optimal system performance it is necessary that the file system configuration and application be compatible with the array configuration.

Your choice of segment size directly affects the performance of the disk array. Choose this parameter in concert with the choice of the parameters used when building the file system on the device. In general, the segment size determines how much data from a single I/O will be stored on a single disk within the array. A smaller value will involve more of the disks with the I/O, whereas a larger value will involve fewer disks. If input/output operations tend to be very long, the involvement of multiple disks may hasten the completion of each I/O. In this case the access time is the same as a single disk, but the disk data transfer time is shared across the set of disks. If input/output operations are short, the access time will dominate relative to the disk data transfer time, and more input/output operations may be processed in parallel by involving fewer disks in each I/O. In all cases the relative locality of data and the access pattern will affect the performance. For highly sequential data, it may be advantageous to locate the data for a single I/O on a single disk, to take advantage of read-ahead caching within each disk.

Configurations for the HP C2430 disk array should enable the automatic data reconstruction LUN flag as part of the configuration specification.

Supported Array Products:

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

`newarray` was developed by HP.

SEE ALSO

arraytab(4), cfl(1M), buildfs(1M), fs(4), mkfs(1M), sss(1M), dcc(1M).

NAME

newfs (generic) - construct a new file system

SYNOPSIS

/usr/sbin/newfs [**-F** *FStype*] [**-o** *specific_options*] [**-V**] *special*

DESCRIPTION

The **newfs** command is a "friendly" front-end to the **mkfs** command (see *mkfs*(1M)). The **newfs** command calculates the appropriate parameters and then builds the file system by invoking the **mkfs** command.

special represents a character (raw) special device.

Options

newfs recognizes the following options:

-F *FStype*	Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.
-o *specific_options*	Specify options specific to the file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for an *FStype*-specific module of the command. See the file system specific manual entries for a description of the *specific_options* that are supported, if any.
-V	Echo the completed command line, but perform no other actions. The command line is generated by incorporating the specified options and arguments and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

n

EXAMPLES

Execute the **newfs** command to create an HFS file system on **/dev/rdsk/c1t0d2**

```
newfs -F hfs /dev/rdsk/c1t0d2
```

AUTHOR

newfs was developed by HP and the University of California, Berkeley.

FILES

/etc/default/fs	File that specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

fsck(1M), fstyp(1M), mkfs(1M), newfs_*FStype*(1M), fstab(4), fs_wrapper(5).

NAME

newfs (hfs) - construct a new HFS file system

SYNOPSIS

/usr/sbin/newfs [**-F hfs**] [**-B**] [**-d**] [**-L**|**-S**] [**-O** *disk_type*] [**-R** *swap*] [**-v**] [**-V**]
[*mkfs-options*] *special*

DESCRIPTION

The **newfs** command builds a file system by invoking the **mkfs** command.

The **newfs** command creates the file system with a rotational delay value of zero (see *tunefs*(1M)).

special represents a character (raw) special device.

Options

newfs recognizes the following options:

-F hfs	Specify the HFS file system type.
-B	Reserve space for boot programs past the end of the file system. If file **/usr/lib/uxbootlf** is present on the system then sufficient space to accommodate that file is reserved, otherwise 691 KB sectors are reserved. This option decreases the size of the file system to be created. This option cannot be used if the **-s** option is given; see "mkfs Options" below.
-d	This option allows the **newfs** command to make the new file system in an ordinary file. In this case, *special* is the name of an existing file in which to create the file system. The **-s** option (see "mkfs Options") must be provided with this option.
-L|**-S**	There are two types of HFS file systems, distinguished mainly by directory formats that place different limits on the length of file names. If **-L** is specified, build a long-file-name file system that allows directory entries (file names) to be up to **MAXNAMLEN** (255) bytes long. If **-S** is specified, build a short-file-name file system that allows directory entries (file names) to be up to **DIRSIZ** (14) bytes long. If neither **-L** nor **-S** is specified, build a file system of the same type as the root file system.
-O *disk_type*	Use disk parameters from the entry for the named disk type in **/etc/disktab**. This option is provided for backward compatibility with previous HP-UX releases. Any parameters specified in the command line will override the corresponding values in **/etc/disktab**. Any values not given in the command line or in **/etc/disktab** will be defaulted.
-R *swap*	Reserve *swap* megabytes (MB) of swap space past the end of the file system. This option decreases the size of the file system to be created by the given amount. This option cannot be used if the **-s** option is given; see "mkfs Options" below.
-v	Verbose; the **newfs** command prints out its actions, including the parameters passed to the **mkfs** command.
-V	Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Both the **-R** and **-B** options can be given in the same command line. In this case, both the requested swap space and the space needed for boot programs are reserved. These options are for use when the file system size defaults to the size of the entire disk.

mkfs Options

The *mkfs-options* argument can be zero or more of the following options that can be used to override default values passed to the **mkfs** command:

-b *blksize* The primary block size for files on the file system. Valid values are: 4096, 8192, 16384, 32768, and 65536. The default value is 8192 bytes.

-c *cylinders_per_group*
The number of disk cylinders per cylinder group. This number must be in the range 1

n

to 32. The default value is 16 cylinders per group.

-f *fragsize* The fragment size for files on the file system. *fragsize* represents the smallest amount of disk space to be allocated to a file. It must be a power of two no smaller than **DEV_BSIZE** and no smaller than one-eighth of the file system block size. The default value is 1024 bytes.

-i *number_of_bytes_per_inode*
The density of inodes in the file system specified as the number of bytes per inode. The default is 6144 bytes per inode.

This number should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used; if more inodes are desired, a smaller number should be used.

Note: The number of inodes that will be created in each cylinder group of a file system is approximately the size of the cylinder group divided by the number of bytes per inode, up to a limit of 2048 inodes per cylinder group. If the size of the cylinder group is large enough to reach this limit, the default number of bytes per inode will be increased.

-m *free_space_percent*
The minimum percentage of free disk space allowed. The default value is 10 percent.

Once the file system capacity reaches this threshold, only users with appropriate privileges can allocate disk blocks.

-r *revolutions_per_minute*
The disk speed in revolutions per minute (rpm). The default value is 3600 revolutions per minute.

-s *size* The number of **DEV_BSIZE** blocks in the file system. **DEV_BSIZE** is defined in **<sys/param.h>**. The default value is the size of the entire disk or disk section minus any swap or boot space requested. See *mkfs_hfs*(1M) for limits on the size of HFS file systems.

-t *tracks_per_cylinder*
The number of tracks per cylinder. The default value depends on the size of the file system. For file systems of less than 500 MB, the default is 7; for file systems between 500 MB and 1 GB, the default is 12; for file systems larger than 1 GB the default is 16.

-o *specific_options*
Specify a list of comma separated suboptions and/or keyword/attribute pairs from the list below.

largefiles |**nolargefiles**
Controls the **largefile featurebit** for the file system. The default is **nolargefiles**. This means the bit is not set and files created on the file system will be limited to less than 2 gigabytes in size. If **largefiles** is specified, the bit is set and the maximum size for files created on the file system is not limited to 2 gigabytes (see *mount_hfs*(1M) and *fsadm_hfs*(1M)).

Access Control Lists

Every file with one or more optional ACL entries consumes an extra (continuation) inode. If you anticipate significant use of ACLs on a new file system, you can allocate more inodes by reducing the value of the argument to the **-i** option appropriately. The small default value typically causes allocation of many more inodes than are actually necessary, even with ACLs. To evaluate the need for extra inodes, run the **bdf -i** command on existing file systems. For more information on access control lists, see *acl*(5).

EXAMPLES

Execute the **newfs** command to create an HFS file system on a non-LVM disk **/dev/rdsk/c1t0d2** and reserve 40 megabytes of swap space.

```
newfs -F hfs -R 40 /dev/rdsk/c1t0d2
```

Create an HFS file system within a logical volume, **my_lvol**, whose size is identical to that of the logical volume. (Note the use of the character (raw) special device.)

```
newfs -F hfs /dev/vg01/rmy_lvol
```

WARNINGS

The old **-F** option, from prior releases of *newfs*(1M), is no longer supported.

newfs(1M) cannot be executed specifying creation of a file system on a whole disk if that disk was previously used as an LVM disk. If you wish to do this, use *mediainit*(1) to reinitialize the disk first.

AUTHOR

newfs was developed by HP and the University of California, Berkeley.

FILES

/etc/disktab	
/etc/fstab	Static information about the file systems.

SEE ALSO

bdf(1M), fsadm_hfs(1M), mkboot(1M), mkfs(1M), mkfs_hfs(1M), mount_hfs(1M), newfs(1M), tunefs(1M), disktab(4), fs(4), acl(5).

NAME

newfs (vxfs) - construct a new VxFS file system

SYNOPSIS

/usr/sbin/newfs -F vxfs [**-V**] [**-v**] [**-R** *swap*] [**-l**] [**-B**] [**-O** *disk_type*] [*mkfs_vxfs_options*] *special*

DESCRIPTION

The **newfs -F vxfs** command builds a VxFS file system by invoking **mkfs**. In addition to file system type specification, **newfs** takes a variety of options, listed below. A character (raw) file, *special*, (for example, **/dev/rdsk/c0t6d0**) must also be specified.

Options

newfs recognizes the following options:

-F vxfs Specify the file-system type **vxfs**, required for a VxFS file system.

-V Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab.** This option allows the user to verify the command line.

-v Verbose. **newfs** prints out its actions, including the parameters passed to **mkfs**.

-l Set the **largefile compatibility bit** for the file system, allowing files larger than 2 Gbytes to be created. (See the **-o largefiles** option in the *mkfs_vxfs*(1M) manual page.)

-R *swap* Reserve *swap* Mbytes of swap space past the end of the file system. This option decreases the size of the file system to be created by the given number of Mbytes. This option cannot be used if the file-system size is also specified using **-s** (see **mkfs_vxfs** options below).

-B Reserve space for boot programs past the end of the file system. If file **/usr/lib/uxbootlf** is present on the system, sufficient space to accommodate that file is reserved; otherwise 691 Kbytes are reserved. This option decreases the size of the file system being created. This option cannot be used if the file-system size is also specified using **-s** (see **mkfs_vxfs** options below).

-O *disk_type* Use disk parameters from the entry for the named *disk_type* in **/etc/disktab**. This option is provided for backward compatibility with previous HP-UX releases. Any parameters specified on the command line will override the corresponding values in **/etc/disktab**. Any values not specified in the command line and not shown in **/etc/disktab** will be defaulted.

Both **-R** and **-B** options may be given in the same command line. In this case, both the requested swap space and the space needed for boot programs are reserved. These options are used when the file system size is defaulted to the size of the entire disk.

mkfs_vxfs Options

The following additional command-line options can be used to override default parameters passed to **mkfs_vxfs:**

-s *size* File system size in **DEV_BSIZE** blocks (defined in **<sys/param.h>**). The default value used is the size of the entire disk or disk section, minus any swap or boot space requested. The *size* specifies the number of sectors in the file system. By default, size is specified in units of DEV_BSIZE sectors. However, the letter **k**, **m**, or **g** can be appended to the number to indicate that the value is in kilobytes, megabytes, or gigabytes, respectively.

-b *block_size* File system block size in bytes. The default value used is 1024 bytes.

-o *largefiles* | *nolargefiles*
Controls the **largefile compatibility bit** for the file system. By default the bit is not set, and files created on the file system will be limited to less than 2 gigabytes in size. If **largefiles** is specified, the bit is set and the maximum file size for files created on the file system is not limited to 2 gigabytes (see *mkfs_vxfs*(1M), *mount_vxfs*(1M) and *fsadm_vxfs*(1M)). This option is only valid for a Version 3 disk layout. The default is *nolargefiles*, although the default may change in the future.

EXAMPLES

Execute **newfs** to create a VxFS file system on **/dev/rdsk/c1t5d0** and reserve 40 Mbytes of swap space.

```
newfs -F vxfs -R40 /dev/rdsk/c1t5d0
```

FILES

/etc/disktab	Disk description file.
/etc/fstab	Static information about the file systems.

SEE ALSO

mkfs(1M), mkfs_vxfs(1M), newfs(1M), disktab(4).

NAME

newkey - create a new Diffie-Hellman key pair in the publickey database

SYNOPSIS

newkey -h *hostname* [**-s nisplus** | **nis** | **files**]

newkey -u *username* [**-s nisplus** | **nis** | **files**]

DESCRIPTION

newkey establishes new public keys for users and machines on the network. These keys are needed when using secure RPC or secure NFS service.

newkey prompts for a password for the given *username* or *hostname* and then creates a new public/secret Diffie-Hellman 192 bit key pair for the user or host. The secret key is encrypted with the given password. The key pair can be stored in the **/etc/publickey** file, the NIS **publickey** map, or the NIS+ **cred.org_dir** table.

newkey consults the **publickey** entry in the name service switch configuration file (see *nsswitch.conf*(4)) to determine which naming service is used to store the secure RPC keys. If the **publickey** entry specifies a unique name service, **newkey** will add the key in the specified name service. However, if there are multiple name services listed, **newkey** cannot decide which source to update and will display an error message. The user is required to specify the source explicitly with the **-s** option.

In the case of NIS, **newkey** should be run by the superuser on the master NIS server for that domain. In the case of NIS+, **newkey** should be run by the superuser on a machine which has permission to update the **cred.org_dir** table of the new user/host domain.

In the case of NIS+, *nisaddcred*(1M) should be used to add new keys.

Options

-h *hostname*	Create a new public/secret key pair for the privileged user at the given *hostname*. Prompts for a password for the given *hostname*.
-u *username*	Create a new public/secret key pair for the given *username*. Prompts for a password for the given *username*.
-s nisplus **-s nis** **-s files**	Update the database in the specified source: **nisplus** (for NIS+), **nis** (for NIS), or **files**. Other sources may be available in the future.

n

AUTHOR

newkey was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), keylogin(1), nisaddcred(1M), nisclient(1M), nsswitch.conf(4), publickey(4).

NAME

nfsd, biod - NFS daemons

SYNOPSIS

/usr/sbin/nfsd [*nservers*]

/usr/sbin/biod [*nservers*]

DESCRIPTION

nfsd starts the NFS server daemons that handle client file system requests (see *nfs*(7)). *nservers* is the number of file system request daemons that start. This number should be determined by the load expected on the server system. To obtain the best performance in most cases, set *nservers* to four.

biod starts *nservers* asynchronous block I/O daemons. This command is used on an NFS client to buffer cache handle read-ahead and write-behind. *nservers* is a number greater than zero. For best performance, set *nservers* to four.

AUTHOR

nfsd was developed by Sun Microsystems, Inc.

SEE ALSO

mountd(1M), exports(4).

NAME

nfsstat - Network File System statistics

SYNOPSIS

nfsstat [**-cmnrsz**]

AVAILABILITY

This program is available with the **Networking** software installation option. Refer to *install*(1M) for information on how to install optional software.

DESCRIPTION

nfsstat displays statistical information about the NFS (Network File System) and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is

```
nfsstat -cnrs
```

That is, display everything, but reinitialize nothing.

OPTIONS

-c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **-n** and **-r** options to print client NFS or client RPC information only.

-m Display statistics for each NFS mounted file system. This includes the server name and address, mount flags, current read and write sizes, the retransmission count, and the timers used for dynamic retransmission. The **srtt** value contains the smoothed round trip time, the **dev** value contains the estimated deviation, and the **cur** value is the current backed-off retransmission value.

-n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **-c** and **-s** options to print client or server NFS information only.

-r Display RPC information.

-s Display server information.

-z Zero (reinitialize) statistics. This option is for use by the super-user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the following fields:

calls	The total number of RPC calls received.
badcalls	The total number of calls rejected by the RPC layer (the sum of **badlen** and **xdrcall** as defined below).
nullrecv	The number of times an RPC call was not available when it was thought to be received.
badlen	The number of RPC calls with a length shorter than a minimum-sized RPC call.
xdrcall	The number of RPC calls whose header could not be XDR decoded.

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made. The client RPC display includes the following fields:

calls	The total number of RPC calls made.
badcalls	The total number of calls rejected by the RPC layer.
retrans	The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server.
badxid	The number of times a reply from a server was received which did not correspond to any outstanding call.
timeout	The number of times a call timed out while waiting for a reply from the server.
wait	The number of times a call had to wait because no client handle was available.
newcred	The number of times authentication information had to be refreshed.
timers	The number of times the calculated time-out value was greater than or equal to the minimum specified time-out value for a call.

The client NFS display shows the number of calls sent and rejected, as well as the number of times a **CLIENT** handle was received (**nclget**), the number of times a call had to sleep while awaiting a handle (**nclsleep**), as well as a count of the various calls and their respective percentages.

AUTHOR

`nfsstat` was developed by Sun Microsystems, Inc.

NAME

nis_cachemgr - maintains a cache containing location information about NIS+ servers

SYNOPSIS

/usr/sbin/nis_cachemgr [**-i**] [**-n**] [**-v**]

DESCRIPTION

The **nis_cachemgr** daemon maintains a cache of the NIS+ directory objects. The cache contains location information necessary to contact the NIS+ servers that serve the various directories in the name space. This includes transport addresses, information neeeded to authenticate the server, and a time to live field which gives a hint on how long the directory object can be cached. The cache helps to improve the performance of the clients that are traversing the NIS+ name space. **nis_cachemgr** should be running on all the machines that are using NIS+. However, it is not required that the **nis_cachemgr** program be running in order for NIS+ requests to be serviced.

The cache maintained by this program is shared by all the processes that access NIS+ on that machine. The cache is maintained in a file that is memory mapped (see *mmap* (2)) by all the processes. On startup, **nis_cachemgr** initializes the cache from the cold start file (see *nisinit*(1M)) and preserves unexpired entries that already exist in the cache file. Thus, the cache survives machine reboots.

The **nis_cachemgr** program is normally started from a system startup script.

Note: The **nis_cachemgr** program makes NIS+ requests under the NIS+ principal name of the host on which it runs. Before running **nis_cachemgr**, security credentials for the host should be added to the **cred.org_dir** table in the host's domain using *nisaddcred*(1M). Credentials of type DES will be needed if the NIS+ service is operating at security level 2 (see *rpc.nisd*(1M)). See the *WARNINGS* section, below. Additionally, a '**keylogin -r**' needs to be done on the machine.

nisshowcache can be used to look at the cached objects.

Options

-i Force **nis_cachemgr** to ignore the previous cache file and reinitialize the cache from just the cold start file. By default, the cache manager initializes itself from both the cold start file and the old cache file, thereby maintaining the entries in the cache across machine reboots.

-n Run **nis_cachemgr** in an *insecure* mode. By default, before adding a directory object to the shared cache, on the request of another process on the machine, it checks the encrypted signature on the request to make sure that the directory object is a valid one and is sent by an authorized server. In this mode, **nis_cachemgr** adds the directory object to the shared cache without making this check.

-v This flag sets *verbose* mode. In this mode, the **nis_cachemgr** program logs not only errors and warnings, but also additional status messages. The additional messages are logged using *syslog*(3C) with a priority of **LOG_INFO**.

DIAGNOSTICS

The **nis_cachemgr** daemon logs error messages and warnings using syslog (see *syslog*(3C)). Error messages are logged to the **DAEMON** facility with a priority of **LOG_ERR**, and warning messages with a priority of **LOG_WARNING**. Additional status messages can be obtained using the **-v** option.

WARNINGS

If the host principal does not have the proper security credentials in the **cred.org_dir** table for its domain, then running this program without the **'-n'** insecure mode option may significantly *degrade* the performance of processes issuing NIS+ requests.

FILES

/var/nis/NIS_SHARED_DIRCACHE	the shared cache file
/var/nis/NIS_COLD_START	the coldstart file
/etc/init.d/rpc	initialization scripts for NIS+

AUTHOR

nis_cachemgr was developed by Sun Microsystems, Inc.

SEE ALSO

keylogin(1), nisaddcred(1M), nisinit(1M), nisshowcache(1M), rpc.nisd(1M), mmap(2), syslog(3C), nisfiles(4).

NAME

nisaddcred - create NIS+ credentials

SYNOPSIS

nisaddcred [**-p** *principal*] [**-P** *nis_principal*] [**-l** *login_password*] *auth_type* [*domain_name*]

nisaddcred -r [*nis_principal*] [*domain_name*]

DESCRIPTION

The **nisaddcred** command is used to create security credentials for NIS+ principals. NIS+ credentials serve two purposes. The first is to provide authentication information to various services; the second is to map the authentication service name into an NIS+ principal name.

When the **nisaddcred** command is run, these credentials get created and stored in a table named **cred.org_dir** in the default NIS+ domain. If *domain_name* is specified, the entries are stored in the **cred.org_dir** of the specified domain. Note that the credentials of normal users must be stored in the same domain as their passwords.

It is simpler to add credentials using *nisclient*(1M) because it obtains the required information itself. *nispopulate*(1M) can also be used to add credentials for entries in the **hosts** and the **passwd** NIS+ tables.

NIS+ principal names are used in specifying clients that have access rights to NIS+ objects. For more details, refer to the "Principal Names" subsection of the *nis+*(1) manual page. See *nischmod*(1), *nischown*(1), *nis_objects*(3N), and *nis_groups*(3N). Various other services can also implement access control based on these principal names.

The **cred.org_dir** table is organized as follows :

cname	auth_type	auth_name	public_data	private_data
fred.foo.com.	LOCAL	2990	10,102,44	
fred.foo.com.	DES	unix.2990@foo.com	098...819	3b8...ab2

The **cname** column contains a canonical representation of the NIS+ principal name. By convention, this name is the login name of a user or the host name of a machine, followed by a dot ("."), followed by the fully qualified "home" domain of that principal. For users, the home domain is defined to be the domain where their **DES** credentials are kept. For hosts, their home domain is defined to be the domain name returned by the *domainname*(1) command executed on that host.

There are two types of *auth_type* entries in the **cred.org_dir** table: those with authentication type **LOCAL** and those with authentication type DES. *auth_type*, specified on the command line in upper or lower case, should be either *local* or *des*.

Entries of type LOCAL are used by the NIS+ service to determine the correspondence between fully qualified NIS+ principal names and users identified by UIDs in the domain containing the **cred.org_dir** table. This correspondence is required when associating requests made using the **AUTH_SYS** RPC authentication flavor (see *rpc_clnt_auth*(3N)) to an NIS+ principal name. It is also required for mapping a UID in one domain to its fully qualified NIS+ principal name whose home domain may be elsewhere. The principal's credentials for any authentication flavor may then be sought for within the **cred.org_dir** table in the principal's home domain (extracted from the principal name). The same NIS+ principal may have LOCAL credential entries in more than one domain. Only users, and not machines, have LOCAL credentials. In their home domain, users of NIS+ should have both types of credentials.

The *auth_name* associated with the LOCAL type entry is a UID that is valid for the principal in the domain containing the **cred.org_dir** table. This may differ from that in the principal's home domain. The public information stored in *public_data* for this type contains a list of GIDs for groups in which the user is a member. The GIDs also apply to the domain in which the table resides. There is no private data associated with this type. Neither a UID nor a principal name should appear more than once among the LOCAL entries in any one **cred.org_dir** table.

The DES *auth_type* is used for Secure RPC authentication (see *secure_rpc*(3N)).

The authentication name associated with the DES *auth_type* is a Secure RPC *netname*. A Secure RPC netname has the form **unix.***id*@*domain*, where *domain* must be the same as the domain of the principal. For principals that are users, the *id* must be the UID of the principal in the principal's home domain. For principals that are hosts, the *id* is the host's name. In Secure RPC, processes running under effective UID

0 (root) are identified with the host principal. Unlike LOCAL, there cannot be more than one DES credential entry for one NIS+ principal in the NIS+ namespace.

The public information in an entry of authentication type DES is the public key for the principal. The private information in this entry is the private key of the principal encrypted by the principal's network password.

User clients of NIS+ should have credentials of both types in their home domain. In addition, a principal must have a LOCAL entry in the **cred.org_dir** table of each domain from which the principal wishes to make authenticated requests. A client of NIS+ that makes a request from a domain in which it does not have a LOCAL entry will be unable to acquire DES credentials. An NIS+ service running at security level 2 or higher will consider such users unauthenticated and assign them the name *nobody* for determining access rights.

This command can only be run by those NIS+ principals who are authorized to add or delete the entries in the **cred** table.

If credentials are being added for the caller itself, **nisaddcred** automatically performs a keylogin for the caller.

Options

-p *principal* Use the principal name *principal* to fill the *auth_name* field for this entry. For LOCAL credentials, the name supplied with this option should be a string specifying a UID. For DES credentials, the name should be a Secure RPC netname of the form **unix.***id*@*domain*, as described earlier. If the **-p** option is not specified, the *auth_name* field is constructed from the effective UID of the current process and the name of the local domain.

-P *nis_principal*
Use the NIS+ principal name *nis_principal*. This option should be used when creating LOCAL credentials for users whose home domain is different from the local machine's default domain.

Whenever the **-P** option is not specified, **nisaddcred** constructs a principal name for the entry as follows. When it is not creating an entry of type LOCAL, **nisaddcred** calls **nis_local_principal**, which looks for an existing LOCAL entry for the effective UID of the current process in the **cred.org_dir** table and uses the associated principal name for the new entry. When creating an entry of authentication type LOCAL, **nisaddcred** constructs a default NIS+ principal name by taking the login name of the effective UID for its own process and appending to it a dot (".") followed by the local machine's default domain. If the caller is a superuser, the machine name is used instead of the login name.

-l *login_password*
Use the *login_password* specified as the password to encrypt the secret key for the credential entry. This overrides the prompting for a password from the shell. This option is intended for administration scripts only. Prompting guarantees not only that no one can see your password on the command line using *ps*(1), but it also checks to make sure you have not made any mistakes. **NOTE:** *login_password* does not really HAVE to be the user's password, but if it is, it simplifies logging in.

-r [*nis_principal*]
Remove all credentials associated with the principal *nis_principal* from the **cred.org_dir** table. This option can be used when removing a client or user from the system. If *nis_principal* is not specified, the default is to remove credentials for the current *user*. If *domain_name* is not specified, the operation is executed in the default NIS+ domain.

RETURN VALUE

This command returns **0** on success and **1** on failure.

EXAMPLES

Add a LOCAL entry with a UID **2970** for the NIS+ principal name **fredw.some.domain**:

```
nisaddcred -p 2970 -P fredw.some.domain. local
```

Note that credentials are always added in the **cred.org_dir** table in the domain where **nisaddcred** is run, unless *domainname* is specified as the last parameter on the command line. If credentials are being

n

added from the domain server for its clients, then *domainname* should be specified. The caller should have adequate permissions to create entries in the **cred.org_dir** table.

The system administrator can add a DES credential for the same user:

```
nisaddcred -p unix.2970@some.domain \
           -P fredw.some.domain. des
```

Here, **2970** is the UID assigned to the user, **fredw**. **some.domain** comes from the user's home domain, and **fredw** comes from the password file. Note that DES credentials can be added only after the LOCAL credentials have been added.

Note that the secure RPC netname does not end with a dot ("."), while the NIS+ principal name (specified with the **-P** option) does. This command should be executed from a machine in the same domain as the user.

Add a machine's DES credentials in the same domain:

```
nisaddcred -p unix.foo@some.domain \
           -P foo.some.domain. des
```

Note that no LOCAL credentials are needed in this case.

Add a LOCAL entry with the UID of the current user and the NIS+ principal name of **tony.some.other.domain**:

```
nisaddcred -P tony.some.other.domain. local
```

You can list the **cred** entries for a particular principal with *nismatch*(1).

AUTHOR

nisaddcred was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), keylogin(1), nis+(1), nischmod(1), nischown(1), nismatch(1), nistbladm(1), nisclient(1M), nispopulate(1M), nis_local_names(3N), rpc_clnt_auth(3N), secure_rpc(3N), nis_objects(3N), nis_groups(3N).

NOTES

The **cred.org_dir** NIS+ table replaces the maps *publickey.byname* and *netid.byname* used in NIS (YP).

NAME

nisaddent - create NIS+ tables from corresponding /etc files or NIS maps

SYNOPSIS

/usr/lib/nis/nisaddent [**-D** *defaults*] [**-Parv**] [**-t** *table*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent [**-D** *defaults*] [**-Paprmv**] **-f** *file* [**-t** *table*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent [**-D** *defaults*] [**-Parmv**] [**-t** *table*] **-y** *ypdomain* [**-Y** *map*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent -d [-AMq] [**-t** *table*] *type* [*nisdomain*]

DESCRIPTION

nisaddent creates entries in NIS+ tables from their corresponding **/etc** files and NIS maps. This operation is customized for each of the standard tables that are used in the administration of HP-UX systems. The *type* argument specifies the type of the data being processed. Legal values for this type are one of **aliases**, **bootparams**, **ethers**, **group**, **hosts**, **netid**, **netmasks**, **networks**, **passwd**, **protocols**, **publickey**, **rpc**, **services**, **shadow**, or **timezone** for the standard tables, or **key-value** for a generic two-column (key, value) table. For a site specific table, which is not of **key-value** type, one can use *nistbladm*(1) to administer it.

The NIS+ tables should have already been created by *nistbladm*(1), *nissetup*(1M), or *nisserver*(1M).

It is easier to use *nispopulate*(1M) instead of **nisaddent** to populate the system tables.

By default, **nisaddent** reads from the standard input and adds this data to the NIS+ table associated with the *type* specified on the command line. An alternate NIS+ table may be specified with the **-t** option. For type **key-value**, a table specification is required.

Note that the data *type* can be different from the table name (**-t**). For example, the automounter tables have **key-value** as the table type.

Although, there is a *shadow* data type, there is no corresponding *shadow* table. Both the shadow and the passwd data are stored in the passwd table itself.

Files may be processed using the **-f** option, and NIS version 2 (YP) maps may be processed using the **-y** option. The merge option is not available when reading data from standard input.

When a *ypdomain* is specified, the **nisaddent** command takes its input from the **dbm** files for the appropriate NIS map (**mail.aliases**, **bootparams**, **ethers.byaddr**, **group.byname**, **hosts.byaddr**, **netid.byname**, **netmasks.byaddr**, **networks.byname**, **passwd.byname**, **protocols.byname**, **publickey.byname**, **rpc.bynumber**, **services.byname**, or **timezone.byname**). An alternate NIS map may be specified with the **-Y** option. For type **key-value**, a map specification is required. The map must be in the **/var/yp/***ypdomain* directory on the local machine. Note that *ypdomain* is case sensitive. *ypxfr*(1M) can be used to get the NIS maps.

If a *nisdomain* is specified, **nisaddent** operates on the NIS+ table in that NIS+ domain; otherwise the default domain is used.

In terms of performance, loading up the tables is fastest when done through the dbm files (**-y**).

Options

-a	Add the file or map to the NIS+ table without deleting any existing entries. This option is the default. Note that this mode only propagates additions and modifications, not deletions.
-d	Dump the NIS+ table to the standard output in the appropriate format for the given *type*. For tables of type **key-value**, use *niscat*(1) instead. To dump the **cred** table, dump the **publickey** and the **netid** types.
-f *file*	Specify that *file* should be used as the source of input (instead of the standard input).
-m	Merge the file or map with the NIS+ table. This is the most efficient way to bring an NIS+ table up to date with a file or NIS map when there are only a small number of changes. This option adds entries that are not already in the database, modifies entries that already exist (if changed), and deletes any entries that are not in the source. Use the **-m** option whenever the database is large and replicated, and the map being loaded differs only in a few entries. This option reduces the number of update messages that have to be sent to the replicas. Also see the **-r** option.

-p Process the password field when loading password information from a file. By default, the password field is ignored because it is usually not valid (the actual password appears in a shadow file).

-q Dump tables in "quick" mode. The default method for dumping tables processes each entry individually. For some tables (e.g., hosts), multiple entries must be combined into a single line, so extra requests to the server must be made. In "quick" mode, all of the entries for a table are retrieved in one call to the server, so the table can be dumped more quickly. However, for large tables, there is a chance that the process will run out of virtual memory and the table will not be dumped.

-r Replace the file or map in the existing NIS+ table by first deleting any existing entries, and then add the entries from the source (**/etc** files, or NIS+ maps). This option has the same effect as the **-m** option. The use of this option is *strongly* discouraged due to its adverse impact on performance, unless there are a large number of changes.

-t *table* Specify that *table* should be the NIS+ table for this operation. This should be a relative name as compared to your default domain or the *domainname* if it has been specified.

-v Verbose.

-y *ypdomain*
Use the **dbm** files for the appropriate NIS map, from the NIS domain *ypdomain*, as the source of input. The files are expected to be on the local machine in the **/var/yp/***ypdomain* directory. If the machine is not an NIS server, use *ypxfr*(1M) to get a copy of the **dbm** files for the appropriate map.

-A All data. This option specifies that the data within the table and all of the data in tables in the initial table's concatenation path be returned.

-D *defaults*
This option specifies a different set of defaults to be used during this operation. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time*
This token sets the default time to live for objects that are created by this command. The value *time* is specified in the format as defined by the *nischttl*(1) command. The default is 12 hours.

owner=*ownername*
This token specifies that the NIS+ principal *ownername* should own the created object. The default for this value is the principal who is executing the command.

group=*groupname*
This token specifies that the group *groupname* should be the group owner for the object that is created. The default is **NULL**.

access=*rights*
This token specifies the set of access rights that are to be granted for the given object. The value *rights* is specified in the format as defined by the *nischmod*(1) command. The default is **----rmcdr---r---**.

-M Master server only. This option specifies that lookups should be sent to the master server. This guarantees that the most up-to-date information is seen at the possible expense that the master server may be busy, or that it may be made busy by this operation.

-P Follow concatenation path. This option specifies that lookups should follow the concatenation path of a table if the initial search is unsuccessful.

-Y *map* Use the **dbm** files for *map* as the source of input.

EXAMPLES

Add the contents of **/etc/passwd** to the **passwd.org_dir** table:

```
cat /etc/passwd | nisaddent passwd
```

Add the shadow information (note that the table type here is **shadow**,**not passwd**, even though the actual information is stored in the **passwd** table):

```
cat /etc/shadow | nisaddent shadow
```

Replace the **hosts.org_dir** table with the contents of **/etc/hosts** (in verbose mode):

```
nisaddent -rv -f /etc/hosts hosts
```

Merge the **passwd** map from **myypdomain** with the **passwd.org_dir.nisdomain** table (in verbose mode) (the example assumes that the **/var/yp/myypdomain** directory contains the **yppasswd** map.):

```
nisaddent -mv -y myypdomain passwd nisdomain
```

Merge the **auto.master** map from **myypdomain** with the **auto_master.org_dir** table:

```
nisaddent -m -y myypdomain -Y auto.master \
          -t auto_master.org_dir key-value
```

Dump the **hosts.org_dir** table:

```
nisaddent -d hosts
```

EXTERNAL INFLUENCES

Environment Variables

NIS_DEFAULTS This variable contains a default string that will override the NIS+ standard defaults. If the **-D** switch is used, those values will then override both the **NIS_DEFAULTS** variable and the standard defaults.

NIS_PATH If this variable is set, and neither the *nisdomain* nor the *table* is fully qualified, each directory specified in **NIS_PATH** will be searched until the table is found (see *nisdefaults*(1)).

RETURN VALUE

nisaddent returns **0** on success and **1** on failure.

AUTHOR

nisaddent was developed by Sun Microsystems, Inc.

SEE ALSO

niscat(1), nischmod(1), nisdefaults(1), nistbladm(1), nispopulate(1M), nisserver(1M), nissetup(1M), ypxfr(1M), hosts(4), passwd(4).

NAME

nisclient - initialize NIS+ credentials for NIS+ principals

SYNOPSIS

/usr/lib/nis/nisclient -c [**-x**] [**-o**] [**-v**] [**-l** *network_password*]
[**-d** *NIS+_domain*] *client_name* **. . .**

/usr/lib/nis/nisclient -i [**-x**] [**-v**] **-h** *NIS+_server_host*
[**-a** *NIS+_server_addr*] [**-d** *NIS+_domain*] [**-S 0**|**2**]

/usr/lib/nis/nisclient -u [**-x**] [**-v**]

/usr/lib/nis/nisclient -r [**-x**]

DESCRIPTION

The **nisclient** shell script can be used to:

- create NIS+ credentials for hosts and users
- initialize NIS+ hosts and users
- restore the network service environment

NIS+ credentials are used to provide authentication information of NIS+ clients to NIS+ service.

Use the first synopsis (**-c**) to create individual NIS+ credentials for hosts or users. You must be logged in as a NIS+ principal in the domain for which you are creating the new credentials. You must also have write permission to the local "cred" table. The *client_name* argument accepts any valid host or user name in the NIS+ domain (for example, the *client_name* must exist in the **hosts** or **passwd** table). **nisclient** verifies each *client_name* against both the **hosts** and **passwd** tables, then adds the proper NIS+ credentials for hosts or users. Note that if you are creating NIS+ credentials outside of your local domain, the host or user must exist in the **hosts** or **passwd** tables in both the local and remote domains.

By default, **nisclient** will not overwrite existing entries in the credential table for the hosts and users specified. To overwrite, use the **-o** option. After the credentials have been created, **nisclient** will print the command that must be executed on the client machine to initialize the host or the user. The **-c** option requires a network password for the client which is used to encrypt the secret key for the client. You can either specify it on the command line with the **-l** option or the script will prompt you for it. You can change this network password later with *nispasswd*(1) or *chkey*(1).

n

nisclient -c is not intended to be used to create NIS+ credentials for all users and hosts that are defined in the passwd and hosts tables. To define credentials for all users and hosts, use *nispopulate*(1M).

Use the second synopsis (**-i**) to initialize a NIS+ client machine. **-i** The option can be used to convert machines to use NIS+ or to change the machine's domainname. You must be logged in as super-user on the machine that is to become a NIS+ client. Your administrator must have already created the NIS+ credential for this host by using **nisclient -c** or **nispopulate -C**. You will need the network password your administrator created. **nisclient** will prompt you for the network password to decrypt your secret key and then for this machine's root login password to generate a new set of secret/public keys. If the NIS+ credential was created by your administrator using **nisclient -c**, then you can simply use the initialization command that was printed by the **nisclient** script to initialize this host instead of typing it manually.

To initialize an unauthenticated NIS+ client machine, use the **-i** option with the **-S 0**. With these options, the **nisclient -i** option will not ask for any passwords.

During the client initialization process, files that are being modified are backed up as *files*.no_nisplus. The files that are usually modified during a client initialization are: **/etc/rc.config.d/namesvrs**, **/etc/nsswitch.conf**, **/etc/hosts**, and, if it exists, **/var/nis/NIS_COLD_START**. Note that a file will not be saved if a backup file already exists.

The **-i** option does not set up an NIS+ client to resolve hostnames using DNS. Please refer to the DNS documentation for information on setting up DNS. (See *resolver*(4)).

Use the third synopsis (**-u**) to initialize a NIS+ user. You must be logged in as the user on a NIS+ client machine in the domain where your NIS+ credentials have been created. Your administrator should have already created the NIS+ credential for your username using **nisclient -c** or *nispopulate*(1M). You will need the network password your administrator used to create the NIS+ credential for your username. **nisclient** will prompt you for this network password to decrypt your secret key and then for your login password to generate a new set of secret/public keys.

Use the fourth synopsis (**-r**) to restore the network service environment to whatever you were using before **nisclient -i** was executed. You must be logged in as super-user on the machine that is to be restored. The restore will only work if the machine was initialized with **nisclient -i** because it uses the backup files created by the **-i** option.

Reboot the machine after initializing a machine or restoring the network service.

Options

-a *NIS+_server_addr*	Specifies the IP address for the NIS+ server. This option is used *only* with the **-i** option.
-c	Adds **DES** credentials for NIS+ principals.
-d *NIS+_domain*	Specifies the NIS+ domain where the credential should be created when used in conjuction with the **-c** option. It specifies the name for the new NIS+ domain when used in conjuction with the **-i** option. The default is your current domainname.
-h *NIS+_server_host*	Specifies the NIS+ server's hostname. This option is used *only* with the **-i** option.
-i	Initializes an NIS+ client machine.
-l *network_password*	Specifies the network password for the clients. This option is used *only with the* **-c** option. If this option is not specified, the script will prompt you for the network password.
-o	Overwrite existing credential entries. The default is not to overwrite. This is used *only with the* **-c** option.
-r	Restores the network service environment.
-S 0|**2**	Specifies the authentication level for the NIS+ client. Level **0** is for unauthenticated clients and level **2** is for authenticated (**DES**) clients. The default is to set up with level **2** authentication. This is used *only* with the **-i** option. **nisclient** always uses level **2** authentication (**DES**) for both **-c** and **-u** options. There is no need to run **nisclient** with **-u** and **-c** for level **0** authentication.
-u	Initializes an NIS+ user.
-v	Runs the script in verbose mode.
-x	turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.

n

EXAMPLES

To add the **DES** credential for host *hpws* and user *fred* in the local domain:

```
/usr/lib/nis/nisclient -c hpws fred
```

To add the **DES** credential for host *hpws* and user *fred* in domain *xyz.hp.com.*:

```
/usr/lib/nis/nisclient -c -d xyz.hp.com. hpws fred
```

To initialize host *hpws* as an NIS+ client in domain *xyz.hp.com.* where *nisplus_server* is a server for the domain *xyz.hp.com.*:

```
/usr/lib/nis/nisclient -i -h nisplus_server -d xyz.hp.com.
```

The script will prompt you for the IP address of *nisplus_server* if the server is not found in the **/etc/hosts** file. The **-d** option is needed only if your current domain name is different from the new domain name.

To initialize host hpws as an unauthenticated NIS+ client in domain xyz.hp.com. where nisplus_server is a server for the domain xyz.hp.com.:

```
/usr/lib/nis/nisclient -i -S 0 -h nisplus_server -d xyz.hp.com. \
    -a 129.140.44.1
```

To initialize user *fred* as an NIS+ principal, log in as user *fred* on an NIS+ client machine.

```
/usr/lib/nis/nisclient -u
```

FILES

/var/nis/NIS_COLD_START

This file contains a list of servers, their transport addresses, and their Secure

RPC public keys that serve the machines default domain.

/etc/defaultdomain
the system default domainname

/etc/nsswitch.conf
configuration file for the name-service switch

/etc/hosts local host name database

AUTHOR

nisclient was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), keylogin(1), nis+(1), nispasswd(1), keyserv(1M), nisaddcred(1M), nisinit(1M), nispopulate(1M), nsswitch.conf(4), resolver(4).

NAME

nisinit - NIS+ client and server initialization utility

SYNOPSIS

nisinit -r

nisinit -p Y|D|N *parent_domain host* ...

nisinit -c -H *host* | **-B** | **-C** *coldstart*

DESCRIPTION

nisinit initializes a machine to be a NIS+ client or an NIS+ root master server. It may be easier to use *nisclient*(1M) or *nisserver*(1M) to accomplish this same task.

Options

-r Initialize the machine to be a NIS+ root server. This option creates the file **/var/nis/root.object** and initializes it to contain information about this machine. It uses the **sysinfo()** system call to retrieve the name of the default domain.

To initialize the machine as an NIS+ root server, it is advisable to use the **-r** option of *nisserver*(1M), instead of using **nisinit -r**.

-p Y | D | N *parent_domain host* ...
This option is used on a root server to initialize a **/var/nis/parent.object** to make this domain a part of the namespace above it. Only root servers can have parent objects. A parent object describes the namespace "above" the NIS+ root. If this is an isolated domain, this option should not be used. The argument to this option tells the command what type of name server is serving the domain above the NIS+ domain. When clients attempt to resolve a name that is outside of the NIS+ namespace, this object is returned with the error **NIS_FOREIGNNS** indicating that a name space boundary has been reached. It is up to the client to continue the name resolution process.

The parameter *parent_domain* is the name of the parent domain in a syntax that is native to that type of domain. The list of host names that follow the domain parameter are the names of hosts that serve the parent domain. If there is more than one server for a parent domain, the first host specified should be the master server for that domain.

n

Y Specifies that the parent directory is a NIS version 2 domain.

D Specifies that the parent directory is a DNS domain.

N Specifies that the parent directory is another NIS+ domain. This option is useful for connecting a pre-existing NIS+ subtree into the global namespace.

Note that in the current implementation, the NIS+ clients do not take advantage of the **-p** feature. Also, since the parent object is currently not replicated on root replica servers, it is recommended that this option not be used.

-c Initializes the machine to be a NIS+ client. There are three initialization options available: initialize by coldstart, initialize by hostname, and initialize by broadcast. The most secure mechanism is to initialize from a trusted coldstart file. The second option is to initialize using a hostname that you specify as a trusted host. The third method is to initialize by broadcast and it is the least secure method.

-C *coldstart*
Causes the file *coldstart* to be used as a prototype coldstart file when initializing a NIS+ client. This coldstart file can be copied from a machine that is already a client of the NIS+ namespace. For maximum security, an administrator can encrypt and encode (with *uuencode*(1)) the coldstart file and mail it to an administrator bringing up a new machine. The new administrator would then decode (with **uudecode()**), decrypt, and then use this file with the **nisinit** command to initialize the machine as an NIS+ client. If the coldstart file is from another client in the same domain, the **nisinit** command may be safely skipped and the file copied into the **/var/nis** directory as **/var/nis/NIS_COLD_START**.

-H *hostname*
Specifies that the host *hostname* should be contacted as a trusted NIS+ server. The **nisinit** command will iterate over each transport in the **NETPATH** environment variable and attempt to contact *rpcbind*(1M) on that machine. This hostname *must* be reachable from the client without the name service running. For IP networks this means that there must be an entry in

/etc/hosts for this host when **nisinit** is invoked.

-B Specifies that the **nisinit** command should use an IP broadcast to locate a NIS+ server on the local subnet. Any machine that is running the NIS+ service may answer. No guarantees are made that the server that answers is a server of the organization's namespace. If this option is used, it is advisable to check with your system administrator that the server and domain served are valid. The binding information can be dumped to the standard output using the *nisshowcache*(1M) command.

Note that **nisinit -c** will just enable navigation of the NIS+ name space from this client. To make NIS+ your name service, modify the file **/etc/nsswitch.conf** to reflect that. See *nsswitch.conf*(4) for more details.

RETURN VALUE

nisinit returns **0** on success and **1** on failure.

EXAMPLES

This example initializes the machine as an NIS+ client using the host *freddy* as a trusted server.

```
nisinit -cH freddy
```

This example sets up a client using a trusted coldstart file.

```
nisinit -cC /tmp/colddata
```

This example sets up a client using an IP broadcast.

```
nisinit -cB
```

This example sets up a root server.

```
nisinit -r
```

EXTERNAL INFLUENCES

Environment Variables

NETPATH This environment variable may be set to the transports to try when contacting the NIS+ server (see *netconfig*(4)). The client library will only attempt to contact the server using connection oriented transports.

FILES

/var/nis/NIS_COLD_START
This file contains a list of servers, their transport addresses, and their Secure RPC public keys that serve the machine's default domain.

/var/nis/*hostname***/root.object**
This file describes the root object of the NIS+ namespace. It is a standard XDR-encoded NIS+ directory object that can be modified by authorized clients using the **nis_modify()** interface.

/var/nis/*hostname***/parent.object**
This file describes the namespace that is logically above the NIS+ namespace. The most common type of parent object is a DNS object. This object contains contact information for a server of that domain.

/etc/hosts Internet host table.

AUTHOR

nisinit was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), uuencode(1), nisclient(1M), nisserver(1M), nisshowcache(1M), hosts(4), netconfig(4), nisfiles(4).

NAME

nislog - display the contents of the NIS+ transaction log

SYNOPSIS

/usr/sbin/nislog [**-h** *num* | **-t** *num*] [**-v**] [*directory* ...]

DESCRIPTION

nislog displays the contents of the NIS+ server transaction log on the standard output. This command can be used to track changes in the namespace. The **/var/nis/***hostname***.log** file contains the transaction log maintained by the NIS+ server. *hostname* is the string returned by **uname -n**. When updates occur, they are logged to this file and then propagated to replicas as log transactions. When the log is checkpointed, updates that have been propagated to the replicas are removed.

The **nislog** command can only be run on an NIS+ server by superuser. It displays the log entries for that server only.

If *directory* is not specified, the entire log is searched. Otherwise, only those log entries that correspond to the specified directories are displayed.

Options

-h [*num*] Display *num* transactions from the "head" of the log. If the numeric parameter is omitted, it is assumed to be **1**. If the numeric parameter is **0**, only the log header is displayed.

-t [*num*] Display *num* transactions from the "tail" of the log. If the numeric parameter is omitted, it is assumed to be **1**. If the numeric parameter is **0**, only the log header is displayed.

-v Verbose mode.

FILES

/var/nis/*hostname***.log**
transaction log

AUTHOR

nislog was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), uname(1), rpc.nisd(1M), nisfiles(4).

NAME

nisping - send ping to NIS+ servers

SYNOPSIS

/usr/lib/nis/nisping [**-uf**] [**-H** *hostname*] [**-r** | *directory*]

/usr/lib/nis/nisping -C [**-a**] [**-H** *hostname*] [*directory*]

DESCRIPTION

In the first *SYNOPSIS* line, the **nisping** command sends a **ping** to all replicas of a NIS+ directory. Once a replica receives a ping, it will check with the master server for the directory to get updates. Prior to pinging the replicas, this command attempts to determine the last update "seen" by a replica and the last update logged by the master. If these two timestamps are the same, the ping is not sent. The **-f** (force) option will override this feature.

Under normal circumstances, NIS+ replica servers get the new information from the master NIS+ server within a short time. Therefore, there should not be any need to use **nisping**.

In the second *SYNOPSIS* line, the **nisping -C** command sends a checkpoint request to the servers. If no *directory* is specified, the home domain, as returned by *nisdefaults*(1), is checkpointed. If all directories, served by a given server, have to be checkpointed, then use the **-a** option.

On receiving a checkpoint request, the servers would commit all the updates for the given *directory* from the table log files to the database files. This command, if sent to the master server, will also send updates to the replicas if they are out of date. This option is needed because the database log files for NIS+ are not automatically checkpointed. **nisping** should be used at frequent intervals (such as once a day) to checkpoint the NIS+ database log files. This command can be added to the *crontab*(1) file. If the database log files are not checkpointed, their sizes will continue to grow.

Options

-a	Checkpoint all directories on the server.
-C	Send a request to checkpoint, rather than a ping, to each server. The servers schedule to commit all the transactions to stable storage.
-H *hostname*	Only the host *hostname* is sent the ping, checked for an update time, or checkpointed.
-f	Force a ping, even though the timestamps indicate there is no reason to do so. This option is useful for debugging.
-r	This option can be used to update or get status about the root object from the root servers, especially when new root replicas are added or deleted from the list. If used without **-u** option, **-r** will send a ping request to the servers serving the root domain. When the replicas receive a ping, they will update their root object if needed. The **-r** option can be used with all other options except with the **-C** option; the root object need not be checkpointed.
-u	Display the time of the last update; no servers are sent a ping.

n

RETURN VALUE

-1	No servers were contacted, or the server specified by the **-H** switch could not be contacted.
0	Success.
1	Some, but not all, servers were successfully contacted.

EXAMPLES

This example pings all replicas of the default domain:

```
nisping
```

Note that this example will not ping the the **org_dir** and **group_dir** subdirectories within this domain.

This example pings the server *example* which is a replica of the *org_dir.foo.com.* directory:

```
nisping -H example org_dir.foo.com.
```

This example checkpoints all servers of the *org_dir.bar.com.* directory.

```
nisping -C org_dir.bar.com.
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found.

AUTHOR

nisping was developed by Sun Microsystems, Inc.

SEE ALSO

crontab(1), nisdefaults(1), nislog(1M), nisfiles(4).

NOTES

If the server specified by the **-H** option does not serve the directory, then no ping is sent.

NAME

nispopulate - populate the NIS+ tables in a NIS+ domain

SYNOPSIS

/usr/lib/nis/nispopulate -Y [**-x**] [**-f**] [**-n**] [**-u**] [**-v**] [**-S 0**|**2**] [**-l** *network_passwd*] [**-d** *NIS+_domain*] **-h** *NIS_server_host* [**-a** *NIS_server_addr*] **-y** *NIS_domain* [*table*] ...

/usr/lib/nis/nispopulate -F [**-x**] [**-f**] [**-u**] [**-v**] [**-S 0**|**2**] [**-d** *NIS+_domain*] [**-l** *network_passwd*] [**-p** *directory_path*] [*table*] ...

/usr/lib/nis/nispopulate -C [**-x**] [**-f**] [**-v**] [**-d** *NIS+_domain*] [**-l** *network_passwd*] [*hosts* | *passwd*]

DESCRIPTION

The **nispopulate** shell script can be used to populate NIS+ tables in a specified domain from their corresponding files or NIS maps. **nispopulate** assumes that the tables have been created either through *nisserver*(1M) or *nissetup*(1M).

The table argument accepts standard names that are used in the administration of HP-UX systems and non-standard *key-value* type tables. See *nisaddent*(1M) for more information on *key-value* type tables. If the table argument is not specified, **nispopulate** will automatically populate each of the standard tables. These standard (default) tables are: **auto_master**, **auto_home**, **ethers**, **group**, **hosts**, **networks**, **passwd**, **protocols**, **services**, **rpc**, **netmasks**, **bootparams**, **netgroup**, **aliases** and **shadow**. Note that the **shadow** table is only used when populating from files. The non-standard tables that **nispopulate** accepts are those of *key-value* type. These tables must first be created manually with the *nistbladm*(1) command.

Use the first *SYNOPSIS* (**-Y**) to populate NIS+ tables from NIS maps. **nispopulate** uses *ypxfr*(1M) to transfer the NIS maps from the NIS servers to the **/var/yp/** *NIS_domain* directory on the local machine. Then, it uses these files as the input source. Note that *NIS_domain* is case sensitive. Make sure there is enough disk space for that directory.

Use the second *SYNOPSIS* (**-F**) to populate NIS+ tables from local files. **nispopulate** will use those files that match the table name as input sources in the current working directory or in the specified directory.

Note that when populating the **hosts** and **passwd** tables, **nispopulate** will automatically create the NIS+ credentials for all users and hosts that are defined in the **hosts** and **passwd** tables, respectively. A network passwd is required to create these credentials. This network password is used to encrypt the secret key for the new users and hosts. This password can be specified using the **-l** option or it will use the default password, "nisplus". **nispopulate** will not overwrite any existing credential entries in the credential table. Use *nisclient*(1M) to overwrite the entries in the **cred** table. It creates both LOCAL and DES credentials for users, and only DES credentials for hosts. To disable automatic credential creation, specify the **-S 0** option.

The third *SYNOPSIS* (**-C**) is used to populate NIS+ credential table with level 2 authentication (**DES**) from the passwd and hosts tables of the specified domain. The valid table arguments for this operation are passwd and hosts. If this argument is not specified then it will use both passwd and hosts as the input source.

If **nispopulate** was earlier used with **-S 0** option, then no credentials were added for the hosts or the users. If later the site decides to add credentials for all users and hosts, then this (**-C**) option can be used to add credentials.

Options

-a *NIS_server_addr*
: specifies the IP address for the NIS server. This option is *only* used with the **-Y** option.

-C
: populate the NIS+ credential table from passwd and hosts tables using **DES** authentication (security level 2).

-d *NIS+_domain.*
: specifies the NIS+ domain. The default is the local domain.

-F
: populates NIS+ tables from files.

-f
: forces the script to populate the NIS+ tables without prompting for confirmation.

n

-h *NIS_server_host*	specifies the NIS server hostname from where the NIS maps are copied. This is *only* used with the **-Y** option. This host must already exist in either the NIS+ **hosts** table or **/etc/hosts** file. If the hostname is not defined, the script will prompt you for its IP address, or you can use the **-a** option to specify the address manually.
-l *network_passwd*	specifies the network password for populating the NIS+ credential table. This is *only* used when you are populating the **hosts** and **passwd** tables. The default passwd is **nisplus**.
-n	does not overwrite local NIS maps in **/var/yp/***NISdomain* directory if they already exist. The default is to overwrite the existing NIS maps in the local **/var/yp/***NISdomain* directory. This is *only* used with the **-Y** option.
-p *directory_path*	specifies the directory where the files are stored. This is *only* used with the **-F** option. The default is the current working directory.
-S 0|**2**	specifies the authentication level for the NIS+ clients. Level **0** is for unauthenticated clients and no credentials will be created for users and hosts in the specified domain. Level 2 is for authenticated (**DES**) clients and DES credentials will be created for users and hosts in the specified domain. The default is to set up with level **2** authentication (**DES**). There is no need to run **nispopulate** with **-C** for level **0** authentication.
-u	updates the NIS+ tables (ie., adds, deletes, modifies) from either files or NIS maps. This option should be used to bring an NIS+ table up to date when there are only a small number of changes. The default is to add to the NIS+ tables without deleting any existing entries. Also, see the **-n** option for updating NIS+ tables from existing maps in the **/var/yp** directory.
-v	runs the script in verbose mode.
-x	turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.
-Y	populate the NIS+ tables from NIS maps.
-y *NIS_domain*	specifies the NIS domain to copy the NIS maps from. This is *only* used with the **-Y** option. The default domainname is the same as the local domainname.

EXTERNAL INFLUENCES

TMPDIR

nispopulate normally creates temporary files in the directory **/tmp**. You may specify another directory by setting the environment variable **TMPDIR** to your chosen directory. If **TMPDIR** is not a valid directory, then **nispopulate** will use **/tmp**.

EXAMPLES

To populate all the NIS+ standard tables in the domain *xyz.hp.com.* from NIS maps of the *yp.hp.com* domain as input source where host *yp_host* is a YP server of *yp.hp.com* :

```
/usr/lib/nis/nispopulate -Y -y yp.hp.com -h yp_host -d xyz.hp.com.
```

To update all of the NIS+ standard tables from the same NIS domain and hosts shown above:

```
/usr/lib/nis/nispopulate -Y -u -y yp.hp.com \
     -h yp_host -d xyz.hp.com.
```

To populate the **hosts** table in domain *xyz.hp.com.* from the hosts file in the **/var/nis/files** directory using "somepasswd" as the network password for key encryption:

```
/usr/lib/nis/nispopulate -F -p /var/nis/files -l somepasswd hosts
```

To populate the passwd table in domain xyz.hp.com. from the passwd file in the **/var/nis/files** directory without automatically creating the NIS+ credentials:

```
/usr/lib/nis/nispopulate -F -p /var/nis/files -d xys.hp.com. \
     -S 0 passwd
```

To populate the credential table in domain xyz.hp.com. for all users defined in the passwd table.

```
/usr/lib/nis/nispopulate -C -d xys.hp.com. passwd
```

To create and populate a non-standard key-value type NIS+ table, "private", from the file **/var/nis/files/private**: (nispopulate assumes that the private.org_dir key-value type table has already been created).

```
/usr/bin/nistbladm -D access=og=rmcd,nw=r \
    -c private key=S,nogw= value=,nogw= private.org.dir
/usr/lib/nis/nispopulate -F -p /var/nis/files private
```

FILES

/etc/hosts	local host name database
/var/yp	NIS(YP) domain directory
/var/nis	NIS+ domain directory
/tmp	

AUTHOR

nispopulate was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nistbladm(1), nisaddcred(1M), nisaddent(1M), nisclient(1M), nisserver(1M), nissetup(1M), rpc.nisd(1M), ypxfr(1M).

NAME

nisserver - set up NIS+ servers

SYNOPSIS

/usr/lib/nis/nisserver -r [**-x**] [**-f**] [**-v**] [**-Y**] [**-d** *NIS+_domain*] [**-g** *NIS+_groupname*] [**-l** *network_passwd*]

/usr/lib/nis/nisserver -M [**-x**] [**-f**] [**-v**] [**-Y**] **-d** *NIS+_domain* [**-g** *NIS+_groupname*] [**-h** *NIS+_server_host*]

/usr/lib/nis/nisserver -R [**-x**] [**-f**] [**-v**] [**-Y**] [**-d** *NIS+_domain*] [**-h** *NIS+_server_host*]

DESCRIPTION

The **nisserver** shell script can be used to set up a root master, non-root master, and replica NIS+ servers with level 2 security (**DES**).

When setting up a new domain, this script creates the NIS+ directories (including **groups_dir** and **org_dir**) and system table objects for the domain specified. It does not populate the tables. You will need to use *nispopulate*(1M) to populate the tables.

Use the first *SYNOPSIS* (**-r**) to set up a root master server. You must be logged in as super-user on the server machine.

Use the second *SYNOPSIS* (**-M**) to set up a non-root master server for the specified domain. You must be logged in as an NIS+ principal on a NIS+ machine and have create permission to the parent directory of the domain that you are setting up. The new non-root master server machine must already be an NIS+ client (see *nisclient*(1M)) and have the **rpc.nisd** daemon running (see *rpc.nisd*(1M)).

Use the third *SYNOPSIS* (**-R**) to set up a replica server for both root and non-root domains. You must be logged in as an NIS+ principal on an NIS+ machine and have create permission to the parent directory of the domain that you are replicating. The new replica server machine must already be an NIS+ client (see *nisclient*(1M)) and have the **rpc.nisd** daemon running (see *rpc.nisd*(1M)).

n

Options

-d *NIS+_domain*	specifies the name for the NIS+ domain. The default is your local domain.
-f	forces the NIS+ server setup without prompting for confirmation.
-g *NIS+_groupname*	specifies the NIS+ group name for the new domain. This option is not valid with **-R** option. The default group is **admin**.*domain*.
-h *NIS+_server_host*	specifies the hostname for the NIS+ server. It must be a valid host in the local domain. Use a fully qualified hostname (for example, hostx.xyz.hp.com.) to specify a host outside of your local domain. This option is ONLY used for setting up non-root master or replica servers. The default for non-root master server setup is to use the same list of servers as the parent domain. The default for replica server setup is the local hostname.
-l *network_password*	specifies the network password with which to create the credentials for the root master server. This option is ONLY used for master root server setup (**-r**option). If this option is not specified, the script will prompt you for the login password.
-r	sets up the server as a root master server. Use the **-R** option to set up a root replica server.
-v	runs the script in verbose mode.
-x	turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.
-M	sets up the specified host as a master server. Make sure that *rpc.nisd*(1M) is running on the new master server before this command is executed.
-R	sets up the specified host as a replica server. Make sure that *rpc.nisd*(1M) is running on the new replica server.
-Y	sets up an NIS+ server with NIS-compatibility mode. The default is to set up the server without NIS-compatibility mode.

EXAMPLES

To set up a root master server for domain *hp.com.* :

```
root_server# /usr/lib/nis/nisserver -r -d hp.com.
```

For the following examples make sure that the new servers are NIS+ clients and **rpc.nisd** is running on these hosts before executing **nisserver**.

To set up a replica server for domain *hp.com.* on host *hpreplica* :

```
root_server# /usr/lib/nis/nisserver -R -d hp.com. -h hpreplica
```

To set up a non-root master server for domain *xyz.hp.com.* on host *hpxyz* with the NIS+ groupname as *admin-mgr.xyz.hp.com.* :

```
root_server# /usr/lib/nis/nisserver -M -d xyz.hp.com. \
     -h hpxyz -g admin-mgr.xyz.hp.com.
```

To set up a non-root replica server for domain *xyz.hp.com.* on host *hpabc*:

```
hpxyz# /usr/lib/nis/nisserver -R -d xyz.hp.com. -h hpabc
```

AUTHOR

nisserver was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nisgrpadm(1), nismkdir(1), nisaddcred(1M), nisclient(1M), nisinit(1M), nismkdir(1), nispopulate(1M), nissetup(1M), rpc.nisd(1M).

n

NAME

nissetup - initialize a NIS+ domain

SYNOPSIS

/usr/lib/nis/nissetup [**-Y**] [*domain*]

DESCRIPTION

nissetup is a shell script that sets up a NIS+ domain to serve clients that wish to store system administration information in a domain named *domain*. This domain should already exist prior to executing this command (see *nismkdir*(1) and *nisinit*(1M)).

A NIS+ domain consists of a NIS+ directory and its subdirectories: **org_dir** and **groups_dir**. **org_dir** stores system administration information and **groups_dir** stores information for group access control.

nissetup creates the subdirectories **org_dir** and **groups_dir** in *domain*. Both subdirectories will be replicated on the same servers as the parent domain. After the subdirectories are created, **nissetup** creates the default tables that NIS+ serves. These are **auto_master**, **auto_home**, **bootparams**, **cred**, **ethers**, **group**, **hosts**, **mail_aliases**, **netmasks**, **networks**, **passwd**, **protocols**, **rpc**, **services**, and **timezone**. The **nissetup** script uses the *nistbladm*(1) command to create these tables. The script can be easily customized to add site specific tables that should be created at setup time.

This command is normally executed just once per domain.

Options

-Y Specify that the domain will be served as both a NIS+ domain as well as an NIS domain using the backward compatibility flag. This will set up the domain to be less secure by making all the system tables readable by unauthenticated clients as well.

AUTHOR

nissetup was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nismkdir(1), nistbladm(1), nisaddent(1M), nisinit(1M), nisserver(1M).

NOTES

While this command creates the default tables, it does not initialize them with data. This is accomplished with the *nisaddent*(1M) command.

It is easier to use the *nisserver*(1M) script to create subdirectories and the default tables.

NAME

nisshowcache - NIS+ utility to print out the contents of the shared cache file

SYNOPSIS

/usr/lib/nis/nisshowcache [**-v**]

DESCRIPTION

nisshowcache prints out the contents of the per-machine NIS+ directory cache that is shared by all processes accessing NIS+ on the machine. By default, **nisshowcache** only prints out the directory names in the cache along with the cache header. The shared cache is maintained by *nis_cachemgr*(1M).

Options

-v Verbose mode. Print out the contents of each directory object, including information on the server name and its universal addresses.

DIAGNOSTICS

Error messages are sent to the *syslogd*(1M) daemon.

FILES

/var/nis/NIS_SHARED_DIRCACHE

AUTHOR

nisshowcache was developed by Sun Microsystems, Inc.

SEE ALSO

nis_cachemgr(1M), syslogd(1M), nisfiles(4).

NAME

nisstat - report NIS+ server statistics

SYNOPSIS

/usr/lib/nis/nisstat [**-H** *host*] [*directory*]

DESCRIPTION

The **nisstat** command queries a NIS+ server for various statistics about its operations. These statistics may vary between implementations and from release to release. Not all statistics are available from all servers. Requesting a statistic from a server that does not support that statistic is never fatal, it simply returns 'unknown statistic.'

By default, statistics are fetched from the server(s) of the NIS+ directory for the default domain. If *directory* is specified, servers for that directory are queried.

Supported statistics for this release are as follows:

root server This reports whether the server is a root server.

NIS compat mode This reports whether the server is running in NIS compat mode.

DNS forwarding in NIS mode
This reports whether the server in NIS compat mode will forward host lookup calls to DNS.

security level This reports the security level of this server.

serves directories This lists the directories served by this server.

Operations This statistic returns results in the form:

OP=*opname***:C=***calls***:E=***errors***:T=***micros*

Where *opname* is replaced by the RPC procedure name or operation, *calls* is the number of calls to this procedure that have been made since the server started running. *errors* is the number of errors that have occurred while processing a call, and *micros* is the average time in microseconds to complete the last 16 calls.

Directory Cache This statistic reports the number of calls to the internal directory object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.

Group Cache This statistic reports the number of calls to the internal NIS+ group object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.

Static Storage This statistic reports the number of bytes the server has allocated for its static storage buffers.

Dynamic Storage This statistic reports the amount of heap the server process is currently using.

Uptime This statistic reports the time since the service has been running.

Options

-H *host* Normally all servers for the directory are queried. With this option, only the machine named *host* is queried. If the named machine does not serve the directory, no statistics are returned.

directory If specified, servers for that directory are queried.

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found (see *nisdefaults*(1)).

AUTHOR

nisstat was developed by Sun Microsystems, Inc.

SEE ALSO

nisdefaults(1).

NAME

nisupdkeys - update the public keys in a NIS+ directory object

SYNOPSIS

/usr/lib/nis/nisupdkeys [**-a** | **-C**] [**-H** *host*] [*directory*]

/usr/lib/nis/nisupdkeys -s [**-a** | **-C**] **-H** *host*

DESCRIPTION

This command updates the public keys in an NIS+ directory object. When the public key for a NIS+ server is changed, the new key must be propagated to all directory objects that reference that server.

nisupdkeys reads a directory object and attempts to get the public key for each server of that directory. These keys are placed in the directory object and the object is then modified to reflect the new keys.

If *directory* is present, the directory object for that directory is updated. Otherwise the directory object for the default domain is updated.

On the other hand, **nisupdkeys -s** gets a list of all the directories served by *host* and updates those directory objects. This assumes that the caller has adequate permission to change all the associated directory objects. The list of directories being served by a given server can also be obtained by *nisstat*(1M).

Before you do this operation, make sure that the new address/public key has been propagated to all replicas.

Options

-a	Update the universal addresses of the NIS+ servers in the directory object. Currently, this only works for the TCP/IP family of transports. This option should be used when the IP address of the server is changed. The server's new address is resolved using **gethostbyname()** on this machine. The **/etc/nsswitch.conf** file must point to the correct source for the *hosts* entry for this resolution to work.
-C	Specify to clear rather than set the public key. Communication with a server that has no public key does not require the use of secure RPC.
-H *host*	Limit key changes only to the server named *host*. If the hostname is not a fully qualified NIS+ name, then it is assumed to be a host in the default domain. If the named host does not serve the directory, no action is taken.
-s	Update all the NIS+ directory objects served by the specified server. This assumes that the caller has adequate access rights to change all the associated directory objects. If the NIS+ principal making this call does not have adequate permissions to update the directory objects, those particular updates will fail and the caller will be notified. If the **rpc.nisd** on *host* cannot return the list of servers it serves, the command will print an error message. The caller would then have to invoke **nisupdkeys** multiple times (as in the first *SYNOPSIS*), once per NIS+ directory that it serves.

EXAMPLES

The following example updates the keys for servers of the *foo.bar.* domain.

```
nisupdkeys foo.bar.
```

This example updates the key for host *fred* which serves the *foo.bar.* domain.

```
nisupdkeys -H fred foo.bar.
```

This example clears the public key for host *wilma* in the *foo.bar.* directory.

```
nisupdkeys -CH wilma foo.bar.
```

This example updates the public key in all directory objects that are served by the host *wilma*.

```
nisupdkeys -s -H wilma
```

AUTHOR

nisupdkeys was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), niscat(1), nisaddcred(1M), gethostent(3N), nis_objects(3N).

NOTES

The user executing this command must have modify access to the directory object for it to succeed. The existing directory object can be displayed with the *niscat*(1) command using the **-o** option.

This command does not update the directory objects stored in the **NIS_COLD_START** file on the NIS+ clients.

If a server is also the root master server, then **nisupdkeys -s** cannot be used to update the root directory.

NAME

ntpdate - set the date and time via NTP

SYNOPSIS

ntpdate [**-bdos**] [**-a** *key#*] [**-e** *authdelay*] [**-k** *keyfile*] [**-p** *samples*] [**-t** *timeout*] *server* ...

DESCRIPTION

ntpdate sets the local date and time by polling the Network Time Protocol server(s) on the host(s) given as arguments to determine the correct time. It must be run as root on the local host. A number of samples are obtained from each of the servers specified and the standard NTP clock filter and selection algorithms are applied to select the best of these. Typically, **ntpdate** can be inserted in the startup script to set the time of day at boot time and/or can be run from time-to-time via *cron*(1M). Note that **ntpdate**'s reliability and precision will improve dramatically with greater numbers of servers. While a single server may be used, better performance and greater resistance to insanity on the part of any one server will be obtained by providing at least three or four servers, if not more.

Time adjustments are made by **ntpdate** in one of two ways. If **ntpdate** determines your clock is off by more than 0.5 seconds it will simply step the time by calling *settimeofday*(2). If the error is less than 0.5 seconds, however, it will by default slew the clock's time via a call to *adjtime*(2) with the offset. The latter technique is less disruptive and more accurate when the offset is small, and works quite well when **ntpdate** is run by *cron*(1M) every hour or two. The adjustment made in the latter case is actually 50% larger than the measured offset since this will tend to keep a badly drifting clock more accurate (at some expense to stability, though this tradeoff is usually advantageous). At boot time, however, it is usually better to always step the time. This can be forced in all cases by specifying the **-b** switch on the command line. The **-s** switch tells **ntpdate** to log its actions via the *syslog*(3C) facility rather than to the standard output, a useful option when running the program from *cron*(1M).

The **-d** flag may be used to determine what **ntpdate** will do without it actually doing it. Information useful for general debugging will also be printed. By default **ntpdate** claims to be an NTP version 3 implementation in its outgoing packets. As some older software will decline to respond to version 3 queries, the **-o** switch can be used to force the program to poll as a version 2 implementation instead.

The number of samples **ntpdate** acquires from each server can be set to between 1 and 8 inclusive using the **-p** switch. The default is 4. The time it will spend waiting for a response can be set using the **-t** switch, and will be rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

ntpdate will authenticate its transactions if need be. The **-a** switch specifies that all packets should be authenticated using the key number indicated. The **-k** switch allows the name of the file from which the keys may be read to be modified from the default of **/etc/ntp.keys**. This file should be in the format described in *xntpd*(1M). The **-e** option allows the specification of an authentication processing delay, in seconds (see *xntpd*(1M) for details). This number is usually small enough to be negligible for **ntpdate**'s purposes, though specifying a value may improve timekeeping on very slow CPU's.

ntpdate will decline to set the date if an NTP server daemon (e.g. *xntpd*(1M)) is running on the same host. When running **ntpdate** on a regular basis from *cron*(1M) as an alternative to running a daemon, doing so once every hour or two will result in precise enough timekeeping to avoid stepping the clock.

FILES

/etc/ntp.keys	Contains the encription keys used by **ntpdate**.

SEE ALSO

xntpd(1M), syslog(3C),
DARPA Internet Request For Comments RFC1035 Assigned Numbers.

AUTHOR

ntpdate was developed by Dennis Ferguson at the University of Toronto

NAME

ntpq - standard Network Time Protocol query program

SYNOPSIS

ntpq [**-inp**] [**-c** *command*] [*host*] [...]

DESCRIPTION

ntpq is used to query NTP servers about current state and to request changes in that state. The program may be run either in interactive mode or controlled using command line arguments. Requests to read and write arbitrary variables can be assembled, with raw and pretty-printed output options being available. **ntpq** can also obtain and print a list of peers in a common format by sending multiple queries to the server.

If one or more request options is included on the command line when **ntpq** is executed, each of the requests will be sent to the NTP servers running on each of the hosts given as command line arguments, or on *localhost* by default. **ntpq** will run in the interactive mode if no request options are given. It will attempt to read interactive format commands from the standard input and execute these commands on the NTP server running on the first host given on the command line, again defaulting to *localhost* when no other host is specified. **ntpq** will prompt for commands if the standard input is a terminal device.

ntpq uses NTP mode 6 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network which permits it. Note that since NTP is a UDP protocol this communication will be somewhat unreliable, especially over large distances in terms of network topology. **ntpq** makes one attempt to retransmit requests, and will time requests out if the remote host is not heard from within a suitable timeout time.

Command line options are described following.

-c The following *command* argument is interpreted as an interactive format command and is added to the list of commands to be sent to and executed on the specified host(s). Multiple **-c** options may be given.

-i Force **ntpq** to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

-n Output all host addresses in dotted-quad numeric format rather than converting to the canonical host names.

-p Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the **peers** interactive command.

INTERACTIVE INTERNAL COMMANDS

Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. The output of a command is normally sent to the standard output, but optionally the output of individual commands may be sent to a file by appending a **>** followed by a file name, to the command line.

A number of interactive format commands are executed entirely within the **ntpq** program itself and do not result in NTP mode 6 requests being sent to a server. These are described following.

? [*command_keyword*]

A **?** by itself will print a list of all the command keywords known to this incarnation of **ntpq**. A **?** followed by a command keyword will print function and usage information about the command.

timeout *millseconds*

Specify a timeout period for responses to server queries. The default is about 5000 milliseconds. Note that since **ntpq** retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

host *hostname*

Set the host to which future queries will be sent. *Hostname* may be either a host name or a numeric address.

keyid *#*

This command allows the specification of a key number to be used to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose.

passwd

This command prompts you to type in a password (which will not be echoed) which will be used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose if such requests are to be successful.

hostnames yes|no

If *yes* is specified, host names are printed in information displays. If *no* is given, numeric addresses are printed instead. The default is *yes* unless modified using the command line **-n** switch.

raw

Causes all output from query commands to be printed as received from the remote server. The only formating/interpretation done on the data is to transform nonascii data into a printable (but barely understandable) form.

cooked

Causes output from query commands to be **cooked**. Variables which are recognized by the server will have their values reformatted for human consumption.

ntpversion 1|2|3

Sets the NTP version number which **ntpq** claims in packets. Defaults to **3**. Note that mode 6 control messages (and modes, for that matter) didn't exist in NTP version 1. There appear to be no servers left which demand version 1.

version

Display the version of **ntpq**.

keytype m|d

set the authentication type to md5 [**m**] or des [**d**].

authenticate yes|no

Normally **ntpq** does not authenticate requests unless they are write requests. The command **authenticate yes** causes **ntpq** to send authentication with all requests it makes.

debug more|less|no

Turns internal query program debugging on and off.

quit

Exit **ntpq**.

CONTROL MESSAGE COMMANDS

Each peer known to an NTP server has a 16 bit integer *association identifier* assigned to it. NTP control messages which carry peer variables must identify the peer to which the values correspond by including its association ID. An association ID of 0 is special, and indicates the variables are system variables, whose names are drawn from a separate name space.

Control message commands result in one or more NTP mode 6 messages being sent to the server, and cause the data returned to be printed in some format. Most commands currently implemented send a single message and expect a single response. The current exceptions are the **peers** command, which will send a preprogrammed series of messages to obtain the data it needs, and the **mreadlist** and **mreadvar** commands, which will iterate over a range of associations.

associations

Obtains and prints a list of association identifiers and peer statuses for in-spec peers of the server being queried. The list is printed in columns. The first of these is an index numbering the associations from 1 for internal use; the second is the actual association identifier returned by the server; and the third is the status word for the peer. This is followed by a number of columns containing data decoded from the status word. Note that the data returned by the **associations** command is cached internally in **ntpq**. The index can be use when dealing with servers that use association identifiers which are hard for humans to type. The form *&index* may be used for any subsequent commands that require an association identifier as an argument.

lassocations

Obtains and prints a list of association identifiers and peer statuses for all associations for which the server is maintaining state. This command differs from the **associations** command only for servers which retain state for out-of-spec client associations (i.e. fuzzballs). Such associations are normally omitted from the display when the **associations** command is used, but are included in the output of **lassociations**

passociations

Prints association data concerning in-spec peers from the internally cached list of associations. This command performs identically to the **associations** except that it displays the internally stored data rather than making a new query.

lpassociations

Print data for all associations, including out-of-spec client associations, from the internally cached list of associations. This command differs from **passociations** only when dealing with fuzzballs.

pstatus *assocID*

Sends a read status request to the server for the given association. The status value, the names, and values of the peer variables returned will be printed.

readvar [*assocID*] [<*variable_name*>[,...]]

Requests that the values of the specified variables be returned by the server by sending a read variables request. If the association ID is omitted or is **0** the variables are system variables; otherwise they are peer variables and the values returned will be those of the corresponding peer. Omitting the variable list will send a request with no data which should induce the server to return a default display.

rv [*assocID*] [<*variable_name*>[,...]]

An easy-to-type short form for the **readvar** command.

writevar *assocID* <*variable_name*>=<*value*>[,...]

Like the **readvar** request, except the specified variables are written instead of read.

readlist [*assocID*]

n

Requests that the values of the variables in the internal variable list be returned by the server. If the association ID is omitted or is 0, the variables are assumed to be system variables; otherwise they are treated as peer variables. If the internal variable list is empty a request is sent without data, which should induce the remote server to return a default display.

rl [*assocID*]

An easy-to-type short form of the **readlist** command.

mreadvar *assocID assocID* [<*variable_name*>[,...]]

Like the **readvar** command except the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent **associations** command.

mrv *assocID assocID* [<*variable_name*>[,...]]

An easy-to-type short form of the **mreadvar** command.

mreadlist *assocID assocID*

Like the **readlist** command except the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent **associations** command.

mrl *assocID assocID*

An easy-to-type short form of the **mreadlist** command.

clocklist [*assocID*]

Requests for the server's clock variables to be sent. Servers which have a radio clock or other external synchronization will respond positively to this. If the association identifier is omitted or is **0**, the request is for the variables of the **system clock** and will generally get a positive response from all servers with a clock. If the server treats clocks as pseudo-peers, and hence can possibly have more than one clock connected at once, referencing the appropriate peer association ID will show the variables of a particular clock.

clockvar [*assocID*] [<*variable_name*>[,...]]

Requests that a list of the server's clock variables be sent. Servers which have a radio clock or other external synchronization will respond positively to this. If the association identifier is omitted or is **0**, the request is for the variables of the **system clock** and will generally get a positive response from all servers with a clock. If the server treats clocks as pseudo-peers, and hence can possibly have more than one clock connected at once, referencing the appropriate peer association ID will show the variables of a particular clock. Omitting the variable list will cause the server to return a default variable display.

cv [*assocID*] [<*variable_name*>[,...]]

An easy-to-type short form of the **clockvar** command.

peers

Obtains a list of in-spec peers of the server, along with a summary of each peer's state. Summary information includes the address of the remote peer, the reference ID (0.0.0.0 if the reference ID is unknown), the stratum of the remote peer, the polling interval, in seconds, the reachability register, in octal, and the current estimated delay, offset and dispersion of the peer, all in seconds. In addition, the character in the left margin indicates the fate of this peer in the clock selection algorithm. Characters only appear beside peers which were included in the final stage of the clock selection algorithm. A **.** indicates that this peer was cast off in the falseticker detection, while a **+** indicates that the peer made it through. A ***** denotes the peer with which the server is currently synchronizing. Note that since the **peers** command depends on the ability to parse the values in the responses it gets, it may fail to work from time to time with servers that poorly control the data formats.

The contents of the host field may be one of four forms. It may be a host name, an IP address, a reference clock implementation name with its parameter, or **REFCLK**(<*implementation number*>, <*parameter*>). On **hostnames no,** only IP-addresses will be displayed.

lpeers

Like **peers,** except a summary is printed of all associations for which the server is maintaining state. This can produce a much longer list of peers from fuzzball servers.

opeers

An old form of the **peers** command with the reference ID replaced by the local interface address.

lopeers

An old form of the **lpeers** command with the reference ID replaced by the local interface address.

FILES

/etc/ntp.keys	Contains the encription keys used for authentication.

AUTHOR

ntpq was developed by Dennis Ferguson at the University of Toronto.

SEE ALSO

ntpdate(1M), xntpd(1M),
DARPA Internet Request For Comments RFC1035 Assigned Numbers.

NAME

ocd - outbound connection daemon used by DDFA software

SYNOPSIS

ocd -f*pseudonym* **-n***node_name* [**-b***board_no*] [**-c***config_file*] [**-l***log_level*] [**-p***port_no*]

DESCRIPTION

The Outbound Connection Daemon (**ocd**) is part of the Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software. It manages the connection and data transfer to the remote terminal server port. It can be spawned from the Dedicated Port Parser (**dpp**) or run directly from the shell.

For performance reasons, **ocd** does not have a debug mode. However, a version called **ocdebug** with debug facilities is available.

See *ddfa*(7) for more information on how to configure the DDFA software and for an explanation of how it works.

ocd logs important messages and error conditions to **/var/adm/syslog**.

Options

ocd recognizes the following options:

-b*board_no* The board number of a DTC. If it is omitted, the port number option must contain the full TCP service port address. The **-b** and **-p** options must not be used if the IP address given in the **-n** option is the IP address of a port.

If the **-n** option explicitly names a terminal server port, the **-b** option is not needed.

-c*config_file* Specify the name (including the absolute path) of the configuration file used to profile the terminal server port. If this option is omitted, the default values specified in the default **pcf** file (**/usr/examples/ddfa/pcf**) are used. If the file specified does not exist, an error message is logged and the following values are used (note that the values for **open_tries** and **open_timer** are different from the default values):

```
telnet_mode:      enable
timing_mark:      enable
telnet_timer:     120
binary_mode:      disable
open_tries:       0
open_timer:       0
close_timer:      0
status_request:   disable
status_timer:     30
eight_bit:        disable
tcp_nodelay:      enable
```

-f*pseudonym* The absolute or relative path to the device file that is linked by the software to the reserved **pty**. Applications use *pseudonym* and not the dynamically allocated **pty** slave.

-l*log_level* Specify the logging level. It determines the severity of messages sent to **/var/adm/syslog**. The logging levels (and how they relate to system logging levels) are as follows:

0	Log only LOG_CRIT messages.
1	Log only LOG_CRIT and LOG_ERR messages.
2	Log only LOG_CRIT, LOG_ERR, and LOG_WARNING messages.
3	Log all messages.

If this option is omitted, the logging level is set to 1.

-n*node_name* The IP address of the terminal server or the port.

-p*port_no* A DTC port number or, if the **-b** option is omitted, the TCP port service address that will be used by the software to access the port. If the value is omitted, the value **23** (Telnet) is used by default.

In order to shutdown every **ocd** running without restarting them, the following command can be executed:

```
kill -15 ‘ps -e | grep ocd | awk ’{print $1}’‘
```

WARNINGS

In order to ensure that commands (such as **ps**) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, in order to ensure that commands (such as **w**, **passwd**, **finger**, and **wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
stty exta <&1 2>/dev/null
```

The printer interface scripts reside in the directory **/etc/lp/interface**. The line must be added just prior to the final **exit** command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

FILES

/usr/examples/ddfa/dp
/usr/examples/ddfa/pcf
/usr/sbin/dpp
/usr/sbin/ocd
/usr/sbin/ocdebug
/var/adm/dpp_login.bin
/var/adm/utmp.dfa

SEE ALSO

dpp(1M), ocdebug(1M), syslog(3C), dp(4), pcf(4), ddfa(7).

NAME

ocdebug - outbound connection daemon debug utility used by DDFA software

SYNOPSIS

ocdebug -f*pseudonym* **-n***node_name* [**-b***board_no*] [**-c***config_file*] [**-d***debug_level*] [**-l***log_level*] [**-p***port_no*]

DESCRIPTION

The **ocdebug** daemon is the debugging version of the Outbound Connection Daemon (**ocd**). **ocd** is part of the Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software. It manages the connection and data transfer to the remote terminal server port.

See *ddfa*(7) for more information on how to configure the DDFA software and for an explanation of how it works.

Debugging may be toggled interactively by sending the **SIGUSR1** signal to the process using:

kill -16 *pid*.

ocdebug logs important messages and error conditions to **/var/adm/syslog**. Debug messages are logged to the file **/var/adm/ocd***pid* and the file name is displayed at the start of debugging.

Options

ocdebug recognizes the following options. Apart from the **-d** option they are the same as the **ocd** options.

-b*board_no* Specify the board number of a DTC. If it is omitted, the port number option must contain the full TCP service port address. The **-b** and **-p** options must not be used if the IP address given in the **-n** option is the IP address of a port.

If the **-n** option explicitly names a terminal server port, the **-b** option is not needed.

-c*config_file* Specify the name (including the absolute path) of the configuration file used to profile the terminal server port. If this value is omitted, the values specified in the default **pcf** file (**/usr/examples/ddfa/pcf**) are used. If the file specified does not exist, an error message is logged and the following values are used (note that the values for *open_tries* and *open_timer* are different from the default values):

```
telnet_mode:      enable
timing_mark:      enable
telnet_timer:     120
binary_mode:      disable
open_tries:       0
open_timer:       0
close_timer:      0
status_request:   disable
status_timer:     30
eight_bit:        disable
tcp_nodelay:      enable
```

-d*debug_level* Specify the level of debugging. Levels can be added together to accumulate debugging functions. For example, **-d7** enables all levels and **-d3** enables only the first two levels. The levels are:

0	No debug messages.
1	Trace procedure entry/exit logged.
2	Additional tracking messages logged.
4	Data structures dumped.

-f*pseudonym* Specify the absolute or relative path to the device file, which is linked by the software to the reserved **pty**. Applications use the pseudonym and not the dynamically allocated **pty** slave.

-l*log_level* Specify the logging level. It determines the severity of messages sent to **/var/adm/syslog**. The logging levels (and how they relate to system logging levels) are as follows:

0 Log only LOG_CRIT messages.

O

1	Log only LOG_CRIT and LOG_ERR messages.
2	Log only LOG_CRIT, LOG_ERR, and LOG_WARNING messages.
3	Log all messages.

If it is omitted, the logging level is set to 1.

-n*node_name* Specify the IP address of the terminal server or the port.

-p*port_no* Specify a DTC port number or, if the **-b** option is omitted, the TCP port service address that will be used by the software to access the port. If the value is omitted, the value **23** (Telnet) is used by default.

In order to shutdown every **ocd** running without restarting them, the following command can be executed:

```
kill -15 ‘ps -e | grep ocd | awk ’{print $1}’‘
```

WARNINGS

In order to ensure that commands (such as *ps*) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, in order to ensure that commands (such as **w**, **passwd**, **finger**, and **wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
stty exta <&1 2>/dev/null
```

The printer interface scripts reside in the directory **/etc/lp/interface**. The line must be added just prior to the final ’exit’ command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

FILES

/usr/examples/ddfa/dp
/usr/examples/ddfa/pcf
/usr/sbin/dpp
/usr/sbin/ocd
/usr/sbin/ocdebug
/var/adm/dpp_login.bin
/var/adm/ocd*pid*
/var/adm/syslog
/var/adm/utmp.dfa

SEE ALSO

dpp(1M), ocd(1M), syslog(3C), dp(4), pcf(4), ddfa(7).

NAME

opx25 - execute HALGOL programs

SYNOPSIS

/usr/lbin/uucp/X25/opx25 [**-f** *scriptname*] [**-c** *char*] [**-o***file-descriptor*] [**-i***file-descriptor*] [**-n***string*] [**-d**] [**-v**]

DESCRIPTION

HALGOL is a simple language for communicating with devices such as modems and X.25 PADs. It has simple statements similar to **send** *xxx* and **expect** *yyy* that are described below.

Options:

opx25 recognizes the following options:

-f *script*	Causes **opx25** to read script as the input program. If **-f** is not specified, **opx25** reads the standard input as a script.
-c *char*	Causes **opx25** to use *char* as the first character in the input stream instead of actually reading it from the input descriptor. This is useful sometimes when the program that calls **opx25** is forced to read a character but then cannot "unread" it.
-o *number*	Causes **opx25** to use *number* for the output file descriptor (i.e., the device to use for **send**). The default is 1.
-i *number*	Causes **opx25** to use 'number' for the input file descriptor (ie, the device to use for 'expect'). The default is 0.
-n *string*	Causes **opx25** to save this string for use when **\#** is encountered in a **send** command.
-d	Causes **opx25** to turn on debugging mode.
-v	Causes **opx25** to turn on verbose mode.

An **opx25** script file contains lines of the following types:

(empty)	Empty lines are ignored.
/	Lines beginning with a slash (**/**) are ignored (comments)
ID	*ID* denotes a label, and is limited to alphanumerics or _.
send *string*	*string* must be surrounded by double quotes. The text is sent to the device specified by the **-o** option. Non-printable characters are represented as in C; i.e., as \DDD, where DDD is the octal ascii character code. **\#** in a send string is the string that followed the **-n** option.
break	Send a break "character" to the device.
expect *number string*	Here *number* is how many seconds to wait before giving up. 0 means wait forever, but this is not advised. Whenever *string* appears in the input within the time allotted, the command succeeds. Thus, it is not necessary to specify the entire string. For example, if you know that the PAD will send several lines followed by an **@** prompt, you could just use **@** as the string.
run *program args*	The program (**sleep**, **date**, etc.) is run with the args specified. Do not use quotes here. Also, the program is invoked directly (using **execp**), so wild cards, redirection, etc. are not possible.
error *ID*	If the most recent expect or run encountered an error, go to the label *ID*.
exec *program args*	Similar to **run**, but does not fork.
echo *string*	Similar to **send**, but goes to standard error instead of to the device.
set debug	Sets the program in debug mode. It echoes each line to **/tmp/opx25.log**, as well as giving the result of each expect and run. This can be useful for writing new scripts. The command **set nodebug** disables this feature.

set log Sends subsequent incoming characters to **/var/uucp/.Log/LOGX25**. This can be used in the ***.in** file as a security measure, because part of the incoming data stream contains the number of the caller. There is a similar feature in **getx25**; it writes the time and the login name into the same logfile. The command **set nolog** disables this feature.

set numlog Similar to **set log**, but better in some cases because it sends only digits to the log file, and not other characters. The command **set nonumlog** disables this feature.

timeout *number*
Sets a global timeout value. Each expect uses time in the timeout reservoir; when this time is gone, the program gives up (exit 1). If this command is not used, there is no global timeout. Also, the global timeout can be reset any time, and a value of 0 turns it off.

exit *number* Exits with this value. 0 is success; anything else is failure.

To perform a rudimentary test of configuration files, run **opx25** by hand, using the **-f** option followed by the name of the script file. **opx25** then sends to standard output and expects from standard input; thus you can type the input, observe the output, and use the **echo** command to see messages. See the file **/usr/lbin/uucp/X25/ventel.out** for a good example of HALGOL programming.

AUTHOR

opx25 was developed by HP.

SEE ALSO

getx25(1), uucp(1).

NAME

ospf_monitor - monitor OSPF (Open Shortest Path First protocol) gateways

SYNOPSIS

ospf_monitor *mon_db_file*

DESCRIPTION

Use the **ospf_monitor** command to query OSPF routers. The **ospf_monitor** command operates in interactive mode. It allows the user to query the various OSPF routers to provide detailed information on IO statistics, error logs, link-state data bases, AS external data bases, the OSPF routing table, configured OSPF interfaces, and OSPF neighbors.

mon_db_file is the complete pathname of a database composed of records configuring destinations for **ospf_monitor** remote commands. Each destination record is a single-line entry which lists the destination IP address, the destination hostname, and an OSPF authentication key (if authentication is activated by the destination). Since authentication keys may be present in the destination records, it is recommended that general access to this database be restricted.

Refer to RFC-1583 (OSPF Specification, version 2) for details about OSPF database and packet formats.

COMMANDS

Upon entering interactive mode, **ospf_monitor** presents this prompt:

[#] dest command params >

From this prompt, you can enter any of the **ospf_monitor** interactive commands. Interactive commands can be interrupted at any time via a keyboard interrupt. Note that the command line length must be less than 200 characters.

Local Commands

?	Display all local commands and their functions.
?R	Display all remote commands and their functions.
d	Display all configured destinations. This command displays *dest_index*, the IP address, and the hostname of all potential **ospf_monitor** command destinations configured in *mon_db_file*.
h	Display the command history buffer showing the last 30 interactive commands.
x	Exit the **ospf_monitor** program.
@ *remote_command*	Send *remote_command* to the same (previous) destination.
@*dest_index remote_command*	Send *remote_command* to configured destination *dest_index*.
F *filename*	Send all **ospf_monitor** output to *filename*.
S	Send all **ospf_monitor** output to stdout.

O

Remote Commands

a *area_id type ls_id adv_rtr*

Display link state advertisement. *area_id* is the OSPF area for which the query is directed. *adv_rtr* is the router-id of the router which originated this link state advertisement. *type* specifies the type of advertisement to request and should be specified as follows:

1 Request the router links advertisements. They describe the collected states of the router's interfaces. For this type of request, the *ls_id* field should be set to the originating router's Router ID.

2 Request the network links advertisements. They describe the set of routers attached to the network. For this type of request, the *ls_id* field should be set to the IP interface address of the network's Designated Router.

3 Request the summary link advertisements describing routes to networks. They describe inter-area routes, and enable the condensing of routing information at area borders. For this type of request, the *ls_id* field should be set to the destination network's IP address.

4 Request the summary link advertisements describing routes to AS boundary routers. They describe inter-area routes, and enable the condensing of routing information at area borders. For this type of request, the *ls_id* field should be set to the Router ID of the described AS boundary router.

5 Request the AS external link advertisements. They describe routes to destinations external to the Autonomous System. For this type of request, the *ls_id* field should be set to the destination network's IP address.

c Display cumulative log. This log includes input/output statistics for monitor request, hello, data base description, link-state request, link-state update, and link-state ack packets. Area statistics are provided which describe the total number of routing neighbors and number of active OSPF interfaces. Routing table statistics are summarized and reported as the number of intra-area routes, inter-area routes, and AS external data base entries.

e Display cumulative errors. This log reports the various error conditions which can occur between OSPF routing neighbors and shows the number of occurrences for each.

h Display the next hop list. This list of valid next hops is mostly derived from the SPF calculation.

l [*retrans*]
Display the link-state database (except for ASE's). This table describes the routers and networks making up the AS. If *retrans* is non-zero, the retransmit list of neighbors held by this lsdb structure will be printed.

A [*retrans*]
Display the AS external data base entries. This table reports the advertising router, forwarding address, age, length, sequence number, type, and metric for each AS external route. If *retrans* is non-zero, the retransmit list of neighbors held by this lsdb structure will be printed.

o [*which*]
Display the OSPF routing table. This table reports the AS border routes, area border routes, summary AS border routes, networks, summary networks, and AS external networks currently managed via OSPF. If *which* is omitted, all of the above will be listed. If specified, the value of *which* (between 1 and 63) specifies that only certain tables should be displayed. The appropriate value is determined by adding up the values for the desired tables from the following list:

1	Routes to AS border routers in this area.
2	Routes to area border routers for this area.
4	Summary routes to AS border routers in other areas.
8	Routes to networks in this area.
16	Summary routes to networks in other areas.
32	AS routes to non-OSPF networks.

I Display all interfaces. This report shows all interfaces configured for OSPF. Information reported includes the area, interface IP address, interface type, interface state, cost, priority, and the IP address of the DR and BDR for the network.

N Display all OSPF routing neighbors. Information reported includes the area, local interface address, router ID, neighbor IP address, state, and mode.

V Display Gated version information.

AUTHOR

Rob Coltun of University of Maryland

Jeffrey C. Honig of Cornell University

SEE ALSO

gated(1M), gdc(1M), ripquery(1M), gated.conf(4).

GateD Documentation

GateD Configuration Guide

NAME

/usr/sbin/owners - lists owners of outgoing network connections

SYNOPSIS

owners

DESCRIPTION

owners displays a list of established network connections which originate on this system, and indicates the owners of each connection using the **identd** running on this system.

SEE ALSO

sendmail(1M).

NAME

pcnfsd - PC-NFS authentication and print request server

SYNOPSIS

/usr/sbin/rpc.pcnfsd

DESCRIPTION

pcnfsd is an RPC server that supports ONC clients on PC (DOS, OS/2, Macintosh, and other) systems. This describes version two of the **pcnfsd** server.

pcnfsd can be started from the **/sbin/init.d/nfs.server** startup script by setting the **PCNFS_SERVER** variable to 1 in **/etc/rc.config.d/nfsconf**, or from the **inetd** daemon (see *inetd*(1M)). It reads the configuration file **/etc/pcnfsd.conf**, if present, and services RPC requests directed to program number 150001. The **pcnfsd** daemon now supports version 1 and version 2 of the PCNFSD protocol.

The requests serviced by **pcnfsd** fall into three categories: authentication, printing, and other. Only the authentication and printing categories have administrative significance.

Authentication

When **pcnfsd** receives a **PCNFSD_AUTH** or **PCNFSD2_AUTH** request, it will "log in" the user by validating the user name and password, returning the corresponding user ID, group IDs, home directory, and umask. It will also append a record to the **wtmp** data base (see *wtmp*(4)). If you do not want PC "logins" recorded in this way, add a line to the **/etc/pcnfsd.conf** file in the form:

wtmp off

By default, **pcnfsd** will only allow authentication or print requests for users with user IDs in the range 101 to 60002 (this corresponds, in SVR4, to the range for nonsystem accounts). To override this, add a line to the **/etc/pcnfsd.conf** file in the form:

uidrange *range* [**,** *range*]...

where each *range* is a user ID number in the form

uid

or an inclusive range of user ID numbers in the form

*uid***-***uid*

NOTE: pcnfsd will deny authentication if the **/etc/shells** file is incorrectly setup.

Printing

pcnfsd supports a printing model that uses NFS to transfer print data from the client to the server. The client system issues a **PCNFSD_PR_INIT** or **PCNFSD2_PR_INIT** request, and the server returns the path to a spool directory that is exported by NFS for use by the client. **pcnfsd** creates a subdirectory for each client. By default, the parent directory is **/var/spool/pcnfs**, and the name of each subdirectory is the same as its client's host name. To use a different parent directory, add a line to the **/etc/pcnfsd.conf** file in the form:

spooldir *path*

Once a client has mounted the spool directory using NFS, and transferred print data to a file in that directory, it will issue a **PCNFSD_PR_START** or **PCNFSD2_PR_START** request. **pcnfsd** handles most print-related requests by constructing a command based on the printing services of the server's operating system, and executing that command using the identity of the PC user. Because this involves set-user-ID privileges, **pcnfsd** must be run as **root**.

Every print request from a client includes the name of the printer to be used. This name corresponds to a printer that has been configured into the line printer spooling system using the **lpadmin** command.

To process print data in a special way (for example, to print it in landscape mode, or to print it in duplex mode), define a new printer and arrange for the client to print to that printer. There are two ways to define the new printer:

- You can add a new printer to the line printer spooling system that uses a different printer model script, and arrange for the client to use the new printer. Do this using the **lpadmin** command (see *lpadmin*(1m)).

p

- **pcnfsd** includes a mechanism to define virtual printers known only to **pcnfsd** clients. Each of these printers is defined by an entry in the file **/etc/pcnfsd.conf** using the following format:

  printer *name alias-for command*

  with the following values:

name	The name of the printer, as it will be referred to in print requests from clients.
alias-for	The corresponding name for the printer, as it is defined in the line printer spooling system. For example, a request to display the queue for *name* will be translated into the corresponding request for the printer *alias-for*. If you have defined a printer within **pcnfsd** that has no corresponding printer defined in the line printer spooling system, use a single hyphen (**-**) for this field. For an example, see the definition of the printer **test** in the examples section, below.
command	A command that will be executed whenever a file is printed on *name*. This command is executed by the POSIX shell, **/usr/bin/sh** using the **-c** option. For complex operations, construct an executable shell program and execute that in *command*.

Within *command* the following tokens will be replaced:

Token	Substitution
$FILE	Replaced by the full path name of the print data file. When the command has been executed, the file will be unlinked.
$USER	Replaced by the user name of the user logged in to the client system.
$HOST	Replaced by the host name of the client system.

Reconfiguration

By checking the modification time (and contents) of the file **/var/spool/lp/pstatus**, **pcnfsd** will detect when printers have been added or deleted, and will rebuild its list of valid printers. However, **pcnfsd** does not monitor the file **/etc/pcnfsd.conf** for updates; if you change this file, you must kill and restart **pcnfsd** for the changes to take effect.

EXAMPLES

Given the following entries for the file **/etc/pcnfsd.conf**:

```
printer abc lj lp -dlj -oraw
printer test - /usr/bin/cp $FILE /usr/tmp/$HOST-$USER
```

If a user on a client system prints a job on printer **abc**, the request will be sent to destination **lj** in raw mode.

If the client requests a list of the print queue for printer **abc**, the **pcnfsd** daemon will translate this into a request for a listing for printer **lj**.

Printer **test** is used only for testing. Any file sent to this printer will be copied into the directory **/usr/tmp**. Any request to list the queue, check the status, etc., of printer **test** will be rejected because *alias-for* has been specified as a hyphen (**-**).

FILES

/etc/pcnfsd.conf
/etc/rc.config.d/nfsconf
/var/spool/lp/pstatus
/var/spool/pcnfs
/etc/shells

SEE ALSO

lp(1), lpstat(1), inetd(1M), lpadmin(1M), wtmp(4).

NAME

pcserver - Basic Serial and HP AdvanceLink server

SYNOPSIS

pcserver [**-n**] [**-l** [*log_file*]] [**-v**]

DESCRIPTION

pcserver is the hostside server program for Basic Serial and AdvanceLink, and is started and terminated by an application program running on a PC.

pcserver supports both the Basic Serial and the AdvanceLink protocols.

Basic Serial offers a library of routines that support a variety of services between a PC and a serially connected host computer, including file transfers and remote interprocess communications.

AdvanceLink is a terminal emulation program that also supports file transfers between a PC and host system over various physical connections.

Options

The following options are recognized by *pcserver*:

-l [*logfile*] This option is now obsolete, but is retained for compatibility with earlier versions of software. Logging is now controlled by the presence or absence of the server.pro file as described in NOTES, below. Enables packet logging and records **pcserver** messages to a specified log file (for debugging). If *logfile* is not specified, the file **s-log** is used in the default logging directory, as defined in the **server.pro** file. **pcserver** looks for a local version of server.pro in the user's home directory. If none is found, it will look for a system-wide version as **/var/adm/server.pro** or **/usr/adm/server.pro**. If the **logfile** exists, logging is appended to it. If the file does not exist, logging is disabled.

-n Informs **pcserver** that a "netmode" for data encryption should be used during special operations (for example, a netmode is needed to mask device control characters when a PAD is being used). The details of the netmode are then negotiated between the **pcserver** and the PC application. For a more comprehensive discussion on netmode, see *Using Basic Serial Connection Files*.

-v Causes **pcserver** to print its version number to standard output and quit.

pcserver is designed to be invoked by a PC application program rather than from the command line. In order for the connection to be correctly established, the PC and host port must be properly configured.

If you are using **pcserver** to manage a session between a PC and a hostside application (via Basic Serial), you may need to use a Basic Serial connection file to actually log in to your account. Establishing connections using Basic Serial connection files is a sensitive operation. Before attempting to use them, you should read the manual *Using Basic Serial Connection Files*.

If you are using **pcserver** to transfer files between a PC and a host machine via Advancelink, use the following AdvanceLink commands:

```
&HOSTCOPY "pcserver"
&TERMINATOR "$"
```

If your prompt does not end with **$**, replace the **$** in the terminator command with the last character in your normal prompt.

To permanently configure AdvanceLink for the HP-UX version of **pcserver**, refer to the *Using AdvanceLink* manual for more information.

NOTES

Packet logging is controlled by the presence or absence of the file **server.pro**

pcserver looks for a local version of **server.pro** in the user's home directory. If none is found, it will look for a system-wide version as **/var/adm/server.pro** or **/usr/adm/server.pro**.

If no logging file is found in these directories, logging is not performed. A commented example of a **server.pro** may be found in **/usr/newconfig/var/adm/server.pro.ex** or **/usr/adm/server.pro.ex**. To make use of this file, copy it to the active file name, **server.pro**, in one of the previously mentioned directory locations.

p

If your screen displays a **`Command not found`** message when you choose START TRANSFER from AdvLink, either **`pcserver`** has not yet been installed on your HP-UX system, or it has been installed in a directory that is not part of your current path.

HP-UX treats files containing binary or ASCII data identically. Therefore it is up to the user to specify the desired file type when using **`pcserver`** to transfer files with Advancelink. The difference between the two is that during ASCII transfers, **`pcserver`** maps HP-UX line-feed characters to the MS-DOS carriage-return/line-feed pair. This produces incorrect results when transferring a binary file as an ASCII file.

Also, older versions of AdvanceLink show totally inaccurate estimates for file transfer times. This does not interfere with the actual transfer.

If the PC is reset while a transfer is taking place, it may temporarily appear to be a "dead" terminal port. This is no cause for alarm; left to its own devices, **`pcserver`** will restore the port in a short time. In the worst case, it could take six timeout periods (6×20 = 120 seconds). For faster response, press the Break key a few times to terminate **`pcserver`** immediately.

FILES

`/usr/bin/pcserver`	the executable program
`/var/adm/server.pro`	system-wide logging profile
`/usr/adm/server.pro`	system-wide logging profile
`$HOME/server.pro`	local logging profile
`/usr/newconfig/var/adm/server.pro.ex`	commented inactive example of server.pro
`/usr/adm/server.pro.ex`	commented inactive example of server.pro

SEE ALSO

Using AdvanceLink	Describes protocol and how to use AdvanceLink.
Using Basic Serial Connection Files	Describes Basic Serial and how connection files should be used.

p

NAME

pdc - processor-dependent code (firmware)

DESCRIPTION

pdc is the firmware that implements all processor-dependent functionality, including initialization and self-test of the processor. Upon completion, it loads and transfers control to the initial system loader (*isl*(1M)). Firmware behavior varies somewhat, depending on the hardware series as described below.

Series 800 Behavior

To load *isl* from an external medium, *pdc* must know the particular device on which *isl* resides. Typically the device is identified by the Primary Boot Path that is maintained by *pdc* in Stable Storage. A *path* specification is a series of decimal numbers each suffixed by '/', indicating bus converters, followed by a series of decimal numbers separated by '.', indicating the various card and slot numbers and addresses. The first number, not specifying a bus converter, is the MID-BUS module number (that is, slot number times four) and followed by the CIO slot number. If the CIO slot contains an HP-IB card, the next number is the HP-IB address, followed by the unit number of the device if the device supports units. If the CIO slot contains a terminal card, the next number is the port number, which must be zero for the console.

When the processor is reset after initialization and self-test complete, *pdc* reads the Console Path from Stable Storage, and attempts to initialize the console device. If the initialization fails, *pdc* attempts to find and initialize a console device. Algorithms used to find a console device are model-dependent. *pdc* then announces the Primary Boot, Alternate Boot, and Console Paths.

If *autoboot* (see *isl*(1M)) is enabled, *pdc* provides a 10-second delay, during which time the operator can override the *autoboot* sequence by typing any character on the console. If the operator does not interrupt this process, *pdc* initializes and reads *isl* from the Primary Boot Path. On models that support autosearch, if this path is not valid and *autosearch* (see *isl*(1M)) is enabled, *pdc* then searches through the MID-BUS modules and CIO slots to find a bootable medium. Currently, autosearch is only implemented on the model 825.

If the *autoboot* sequence is unsuccessful, overridden by the operator, or not enabled in the first place, *pdc* interactively prompts the operator for the Boot Path to use. Any required path components that are not supplied default to zero.

The Primary Boot, Alternate Boot, and Console Paths as well as *autoboot* and *autosearch* enable can be modified via *isl*.

Series 700 Behavior

To load *isl* from an external medium, *pdc* must know the particular device on which *isl* resides. Typically the device is identified by the Primary Boot Path that is maintained by *pdc* in Stable Storage. A *path* specification is an I/O subsystem mnemonic that varies according to hardware model.

When the processor is reset after initialization and self-test complete, *pdc* reads the Console Path from Stable Storage, and attempts to initialize the console device. If the initialization fails, *pdc* attempts to find and initialize a console device. Algorithms used to find a console device vary according to hardware model.

If *autoboot* and *autosearch* (see *isl*(1M)) are enabled, *pdc* waits for approximately 10 seconds during which time the operator can override the *autoboot* sequence pressing and holding the ESC (escape) key on the console.

The system then begins a search for potentially bootable devices. If allowed to complete, a list of potentially bootable devices is displayed, labeled with abbreviated path identifiers (P0, P1, etc). A simple menu is then displayed where the user can:

- Boot a specific device, using the abbreviated path identifier, or the full mnenomic.
- Start a device search where the contents are searched for IPL images (note the first search only identified devices and did not check the contents).
- Enter the boot administration level.
- Exit the menu and return to autobooting
- Get help on choices

The search of potentially bootable devices can be aborted by pressing and holding the escape key. The search for device contents can also be aborted by pressing and holding the escape key.

If the operator does not interrupt the search process, *pdc* initializes and reads *isl* from the Primary Boot Path.

If the *autoboot* sequence is unsuccessful, overridden by the operator, or not enabled in the first place, *pdc* executes the device search and enters the menu described above.

The Primary Boot, Alternate Boot, and Console Paths as well as *autoboot* and *autosearch* enable can be modified via *isl* or at the pdc boot administration level.

SEE ALSO

boot(1M) isl(1M).

NAME

pddcesetup - configure DCE for the HPDPS

SYNOPSIS

pddcesetup [**force**]

DESCRIPTION

The **pddcesetup** command is used to configure DCE information for the HP Distributed Print Service (HPDPS).

pddcesetup must be run on each HP-UX host in your DCE cell that will execute the HPDPS in the Extended Environment. If a host will run the HPDPS in the Basic Environment only, not in the Extended Environment, then execution of **pddcesetup** is not needed on that host. If you do not intend to execute the HPDPS in a DCE cell, or do not wish to use DCE services in conjunction with the HPDPS, then each host must execute the HPDPS in the Basic Environment, and execution of **pddcesetup** is not needed on any of the hosts in your network. **pddcesetup** must be executed once on each host in your DCE cell that will execute an HPDPS client daemon, spooler, or supervisor in the Extended Environment. **pddcesetup** must be executed before starting any of these HPDPS components on that host.

The first time that **pddcesetup** executes in your DCE cell, it will prompt for and create various DCE security identities and CDS namespace entries that are then configured for the entire cell. On subsequent executions of **pddcesetup**, only local information needed by the local host is configured; the DCE security identities and CDS namespace entries have already been created for the cell.

If the parameter **force** is given, then **pddcesetup** will give you the option to fully recreate security identities and CDS directories. This option is useful if DCE has been only partially configured, or if configuration is accidentally removed after creation. If the **force** parameter is not given, then **pddcesetup** quickly checks to see if it appears the DCE information has already been configured, and if so, **pddcesetup** does not attempt to configure any of this information.

The host on which **pddcesetup** is executed must already be configured as a DCE client or server, using DCE command-line utilities or the SAM program. You must also be DCE logged in using an account with sufficient administrator privileges to modify DCE security and namespace information. The DCE login of the cell administrator for your cell is normally used for this purpose (the default DCE login name is **cell_admin**).

The HPDPS DCE information created by **pddcesetup** is:

- Accounts adm_user and pd_server.
- Principals adm_user and pd_server.
- Groups pd_admin and pd_operator.
- CDS namespace directories and links.
- Initial Access Control List entries.
- Local key table entries for principal pd_server.

EXAMPLES

An example execution follows. This example illustrates the execution of pddcesetup for the first time in a DCE cell.

```
# pddcesetup

Checking whether your host is configured in a DCE cell.

Verifying your DCE login.
You are DCE logged in as <your login>.

Checking whether HPDPS security identities have already been configured
in your cell.

DCE security identities needed for the HPDPS have not yet been
configured in your DCE cell.  The security identities that must be
configured are:

 Accounts   adm_user and pd_server
 Principals adm_user and pd_server
 Groups     pd_admin and pd_operator
```

```
Are you ready to configure these identities now (y/n)? y

The new groups and accounts about to be created must be members of a DCE
"organization".  You may use an existing organization if you have
already defined one.

Do you wish to create a new DCE organization (y/n)? y
Please enter the name of the new organization: <organization name>

Creating organization <organization name>.

Creating group pd_admin.

Creating group pd_operator.

Creating principal pd_server.

Creating principal adm_user.

Adding new principals to groups and organizations.

pddcesetup is ready to create DCE accounts pd_server and adm_user.
To accomplish this, you must enter the password for the DCE account under
which you are currently logged in.  If not entered correctly, an attempt
to create the new accounts will generate the error message "data
integrity error".

Please enter the password for your current DCE login account: <password>

Please choose a unique password for new account adm_user.
This account will be used by HPDPS administrators.

Please enter the password for account adm_user: <password>
Please re-enter the password for account adm_user: <password>

Creating account adm_user.

Please choose a unique password for the pd_server account.
This password is used by the HPDPS client daemon, spooler, and
supervisor to automatically DCE login to account pd_server.

Please enter the password for account pd_server: <password>
Please re-enter the password for account pd_server: <password>

Creating account pd_server.

Creating HPDPS directories and links in the DCE CDS namespace.

Creating initial HPDPS Access Control List entries.

Adding entry for pd_server to the local key table.

pddcesetup: DCE setup is complete.
```

SEE ALSO

dce_config(1M), dce_login(1M)
HP Distributed Print Service Administration Guide

NAME

pdfck - compare Product Description File to File System

SYNOPSIS

pdfck [**-n**] [**-r** *alternate_root*] *PDF*

DESCRIPTION

pdfck is a program that compares the file descriptions in a PDF (Product Description File) to the actual files on the file system. It is intended as a tool to audit the file system and detect corruption and/or tampering. Differences found are reported in the format described in the *pdfdiff*(1M) manual entry. (Size growth (**-p** option) is not reported.) For a detailed explanation of the PDF fields see *pdf*(4). The command

```
pdfck -r /pseudoroot /system/AL_CORE/pdf
```

is roughly equivalent to

```
mkpdf -r /pseudoroot /system/AL_CORE/pdf - | \
pdfdiff /system/AL_CORE/pdf -
```

Options

pdfck recognizes the following options:

-n Compare numerical representation of user id *uid* and group id *gid* of each file, instead of the usual text representation. If owner or group is recorded in the PDF as a name, look the name up in the **/etc/passwd** or **/etc/group** file, respectively, to find the id number.

-r *alternate_root* *alternate_root* is a string that is prefixed to each pathname in the prototype when the filesystem is being searched for that file. Default is NULL.

EXAMPLES

The following output indicates tampering with **/usr/bin/cat**:

```
/usr/bin/cat: mode(-r-xr-xr-x -> -r-sr-xr-x)(became suid), size(27724 -> 10345),
      checksum(1665 -> 398)
```

WARNING

Use of PDFs is discouraged since this functionality is obsolete and is being replaced with Software Distributor (see *sd*(4)).

FILES

/system/*fileset_name***/pdf** Product Description File of fileset called *fileset_name*.

SEE ALSO

mkpdf(1M), pdfdiff(1M), pdf(4).

NAME

pdfdiff - compare two Product Description Files

SYNOPSIS

pdfdiff [**-n**] [**-p** *percent*] *pdf1 pdf2*

DESCRIPTION

pdfdiff is a program that compares two PDFs (Product Description Files). The PDFs can be generated using the **mkpdf** command (see *mkpdf*(1M)). Individual fields in the PDFs are compared, and differences found in these fields are reported. For a detailed explanation of the PDF fields see *pdf*(4).

The report format is:

pathname: *diff_field*[(*details*)][,...]

diff_field is one of the field names specified in *pdf*(4). The format of *details* is "*oldvalue* –> *newvalue*" and may include an additional "(*added description*)".

A summary of total product growth in bytes, **DEV_BSIZE** disk blocks, and the percentage change in disk blocks is reported. This summary includes growth of all files, including those for which growth did *not* exceed the threshhold *percent*. Format of the growth summary is:

Growth: *x* bytes, *y* blocks (*z*%)

Options

pdfdiff recognizes the following options:

-n Compare numerical representation of user ID *uid* and group ID *gid* of each file, instead of the usual text representation. If owner or group is recorded in the PDF as a name, look the name up in the **/etc/passwd** or **/etc/group** file, respectively, to find the ID number.

-p *percent* specifies a threshhold percentage for file growth. Files having a net size change greater than or equal to this percentage are reported. A decrease in size is reported as a negative number. If **-p** is not specified, a default value of zero percent is used.

EXAMPLES

The following output results when the **/usr/bin/cat** entry in the example from *pdf(4)* is different in the compared PDF:

```
/usr/bin/cat: mode(-r-xr-xr-x -> -r-sr-xr-x)(became suid), size(27724 -> 10345),
      checksum(1665 -> 398)
Growth: -17379 bytes, -17 blocks (-4%)
```

WARNING

Use of PDFs is discouraged since this functionality is obsolete and is being replaced with Software Distributor (see *sd*(4)).

FILES

/system/*fileset_name***/pdf**

SEE ALSO

mkpdf(1M), pdfck(1M), pdf(4).

NAME

pdgwcfg - configures HPDPS gateway printers in a Basic environment

SYNOPSIS

pdgwcfg [-a|-m] [-h] [-p] [-v]

DESCRIPTION

The **pdgwcfg** utility simplifies the configuration of HPDPS gateway printers in a Basic (non-DCE Extended) environment by reading an administrator-supplied configuration file **/etc/pdgwcfg.conf** (see *pdgwcfg.conf*(4)). It creates-enables and/or deletes gateway printers as appropriate. Gateway printers are similar to "remote printers" provided by the LP spooler, allowing access to a printer in a foreign (DCE Extended or Basic) environment.

You must have super-user privileges to invoke the utility. The default behavior of the utility will not modify any previously-created gateway printers listed in **/etc/pdgwcfg.conf**. Any new entries will be created and any gateway printers not listed will be deleted.

All output is sent to **$PDBASE/pdgwcfg/error.log** (or **/var/opt/pd/pdgwcfg/error.log** by default). Previous error logs are retained in separate files in this directory for reference and, if used extensively, the directory may need to be cleaned-up periodically. If a severe error is encountered that causes a premature abort, an error message is also sent to stderr.

Options

pdgwcfg uses the following options:

-a Retain all previously-created gateway printers, even if they are no longer listed in **/etc/pdgwcfg.conf**. No gateway printers will be deleted. The administrator must manually remove any unwanted gateway printers.

-m Retain any manually-created gateway printers. These would not contain the text **PDGWCFG-MARKER** in the *descriptor* attribute which is placed there by the **pdgwcfg** utility when it creates a gateway printer. This option is intended for scenarios where **/etc/pdgwcfg.conf** is not used exclusively for gateway printer configuration (e.g. a local sysadmin also creates gateway printers without the utility).

-h Provides invocation syntax help.

-p Preview mode. No changes to the gateway configuration will actually be made. For best results, the local HPDPS system should be running so that the utility can query the system to determine which gateway printers, if any, would be created/deleted.

-v Verbose mode. Provides more extensive output in the **error.log**.

RETURN VALUE

pdgwcfg exits with one of the following values:

0	Successful completion.
1	Failure.

EXTERNAL INFLUENCES

Environment Variables

PATH needs to include at a minimum **/usr/bin:/opt/pd/bin**.

PDBASE affects the location of the **error.log**

EXAMPLES

If it is desirable to have a single configuration file distributed across multiple systems, there are various ways to distribute the configuration file and have the utility invoked to configure a system (e.g. *swinstall*(1M), *rdist*(1), etc.). Each method should be weighed against the administration and security concerns of your particular environment.

The below is just an example using *rdist*(1) and is not intended to be a recommendation.

```
# sample distfile for use with rdist
# copies /etc/pdgwcfg.conf and invokes the pdgwcfg utility
# invoke as 'rdist -b -h -f distfile'
HOSTS = ( host7 host8 )
```

```
FILES = ( /etc/pdgwcfg.conf )
${FILES} -> ${HOSTS}
        install ;
        special /etc/pdgwcfg.conf \
     " PATH=/usr/bin:/opt/pd/bin;pdgwcfg" ;
```

To update the configuration on host8 only, one would invoke:

```
rdist -b -h -f distfile -m host8
```

WARNINGS

By default, any gateway printer not listed in **/etc/pdgwcfg.conf** will be removed. **pdgwcfg** does not check for entry modifications in **/etc/pdgwcfg.conf**. See *pdgwcfg.conf*(4) for possible ways to accomplish modifications.

If the descriptor attribute is modified, then the **-m** option will not consider these gateway printers as candidates for deletion because it overrides the **PDGWCFG-MARKER** marker that **pdgwcfg** would have placed in that attribute.

AUTHOR

pdgwcfg was developed by HP.

SEE ALSO

pdgwcfg.conf(4), pdcreate(1), and the *HP Distributed Print Service Administration Guide* (re: gateway printers)

NAME
pdstartclient - start the HPDPS client daemon

SYNOPSIS
pdstartclient [**-l** *locale*] [**-p** *port*] [**-q**]

DESCRIPTION
The **pdstartclient** utility is issued by an administrator to start the HPDPS client daemon.

Options
The **pdstartclient** utility uses the following flag:

-l *locale* Allows you to specify the locale for HPDPS messages in a specific language.

-p *port* Allows you to specify the port number when starting a HPDPS client in a locale other than the default locale. The port number you assign must not conflict with port numbers in use by other processes. The file **/etc/services** lists the port numbers reserved by other processes.

-q Allows you to query the status whether the daemon is running or not running without starting the daemon.

EXAMPLES
Start a Daemon
To start the daemon, enter the following:

```
pdstartclient
```

Start a Daemon in a Different Locale
To start the daemon in a Japanese locale and assign port number 1411, enter the following:

```
pdstartclient -l ja_JP.SJIS -p 1411
```

Query the Status of a Daemon
To query the status of a daemon, enter the following:

```
pdstartclient -q; echo $?
```

If the daemon is running, you will receive the following message:

```
The HPDPS daemon is already running
0
```

If the daemon is not running, you will receive a **1**.

To query the status of a daemon running in locale **ja_JP.SJIS**, enter the following:

```
pdstartclient -q -l ja_JP.SJIS
```

To query the status of a daemon running on port 1411, enter the following:

```
pdstartclient -q -p 1411
```

SEE ALSO
pdstopd(1M), pdstartspl(1M), pdstartsuv(1M).

NAME

pdstartspl - create or restart an HPDPS spooler

SYNOPSIS

pdstartspl [**-F**] *ServerName*

DESCRIPTION

The **pdstartspl** utility is issued by an administrator to create or restart a spooler. A spooler represents the server that manages the validation, routing, and scheduling of jobs. A spooler contains logical printers and queues.

You can restart a spooler after it is terminated by issuing this same utility.

When you create or restart a spooler, you must specify its name.

Options

The **pdstartspl** utility uses the following flag:

-F Bypasses (does not display) prompts; force creation of a new spooler

Arguments

The argument value identifies the specific object to which the utility applies.

The valid argument value for the **pdstartspl** utility is:

ServerName
Assigns a name to a new spooler or specifies the name of the spooler to restart.

EXAMPLES

Create or Restart a Spooler

To create or restart a spooler, **spool1**, enter the command:

```
pdstartspl spool1
```

SEE ALSO

pdstartclient(1M), pdstartsuv(1M), pdshutdown(1)

p

NAME

pdstartsuv - create or restart an HPDPS supervisor

SYNOPSIS

pdstartsuv [**-F**] *ServerName*

DESCRIPTION

The **pdstartsuv** utility is issued by an administrator to create or restart a supervisor. A supervisor receives jobs from a spooler and manages the printing process. A single supervisor may contain many physical printers. The supervisor may be started on a different HP-UX processor from the spooler, but it must be able to communicate with its printer devices.

If the supervisor already exists but is shut down, the **pdstartsuv** utility restarts it.

Options

The **pdstartsuv** utility uses the following flag:

-F Bypasses (does not display) prompts; force creation of a new supervisor.

Arguments

The argument value identifies the specific object to which the utility applies.

The valid argument value for the **pdstartsuv** utility is:

ServerName
Assigns a name to a new supervisor or specifies the name of the supervisor to restart.

EXAMPLES

Create or Restart a Supervisor

To create or restart a supervisor, **super1**, enter the command:

```
pdstartsuv super1
```

SEE ALSO

pdstartclient(1M), pdstartspl(1M), pdshutdown(1).

NAME

pdstopd - stop the HPDPS client daemon

SYNOPSIS

pdstopd

DESCRIPTION

The **pdstopd** utility is issued by an administrator to stop the HPDPS client daemon.

EXAMPLES

Stopping a Daemon

To stop the daemon, enter the command:

```
pdstopd
```

To stop the daemon running in a specific locale, such as **ja_JP.SJIS** enter the following:

```
export LC_ALL=ja_JP.SJIS
pdstopd
export LC_ALL=
```

SEE ALSO

pdstartclient(1M), pdstartspl(1M), pdstartsuv(1M).

NAME

pfs_exportfs - export and unexport directories to PFS clients

SYNOPSIS

/usr/sbin/pfs_exportfs [**-a -u -v**] [*pathname*]

DESCRIPTION

pfs_exportfs makes a local directory or filename available for mounting over the network by PFS clients. It is recommended that a command to invoke **pfs_exportfs** at boot time be added to *rc*(1M). **pfs_exportfs** uses information contained in the **/etc/pfs_exports** file to export *pathname* (which must be specified as a full pathname). The superuser can run **pfs_exportfs** at any time to alter the list or characteristics of exported directories and filenames. Directories and files that are currently exported are listed in the file **/etc/pfs_xtab**.

With no options or arguments, **pfs_exportfs** prints out the list of directories and filenames currently exported.

Options

-a All. Export all pathnames listed in **/etc/pfs_exports**, or if **-u** is specified, unexport all of the currently exported pathnames.

-u Unexport the indicated pathnames.

-v Verbose. Print each directory or filename as it is exported or unexported.

AUTHOR

psf_exportfs was developed by Young Minds, Inc.

FILES

/etc/pfs_exports	static export information
/etc/pfs_xtab	current state of exported pathnames

SEE ALSO

pfs_exports(5).

NAME

pfs_mount, pfs_umount - mount and unmount CD-ROM file systems

SYNOPSIS

pfs_mount

pfs_mount [**-v -f -n**] [**-t** *type*] [**-x** *xlat*] [**-o** *options*] *filesystem directory*

pfs_mount [**-v -f -n**] [**-x** *xlat*] [**-o** *options*] *filesystem* | *directory*

pfs_umount [**-v**] *filesystem* | *directory*

DESCRIPTION

pfs_mount attaches a named *filesystem* to the file system hierarchy at the pathname location *directory*, which must already exist. If *directory* has any contents prior to the **mount** operation, these remain hidden until the *filesystem* is once again unmounted. If *filesystem* is of the form *host*:*pathname*, it is assumed to be a remote file system.

In the case of a local mount, **pfs_mount** probes the specified device to determine the file system type. It then contacts the local mountd to register the specified *directory* as a valid mounted file system. **pfs_mountd.rpc** will reply with the address of the **pfsd.rpc** who will be handling all requests for files on that *directory*.

Remote mounts are very similar, except that both the local and remote mount daemons will be contacted. The remote mount daemon will supply the pfs server address, and the local mount daemon will be contacted to register the mount.

pfs_umount unmounts a currently mounted file system, which can be specified as a either *directory* or a *filesystem* .

pfs_umount contacts the local mount daemon to determine what actions should be taken to perform the unmount. If the file system was originally remotely mounted, the remote mount daemon is informed of the unmount, and the file system is unmounted. Otherwise, it is simply unmounted.

pfs_mount and **pfs_umount** maintain a table of mounted file systems in **/etc/mtab**, described in *pfs_fstab*(5). If invoked without an argument, pfs_mount displays the contents of this table. If invoked with either a *file*system or a *directory* only, mount searches the file /etc/pfs_fstab for a matching entry, and mounts the file system indicated in that entry on the indicated directory.

pfs_mount Options

p

-v Verbose. Display a message indicating each file system being mounted.

-f Fake an /etc/mtab entry, but do not actually mount any file systems.

-n Mount the file system without making an entry in /etc/mtab.

-x *xlat* Filename translation options. Any combination can be specified, although some combinations do not make sense (i.e. dot_version and no_version).

- **no_version** will suppress the printing of the version number (and semicolon) at the end of iso9660 and high sierra filenames.
- **dot_version** replaces the version number (and semicolon) with a period followed by the version number.
- **lower_case** Converts upper to lower case on all file (and directory) names. version number.
- **unix** Shorthand for no_version and lower_case

-t *type* Force the CD-ROM to be mounted as the specified type, if possible. Accepted types are:

- **iso9660** will cause the mount program to attempt to mount the CD-ROM image using the iso9660 specifications. If the CD image is not iso9660 compatible, the mount fails. Note that if the CD image is also RockRidge compliant, and the -t iso9660 option is not specified, the CD-ROM image will be mounted with Rock-Ridge extensions enabled.
- **hsfs** will cause the mount program to attempt to mount the CD-ROM image using the high sierra specifications. If the CD image is not hsfs compatible, the mount fails.

rrip will cause the mount program to attempt to mount the CD-ROM image using the RockRidge Interchange specifications. If the CD image is not rrip compatible, the mount fails. Note, that if the CR-ROM image is supports the Rock-Ridge Interchange Protocol, and the CR-ROM image is mounted with rrip, the translation options are suppressed.

Note that these get entered into the **/etc/mtab** and **/etc/pfs_fstab** with a **pfs-** preceding the type. This is to avoid confusing other programs which may scan the **/etc/mtab** looking for types of the same name.

-o *options* Specify file system *options* as a list of comma-separated words from the list below.

options valid on *all* file systems:

ro	Even if not specified, the read-only option is implied.
suid | **nosuid**	Setuid execution allowed or disallowed.
bg | **fg**	If the first attempt fails, retry in the background, or, in the foreground.
retry=*n*	The number of times to retry the mount operation.
rsize=*n*	Set the read buffer size to *n* bytes.
timeo=*n*	Set the PFS timeout to *n* tenths of a second.
retrans=*n*	The number of PFS retransmissions.
soft | **hard**	Return an error if the server does not respond, or continue the retry request until the server responds.
intr	Allow keyboard interrupts on hard mounts.

The defaults are:

```
suid,fg,retry=10000,timeo=7,rsize=2048,retrans=3,hard,\
acsize=1037,bcsize=100,lcsize=500
```

options specific to **iso9660** and **hsfs** file systems:

xlat=xlat_flags
xlat_flags is a colon (:) separated list of translation options. Currently supported are no_version, dot_version, lower_case, and unix. They allow you to perform the same translations options the *-x* flag does. The *-x* flag remains for backward compatibility. It is suggested that you use the *xlat*= option flag as they can be placed in the *pfs_fstab* file.

p

pfs_umount Options

-v Verbose. Display a message indicating each file system as it is unmounted.

Background vs. Foreground

Filesystems mounted with the **bg** option indicate that **mount** is to retry in the background if the server's mount daemon (*pfs_mountd*(1M)) does not respond. **mount** retries the request up to the count specified in the **retry=***n* option. Once the file system is mounted, each PFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=***n* option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Interrupting Processes With Pending PFS Requests

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system.

Attributes Cache

The server's attribute cache retains file attribute information on requests that have been made. This provides faster access to entries which have previously been decoded.

Lookup Cache

The **Lookup**Cache holds information about the sequential nature of the directory entries. This cache stores the location of the next directory entry. When a request comes in for a directory entry, if the preceding directory entry had been accessed earlier, this location is examined first to see if the directory entry being requested matches the directory entry at that location.

Block Cache

This cache holds raw 8k blocks of recently accessed data.

EXAMPLES

To mount a CD-ROM disk:

```
pfs_mount /dev/sr0 /cd-rom
```

To mount a remote file system:

```
pfs_mount serv:/cd-rom /cd-rom
```

To fake an entry for iso9660 on /cd-rom:

```
pfs_mount -f -t iso9660 /dev/sr0 /cd-rom
```

To hard mount a remote file system:

```
pfs_mount -o hard serv:/cd-rom /cd-rom
```

AUTHOR

pfs_mount was developed by Young Minds, Inc.

FILES

/etc/mtab	table of mounted file systems
/etc/pfs_fstab	table of pfs file systems

SEE ALSO

fstab(5), mtab(5), pfs_fstab(5), pfs_mountd(1M), pfsd(1M).

BUGS

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers*, rather than being mounted on top of the symbolic link itself.

On Pioneer six disc changers (and perhaps other drives) if you mount the file system using the block device driver (**/dev/sr0** for Sun), the Pioneer returns information to the driver indicating there is no data, causing the mount to fail. Either mount the file system again (which will should succeed), or use the raw device driver (**/dev/rsr0** for Sun).

NAME

pfs_mountd, pfs_mountd.rpc - PFS mount request server

SYNOPSIS

/usr/etc/pfs_mountd

DESCRIPTION

This program is available with the *Portable File System Package (PFS)*. **pfs_mountd** is an RPC server that answers file system mount requests. In the case of remote mount requests, it reads the file **/etc/pfs_xtab**, described in *pfs_exports*(5), to determine which file systems are available for mounting by which machines.

It is recommended that the **pfs_mountd** daemon be invoked by *rc*(1M). It must be invoked in the background.

The **pfs** mount daemon is composed of two programs: **pfs_mountd** and **pfs_mountd.rpc**. The **pfs_mountd.rpc** program should **not** be run directly. It is invoked by the **pfs_mountd** program.

The mount daemon assigns servers to mounted file systems in a round-robin fashion. For example, if there are four pfs daemons, and four **pfs_mount**'s are performed, each daemon will be serving a different mount.

Options

-v Verbose. Show version number, etc.

AUTHOR

pfs_mountd was developed by Young Minds, Inc.

FILES

/etc/pfs_xtab

SEE ALSO

pfs_exports(5), rc(1M).

NAME

pfsd, pfsd.rpc - PFS daemon

SYNOPSIS

pfsd [*nservers*] [**-v**] [**-o** *options*]

DESCRIPTION

pfsd starts the daemons that handle client filesystem requests. *nservers* is the number of file system server daemons to start. This number should be based on the load expected on this server. The load is defined by the number of mounted file systems.

Mounts are distributed in a round-robin fashion to the **pfsd** daemons.

It is recommended that the **pfsd** daemon be invoked by *rc*(1M). It must be invoked in the background.

The **PFS**daemon is composed of two programs: **pfsd** and **pfsd.rpc**. The **pfsd.rpc** program should **not** be run directly. It is invoked by the **pfsd** program.

Options

-v	Verbose. Show version number, etc.
-o *options*	Specify filesystem *options* using a comma-separated list from the following:

acsize=*n*	The number of entries to keep in the attribute cache (1390 bytes per entry).
bcsize=*n*	The number of entries to keep in the block cache (8244 bytes per entry).
lcsize=*n*	The number of entries to keep in the lookup cache (56 bytes per entry).

The defaults are: **acsize=200,bcsize=25,lcsize=100**

Attributes Cache

The server's attribute cache retains file attribute information on requests that have been made. This provides faster access to entries which have previously been decoded.

Lookup Cache

The lookup cache holds information about the sequential nature of the directory entries. This cache stores the location of the next directory entry. When a request comes in for a directory entry, if the preceding directory entry had been accessed earlier, this location is examined first to see if the directory entry being requested matches the directory entry at that location.

Block Cache

This cache holds raw 8k blocks of recently accessed data.

EXAMPLES

To start a pfs daemon with a 400 entry attribute cache:

```
pfsd -o acsize=400 &
```

To start 4 pfs daemons with the default cache sizes:

```
pfsd 4 &
```

WARNINGS

It is not a good idea to have the cache sizes of the **pfsd** exceed the amount of physical memory (or actually a small portion thereof). If the **pfsd** spends excessive amounts of time swapping to and from disk, the benefits of the caching are diminished.

Specifying cache which consume more virtual memory than available will cause the daemon to die with a vitual memory error.

AUTHOR

pfsd was developed by Young Minds, Inc.

SEE ALSO

pfs_mountd(1M).

NAME

ping - send ICMP Echo Request packets to network host

SYNOPSIS

ping [**-oprv**] [**-i** *address*] [**-t** *ttl*] *host* [**-n** *count*]

ping [**-oprv**] [**-i** *address*] [**-t** *ttl*] *host* *packet-size* [[**-n**] *count*]

DESCRIPTION

The **ping** command sends ICMP Echo Request (ECHO_REQUEST) packets to *host* once per second. Each packet that is echoed back via an ICMP Echo Response packet is written to the standard output, including round-trip time.

ICMP Echo Request datagrams ("pings") have an IP and ICMP header, followed by a **struct timeval** (see *gettimeofday*(2)) and an arbitrary number of "pad" bytes used to fill out the packet. The default datagram length is 64 bytes, but this can be changed by using the *packet-size* option.

Options

The following options and parameters are recognized by **ping**:

-i *address* If *host* is a multicast address, send multicast datagrams from the interface with the local IP address specified by *address* in “dot” notation (see *inet*(3N)). If the **-i** option is not specified, multicast datagrams are sent from the default interface, which is determined by the route configuration.

-o Insert an IP Record Route option in outgoing packets, summarizing routes taken when the command terminates.

It may not be possible to get the round-trip path if some hosts on the route taken do not implement the IP Record Route option. A maximum of 9 Internet addresses can be recorded due to the maximum length of the IP option area.

-p The new Path MTU information is displayed when a ICMP "Datagram Too Big" message is received from a gateway. The **-p** option must be used in conjunction with a large *packetsize* and with the **-v** option.

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-connected network, an error is returned. This option can be used to ping the local system through an interface that has no route through it, such as after the interface was dropped by **gated** (see *gated*(1M)).

-t *ttl* If *host* is a multicast address, set the time-to-live field in the multicast datagram to *ttl*. This controls the scope of the multicast datagrams by specifying the maximum number of external systems through which the datagram can be forwarded.

If *ttl* is zero, the datagram is restricted to the local system. If *ttl* is one, the datagram is restricted to systems that have an interface on the network directly connected to the interface specified by the **-i** option. If *ttl* is two, the datagram can forwarded through at most one multicast router; and so forth. *Range*: zero to 255. The default value is 1.

-v Verbose output. Show ICMP packets other than Echo Responses that are received.

host Destination to which the ICMP Echo Requests are sent. *host* can be a hostname or an Internet address. All symbolic names specified for *host* are looked up by using **gethostbyname()** (see *gethostent*(3N)). If *host* is an Internet address, it must be in "dot" notation (see *inet*(3N)).

If a system does not respond as expected, the route might be configured incorrectly on the local or remote system or on an intermediate gateway, or there might be some other network failure. Normally, *host* is the address assigned to a local or remote network interface.

If *host* is a broadcast address, all systems that receive the broadcast should respond. Normally, these are only systems that have a network interface on the same network as the local interface sending the ICMP Echo Request.

If *host* is a multicast address, only systems that have joined the multicast group should respond. These may be distant systems if the **-t** option is specified, and there is a multicast router on the network directly connected to the interface specified by the **-i** option.

packet-size	The size of the transmitted packet, in bytes. By default (when *packet-size* is not specified), the size of transmitted packets is 64 bytes. The minimum value allowed for *packet-size* is 8 bytes, and the maximum is 4095 bytes. If *packet-size* is smaller than 16 bytes, there is not enough room for timing information. In that case, the round-trip times are not displayed.
count	The number of packets **ping** will transmit before terminating. *Range*: zero to 2147483647. The default is zero, in which case **ping** sends packets until interrupted.

When using **ping** for fault isolation, first specify a local address for *host* to verify that the local network interface is working correctly. Then specify host and gateway addresses further and further away to determine the point of failure. **ping** sends one datagram per second, and it normally writes one line of output for every ICMP Echo Response that is received. No output is produced if there are no responses. If an optional *count* is given, only the specified number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the command times out (if the *count* option is specified), or if the command is terminated with a **SIGINT**, a brief summary is displayed.

This command is intended for use in testing, managing and measuring network performance. It should be used primarily to isolate network failures. Because of the load it could impose on the network, it is considered discourteous to use **ping** unnecessarily during normal operations or from automated scripts.

AUTHOR

ping was developed in the Public Domain.

FILES

/etc/hosts

SEE ALSO

gethostent(3N), inet(3N).

NAME

pong - send Fibre Channel Light Weight Protocol Echo Request packet

SYNOPSIS

/opt/fc/bin/pong *N_port_address*

DESCRIPTION

The **pong** command sends FC_LWP Echo Request packets to a well known FC_LWP protocol port number at the specified *N_port_address* once per second. Information from each packet that is echoed back to the sending N_port is written to the standard output.

FC_LWP Echo Request are datagrams packets which have an FC-PH header and an FC_LWP header which specifies routing to the FC_LWP protocol port, followed by a FC_LWP protocol port Echo request code. Pad bytes are used to fill out the packet to the fixed size of 64 bytes. **Pong** sends one datagram packet per second until interrupted. No output is produced if there are no responses.

This command is intended for use in testing, managing, and measuring network performance for FC_LWP. It should be used primarily to isolate network failures, and validate simple connectivity.

AUTHOR

pong was developed by the HP.

NAME

power_onoff - timed, automatic system power on, and power off

SYNOPSIS

/usr/sbin/power_onoff -n

/usr/sbin/power_onoff *time* [*date*] [[**next** | **+***increment*] *time_designation*]

DESCRIPTION

power_onoff instructs the UPS monitor (**ups_mond**) to shut down the system, and optionally informs the monitor when to power on the system again. The UPS monitor in turn instructs the uninterruptible power source (UPS) when to turn the power off and on. The UPS monitor then proceeds to shut down the system. The time to restart the system (power on) is specified with **power_onoff** command-line arguments.

Some UPS units limit the time that can elapse between the time the power is turned off and the time it is turned back on. Please see your UPS documentation for information about limitations.

power_onoff requires a UPS that is supported by the UPS monitor (see *ups_mond*(1M)).

Command Line Arguments

The **power_onoff** command has two forms, and recognizes the following arguments:

-n No power on. Causes the system to be shutdown and not be powered back on.

time Can be specified as one, two, or four digits. One- and two-digit numbers represent hours; four digits represent hours and minutes. *time* can also be specified as two numbers separated by a colon (**:**), single quote (**'**), the letter "h" (**h**), a period (**.**), or comma (**,**). A suffix **am** or **pm** can be appended. Otherwise a 24-hour clock time is understood. For example, **0815**, **8:15**, **8'15**, **8h15**, **8.15**, and **8,15** are read as 15 minutes after 8 in the morning. The suffixes **zulu** and **utc** can be used to indicate Coordinated Universal Time. The special names **noon**, **midnight**, **now**, and **next** are also recognized.

date Can be specified as either a day of the week (fully spelled out or abbreviated) or a date consisting of a day, a month, and optionally a year. The day and year fields must be numeric, and the month can be fully spelled out, abbreviated, or numeric. These three fields can be in any order, and be separated by punctuation marks such as **/**, **-**, **.**, or **,**. Two special “days”, **today** and **tomorrow**, are also recognized. If no *date* is given, **today** is assumed if the given time is greater than the current time; **tomorrow** is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.

next or **+***increment* If followed by a *time_designation* of **minutes**, **hours**, **days**, **weeks**, **months**, or **years**, lets the user startup the system when the specified *time_designation* has elapsed. A numerical operator, **+***increment*, enables the user to schedule the startup several hours, days, weeks, months, or years in advance (see EXAMPLES). Using the argument **next** is equivalent to using an *increment* of **+1**. Both plural and singular forms of *time_designation* are accepted.

p

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

Exit code 0 is returned upon successful completion, otherwise non 0 is returned.

DIAGNOSTICS

power_onoff issues diagnostic messages when it encounters syntax errors and out-of-range times.

EXAMPLES

To startup the system at 5:00 am next Tuesday, use

```
power_onoff 5am Tuesday next week
```

To startup the system at 5:30 am tomorrow, use

```
power_onoff 5:30 tomorrow
```

To make your system startup each weekday at 7:30am and shutdown at 5:30pm each week day, use **crontab** to execute the first entry on Monday through Thursday and the second entry on Friday (see *crontab*(1)).

```
power_onoff 7:30 tomorrow
power_onoff 7:30 Monday
```

To startup the system at 8:15 on January 24, use

```
power_onoff 0815 Jan 24
```

To startup the system at 5:15 on January 24, use

```
power_onoff 5:15 Jan 24
```

To startup the system at 9:30 tomorrow, use

```
power_onoff 9:30am tomorrow
```

To startup the system 24 hours from now, use

```
power_onoff now + 1 day
```

To shutdown the system and not start it up, use

```
power_onoff -n
```

WARNINGS

Some UPS units limit the time that can elapse between the time the power is turned off and the time it is turned back on. Please see your UPS documentation for information about limitations.

If the *date* argument begins with a number and the *time* argument is also numeric (and without suffix), the *time* argument should be a four-digit number that can be correctly interpreted as hours and minutes.

Do not use both **next** and **+** *increment* within a single **power_onoff** command; only the first operator is accepted and the trailing operator is ignored. No warning or error is produced.

The power cord must be disconnected before servicing the unit.

AUTHOR

power_onoff was developed by HP.

FILES

/var/tmp/timed_off	fifo for communicating with ups_mond.

SEE ALSO

at(1), cron(1M), crontab(1), queuedefs(4), proto(4), kill(1), sam(1M), ups_mond(1M).

p

NAME

pscan - scan an HP SCSI disk array LUN for parity consistency

SYNOPSIS

pscan -s *system_state device_file*

DESCRIPTION

pscan is a front end script to **scn** designed to be called during system bootup and shutdown. It stores information on the disk array used to indicate improper system shutdown. If the system was not properly shutdown, a parity scan is initiated when the system boots. The values of 0 (operating system booting), and 1 (operating system shutdown) are expected for *system_state*. *device_file* refers to the device file associated with the selected disk array. If multiple hosts (initiators) are connected to the disk array, the file /etc/hpC2400/pscan.initiators needs to be created. This file will contain an integer value from 1 to 8. If the file is not created, pscan will assume that only one host (initiator) is connected to the disk array.

RETURN VALUE

pscan returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **pscan**
- SCSI (device level) communications
- system calls

Error messages generated by pscan:

usage: pscan -s <system_state> <special>
An error in command syntax has occurred. Enter command again with all required arguments, in the order shown.

pscan: LUN # too big
The LUN number, which is derived from the device file name, is out of range.

pscan: Not a raw file
Utilities must be able to open the device file for raw access.

pscan: LUN does not exist
The addressed LUN is not configured, and thus is not known to the array controller.

pscan: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

pscan uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **pscan** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To call pscan on the LUN **/dev/rdsk/c2t0d2** prior to a system shutdown on a series 800:

```
pscan -s 1 /dev/rdsk/c2t0d2
```

To call pscan on the LUN **/dev/rdsk/c2t0d2** during a system bootup on a series 700:

```
pscan -s 0 /dev/rdsk/c2t0d2
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

pscan was developed by HP.

NAME

pushAgent - install the Software Distributor agent on remote systems

SYNOPSIS

/usr/sbin/pushAgent [**-a** *additional_diskspace*] [**-m** *machine_name* | **-t** *target_file*]
[**-x prompt_rootname=**[**true**|**false**]

Remarks:

- *This command applies only to the HP OpenView Software Distributor product. It is not part of the SD-UX command set shipped with the HP-UX operating system.*
- For an overview of all SD commands, see the *sd*(5) manual page by typing:

  man 5 sd

DESCRIPTION

The **pushAgent** command provides the HP OpenView Software Distributor (SD-OV) user with a way to install the SD agent onto remote systems for the first time or to replace SD-UX. The tool is especially useful when configuring a large number of remote systems as SD agents.

The SD supports two separate configurations:

- SD controller systems
- SD agent systems

SD controller systems are created by installing the full SD product from the installation media. SD agent systems are created by using **pushAgent** to install the SD agent on remote systems. Once the SD agent has been installed on a remote system using this tool, the remote system becomes a valid target for software distribution tasks.

The **pushAgent** command has two different basic modes of operation: interactive and command line. Command line mode is entered when either **-m** or **-t** is specified on the command line. If neither of these options is specified, interactive mode is used.

In interactive mode, the program will present a number of alternatives to the user via a terminal-based user interface. On-line help is available from within this tool. Use interactive mode when you only need to install the SD agent onto a few systems. When you select this method, you will be prompted for the name of the remote system. Once the system name has been entered, **pushAgent** will install and configure the SD agent on that remote system. While the installation progresses, the current status of the installation will be displayed. Finally, the success or failure of the installation is shown. If the installation failed, the tool will report why, and may suggest a way for the user to remedy the problem so that the installation can be re-tried later.

p

In command line mode, all machines which need the SD agent are listed via the command line (on the command line itself with the **-m** option or in a batch file with the **-t** option). Install the SD agent via **-m** when you have to install the SD agent on just a few systems, but do not want to use the interactive mode.

Install the SD agent from a batch file when you need to install the SD agent on many systems. When you select this method, you must provide an input file which contains a list of remote systems. The **pushAgent** command reads the file, and serially installs the SD agent on each system listed in the file. While the installation progresses, **pushAgent** displays the current status of the installation. When the installation process has completed to all of the remote systems listed in the input file, a summary screen is displayed which lists the names of the systems whose installation failed.

An example input file for the batch installation method is displayed below:

```
# Accounting Department
hpbob # Bob’s machine
hpfrank # Frank’s machine

# R+D Department
grpserver.co.here.com  # Our server system
15.1.2.3   # IP address of their test machine
```

For both modes, verbose logging information is written to the log file **/tmp/pushAgent.log** (see the DIAGNOSTICS section below).

For both modes, access to the remote machines is first attempted via **remsh**. If **remsh** access fails, **rexec** access is attempted. If **rexec** access is not already set up for the remote machine, **pushAgent** prompts for the remote machine's root password.

If you are using **pushAgent** on an HP-UX 10.x system, you will be prompted for the name of a source depot which contains the SD commands. This depot will most likely be your SD media (on which the SD product was delivered). You will only be asked for the name of the source depot once per invocation of **pushAgent**.

pushAgent expects SD for different architectures to be already loaded on the system. The location is **/tmp/sd**. If you do not have enough space on your **/tmp** file system, you can create a soft link to a file system with enough space.

If the SD product for the target system's architecture is not loaded on the target system, **pushAgent** prompts you for the source depot that contains SD. **pushAgent** then loads SD into **/tmp/sd** on the target system.

pushAgent does not remove **/tmp/sd** so that SD will be available on the system for subsequent pushes. If you need the disk space, you can remove **/tmp/sd** manually.

Options

pushAgent supports the following options:

-a Specifies how much additional disk space must be present on each of the target machines in the file system containing /usr. This is space above and beyond what SD itself will require. The number is specified in Kbytes. The default is 0. This option is useful when you have other software which will be installed after the SD agent, and you want to make sure both it and the SD agent will fit on your remote system.

-m Specifies a single machine to install the SD agent on. Multiple **-m** options are allowed on a single invocation of **pushAgent**.

-t Specifies a file containing a list of machines to install the SD agent on (a batch file).

-x prompt_rootname=
This option is useful when your remote system's root user is something other than **root**. Valid option values are either **true** or **false** (default is **false**). If set to **true**, **pushAgent** will prompt for both the remote root name and the remote root password if access to the remote machine via *remsh* fails. If set to **false**, only the root password will be prompted for (the user **root** is assumed).

EXTERNAL INFLUENCES

Signals

The **pushAgent** command catches the signals SIGHUP, SIGINT, SIGQUIT, and SIGTERM. If any of these signals are received, the **pushAgent** command confirms whether the user wishes to exit. Note that sending a signal to the **pushAgent** command while it is installing the SD agent on a remote system may leave that remote system in an inconsistent state, and is therefore not recommended.

SECURITY

When the installation of the SD agent on remote systems has completed, **pushAgent** performs some basic configuration of the security Access Control Lists (ACLs) on the remote system. Specifically, an entry is added to the remote system which will enable full software distribution access by the super-user on the system which is running the **pushAgent** program. This super-user will be able to perform software distribution tasks to the remote system without having to further configure its ACLs.

DIAGNOSTICS

The **pushAgent** tool supports three log files:

- **/tmp/pushAgent.log**
- **/tmp/pushAgent.old**
- **/tmp/pushAgent.older**

Every time this command is started, a new log file **/tmp/pushAgent.log** is created. The **pushAgent** command records in this log file the names of the remote systems which had the SD agent installed on, the success or failure of that installation, together with a detailed message on the type of failure if the installation failed. In addition, log files for the two previous **pushAgent** sessions are maintained. The log information from these two sessions is saved in the files **/tmp/pushAgent.old** and **/tmp/pushAgent.older**.

RETURN VALUES

The **pushAgent** command returns:

- **1** if an invalid command line option is used or if a batch file is incorrect.
- **2** if the user prematurely exits from the program.
- **0** in all other circumstances.

LIMITATIONS

pushAgent is not supported on SunOS. Refer to *Using HP OpenView Software Distributor on Sun Platforms* for more information on how to distribute the SD agent to remote workstations running SunOS.

pushAgent is not supported for installing to PC controllers.

AUTHOR

pushAgent was developed by the Hewlett-Packard Company.

SEE ALSO

HP OpenView Software Distributor Administrator's Guide, swjob(1M), sd(5), remsh(1), rexec(1).

NAME

pvchange - change characteristics and access path of physical volume in LVM volume group

SYNOPSIS

/usr/sbin/pvchange [**-A** *autobackup*] **-s** *pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-S** *autoswitch pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-x** *extensibility pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-t** *IO_timeout pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-z** *sparepv pv_path*

Remarks

pvchange cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **pvchange** command changes the characteristics and access path of a physical volume (*pv_path*) in a volume group.

On dual controller devices, **pvchange** sets the permission that controls whether or not the system will automatically switch from the current controller to the original controller after the original controller has recovered from a failure. It also permits you to switch manually to a controller on the device other than the current controller.

pvchange sets the allocation permission to add physical extents to the physical volume.

If you have installed the optional HP MirrorDisk/UX software, you can use the **-z** option to designate a spare physical volume to be used to replace an existing physical volume within a volume group when mirroring is in effect, in the event the existing physical volume fails.

Options and Arguments

pvchange recognizes the following options and arguments.

pv_path — The block device path name of a physical volume.

-A *autobackup* — Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** — Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** — Do not back up configuration changes this time.

-s — Manually switch access to the device to the path named by *pv_path*, specifying it as the primary path.

-S *autoswitch* — Set the autoswitch operation for the physical volume *pv_path*. *autoswitch* can have one of the following values:

- **y** — Automatically switch back to the original primary path after a recovery from a failure. This is the default.
- **n** — Do not switch back. Stay on the current controller.

-x *extensibility* — Set the allocation permission to add physical extents to the physical volume *pv_path*. *extensibility* can have the following values:

- **y** — Allow allocation of additional physical extents on the physical volume. This is the default.
- **n** — Prohibit allocation of additional physical extents on the physical volume. However, logical volumes residing on the physical volume are accessible.

-t *IO_timeout* — Set *IO_timeout* for the physical volume to the number of seconds indicated. An *IO_timeout* value of zero (0) causes the system to use the default value supplied by the device driver associated with the physical device. *IO_timeout* is used by the device driver to determine how long to wait for disk transactions to complete before concluding that an IO request can not be completed (and the device is offline or unavailable).

-z *sparepv* This option requires the installation of the optional HP MirrorDisk/UX software. It allows you to change the physical volume specified by *pv_path* into a spare physical volume for its volume group, or change the specified spare physical volume back into a regular physical volume for this volume group. No physical extents from a spare physical volume will be available as part of the "free" pool of extents in the volume group. A spare physical volume will only be used in the event that another physical volume within this volume group becomes unavailable (fails). *sparepv* can have one of the following values:

y Change the specified physical volume to be a "stand-by" spare for its volume group. The specified physical volume must not have extents allocated on it (i.e., no logical volumes residing on it) at the time this command is issued. A stand-by spare physical volume will only be used in the event of a failure of another physical volume -- prior to such a failure, no logical volume is allowed to reside on it.

n Change the specified spare physical volume back into a regular physical volume. If the physical volume was a stand-by spare, then all of the disk space associated with it will be immediately available for use by logical volumes. If the physical volume is an "active" spare, that is, it was previously a stand-by spare but then took over for a failed physical volume, it will simply mark the physical volume as a regular member of its volume group and the logical volumes residing on it will remain unchanged.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).

EXAMPLES

Prohibit the allocation of additional physical extents to a physical volume:

```
pvchange -x n /dev/dsk/c0t0d0
```

Allow the allocation of additional physical extents to a physical volume:

```
pvchange -x y /dev/dsk/c0t0d0
```

Prohibit a switch back to the original primary controller after it has recovered from a failure:

```
pvchange -S n /dev/dsk/c0t0d0
```

Allow a switch back to the original primary controller after it has recovered from a failure:

```
pvchange -S y /dev/dsk/c0t0d0
```

Manually switch a physical volume to use another controller path:

```
pvchange -s /dev/dsk/c2t0d2
```

Set the *IO_timeout* value for a physical volume to 60 seconds:

```
pvchange -t 60 /dev/dsk/c2t0d2
```

Set the *IO_timeout* value for a physical volume to zero (0) to use the driver default:

```
pvchange -t 0 /dev/dsk/c2t0d2
```

Change the (empty) physical volume to become a stand-by spare for the volume group:

```
pvchange -z y /dev/dsk/c2t0d2
```

Change the (active or stand-by) spare physical volume to become a regular member of the volume group:

```
pvchange -z n /dev/dsk/c2t0d2
```

SEE ALSO

pvdisplay(1M).

NAME

pvck - check or repair a physical volume in LVM volume group

SYNOPSIS

/usr/sbin/pvck -y *pv_path*

/usr/sbin/pvck -n *pv_path*

DESCRIPTION

Note: Currently **pvck** is only capable of detecting bad checksums caused by a forward system migration after a backward system migration. It should not be used in other situations.

The **pvck** command examines and repairs LVM data structures on a raw disk (*pv_path*) in a volume group.

Options and Arguments

pvck recognizes the following options and arguments.

-y	Repair problems found.
-n	Report, but do not repair problems.
pv_path	The raw device path name of a physical volume.

RETURN VALUE

pvck returns the following values

0	Either no problems were found or all problems were corrected.
1	Unable to repair.

EXAMPLES

Examine LVM checksums on **/dev/rdsk/c0t6d0** without modifying anything:

```
pvck -n /dev/rdsk/c0t6d0
```

Repair LVM checksums on **/dev/rdsk/c0t6d0** if necessary:

```
pvck -y /dev/rdsk/c0t6d0
```

WARNINGS

pvck should only be run on a device whose volume group has not been activated.

It is designed to repair the root device or devices while the system is booted in maintenance mode ("**hpux -lm**", see *hpux*(1M)).

AUTHOR

pvck was developed by HP.

NAME

pvcreate - create physical volume for use in LVM volume group

SYNOPSIS

/usr/sbin/pvcreate [**-b**] [**-B**] [**-d** *soft_defects*] [**-s** *disk_size*] [**-f**] [**-t** *disk_type*] *pv_path*

DESCRIPTION

The **pvcreate** command initializes a direct access storage device (a raw disk device) for use as a physical volume in a volume group.

If *pv_path* contains a file system and the **-f** option is not specified, **pvcreate** asks for confirmation. The request for confirmation avoids accidentally deleting a file system.

The operation is denied if *pv_path* belongs to another volume group. Only physical volumes not belonging to other volume groups can be created.

After using **pvcreate** to create a physical volume, use **vgcreate** to add it to a new volume group or **vgextend** to add it to an existing volume group (see *vgcreate*(1M) and *vgextend*(1M)).

Disks cannot be added to a volume group until they are properly initialized by **pvcreate**.

pv_path can be made into a bootable disk by specifying the **-B** option, which reserves space on the physical volume for boot-related data. This is a prerequisite for creating root volumes on logical volumes. Refer to *mkboot*(1M) and *lif*(4) for more information.

Options and Arguments

pvcreate recognizes the following options and arguments:

pv_path	The character (raw) device path name of a physical volume.
-b	Read from standard input the numbers that correspond to the indexes of all known bad blocks on the physical volume, *pv_path*, that is being created. Specify the indexes using decimal, octal, or hexadecimal numbers in standard C-language notation, with numbers separated by newline, tab, or formfeed characters. If this option is not used, **pvcreate** assumes that the physical volume contains no bad blocks.
-B	Make a bootable physical volume (i.e., a system disk).
-d *soft_defects*	Specify the minimum number of bad blocks that LVM should reserve in order to perform software bad block relocation. This number can be no larger than 7039. If not specified, one block is reserved for each 8K of data blocks. This option is not supported on HP-IB devices; *soft_defects* is set to 0 when **pvcreate** is executed for an HP-IB device.
-s *disk_size*	Effective size of the physical volume to be created, specified in number of physical sectors.
-f	Force the creation of a physical volume (thus deleting any file system present) without first requesting confirmation.
-t *disk_type*	Retrieve configuration information about the physical volume from the file **/etc/disktab**. *disk_type* specifies the device (for example, hp7959S). The *disk_type* only needs to be specified when **pvcreate** fails to get the size from the underlying disk driver. If the driver successfully returns the size of the device, *disk_type* is ignored.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Create a physical volume on raw device **/dev/rdsk/c1t0d0**, and force the creation without confirmation:

```
pvcreate -f /dev/rdsk/c1t0d0
```

Create a physical volume on raw device **/dev/rdsk/c1t0d0**, specifying that a bad blocks list (7, 13, 95, and 133) must be read from standard input:

```
echo 7 13 95 133 | pvcreate -b /dev/rdsk/c1t0d0
```

FILES

/etc/disktab	Disk geometry and disk partition characteristics for all disk devices on the system

WARNINGS

Check the manufacturer's listing or run diagnostics testing for bad blocks on the device prior to creating a physical volume. If bad blocks are present, use the **-b** option when creating the physical volume.

SEE ALSO

mkboot(1M), vgcreate(1M), vgextend(1M), lif(4).

NAME

pvdisplay - display information about physical volumes within LVM volume group

SYNOPSIS

/usr/sbin/pvdisplay [**-v**] [**-b** *BlockList*] *pv_path* ...

DESCRIPTION

The **pvdisplay** command displays information about each physical volume specified by a *pv_path* parameter.

Options

pvdisplay recognizes the following options:

pv_path The block device path name of a physical volume.

-v For each physical volume, display the logical volumes that have extents allocated on the physical volume and the usage of all the physical extents.

-b *BlockList*
For each block in *BlockList*, display information about the block. *BlockList* is a comma separated list of blocks in **DEV_BSIZE** units.

Display Without –v Option

If you omit the **-v** option, **pvdisplay** displays the characteristics of each physical volume specified by *pv_path*:

--- Physical volumes ---

PV Name The block device path name of the physical volume

VG Name The path name of the volume group

PV Status State of the physical volume (*NOTE*: spare physical volumes are only relevant if you have installed HP MirrorDisk/UX software):

available The physical volume is available and is not a spare physical volume.

available/data spared
The physical volume is available. However, its data still resides on an active spare.

available/active spare
The physical volume is available and is an active spare physical volume. (An active spare is a spare that has taken over for a failed physical volume.)

available/standby spare
The physical volume is a spare, "standing by" in case of a failure on any other physical volume in this volume group. It can only be used to capture data from a failed physical volume.

unavailable The physical volume is unavailable and is not a spare physical volume.

unavailable/data spared
The physical volume is unavailable. However, its data now resides on an active spare, and its data is available if the active spare is available.

unavailable/active spare
The physical volume is unavailable and it's an active spare. Thus, the data on this physical volume in unavailable.

unavailable/standby spare
The physical volume is a spare, "standing by" that is not currently available to capture data from a failed physical volume.

`Allocatable`	Allocation permission for the physical volume
`VGDA`	Number of volume group descriptors on the physical volume
`Cur LV`	Number of logical volumes using the physical volume
`PE Size (Mbytes)`	Size of physical extents on the volume, in megabytes (MB)
`Total PE`	Total number of physical extents on the physical volume
`Free PE`	Number of free physical extents on the physical volume
`Allocated PE`	Number of physical extents on the physical volume that are allocated to logical volumes
`Stale PE`	Number of physical extents on the physical volume that are not current
`IO Timeout`	The IO timeout used by the disk driver when accessing the physical volume. A value of **`default`**, indicates that the driver default IO timeout is being used.
`Spared from PV`	If the physical volume represents an active spare, this field will show the name of the failed physical volume whose data now resides on this spare. This information can be used to manually move the data back to the original physical volume, once it has been repaired. (See *pvmove*(1M)*)*. If it cannot be determined which physical volume that the data came from, this field will instead display **`Missing PV`**. A missing PV would indicate that when the volume group was last activated or reactivated (see *vgchange*(1M)*)*, the "failed" physical volume was not able to attach to the volume group.
`Spared to PV`	If the physical volume represents a failed physical volume, this field will show the name of the active spare physical volume that now contains the data that originally resided on this volume. This information can be used to manually move the data back to the original physical volume (see *pvmove*(1M)*)* once it has been repaired.

Display With −v Option

If **`-v`** is specified, **`pvdisplay`** lists additional information for each logical volume and for each physical extent on the physical volume:

`--- Distribution of physical volume ---`

The logical volumes that have extents allocated on *pv_path*, displayed in column format:

`LV Name`	The block device path name of the logical volume which has extents allocated on *pv_path*.
`LE of LV`	Number of logical extents within the logical volume that are contained on this physical volume
`PE for LV`	Number of physical extents within the logical volume that are contained on this physical volume

`--- Physical extents ---`

The following information for each physical extent, displayed in column format:

`PE`	Physical extent number
`Status`	Current state of the physical extent: **`free`**, **`used`**, or **`stale`**
`LV`	The block device path name of the logical volume to which the extent is allocated
`LE`	Index of the logical extent to which the physical extent is allocated

Display With −b Option

If **`-b`** is specified, **`pvdisplay`** lists additional information for each block specified in *BlockList*.

`--- Block Mapping ---`

The use of blocks on *pv_path*, displayed in column format:

`Block`	The block number relative to the physical volume.
`Status`	The current status of the block: **`free`**, **`used`**, **`structure`**, **`spared`**, or **`unknown`**
`Offset`	The offset of the block relative to the logical volume.
`LV Name`	The block device path name of the logical volume to which the block is allocated.

EXTERNAL INFLUENCES

Environment Variables

`LANG` determines the language in which messages are displayed.

If **`LANG`** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Display the status and characteristics of a physical volume:

```
pvdisplay /dev/dsk/c1t0d0
```

Display the status, characteristics, and allocation map of a physical volume:

```
pvdisplay -v /dev/dsk/c2t0d0
```

SEE ALSO

lvdisplay(1M), pvchange(1M), vgdisplay(1M).

NAME

pvmove - move allocated physical extents from one LVM physical volume to other physical volumes

SYNOPSIS

/usr/sbin/pvmove [**-A** *autobackup*] [**-n** *lv_path*] *source_pv_path*
[*dest_pv_path* ... | *dest_pvg_name* ...]

Remarks

pvmove cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **pvmove** command moves allocated physical extents and the data they contain from a source physical volume, *source_pv_path*, to one or more other physical volumes in the same volume group.

If a destination physical volume or physical volume group is not specified, all physical volumes in the volume group are available as destination volumes for the transfer. **pvmove** selects the proper physical volumes to be used in order to preserve the allocation policies of the logical volume involved.

To limit the transfer to specific physical volumes, specify the name of each physical volume directly with a *dest_pv_path* argument. Optionally, if physical volume groups are defined for the volume group, specify the physical volumes indirectly with one or more *dest_pvg_name* arguments.

source_pv_path must not appear as a *dest_pv_path*.

If *source_pv_path* is a member of a *dest_pv_path*, it is automatically excluded from being a destination physical volume.

pvmove succeeds only if there is enough space on the destination physical volumes to hold all the allocated extents of the source physical volume.

If you have installed HP MirrorDisk/UX on your system and *source_pv_path* is an "active spare" physical volume within a mirrored logical volume, once all of the data has been moved to *dest_pv_path*, the *source_pv_path* physical volume will be returned to a "stand-by" spare physical volume. This is how to "unspare" data once the original failed physical volume has been repaired and is available to receive data.

Options

pvmove recognizes the following options:

dest_pv_path The block device path name of a physical volume. It cannot be the source physical volume. It must be in the same volume group as *source_pv_path*.

dest_pvg_name The name of a physical volume group. It must be in the same volume group as *source_pv_path*.

source_pv_path The block device path name of a physical volume.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the physical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the physical volume belongs.

- **n** Do not back up configuration changes this time.

-n *lv_path* Move only the physical extents allocated to the logical volume specified by *lv_path* that are located on the source physical volume specified by *source_pv_path*.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Move physical extents from **/dev/dsk/c1t0d0** to **/dev/dsk/c2t0d0** and **/dev/dsk/c3t0d0**:

```
pvmove /dev/dsk/c1t0d0 /dev/dsk/c2t0d0 /dev/dsk/c3t0d0
```

If physical volumes **/dev/dsk/c2t0d0** and **/dev/dsk/c3t0d0** are the only ones that belong to physical volume group **PVG0**, the same result can be achieved with the following command:

```
pvmove /dev/dsk/c1t0d0 PVG0
```

Move only the physical extents for logical volume **/dev/vg01/lvol2** that are currently on **/dev/dsk/c1t0d0** to **/dev/dsk/c2t0d0**:

```
pvmove -n /dev/vg01/lvol2 /dev/dsk/c1t0d0 /dev/dsk/c2t0d0
```

SEE ALSO

pvdisplay(1M), vgcfgbackup(1M).

NAME

pwck, grpck - password/group file checkers

SYNOPSIS

/usr/sbin/pwck [**-s**] [**-l**] [*file*]

/usr/sbin/grpck [*file*]

DESCRIPTION

pwck scans the default password file or *file* and reports any inconsistencies to standard error. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program name exist. The criteria for determining a valid login name are described in the *Managing Systems and Workgroups* manual. The default password file is **/etc/passwd**.

grpck verifies all entries in the group file and reports any inconsistencies to standard error. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is **/etc/group**.

Options

pwck recognizes the following options:

-s Check inconsistencies with the Protected Password database. It calls **authck -p**.

-l Check encrypted password lengths that are greater than 8 characters. If **Nis+** is running with Trusted mode, password lengths must not be longer than 8 characters.

DIAGNOSTICS

Group entries in **/etc/group** with no login names are flagged.

AUTHOR

pwck was developed by AT&T and HP.

DEPENDENCIES

NFS:

pwck and **grpck** check only the local password and group files. The Network Information Service database for password and group files is not checked.

FILES

/etc/group
/etc/passwd

SEE ALSO

group(4), passwd(4), authck(1M).

STANDARDS CONFORMANCE

pwck: SVID2, SVID3

grpck: SVID2, SVID3

NAME

pwconv - update secure password facility

SYNOPSIS

pwconv

DESCRIPTION

If the secured password facility is already installed, **pwconv** updates the facility to reflect any changes made in the **/etc/passwd** file.

FILES

/etc/passwd
/tcb/files/auth/*/*

SEE ALSO

vipw(1M), pwck(1M).

NAME

pwgr_stat - Password and Group Hashing and Caching Statistics

SYNOPSIS

/usr/sbin/pwgr_stat

DESCRIPTION

pwgr_stat displays the current status of the **pwgrd** daemon process running on the system. It includes whether or not the daemon is running, how much activity is occurring, as well as statistics for each kind of request serviced by **pwgrd**. Request specific statistics include the number of request and the percent of requests handled by the cache and the hashtables used to service that request. A request may not have both a cache and a hashtable. Such requests will have a **-** for the corresponding hit rate. Requests where no answer was found are not counted in the hit rate.

The display is updated every 2 seconds. Use the **q** key to exit **pwgr_stat**. **pwgr_stat** verifies that **pwgrd** is accessible by issuing a **NULL** request to **pwgrd**, therefore the NULL request count will be increased as long as **pwgr_stat** is running.

FILES

/var/spool/pwgr/daemon	Daemon Unix domain socket.
/var/spool/pwgr/status	Daemon process status file.

AUTHOR

pwgr_stat was developed by the Hewlett-Packard Company.

SEE ALSO

pwgrd(1M).

NAME

pwgrd - Password and Group Hashing and Caching daemon

SYNOPSIS

/usr/sbin/pwgrd [**-d**] [**-l** *logfile*]

DESCRIPTION

pwgrd provides accelerated lookup of password and group information for libc routines like **getpwuid** and **getgrname**. **pwgrd** implements per request type caches and hashtables as appropriate. When the corresponding routine in libc is called, a request is issued to **pwgrd** via a Unix domain socket connection. **pwgrd** determines whether it can satisfy the request, returning the appropriate results to the requesting process.

Options

pwgrd recognizes the following options and command-line arguments:

-d	Debug mode. Do not become a daemon. Issue additional diagnostic messages. Instead of logging message via **syslog**, issue messages to stderr.
-l	*logfile* Logfile. In addition to logging via **syslog**, **pwgrd** will write log messages to *logfile*.

pwgrd modifies its behavior depending on whether or not the local machine is using some form of *NIS* for password or group information. When *NIS* or *NIS+* is being used, the hashtables corresponding to that service are not generated or consulted. Therefore only caching is provided for those requests.

FILES

/etc/rc.config.d/pwgr	Start up configuration variable. Set **PWGR** to **1** if you want **pwgrd** to start on reboot.
/var/spool/pwgr/*	Hash files, status file and daemon Unix domain socket.
/var/spool/sockets/pwgr/*	Client Unix domain sockets.

AUTHOR

pwgrd was developed by the Hewlett-Packard Company.

SEE ALSO

pwgr_stat(1M).

NAME

quot - summarize file system ownership

SYNOPSIS

/usr/sbin/quot [**-F** *FStype*] [**-V**] [**-cfhnv**] [**-o** *FSspecific-options*] *filesystem* ...

/usr/sbin/quot [**-F** *FStype*] [**-V**] [**-cfhnv**] **-a**

DESCRIPTION

The **quot** command displays the number of 1024-byte blocks in the named *filesystem* that are currently owned by each user. *filesystem* is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

Options

quot recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, the file system type is determined from the file **/etc/fstab** by matching *filesystem* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-o *FSspecific-options*
Specify any options specific to the file system.

-a Generate a report for all mounted file systems.

-c Report size rather than user statistics. Generates histogram statistics in 3-column format:

Column 1:	File size in blocks. Sizes are listed in ascending order up to 499 blocks per file. Files occupying 499 or more blocks are counted together on a single line as 499-block files (but column 3 is based on actual number of blocks occupied).
Column 2:	Number of files of size indicated in column 1.
Column 3:	Cumulative total blocks occupied by files counted in current plus all preceding lines.

Use of this option overrides the **-f** and **-v** options.

-f Display number of files and space occupied by each user.

-h Calculate the number of blocks in the file based on file size rather than actual blocks allocated. This option does not account for sparse files (files with holes in them).

-n Accept data from the **ncheck** command (see *ncheck*(1M)) as input. Run the pipeline:

ncheck *device* **| sort +0n | quot -n** *filesystem*

to produce a list of all files and their owners.

-v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

AUTHOR

quot was developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/default/fs	Specifies the default file system type
/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
/etc/passwd	Password file (contains user names)

q

SEE ALSO

quot_*FStype*(1M), du(1), find(1), ls(1), fstyp(1M), mount(1M), ncheck(1M), repquota(1M), fs_wrapper(5), quota(5).

NAME

quot (hfs) - summarize ownership on an HFS file system

SYNOPSIS

/usr/sbin/quot [**-F hfs**] [**-V**] [**-cfhnv**] *filesystem* ...

/usr/sbin/quot [**-F hfs**] [**-V**] [**-cfhnv**] **-a**

DESCRIPTION

The **quot** command displays the number of 1024-byte blocks in the named HFS *filesystem* that are currently owned by each user. *filesystem* is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

Options

quot recognizes the following options:

-F hfs Specify the file system type **hfs**.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab.** This option allows the user to verify the command line. If the options specified are valid, the completed command line is echoed. If the options specified are not valid, an error message is printed.

-a Generate a report for all mounted HFS file systems.

-c Report size rather than user statistics. Generates histogram statistics in 3-column format:

Column 1:	File size in blocks. Sizes are listed in ascending order up to 499 blocks per file. Files occupying 499 or more blocks are counted together on a single line as 499-block files (but column 3 is based on actual number of blocks occupied).
Column 2:	Number of files of size indicated in column 1.
Column 3:	Cumulative total blocks occupied by files counted in current plus all preceding lines.

Use of this option overrides the **-f** and **-v** options.

-f Display number of files and space occupied by each user.

-h Calculate the number of blocks in the file based on file size rather than actual blocks allocated. This option does not account for sparse files (files with holes in them).

-n Accept data from the **ncheck** command (see *ncheck*(1M)) as input. Run the pipeline:

ncheck *device* **| sort +0n | quot -n** *filesystem*

to produce a list of all files and their owners.

-v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

AUTHOR

quot, a disk quota command, was developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
/etc/passwd	Password file (contains user names).

SEE ALSO

quot(1M), du(1), find(1), ls(1), fstyp(1M), mount(1M), ncheck(1M), repquota(1M), quota(5).

NAME

quot(vxfs) - summarize ownership on a VxFS file system

SYNOPSIS

/usr/sbin/quot [**-F vxfs**] [**-V**] [**-cfhnv**] *filesystem* ...

/usr/sbin/quot [**-F vxfs**] [**-V**] [**-cfhnv**] **-a**

DESCRIPTION

The **quot** command displays the number of 1024-byte blocks in the named VxFS *filesystem* that are currently owned by each user. *filesystem* is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

Options

quot recognizes the following options:

-F vxfs
Specifies file system type **vxfs**

-V Validate the command line options, but perform no other action. If the options specified are valid, the complete command line is echoed. If the options specified are not valid, an error message is printed.

-a Generate a report for all mounted file systems.

-c Report size rather than user statistics. Generates histogram statistics in 3-column format:

Column 1:	File size in blocks. Sizes are listed in ascending order up to 499 blocks per file. Files occupying 499 or more blocks are counted together on a single line as 499-block files (but column 3 is based on actual number of blocks occupied).
Column 2:	Number of files of size indicated in column 1.
Column 3:	Cumulative total blocks occupied by files counted in current plus all preceding lines.

Use of this option overrides the **-f** and **-v** options.

-f Display number of files and space occupied by each user.

-h Calculate the number of blocks in the file based on file size rather than actual blocks allocated. This option does not account for sparse files (files with holes in them).

-n Accept *ncheck*(1M) data as input. Run the pipeline

ncheck *device* **| sort +0n | quot -n** *filesystem*

to produce a list of all files and their owners.

-v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/mnttab	Mounted file system table
/etc/passwd	Password file (contains user names).

SEE ALSO

quot(1M), du(1), find(1), fstyp(1M), ls(1), mount(1M), ncheck(1M), repquota(1M), quota(5).

NAME

quotacheck (generic) - file system quota consistency checker

SYNOPSIS

/usr/sbin/quotacheck [**-F** *FStype*] [**-V**] [**-o** *specific-options*] *filesystem* ...

/usr/sbin/quotacheck [**-F** *FStype*] [**-V**] [**-o** *specific-options*] **-a**

DESCRIPTION

The **quotacheck** command examines each file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

quotacheck expects each file system to be checked to have a file named **quotas** in the root directory. If none is present, **quotacheck** reports an error and ignores the file system. **quotacheck** is normally run at mount time from start-up scripts.

filesystem represents a mount point or block special device such as **/dev/dsk/c1t0d2**.

Options

quotacheck recognizes the following options:

-F *Fstype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *filesystem* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-o *specific-options*
Specify options specific to each file system type. *specific-options* is a list of suboptions and/or keyword/attribute pairs intended for a *FStype*-specific module of the command. See the file system specific man pages for a description of the *specific-options* supported, if any.

-a Obtain list of file systems to check from **/etc/fstab**. Only mounted **rw** (or **default**) type file systems with the **quota** option are checked.

q

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **quotacheck** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

quotacheck was developed by HP and the University of California, Berkeley.

FILES

/etc/default/fs	Specifies the default file system type
/etc/fstab	Default list of file systems to check
/etc/mnttab	Mounted file system table
*directory***/quotas**	Quota statistics static storage for file system where *directory* is the file system root as specified to the **mount** command (see *mount*(1M)).

SEE ALSO

fs_wrapper(5), mount(1M), quota(5), quotacheck_*FSType*(1M).

NAME

quotacheck (hfs) - quota consistency checker for HFS file systems

SYNOPSIS

/usr/sbin/quotacheck [**-F hfs**] [**-V**] [**-pPv**] *filesystem* ...

/usr/sbin/quotacheck [**-F hfs**] [**-V**] [**-pPv**] **-a**

DESCRIPTION

The **quotacheck** command examines each HFS file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

quotacheck expects each file system to be checked to have a file named **quotas** in the root directory. If none is present, **quotacheck** reports an error and ignores the file system. **quotacheck** is normally run at mount time from start up scripts.

filesystem represents a mount point or block special device (e.g., **/dev/dsk/c1t0d2**).

Options

quotacheck recognizes the following options:

-F hfs	Specify the file system type **hfs**.
-V	Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab.** This option allows the user to verify the command line.
-a	Obtain list of file systems to check from **/etc/fstab**. Only mounted file systems of type **hfs** and **rw** (or **default**) with the **quota** option are checked.
-v	Indicate the calculated disk quotas for each user on a particular file system. **quota-check** normally reports only those quotas that are modified.
-p	Check file systems in parallel as allowed by equal values in the *pass number* field in **/etc/fstab**.
-P	Preen file systems, checking only those with invalid quota statistics (**quotaoff** and **edquota** commands can invalidate quota statistics as discussed in *quota*(5) — see *quotaoff*(1M) and *edquota*(1M)). Also checks in parallel as in **-p** above.

q

AUTHOR

quotacheck was developed by HP and the University of California, Berkeley.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
directory/**quotas**	Quota statistics static storage for filesystem where *directory* is the file system root as specified to the **mount** command (see *mount*(1M)).

SEE ALSO

mount(1M), quota(5), quotacheck(1M), quotaon(1M), quotaoff(1M).

NAME

quotacheck - VxFS file system quota consistency checker

SYNOPSIS

/usr/sbin/quotacheck [**-F vxfs**] [**-V**] [**-pPv**] *filesystem*...

/usr/sbin/quotacheck [**-F vxfs**] [**-V**] [**-pPv**] -a

DESCRIPTION

Since VxFS maintains quota information in the kernel, **quotacheck** for VxFS syncs quotas from the current system copy to the disk quota file for the VxFS file system.

quotacheck expects each file system to be checked to have a file named **quotas** in the root directory. **quotacheck** is normally run at mount time from start-up scripts.

filesystem represents a mount point or block special device (e.g., **/dev/dsk/c0t0d0)**.

Options

quotacheck recognizes the following options:

-F vxfs	Specify the file-system type **vxfs**.
-V	Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.
-a	Obtain list of file systems to check from **/etc/fstab**. Only mounted **rw** type file systems with the **quota** option are checked.
-v	Reports the file system name before syncing quotas from current system copy to the disk quota file.
-p	This option does nothing, but exists for standards compatibility.
-P	This option does nothing, but exists for standards compatibility.

AUTHOR

quotacheck was developed by HP and the University of California, Berkeley.

FILES

/etc/fstab	default file systems
*directory***/quotas**	quota statistics static storage for filesystem where *directory* is the file system root as specified to **mount** (see *mount*(1M)).

SEE ALSO

quota(5), quotacheck(1M), quotacheck_hfs(1M).

NAME

quotaon, quotaoff - turn HFS file system quotas on and off

SYNOPSIS

/usr/sbin/quotaon [**-v**] *filesystem* ...

/usr/sbin/quotaon [**-v**] **-a**

/usr/sbin/quotaoff [**-v**] *filesystem* ...

/usr/sbin/quotaoff [**-v**] **-a**

Remarks

These commands are provided for compatibility only. Their use is neither required nor recommended because **mount** and **umount** enable and disable quotas cleanly (see *mount*(1M)). See *WARNINGS* below for more information.

DESCRIPTION

The **quotaon** command enables quotas on one or more HFS file systems.

The **quotaoff** command disables quotas on one or more HFS file systems.

filesystem is either the name of the mount point of the file system, or the name of the block device containing the file system. The file systems specified must be currently mounted in order to turn quotas on or off. Also, the file system quota file, **quotas**, must be present in the root directory of each specified file system.

These commands will update the appropriate entries in **/etc/mnttab** to indicate that quotas are on or off for each file system.

When enabling quotas interactively after boot time, the **quotacheck** command should be run immediately afterward (see *WARNINGS* below).

Use **mount** (see *mount*(1M)) to determine whether quotas are enabled on mounted file systems.

Options

The following options affect the behavior described above.

-a Obtain the *filesystem* list from **/etc/fstab**, using entries of type **hfs** and **rw** (or **default**) with the **quota** option (see *fstab*(4)).

-v Generate a message for each file system affected.

q

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **quotaon** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

Using **quotaoff** to disable quotas on a file system causes the system to discontinue tracking quotas for that file system, and marks the **quota clean** flag in the superblock **NOT_OK** (see *fsclean*(1M)). This in turn, forces a **quotacheck** the next time the system is booted. Since quotas are enabled and disabled cleanly by **mount** and **umount** anyway, the use of **quotaon** and **quotaoff** is generally discouraged.

AUTHOR

Disk quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/fstab	Static information about the file systems

`/etc/mnttab`	Mount file system table
*directory***`/quotas`**	Quota statistics storage for the file system, where *directory* is the root of the file system as specified to the **`mount`** command (see *mount*(1M)).

SEE ALSO

fsclean(1M), quotacheck(1M), quotacheck_hfs(1M), quotacheck_vxfs(1M), mount(1M), quota(5).

NAME

rarpc - Reverse Address Resolution Protocol client

SYNOPSIS

rarpc [**-d**] [**-e**|**-s**] [**-n** *count*] *interface_name*

DESCRIPTION

rarpc, the Reverse Address Resolution Protocol client, implements the client portion of the Reverse Address Resolution Protocol (see *SEE ALSO*). It sends RARP requests for the specified interface's hardware address and waits for the response from the RARP server. **rarpc** can be used during boot-time initialization to find the IP address of an interface. To do so, set the **IP_ADDRESS**[*i*] variable of interface *i* with **IP_ADDRESS[** *i* **]=RARP** in **/etc/rc.config.d/netconf**.

Options are:

-d	Print debugging information.
-e	Use ethernet encapsulation only.
-s	Use SNAP encapsulation only.
-n *count*	Transmits count requests and waits for each one to time out before giving up.
interface_name	Identifies the interface to request information about.

If a response is received, it prints the IP address to its standard output. This information can be used to configure the interface as seen in **/sbin/init.d/net**.

If a response is not received, the client will retransmit after 2 seconds, and then after 4 seconds. After that, retransmissions occur every 8 seconds.

RETURN VALUE

Exit status is 1 if the command fails or no RARP response is received. Exit status is 0 and the IP address is printed to standard output if a response is received.

LIMITATIONS

1. The **rarpc** client cannot be run at the same time a **rarpd** daemon is running on the same interface.
2. The **rarpc** client supports only ethernet, 100VG and FDDI network interfaces.

AUTHOR

rarpc was developed by HP.

SEE ALSO

rarpd(1M).

R. Finlayson, T. Mann, J.C. Mogul, M. Theimer, "Reverse Address Resolution Protocol", RFC 903.

NAME

rarpd - Reverse Address Resolution Protocol daemon

SYNOPSIS

rarpd [**-d**] [**-f** *config_file*] [*interface_name*]

DESCRIPTION

rarpd, the Reverse Address Resolution Protocol daemon, implements the server portion of the Reverse Address Resolution Protocol [1]. It responds to RARP requests providing the requested client IP address. Rarpd can be started during boot-time initialization. To do so, set the RARPD variable with **RARPD=1** in **/etc/rc.config.d/netconf**.

Options are:

-d	Print debugging information.
-f *config_file*	Use the specified config_file database instead of **/etc/rarpd.conf**.
interface_name	Respond to requests over just this interface.

The configuration file database contains hardware address to IP address mappings. Other than comment lines (which begin with a '#') and blank lines, all lines are considered client entries. A client entry is of the form:

hardware_address WHITE_SPACE *ip_address*

where *hardware_address* consists of (**:**) colon-separated hexadecimal bytes, and *ip_address* consists of (**.**) dot-seperated decimal bytes. For example:

```
#
# hardware addr   IP addr
#
#   ethernet clients
08:00:09:26:ec:19 15.13.136.68
08:00:09:17:0a:93 15.13.136.74
#
#   100VG clients
08:00:09:63:5d:f5 190.20.30.103
#
#   FDDI clients
08:00:09:09:53:4c 192.20.30.98
```

There must be exactly 6 hardware address bytes. There must be exactly 4 protocol address bytes.

The following signals have the specified effect when sent to the rarpd process using the *kill*(1) command:

SIGHUP	Causes server to read the config file and reload database.
SIGINT	Dumps current data base and cache to **/var/tmp/rarpd.db**.

RETURN VALUE

Exit status is 1 if the command fails, and error messages are written to stderr and/or syslog. Typically, the daemon will continue answering requests until externally interrupted.

LIMITATIONS

1. The rarpd daemon supports only ethernet, 100VG and FDDI network interfaces.
2. The rarpd daemon supports only 4 byte Internet Protocol addresses.
3. The rarpd and rarpc programs cannot be run on the same interface at the same time.

AUTHOR

rarpd was developed by HP.

SEE ALSO

rarpc(1M).

[1] R. Finlayson, T. Mann, J.C. Mogul, M. Theimer, "Reverse Address Resolution Protocol", RFC 903.

NAME

rbootd - remote boot server for RMP clients

SYNOPSIS

/usr/sbin/rbootd [**-a**] [**-l** *loglevel*] [**-L** *logfile*] [**-t** *minutes*] [*landevs*]

DESCRIPTION

rbootd services initial boot-up requests from RMP clients over a local area network. Early s700 workstations and all Datacommunications and Terminal Controllers (DTC/9000) use this RMP protocol and can only communicate with **rbootd** during boot-up. Later s700 workstations (starting with the s712) use the industry standard BOOTP protocol and communicate with *bootpd*(1M). Future s700 workstations will use the BOOTP protocol. See the listings below.

rbootd now acts as a forwarding agent for s700 RMP clients, receiving their RMP boot requests and reformulating them into BOOTP boot requests that are sent to the local **bootpd** daemon. If **bootpd** replies to this boot request, **rbootd** receives the BOOTP reply and produces an RMP reply which is sent to the client. **rbootd** continues to act as the intermediary in this transaction until the client is successfully booted.

rbootd only responds to DTC clients if they are listed in the **map802** file. The **map802** file (a binary file) is created when a DTC is configured by *dtcconfig*(1M) on the host machine.

In order to boot a s700 RMP client run **rbootd** and **bootpd** on the server machine, on the same subnet as the client. If the local **bootpd** daemon is acting as a relay agent, there must also be a remote NFS Diskless server with the necessary boot files and NFS or **tftp** access to those files.

Options

rbootd supports the following options:

-a Append to the **rbootd** log file. By default, starting up **rbootd** truncates the log file.

-l *loglevel* Set the amount of information that will be logged in the log file. **rbootd** supports the following logging levels:

0	Log only **rbootd** startup and termination messages.
1	Log all errors. This is the default logging level.
2	Log rejected boot requests from machines not found in **/etc/bootptab** or **/usr/dtcmgr/map802**.
3	Log all boot requests.

-L *logfile* Specify an alternate file that **rbootd** should use to log status and error messages.

-t *minutes* Grace period before removing inactive temporary files. Meaningful only in the **tftp**-remote configuration. Default is 10 minutes.

landevs Specify the *only* devices that **rbootd** should use to listen for boot requests. The default is all LAN devices. The device names must be of the form **/dev/lan***

New Functionality

Beginning with HP-UX 10.0 **rbootd** has the following behavior:

- **bootpd/bootptab Dependency**:

  rbootd now relies on *bootpd*(1M) to verify the identity of cluster clients and locate the bootable images (from **/etc/bootptab**). RMP clients are thus administered in exactly the same way as new BOOTP clients. The old methods for administering RMP clients (**/etc/clusterconf**, context-dependent files, **/usr/boot/***) are obsolete and no longer work.

  See *bootpd*(1M) and *sam*(1M) for details on configuring cluster clients.

  It is necessary to have the **bootpd** daemon running on the same machine as the **rbootd** daemon.

- **Auto-Discovery**:

  To aid the system administrator, **rbootd** now discovers working ethernet interfaces at startup time and monitors them for boot requests. Alternatively, the system administrator may put a list of up to ten ethernet devices on the command line. Putting device names on the command line means "monitor these devices ONLY". If device names are included on the command line, they must be ethernet interfaces (not X.25, token-ring, etc) and they must be up and running at the time **rbootd** is started. See *lanscan*(1M) and *ifconfig*(1M) to determine the state of system devices. Attempting to have **rbootd** monitor non-ethernet devices will not succeed. The device names must always be of the form **/dev/lan***.

- **Multiple LAN Coverage**:

  rbootd can monitor up to 10 lan devices (depending on hardware) and can boot clients from all of them. Clients are still restricted to booting from their own builtin lan devices.

- **Gateway Booting**:

  RMP clients can now be booted from servers that are not on the same subnet as the client. The RMP boot requests and replies cannot cross gateways, but the repackaged BOOTP requests and replies can. The BOOTP requests and replies are relayed across gateways by **bootpd**. This is known as the remote configuration.

  rbootd uses the NFS or **tftp** mechanism to transfer the necessary files from the remote server to the **rbootd** machine, and then transfers the bootable images to the client in a succession of RMP packets. Thus the remote server must make the necessary files accessible by NFS or **tftp**.

  In the remote-**tftp** case, the boot files are temporarily stored in **/var/rbootd/C0809***, and are removed after a period of inactivity, controlled by the **-t** option. The default is 10 minutes.

- **S800 Servers**:

  S800 machines can now be used as cluster servers, booting s700 clients and DTCs. S800 machines are not supported as cluster clients.

- **Network Install**:

  rbootd now forwards install requests to *instl_bootd*(1M). If there is no appropriate response, **rbootd** will deny the request.

- **Performance Recommendations**:

  Boot from a local server for the fastest boot times. Run the **rbootd** daemon and the **bootpd** server daemon on the same machine, and avoid transferring the boot files by NFS or **tftp**. This is strongly recommended.

  If booting from remote **bootpd** servers (across gateways), use NFS mounts to make the boot files available to the **rbootd** server. See *mount*(1M) for more information. The system administrator can configure local and remote diskless clients in any mix, but it is strongly recommended that the number of remote diskless clients be minimized.

  If booting from remote servers using the **tftp** method, there must also be temporary file space available on the **rbootd** server machine. Generally 6-8 MBytes per diskless client must be available under **/var**, but this number could be larger when booting customized kernels. These temporary files are removed automatically after some period of inactivity, controlled by the **-t** option. The default is 10 minutes.

- **RMP/BOOTP**:

  The RMP clients are the older s700 workstations and all DTCs: workstations: 705, 710, 715, 720, 725, 730, 735, 750, 755

  The BOOTP clients are the s712 and future workstations.

WARNINGS

It is necessary to stop **rbootd** before running **bootpquery** because they use the same reserved port (67/udp).

AUTHOR

rbootd was developed by HP.

FILES

/var/adm/rbootd.log	Default rbootd log file.
/etc/boottab	Bootstrap configuration file.
/etc/opt/dtcmgr/map802	DTC/9000 configuration file.
/opt/dtcmgr/map802	DTC/9000 configuration file.
/var/rbootd/C0809*	Temporary boot files.

Obsoleted Files

`/etc/clusterconf`
`/usr/boot/*`

SEE ALSO

bootpd(1M), instl_bootd(1M), tftpd(1M), mount(1M), sam(1M), dcnodes(1), dtcconfig(1M), dtcnmd(1M), dtcnmp(1M).

NAME

rc - general purpose sequencer invoked upon entering new run level

SYNOPSIS

/sbin/rc

DESCRIPTION

The **rc** shell script is the general sequencer invoked upon entering a new run level via the **init** *N* command (where *N* equals 0-6). The script **/sbin/rc** is typically invoked by the corresponding entry in the file **/etc/inittab** as follows:

```
sqnc:123456:wait:/sbin/rc </dev/console >/dev/console 2>&1
```

/sbin/rc is the startup and shutdown sequencer script. There is only one sequencer script and it handles all of the sequencer directories. This script sequences the scripts in the appropriate sequencer directories in alphabetical order as defined by the shell and invokes them as either startup or kill scripts.

If a transition from a lower to a higher run level (i.e., *init* state) occurs, the start scripts for the new run level and all intermediate levels between the old and new level are executed. If a transition from a higher to a lower run level occurs, the kill scripts for the new run level and all intermediate levels between the old and new level are executed.

If a start script link (e.g., **/sbin/rc***N***.d/S123test**) in sequencer *N* has a stop action, the corresponding kill script should be placed in sequencer *N***-1** (e.g., **/sbin/rc***N***-1.d/K200test**). Actions started in level *N* should be stopped in level *N***-1**. This way, a system shutdown (e.g., transition from level 3 directly to level 0) will result in all subsystems being stopped.

Start and Kill Scripts

In many cases, a startup script will have both a start and a kill action. For example, the *inetd* script starts the Internet daemon in the start case, and kills that process in the stop case. Instead of two separate scripts, only one exists, which accepts both the **start** and **stop** arguments and executes the correct code. In some cases, only a start action will be applicable. If this is the case, and if the **stop** action is specified, the script should produce a usage message and exit with an error. In general, scripts should look at their arguments and produce error messages if bad arguments are present. When a script executes properly, it must exit with a return value of zero. If an error condition exists, the return value must be nonzero.

Naming Conventions

The startup and shutdown scripts (referred to as startup scripts hereafter) exist in the **/sbin/init.d** directory, named after the subsystem they control. For example, the **/sbin/init.d/cron** script controls starting up the **cron** daemon. The contents of sequencer directories consist of symbolic links to startup scripts in **/sbin/init.d**. These symbolic links must follow a strict naming convention, as noted in the various fields of this example:

```
/sbin/rc2.d/S060cron
```

where the fields are defined as follows:

rc2.d The sequencer directory is numbered to reflect the run level for which its contents will be executed. In this case, start scripts in this directory will be executed upon entering run level 2 from run level 1, and kill scripts will be executed upon entering run level 2 from run level 3.

S The first character of a sequencer link name determines whether the script is executed as a start script (if the character is **S**), or as a kill script (if the character is **K**).

060 A three digit number is used for sequencing scripts within the sequencer directory. Scripts are executed by type (start or kill) in alphabetical order as defined by the shell. Although it is not recommended, two scripts may share the same sequence number.

cron The name of the startup script follows the sequence number. The startup script name must be the same name as the script to which this sequencer entry is linked. In this example, the link points to **/sbin/init.d/cron**.

Note that short file name systems require file names of 14 or less characters. This means that the fourth field is limited to 10 or fewer characters.

Scripts are executed in alphabetical order. The entire file name of the script is used for alphabetical ordering purposes.

When ordering start and kill script links, note that subsystems started in any given order should be stopped in the reverse order to eliminate any dependencies between subsystems. This means that kill scripts will generally not have the same numbers as their start script counterparts. For example, if two subsystems must be started in a given order due to dependencies (e.g., **S111house** followed by **S222uses_house**), the kill counterparts to these scripts must be numbered so that the subsystems are stopped in the opposite order in which they were started (e.g., **K555uses_house** followed by **K777house**).

Also keep in mind that kill scripts for a start script in directory **/sbin/rc***N***.d** will reside in **/sbin/rc(***N***-1).d**. For example, **/sbin/rc3.d/S123homer** and **/sbin/rc2.d/K654homer** might be start/kill counterparts.

Arguments

The startup/shutdown scripts should be able to recognize the following four arguments (where applicable):

start The **start** argument is passed to scripts whose names start with **S**. Upon receiving the **start** argument, the script should perform its start actions.

stop The **stop** argument is passed to scripts whose names start with **K**. Upon receiving the **stop** argument, the script should perform its stop actions.

start_msg The **start_msg** argument is passed to scripts whose names start with **S** so that the script can report back a short message indicating what the start action will do. For instance, when the **lp** spooler script is invoked with a **start_msg** argument, it echoes

```
Starting the LP subsystem
```

This string is used by the startup routines. Scripts given just the **start_msg** argument will only print a message and not perform any actions.

stop_msg The **stop_msg** argument is passed to scripts whose names start with **K** so that the script can report back a short message indicating what the stop action will do. For instance, when the **lp** spooler script is invoked with a **stop_msg** argument, it echoes

```
Stopping the LP subsystem
```

This string is used by the shutdown checklist. Scripts given just the **stop_msg** argument will only print a message and not perform any actions.

Script Output

To ensure proper reporting of startup events, startup scripts are required to comply with the following guidelines for script output.

- Status messages, such as

  ```
  starting house daemon
  ```

  must be directed to stdout. All error messages must be directed to stderr.

- Script output, both stdout and stderr, is redirected to log file **/etc/rc.log**, unless the startup checklist mode is set to the raw mode. In this case, all output goes to the console. All error messages should be echoed to stdout or stderr.

- Startup scripts are not allowed to send messages directly to the console, or to start any daemons that immediately write to the console. This restriction exists because these scripts are now started by the **/sbin/rc** checklist wrapper. All script output should go to either stdout or stderr, and thus be captured in a log file. Any console output will be garbled.

RETURN VALUE

The return values for startup scripts are as follows:

0	Script exited without error.
1	Script encountered errors.
2	Script was skipped due to overriding control variables from **/etc/rc.config.d** files, or for other reasons, and did not actually do anything.
4	Script exited without error and started a process in background mode.

r

SEE ALSO

init(1M), shutdown(1M), inittab(4), rc.config(4).

NAME

rcancel - remove requests from a remote line printer spooling queue

SYNOPSIS

/usr/sbin/rcancel [*id* ...] [*printer*] [**-a**] [**-e**] [**-u** *user*]

DESCRIPTION

The **rcancel** command removes a request, or requests, from the spool queue of a remote printer. **rcancel** is invoked by the **cancel** command (see *cancel*(1)).

At least one *id* or the name of a *printer* must be specified.

This command is intended to be used only by the spool system in response to the **cancel** command (see *lp*(1)), and should not be invoked directly.

Options

The **rcancel** command recognizes the following options:

id Specifying a request ID (as returned by **lp** (see *lp*(1)) cancels the associated request (if the request is owned by the user), even if it is currently printing.

printer Name of the printer (for a complete list, use **lpstat** (see *lpstat*(1)). Specifying a *printer* cancels the request which is currently printing on that printer, if the request is owned by the user. If the **-a**, **-e**, or the **-u** option is specified, this option only specifies the printer on which to perform the cancel operation.

-a Remove all requests owned by the user on the specified printer (see *printer*). The owner is determined by the user's login name and host name on the machine where the **lp** command was invoked.

-e Empty the spool queue of all requests for the specified *printer*. This form of invoking **rcancel** is useful only to users with appropriate privileges.

-u *user* Remove any requests queued belonging to that user (or users). This form of invoking **rcancel** is available only to users with appropriate privileges.

AUTHOR

rcancel was developed by the University of California, Berkeley, and HP.

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

enable(1), lp(1), lpstat(1), accept(1M), lpadmin(1M), lpsched(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

NAME

rdpd - router discovery protocol daemon (OBSOLESCENT)

SYNOPSIS

rdpd [**-r** | **-t** | **-v**]

DESCRIPTION

rdpd, the router discover protocol daemon, implements the host portion of the router discovery protocol (see *SEE ALSO*). More specifically **rdpd**:

- solicits router advertisements when it is first started so as to populate the kernel table as soon as possible.
- listens on all ethernet interfaces (that are up) for ICMP router advertisement datagrams.
- adds a default router to the kernel table based on whether the router is a neighbor and has the highest preference among all advertisements received.
- ages the default router entry applied to the kernel table based on the lifetime value found in the advertisement message.

rdpd can be started during boot-time initialization. To do so, see **/etc/rc.config.d/netconf**. (But see *WARNINGS* below.)

Options

rdpd supports the following options:

-r Display the version of **rdpd**.

-t Enable tracing of the following events:

- setting of expiration timer for advertised entry.
- expiration of a router advertisement entry (only the active entry has a timer running).
- add/update of an advertised router to the kernel.
- removal from kernel table of an advertised router.
- reception of a router advertisement from the link.
- transmission of a router solicitation message.
- failure while attempting to transmit a solicitation.

-v Enable verbose tracing, which in addition to the above, traces:

- contents of the router advertisement message received.
- contents of *rdpd* internal statics which includes:
    1. total number of **icmp** messages received,
    2. total number advertisements received,
    3. total number of advertisements with invalid number of addresses field,
    4. total number of advertisements with invalid address size field,
    5. total number of advertisements with invalid message lengths,
    6. total number of advertisements with invalid lifetime fields,
    7. total number of **icmp** messages with number of bytes received <> header length field.

r

LIMITATIONS

1. The maximum number of default routes retained is 10. Only one of which is applied to the kernel routing tables (the one with the highest preference). In the event that the advertised router with the highest preference expires the retained advertised router list will be searched for the highest preference, still current entry and that entry will be applied to the kernel table. This allows for quick recovery from aged advertisements.
2. **rdpd** only becomes aware of link state changes when either a new Router Advertisement message is received or a timer pops to age a currently active default router added by **rdpd**. This may cause a delay between an interface state change (e.g., **ifconfig** down) and any necessary change to the kernel routing table.
3. The "all hosts on subnet" broadcast address is used for sending solicitations instead of either the all-routers multicast or limited-broadcast IP addresses.
4. The limited-broadcast address for inbound Advertisements is assumed.

5. Default routers added via the **route** command can be altered due to Router Advertisements for the same router.
6. Adding default routes via the **route** command can cause unpredictable results and should be avoided.

OBSOLESCENCE

The functionality of **rdpd** has been subsumed in **gated**. See the **routerdiscovery** statements described in *gated.conf*(4). Consequently, **rdpd** may be obsoleted in a future release of HP-UX.

WARNINGS

rdpd should not be used if **routerdiscovery client** is enabled when running **gated**.

AUTHOR

rdpd was developed by HP.

SEE ALSO

gated(1m), gated.conf(4).

[1] Deering, S., "ICMP Router Discovery Messages", RFC 1256

NAME

reboot - reboot the system

SYNOPSIS

/usr/sbin/reboot [**-h**|**-r**] [**-n**|**-s**] [**-q**] [**-t** *time*] [**-m** *message*]

DESCRIPTION

The **reboot** command terminates all currently executing processes except those essential to the system, then halts or reboots the system. When invoked without arguments, **reboot** syncs all disks before rebooting the system.

Options

The **reboot** command recognizes the following options:

-h	Shut down the system and halt.
-r	Shut down the system and reboot automatically (default).
-n	Do not sync the file systems before shutdown.
-s	Sync the file systems before shutdown; for file systems that were cleanly mounted, modify the **fs_clean** flag from **FS_OK** to **FS_CLEAN** (default).
-q	Quick and quiet. Suppress broadcast of warning messages, terminate processes by brute force (with **SIGKILL**) and immediately call **reboot** with arguments as indicated by the other options (see *reboot*(2)). No logging is performed. The **-t** and **-m** options are ignored with this option.
-t *time*	Specify what time **reboot** will bring the system down. *time* can be the word **now** (indicating immediate shutdown) or a future time in one of two formats: **+***number* and *hour***:***min*. The first form brings the system down in *number* minutes; the second brings the system down at the time of day indicated (based on a 24-hour clock).
-m *message*	Display *message* at the terminals of all users on the system at decreasing intervals as reboot *time* approaches. The *message* must not contain any embedded double quotes. options cannot be used together.

At shutdown time a message is written in the file

/etc/shutdownlog

(if it exists), containing the time of shutdown, who ran **reboot**, and the reason.

Only users with appropriate privileges can execute the **shutdown** command.

AUTHOR

reboot was developed by HP and the University of California, Berkeley.

FILES

/etc/shutdownlog	Shutdown log

SEE ALSO

reboot(2).

NAME

remshd - remote shell server

SYNOPSIS

/usr/lbin/remshd [**-ln**]

DESCRIPTION

The **remshd** command is the server for the **rcp**, **rdist** and **remsh** commands, and the **rcmd()** function (see *rcp*(1), *rdist*(1), *remsh*(1), and *rcmd*(3N)). The server provides remote execution facilities with authentication based on privileged port numbers.

The **inetd** daemon calls **remshd** when a service request is received at the port indicated for the **shell** (or **cmd**) service specified in **/etc/services** (see *inetd*(1M) and *services*(4)). When called, **inetd** creates a connection to the service on the client's host. To run **remshd**, the following line should be present in the **/etc/inetd.conf** file:

```
shell  stream  tcp  nowait  root  /usr/lbin/remshd  remshd
```

See *inetd.conf*(4) for more information.

Options

remshd recognizes the following options.

-l Disallow authentication based on the user's **.rhosts** file unless the user is a superuser.

-n Disable transport-level keep-alive messages. Otherwise, the messages are enabled. The keep-alive messages allow sessions to be timed out if the client crashes or becomes unreachable.

Operation

When **remshd** receives a service request, it responds with the following protocol:

1. The server checks the client's source port. If the port is not in the range 512 through 1023, the server aborts the connection.
2. The server reads characters from the connection up to a null (**\0**) byte. It interprets the resulting string as an ASCII number, base 10.
3. If the number is non-zero, it is interpreted as the port number of a secondary stream to be used for standard error. A second connection is then created to the specified port on the client's host. (The source port of this second connection must be also in the range 512 through 1023.) If the first character sent is a null (**\0**), no secondary connection is made, and the standard error from the command is sent to the primary stream. If the secondary connection has been made, **remshd** interprets bytes it receives on that socket as signal numbers and passes them to the command as signals. See *signal*(2).
4. The server checks the client's source address and requests the corresponding host name (see *named*(1M), *gethostbyaddr*(3N), and *hosts*(4)). If it cannot determine the hostname, it uses the dot-notation representation of the host address.
5. The server reads the client's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.
6. The server reads the server's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.
7. The server reads a command to be passed to the shell from the first connection. The command length is limited by the maximum size of the system's argument list.
8. **remshd** then validates the user as follows (all actions take place on the host **remshd** runs on):
   a. It looks up the user account name (retrieved in step 6) in the password file. If it finds it, it performs a **chdir()** to either the user's home directory, if there is one, or to "/."
   b. If either the lookup or **chdir()** fails, the connection is terminated (see *chdir*(2)).
   c. The connection is also terminated if
      - the account accessed is administratively locked;
      - the account accessed is protected by a password and, either the password expired or the account on the client's host is not equivalent to the account accessed;

r

- **remshd** runs on a secure system and the account accessed is not protected by a password.

  For more information on equivalent accounts, see *hosts.equiv*(4).

9. A null byte is returned on the primary connection and the command line is passed to the normal login shell of the user with that shell's **-c** option. The shell inherits the network connections established by **remshd** and assumes the normal user and group permissions of the user.

   remshd uses the following path when executing the specified command:

   ```
   /usr/bin:/usr/ccs/bin:/usr/bin/X11:
   ```

10. If a secondary socket has been set up, **remshd** normally exits when command standard error and secondary socket standard error have both been closed. If no secondary socket was set up, **remshd** has called an *exec*(2) function, launched the command process, and is no longer present.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with standard error after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps before the command execution).

Malformed from address

The first socket connection does not use a reserved port or the client's host address is not an Internet address.

Can't get stderr port

Unable to complete the connection of the secondary socket used for error communication.

Second port not reserved

The secondary socket connection does not use a reserved port.

Locuser too long

The name of the user account on the client's host is longer than 16 characters.

Remuser too long

The name of the user on the server's host is longer than 16 characters.

Command too long

The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect

No password file entry existed for the user name on the server's host, or the authentication procedure described above in step 8 failed.

No remote directory

The **chdir** command to the home directory or "/" on the server's host failed.

Can't make pipe

The pipe needed for the standard error output wasn't created.

No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

system call: *message*

Error in executing the named system call. The message specifies the cause of the failure.

shellname: ...

The user's login shell could not be started. This message is returned on the connection associated with the standard error, and is not preceded by a leading byte with a value of 1. Other messages can be returned by the remote command when it executes.

WARNINGS

The "privileged port" authentication procedure used here assumes the integrity of each host and the connecting medium. This is insecure, but is useful in an "open" environment.

remshd ignores **SIGHUP**, **SIGINT**, **SIGQUIT**, and **SIGTERM**, so these signal numbers can safely be sent to remote commands via the secondary socket provided by **remshd**. Other signal numbers may cause **remshd** to kill itself.

AUTHOR

remshd was developed by the University of California, Berkeley.

FILES

$HOME/.rhosts	User's private equivalence list
/etc/hosts.equiv	List of equivalent hosts

SEE ALSO

remsh(1), inetd(1M), named(1M), rcmd(3N), hosts(4), hosts.equiv(4), inetd.conf(4), inetd.sec(4), services(4).

NAME

remshd - remote shell server

SYNOPSIS

/usr/lbin/remshd [**-ln**]

In Kerberos V5 Network Authentication environments:

/usr/lbin/remshd [**-clnKkRr**]

DESCRIPTION

The **remshd** command is the server for the **rcp**, **rdist** and **remsh** commands, and the **rcmd()** function (see *rcp*(1), *rdist*(1), *remsh*(1), and *rcmd*(3N)).

remshd allows two kinds of authentication methods:

1. Authentication based on privileged port numbers where the client's source port must be in the range 512 through 1023. In this case **remshd** assumes it is operating in normal or non-secure environment.
2. Authentication based on Kerberos V5. In this case **remshd** assumes it is operating in a Kerberos V5 Network Authentication, i.e., secure environment.

The *inetd* daemon invokes **remshd** if a service request is received at ports indicated by **shell** or **kshell** services specified in **/etc/services** (see *inetd*(1M) and *services*(4)). Service requests arriving at the **kshell** port assume a secure environment and expect Kerberos authentication to take place.

To start **remshd** from the *inetd* daemon in a non-secure environment, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
shell  stream  tcp  nowait  root  /usr/lbin/remshd  remshd
```

In a secure environment, **/etc/inetd.conf** must contain an entry:

```
kshell  stream  tcp  nowait  root  /usr/lbin/remshd  remshd -K
```

See *inetd.conf*(4) for more information.

To prevent non-secure access, the entry for **shell** should be commented out in **/etc/inetd.conf**. Any non-Kerberos access will be denied since the entry for the port indicated by **shell** has now been removed or commented out. In a such a situation, a generic error message,

rcmd: connect <hostname> : Connection refused

is displayed. See DIAGNOSTICS for more details. Note: by commenting out the entry for the port, access by other clients such as **rdist** will also be prevented.

Options

remshd recognizes the following options.

-c Ignore checksum verification. This option is used to achieve interoperability between clients and servers using different checksum calculation methods. For example, the checksum calculation in a application developed with Kerberos V5 Beta 4 API is different from the calculation in a Kerberos V5-1.0 application.

-l Disallow authentication based on the user's **.rhosts** file unless the user is a superuser.

-n Disable transport-level keep-alive messages. Otherwise, the messages are enabled. The keep-alive messages allow sessions to be timed out if the client crashes or becomes unreachable.

In a secure environment, **remshd** will recognize the following additional options:

-K Authorization based on Kerberos V5 must succeed or access will be rejected. (see *sis*(5) for details on authorization).

-R Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts must succeed. For more information on equivalent accounts, see *hosts.equiv(4).*

-r Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts (see *hosts.equiv*(4)).

r

2. Authorization based on Kerberos V5.

-k Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authorization based on Kerberos V5.
2. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts.

Note: The **-k** option is ignored when used with **-K**, and the **-r** option is ignored when used with **-R**. Also, if no options are specified, the default option is **-K**.

Operation

When **remshd** receives a service request, it responds with the following protocol:

1. The server checks the client's source port. If the port is not a privileged port, i.e., in the range 512 through 1023, and **remshd** is operating in a non-secure environment, the connection is terminated. In a secure environment, the action taken depends on the command line options:

   -R The source port must be a privileged port otherwise the connection is terminated.

   -r If the source port is not a privileged port then authorization based on Kerberos must succeed or the connection is terminated.

   -k The source port must be a privileged port if Kerberos authorization fails.

   -K No action is taken.

2. The server reads characters from the connection up to a null (**\0**) byte. It interprets the resulting string as an ASCII number, base 10.
3. If the number is non-zero, it is interpreted as the port number of a secondary stream to be used for standard error. A second connection is then created to the specified port on the client's host. (The source port of this second connection will also be checked as specified in item 1.) If the first character sent is a null (**\0**), no secondary connection is made, and the standard error from the command is sent to the primary stream. If the secondary connection has been made, **remshd** interprets bytes it receives on that socket as signal numbers and passes them to the command as signals. See *signal*(2).
4. The server checks the client's source address and requests the corresponding host name (see *named*(1M), *gethostbyaddr*(3N), and *hosts*(4)). If it cannot determine the hostname, it uses the dot-notation representation of the host address.
5. In a secure environment, **remshd** performs authentication based on Kerberos V5. See *sis*(5) for details.
6. The server reads the client's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.
7. The server reads the server's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.
8. The server reads a command to be passed to the shell from the first connection. The command length is limited by the maximum size of the system's argument list.
9. **remshd** then validates the user as follows (all actions take place on the host **remshd** runs on):

   a. It looks up the user account name (retrieved in step 6) in the password file. If it finds it, it performs a **chdir**() to either the user's home directory, if there is one, or to "/."

   b. If either the lookup or **chdir**() fails, the connection is terminated (see *chdir*(2)).

   c. The connection is also terminated if

   - the account accessed is administratively locked;
   - in a non-secure environment, the account accessed is protected by a password and, either the password expired or the account on the client's host is not equivalent to the account accessed.
   - in a secure environment, the command line options decide whether connection is to be terminated.

- **-K** if Kerberos authorization does not succeed the connection is terminated (see *sis*(5) for details on authorization).
- **-R** if the client's host is not equivalent to the account accessed, the connection is terminated.
- **-r** if the account is not equivalent to the account accessed, then Kerberos authorization has to succeed or the connection is terminated.
- **-k** if Kerberos authorization fails, then the account has to be equivalent or the connection is terminated. For more information on equivalent accounts, see *hosts.equiv*(4).

10. A null byte is returned on the primary connection and the command line is passed to the normal login shell of the user with that shell's **-c** option. The shell inherits the network connections established by **remshd** and assumes the normal user and group permissions of the user.

    remshd uses the following path when executing the specified command:

    ```
    /usr/bin:/usr/ccs/bin:/usr/bin/X11:
    ```

11. If a secondary socket has been set up, **remshd** normally exits when command standard error and secondary socket standard error have both been closed. If no secondary socket was set up, **remshd** has called an *exec*(2) function, launched the command process, and is no longer present.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with standard error after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps before the command execution).

Malformed from address

The first socket connection does not use a reserved port or the client's host address is not an Internet address.

Can't get stderr port

Unable to complete the connection of the secondary socket used for error communication.

Second port not reserved

The secondary socket connection does not use a reserved port.

Locuser too long

The name of the user account on the client's host is longer than 16 characters.

Remuser too long

The name of the user on the server's host is longer than 16 characters.

Command too long

The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect

No password file entry existed for the user name on the server's host, or the authentication procedure described above in step 8 failed.

No remote directory

The **chdir** command to the home directory or "/" on the server's host failed.

Can't make pipe

The pipe needed for the standard error output wasn't created.

No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

system call **:** *message*

Error in executing the named system call. The message specifies the cause of the failure.

shellname: ...

The user's login shell could not be started. This message is returned on the connection associated with the standard error, and is not preceded by a leading byte with a value of 1. Other messages can be returned by the remote command when it executes.

rcmd: connect : <hostname>: Connection refused.

This generic message could be due to a number of reasons. One of the reasons could be because the entry for *shell* service is not present in **/etc/inetd.conf**. This entry may have been removed or commented out to prevent non-secure access.

Kerberos specific errors are listed in *sis*(5).

WARNINGS

The integrity of each host and the connecting medium is assumed if the "privileged port" authentication procedure is used in a non-secure environment or if the command line options **-R** or **-r** are used in a secure environment. Although both these methods provide insecure access, they are useful in an "open" environment.

Note also that all information, including any passwords, are passed unencrypted between the two hosts when **remshd** is invoked in a non-secure environment.

remshd ignores **SIGHUP**, **SIGINT**, **SIGQUIT**, and **SIGTERM**, so these signal numbers can safely be sent to remote commands via the secondary socket provided by **remshd**. Other signal numbers may cause **remshd** to kill itself.

AUTHOR

remshd was developed by the University of California, Berkeley.

FILES

$HOME/.rhosts	User's private equivalence list
/etc/hosts.equiv	List of equivalent hosts

SEE ALSO

remsh(1), inetd(1M), named(1M), rcmd(3N), hosts(4), hosts.equiv(4), inetd.conf(4), inetd.sec(4), services(4), sis(5).

r

NAME

renice - alter priority of running processes

SYNOPSIS

renice [**-n** *newoffset*] [**-g**|**-p**|**-u**] *id* ...

DESCRIPTION

The **renice** command alters the system nice value (used in the system scheduling priority) of one or more running processes specified by *id* .... The new system nice value is set to 20 + *newoffset*, and is limited to the range 0 to 39. However if the **UNIX95** environment variable is set, the new system nice value is set to current nice value + *newoffset*. Processes with lower system nice values run at higher system priorities than processes with higher system nice values. The **-l** option of the **ps** command shows the current priority (**PRI**) and nice value (**NI**) for processes. See also *nice*(1).

To reduce the system nice value of a process, or to set it to a value less than 20 (with a negative *newoffset*), a user must have appropriate privileges. Otherwise, users cannot decrease the system nice value of a process and can only increase it within the range 20 to 39, to prevent overriding any current administrative restrictions.

To alter the system nice value of another user's process, a user must have appropriate privileges. Otherwise, users can only affect processes that they own.

Options

renice recognizes the following options. If no **-g**, **-p**, or **-u** option is specified, the default is **-p**.

-g *id* ... Interpret each *id* as a process group ID. All processes in each process group have their system nice value altered. Only users with appropriate privileges can use this option.

-n *newoffset* Change the system nice value of each affected process to 20 + *newoffset*. If the **UNIX95** environment variable is set, the system nice value of each affected process is changed to current nice value + *newoffset*.

If *newoffset* is negative, the system nice value is set to 20 minus the absolute value of *newoffset*. If the **UNIX95** environment variable is set and the *newoffset* is negative, the system nice value is set to current nice value minus the absolute value of *newoffset*. Only users with appropriate privileges can reduce the system nice value or set it to less than 20. If this option is omitted, *newoffset* defaults to 10.

-p *id* ... Interpret each *id* as a process ID. This is the default.

Note: *id* is a process ID as reported by the **ps** command, not a job number (e.g., **%1**), as used by some shells.

-u *id* ... Interpret each *id* as a user name or user ID number. All processes owned by each specified user have their system nice values altered. Only users with appropriate privileges can use this option for user names and IDs other than their own.

RETURN VALUES

renice returns a 0 when successful, and a non-zero value when unsuccessful.

EXTERNAL INFLUENCES

Single-byte character code sets are supported.

DIAGNOSTICS

renice reports the old and new *newoffset* values (system nice value – 20) of the affected processes if the operation requested completes successfully. Otherwise, an error message is displayed to indicate the reason for failure.

However, if the **UNIX95** envionment variable is set, no reporting is done unless the command fails.

EXAMPLES

Use **renice** default values to decrease the priority of process **923**. The *id* type defaults to **-p**, and *newoffset* defaults to **10**, setting the process to a system nice value of 30.

```
renice 923
```

Change the system nice value for all processes owned by user **john** and user **123** to 33 (*newoffset*=13). (Affecting other users processes requires appropriate privileges.)

r

```
renice -n 13 -u john 123
```

Change the system nice value of all processes in process group 20 to **10**. (Lowering the system nice value of a process group requires appropriate privileges.)

```
renice -n -10 -g 20
```

WARNINGS

Users who do not have appropriate privileges cannot reduce the system nice values of their own processes, even if they increased them in the first place.

FILES

/etc/passwd	Maps user names to user ID's

SEE ALSO

nice(1), ps(1), getpriority(2), nice(2).

STANDARDS CONFORMANCE

renice: XPG4

NAME

repquota - summarize file system quotas

SYNOPSIS

/usr/sbin/repquota [**-v**] *filesystem* ...

/usr/sbin/repquota [**-v**] **-a**

DESCRIPTION

The **repquota** command prints a summary of disk usage and quotas for each specified *filesystem*.

filesystem is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

For each user, the current number of files and amount of space (in Kbytes) is printed, along with any quotas created with **edquota** (see *edquota*(1M)).

Options

repquota recognizes the following options:

-a Report on all appropriate file systems in **/etc/fstab**.

-v Report all quotas, even if there is no usage.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **repquota** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, and HP.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
*directory***/quotas**	Quota statistics static storage for the file system, where *directory* is the root of the file system as interpreted by **mount** (see *mount*(1M)).

SEE ALSO

edquota(1M), mount(1M), quota(5).

NAME

restore, rrestore - restore file system incrementally, local or across network

SYNOPSIS

/usr/sbin/restore *key* [*name* ...]

/usr/sbin/rrestore *key* [*name* ...]

DESCRIPTION

The **restore** and **rrestore** commands read tapes previously dumped by the **dump** or **rdump** command (see *dump*(1M) and *rdump*(1M)). Actions taken are controlled by the *key* argument where *key* is a string of characters containing not more than one function letter and possibly one or more function modifiers. One or more *name* arguments, if present, are file or directory names specifying the files that are to be restored. Unless the **h** modifier is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

Function Portion of *key*

The function portion of the key is specified by one of the following letters:

r Read the tape and load into the current directory. **r** should be used only after careful consideration, and only to restore a complete dump tape onto a clear file system, or to restore an incremental dump tape after a full level zero restore. Thus,

```
/usr/sbin/newfs -F hfs /dev/rdsk/c0t0d0
/usr/sbin/mount /dev/dsk/c0t0d0 /mnt
cd /mnt
restore r
```

is a typical sequence to restore a complete dump. Another **restore** or **rrestore** can then be performed to restore an incremental dump on top of this. Note that **restore** and **rrestore** leave a file **restoresymtab** in the root directory of the file system to pass information between incremental restore passes. This file should be removed when the last incremental tape has been restored. A **dump** or **rdump** followed by a **newfs** and a **restore** or **rrestore** is used to change the size of a file system (see *newfs*(1M)).

R **restore** and **rrestore** request a particular tape of a multivolume set on which to restart a full restore (see **r** above). This provides a means for interrupting and restarting **restore** and **rrestore**.

x Extract the named files from the tape. If the named file matches a directory whose contents had been written onto the tape, and the **h** modifier is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the root directory is extracted, which results in the entire contents of the tape being extracted, unless **h** has been specified.

t Names of the specified files are listed if they occur on the tape. If no file argument is given, the root directory is listed, which results in the entire content of the tape being listed, unless **h** has been specified.

s The next argument to **restore** is used as the dump file number to recover. This is useful if there is more than one dump file on a tape.

i This mode allows interactive restoration of files from a dump tape. After reading in the directory information from the tape, **restore** and **rrestore** provide a shell-like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

add [*arg*] The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, it and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). File names on the extraction list are displayed with a leading * when listed by **ls**.

cd [*arg*] Change the current working directory to the specified argument.

delete [*arg*] The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, it and all its descendents are

r

deleted from the extraction list (unless **h** is specified on the command line). The most expedient way to extract files from a directory is to add the directory to the extraction list, then delete unnecessary files.

extract All files named on the extraction list are extracted from the dump tape. **restore** and **rrestore** ask which volume the user wants to mount. The fastest way to extract a few files is to start with the last volume, then work toward the first volume.

help List a summary of the available commands.

ls [*arg*] List the current or specified directory. Entries that are directories are displayed with a trailing **/**. Entries marked for extraction are displayed with a leading *****. If the verbose key is set, the inode number of each entry is also listed.

pwd Print the full path name of the current working directory.

quit **restore** and **rrestore** immediately exit, even if the extraction list is not empty.

set-modes Set the owner, modes, and times of all directories that are added to the extraction list. Nothing is extracted from the tape. This setting is useful for cleaning up after a restore aborts prematurely.

verbose The sense of the **v** modifier is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes **restore** and **rrestore** to print out information about each file as it is extracted.

Function Modifiers

The following function modifier characters can be used in addition to the letter that selects the function desired:

b Specify the block size of the tape in kilobytes. If the **-b** option is not specified, **restore** and **rrestore** try to determine the tape block size dynamically.

f Specify the name of the archive instead of **/dev/rmt/0m**. If the name of the file is **-**, **restore** reads from standard input. Thus, **dump** and **restore** can be used in a pipeline to dump and restore a file system with the command

```
dump 0f - /usr | (cd /mnt; restore xf -)
```

When using **rrestore**, this key should be specified, and the next argument supplied should be of the form *machine***:***device*.

h Extract the actual directory, rather than the files to which it refers. This prevents hierarchical restoration of complete subtrees from the tape, rather than the files to which it refers.

m Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted and one wants to avoid regenerating the complete path name to the file.

v Type the name of each file **restore** and **rrestore** treat, preceded by its file type. Normally **restore** and **rrestore** do their work silently; the **v** modifier specifies verbose output.

y Do not ask whether to abort the operation if **restore** and **rrestore** encounters a tape error. **restore** and **rrestore** attempt to skip over the bad tape block(s) and continue.

rrestore creates a server, either **/usr/sbin/rmt** or **/etc/rmt,** on the remote machine to access the tape device.

DIAGNOSTICS

restore and **rrestore** complain about bad key characters.

restore and **rrestore** complain if a read error is encountered. If the **y** modifier has been specified, or the user responds **y**, **restore** and **rrestore** attempt to continue the restore.

If the dump extends over more than one tape, **restore** and **rrestore** ask the user to change tapes. If the **x** or **i** function has been specified, **restore** and **rrestore** also ask which volume the user wants to mount. The fastest way to extract a few files is to start with the last volume and work towards the first volume.

There are numerous consistency checks that can be listed by **restore** and **rrestore**. Most checks are self-explanatory or can “never happen”. Here are some common errors:

filename: **not found on tape**
: The specified file name was listed in the tape directory but not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file *inumber*, **got** *inumber*
: A file not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

Incremental tape too low
: When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high
: When doing an incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level has been loaded.

Tape read error while restoring *filename*
Tape read error while skipping over inode *inumber*
Tape read error while trying to resynchronize
: A tape read error has occurred. If a file name is specified, the contents of the restored files are probably partially wrong. If restore is skipping an inode or is trying to resynchronize the tape, no extracted files are corrupted, although files may not be found on the tape.

Resync restore, skipped *num* **blocks**
: After a tape read error, **restore** and **rrestore** may have to resynchronize themselves. This message indicates the number of blocks skipped over.

WARNINGS

restore and **rrestore** can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level zero dump (see *dump*(1M)) must be done after a full restore. Since restore runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files are unchanged.

AUTHOR

restore and **rrestore** were developed by the University of California, Berkeley.

FILES

r

/dev/rmt/0m	Default tape drive.
/tmp/rstdr*	File containing directories on the tape.
/tmp/rstmd*	Owner, mode, and time stamps for directories.
./restoresymtab	Information passed between incremental restores.

SEE ALSO

dump(1M), mkfs(1M), mount(1M), newfs(1M), rmt(1M).

NAME

revck - check internal revision numbers of HP-UX files

SYNOPSIS

/usr/old/usr/bin/revck *ref_files*

DESCRIPTION

revck checks the internal revision numbers of lists of files against reference lists. Each *ref_file* must contain a list of absolute path names (each beginning with **/**) and *whatstrings* (revision information strings from **what** — see *what*(1)). Path names begin in column 1 of a line, and have a colon appended to them. Each path name is followed by zero or more lines of *whatstrings*, one per line, each indented by at least one tab (this is the same format in which **what** outputs its results).

For each path name, **revck** checks that the file exists, and that executing **what** on the current path name produces results identical to the *whatstrings* in the reference file. Only the first 1024 bytes of *whatstrings* are checked.

ref_files are usually the absolute path names of the *revlist* files shipped with HP-UX. Each HP-UX software product includes a file named **/system/***product***/revlist** (for example, **/system/97070A/revlist**). The *revlist* file for each product is a reference list for the ordinary files shipped with the product, plus any empty directories on which the product depends.

FILES

/system/*product***/revlist**	lists of HP-UX files and revision numbers

SEE ALSO

what(1).

DIAGNOSTICS

revck is silent except for reporting missing files or mismatches.

WARNINGS

revck produces unpredictable results if a *ref_file* is not in the right format.

r

NAME

rexd - RPC-based remote execution server

SYNOPSIS

/usr/sbin/rpc.rexd [**-l** *log_file*] [**-m** *mountdir*] [**-r**]

DESCRIPTION

rexd is the RPC server for remote command execution. A **rexd** is started by **inetd** when a remote execution request is received (see *inetd*(1M)). **rexd** exits when command execution has completed.

If the user ID (uid) in the remote execution request is assigned to a user on the server, **rexd** executes the command as that user. If no user on the server is assigned to the uid, **rexd** does not execute the command. The **-r** option and **inetd.sec** security file allow for better access control (see *inetd.sec*(4)).

For noninteractive commands, standard output and error file descriptors are connected to sockets. Interactive commands use pseudo terminals for standard input, output, and error (see *pty*(7)).

If the file system specified in the remote execution request is not already mounted on the server, **rexd** uses NFS to mount the file system for the duration of the command execution (see *nfs*(7)). **rexd** mounts file systems with the **nosuid** and **soft** options. For more details on mount options see *mount*(1M). If the server cannot mount the file system, an error message is returned to the client. By default, any mount points required by **rexd** are created below **/var/spool/rexd**. To change the default location, use the **-m** option.

Options

rexd recognizes the following options and command-line arguments:

-l *log_file* Log any diagnostic, warning, and error messages to *log_file*. If *log_file* exists, **rexd** appends messages to the file. If *log_file* does not exist, **rexd** creates it. Messages are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, host name, process ID and name of the function generating the error, and the error message. Note that different RPC services can share a single log file because enough information is included to uniquely identify each error.

-m *mountdir* Create temporary mount points below directory *mountdir*. By default, **rexd** creates temporary mount points below **/var/spool/rexd**. The directory *mountdir* should have read and execute permission for all users (mode 555). Otherwise, **rexd** denies execution for users that do not have read and execute permission.

-r Use increased security checking. When started with the **-r** option, **rexd** denies execution access to a client unless one of the following conditions is met:

- The name of the client host is in **/etc/hosts.equiv** file on the server.
- The user on the server that is associated with the uid sent by the client has an entry in **$HOME/.rhosts** specifying the client name on a line or the client name followed by at least one blank and the user's name.

  For example, assume a user whose login name is **mjk** is assigned to uid 7 on **NODE1** and executes the following **on** command:

  ```
  on NODE2 pwd
  ```

  User **mjk** on **NODE2** must have one of the following entries in **$HOME/.rhosts**:

  ```
  NODE1
  NODE1 mjk
  ```

DIAGNOSTICS

The following is a subset of the messages that could appear in the log file if the **-l** option is used. Some of these messages are also returned to the client.

rexd: could not umount: *dir*
rexd was unable to **umount()** the user's current working file system. See

r

WARNINGS for more details.

rexd: mountdir (*mountdir***) is not a directory**
The path name *mountdir*, under which temporary mount points are created, is not a directory or does not exist.

rexd: *command***: Command not found**
rexd could not find *command*.

rexd: *command***: Permission denied**
rexd was denied permission to execute *command*.

rexd: *command***: Text file busy**
The executable file is currently open for writing.

rexd: *command***: Can't execute**
rexd was unable to execute *command*.

rexd: root execution not allowed
rexd does not allow execution as user **root**.

rexd: User id *uid* **not valid**
The uid *uid* is not assigned to a user on the server.

rexd: User id *uid* **denied access**
rexd was started with the **-r** option and the remote execution request did not meet either of the conditions required by the **-r** option.

rexd: *host* **is not running a mount daemon**
The host *host* on which the user's current working directory is located is not running **mountd**. Therefore, **rexd** is unable to mount the required file system (see *mountd*(1M)).

rexd: not in export list for *file_system*
The host on which the client's current working directory is located does not have the server on the export list for file system *file_system* containing the client's current working directory. Therefore, **rexd** is unable to mount the required file system.

WARNINGS

The client's environment is simulated by **rexd**, but not completely recreated. The simulation of the client's environment consists of mounting the file system containing the client's current working directory (if it is not already mounted) and setting the user's environment variables on the server to be the same as the user's environment variables on the client. Therefore a command run by **rexd** does not always have the same effect as a command run locally on the client.

The **rex** protocol only identifies the client user by sending the uid of the client process and the host name of the client. Therefore, it is very difficult for **rexd** to perform user authentication. If a user on the server is assigned to the uid sent by the client, **rexd** executes the requested command as that user. If no user on the client is assigned to the uid sent by the client, **rexd** returns an error.

The **-r** option has been added to provide increased user authentication. However, the authentication provided is not foolproof, and is limited by the information passed by the **rex** protocol.

In order to simulate the client's environment, **rexd** mounts the file system containing the client's current working directory (if it is not already mounted). This mount is intended to be temporary for the duration of the command.

If **rexd** mounts a file system, it attempts to **umount()** the file system after the command has completed executing. However, if **rexd** receives a **SIGKILL** signal (see *signal*(2)), the file system is not unmounted. The file system remains mounted until the superuser executes the appropriate **umount** command or the server is rebooted.

rexd's attempt to umount the file system can also fail if the file system is busy. The file system is busy if it contains an open file or a user's current working directory. The file system remains mounted until the superuser executes the appropriate **umount** command or the server is rebooted.

For more information on **rexd** security issues, see *Using and Administering NFS Services*. Security issues and their consequences should be considered before configuring **rexd** to run on a system.

FILES

/dev/pty[**pqr**]*****	Master pseudo terminals.
/dev/tty[**pqr**]*****	Slave pseudo terminals.
/dev/ptym/pty[**pqr**]*****	Master pseudo terminals.
/dev/pty/tty[**pqr**]*****	Slave pseudo terminals.
/etc/inetd.conf	Configuration file for *inetd*(1M).
/etc/hosts.equiv	List of equivalent hosts.
$HOME/.rhosts	User's private equivalence list.
/var/spool/rexd/rexd*xxxxx*	Temporary mount points for remote file systems where *xxxxx* is a string of alpha numeric characters.

AUTHOR

rexd was developed by Sun Microsystems, Inc.

SEE ALSO

on(1), inetd(1M), mount(1M), exports(4), inetd.conf(4), inetd.sec(4).

Using and Administering NFS Services

NAME

rexecd - remote execution server

SYNOPSIS

/usr/lbin/rexecd [**-n**]

DESCRIPTION

rexecd is the server for the *rexec*(3N) routine; it expects to be started by the internet daemon (see *inetd*(1M)). **rexecd** provides remote execution facilities with authentication based on user account names and unencrypted passwords.

inetd(1M) calls **rexecd** when a service request is received at the port indicated for the "exec" service specification in **/etc/services**; see *services*(4). To run **rexecd**, the following line should be present in **/etc/inetd.conf**:

```
exec  stream  tcp  nowait  root  /usr/lbin/rexecd  rexecd
```

When a service request is received, the following protocol is initiated:

1. The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's host. If the first character sent is a null (**\0**), no secondary connection is made and the **stderr** of the command is sent to the primary stream. If the secondary connection has been made, **rexecd** interprets bytes it receives on that socket as signal numbers and passes them to the command as signals (see *signal*(2)).
3. A null-terminated user name of not more than 16 characters is retrieved on the initial socket.
4. A null-terminated, unencrypted, password of not more than 16 characters is retrieved on the initial socket.
5. A null-terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. **rexecd** then validates the user as is done by *login*(1). If the authentication succeeds, **rexecd** changes to the user's home directory and establishes the user and group protections of the user. If any of these steps fail, **rexecd** returns a diagnostic message through the connection, then closes the connection.
7. A null byte is returned on the connection associated with **stderr** and the command line is passed to the normal login shell of the user with that shell's **-c** option. The shell inherits the network connections established by **rexecd**.

rexecd uses the following path when executing the specified command:

```
/usr/bin:/usr/ccs/bin:/usr/bin/X11:
```

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

Username too long
The user name is longer than 16 characters.

Password too long
The password is longer than 16 characters.

Command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect
No password file entry for the user name existed or the wrong password was supplied.

`No remote directory`
The **`chdir`** command to the home directory failed.

`No more processes`
The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If the message persists, then the server's host may have a runaway process that is using all the entries in the process table.

shellname`: ...`
The user's login shell could not be started via *exec*(2) for the given reason.

WARNINGS

The password is sent unencrypted through the socket connection.

AUTHOR

`rexecd` was developed by the University of California, Berkeley.

SEE ALSO

remsh(1), inetd(1M), rexec(3N), inetd.conf(4), inetd.sec(4), services(4).

NAME

ripquery - query RIP gateways

SYNOPSIS

ripquery [**-1**] [**-2**] [**-**[**a5**] *authkey*] [**-n**] [**-N** *dest*[**/***mask*]] [**-p**] [**-r**] [**-v**] [**-w** *time*] *gateway* ...

DESCRIPTION

ripquery is used to request all routes known by a RIP gateway by sending a RIP request or POLL command. The routing information in any routing packets returned is displayed numerically and symbolically. **ripquery** is intended to be used as a tool for debugging gateways, not for network management. SNMP is the preferred protocol for network management.

ripquery by default uses the RIP POLL command, which is an undocumented extension to the RIP specification supported by **routed** on SunOS 3.x and later and by **gated** 1.4 and later. The RIP POLL command is preferred over the RIP REQUEST command because it is not subject to Split Horizon and/or Poisoned Reverse. See the RIP RFC for more information.

Options

-1 Send the query as a version 1 packet.

-2 Send the query as a version 2 packet (default).

-[**a5**]*authkey* Specifies the authentication password to use for queries. If **-a** specified, an authentication type of SIMPLE will be used, if **-5** is specified, an authentication type of MD5 will be used; otherwise the default is an authentication type of NONE. Authentication fields in incoming packets will be displayed, but not validated.

-n Prevents the address of the responding host from being looked up to determine the symbolic name.

-N *dest*[**/***mask*]
Specifies that the query should be for the specified *dest***/***mask* instead of complete routing table. The specification of the optional mask implies a version 2 query. Up to 23 requests about specific destinations may be include in one packet.

-p Uses the RIP POLL command to request information from the routing table. This is the default, but is an undocumented extension supported only by some versions of unOS 3.x and later versions of **gated**. If there is no response to the RIP POLL command, the RIP REQUEST command is tried. **gated** responds to a POLL command with all the routes learned via RIP.

-r Used the RIP REQUEST command to request information from the gateway's routing table. Unlike the RIP POLL command, all gateways should support the RIP REQUEST. If there is no response to the RIP REQUEST command, the RIP POLL command is tried. **gated** responds to a REQUEST command with all the routes he announces out the specified interface. Due to limitations in the UDP interface, on systems based on BSD 4.3 Reno or earlier, REQUESTs respond about the interface used to route packets back to the sender. This can be avoided by running **ripquery** on the host being queried.

-v Version information about **ripquery** is displayed before querying the gateways.

-w *time* Specifies the time in seconds to wait for the initial response from a gateway. The default value is 5 seconds.

AUTHORS

Jeffrey C Honig.

SEE ALSO

gated(1M), gdc(1M), ospf_monitor(1M), *GateD Documentation*, *GateD Configuration Guide*.

BUGS

Some versions of Unix do not allow looking up the symbolic name of a subnet.

NAME

rlogind - remote login server

SYNOPSIS

/usr/lbin/rlogind [**-ln**] [**-B** *bannerfile*]

DESCRIPTION

rlogind is the server for the *rlogin*(1) program. It provides a remote login facility with authentication based on privileged port numbers. **rlogind** expects to be executed by the Internet daemon (*inetd*(1M)) when it receives a service request at the port indicated in the services database for **login** using the **tcp** protocol (see *services*(4)).

When a service request is received, the following protocol is initiated by **rlogind**:

1. **rlogind** checks the client's source port. If the port is not in the range 512 through 1023 (a "privileged port"), the server aborts the connection.
2. **rlogind** checks the client's source address and requests the corresponding host name (see *gethostent*(3N), *hosts*(4), and *named*(1M)). If it cannot determine the hostname, it uses the Internet dot-notation representation of the host address.

Once the source port and address have been checked, **rlogind** proceeds with the authentication process described in *hosts.equiv*(4). **rlogind** then allocates a STREAMS based pseudo-terminal (see *ptm*(7), *pts*(7)), and manipulates file descriptors so that the slave half of the pseudo-terminal becomes **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of *login*(1) invoked with the **-f** option if authentication has succeeded. If automatic authentication fails, *login*(1) prompts the user with the normal login sequence. The **-l** option to **rlogind** prevents any authentication based on the user's **.rhosts** file unless the user is logging in as super-user. The **-B <bannerfile>** option to *rlogind* causes the file <bannerfile> to be displayed to incoming rlogin requests.

The **rlogind** process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. The protocol described in *ptm*(7) and *pts*(7) is used to enable and disable flow control via Ctrl-S/Ctrl-Q under the direction of the program running on the slave side of the pseudo-terminal, and to flush terminal output in response to interrupt signals. The login process sets the baud rate and **TERM** environment variable to correspond to the client's baud rate and terminal type (see *environ*(5)).

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

To start **rlogind** from the Internet daemon, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
login   stream   tcp   nowait   root   /usr/lbin/rlogind   rlogind
```

r

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Errors in establishing a connection cause an error message to be returned with a leading byte of 1 through the socket connection, after which the network connection is closed. Any errors generated by the login process or its descendents are passed through by the server as normal communication.

fork: No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

Cannot allocate pty on remote host

The server was unable to obtain a pseudo-terminal for use with the login process. Either all pseudo-terminals were in use, or the pty driver has not been properly set up. Note, the number of slave devices that can be allocated depends on NSTRPTY, a kernel tunable parameter. This can be changed via SAM (see *ptm*(7), *pts*(7)).

Next step: Check the pty configuration of the host where **rlogind** executes.

`Permission denied`
The server denied access because the client was not using a reserved port. This should only happen to interlopers trying to break into the system.

`/usr/bin/login: ...`
The login program could not be started via *exec*(2) for the reason indicated.

Next step: Try to correct the condition causing the problem. If this message persists, contact your system administrator.

WARNINGS

The “privileged port” authentication procedure used here assumes the integrity of each host and the connecting medium. This is insecure, but is useful in an “open” environment. Note that any passwords are sent unencrypted through the socket connection.

AUTHOR

`rlogind` was developed by the University of California, Berkeley.

FILES

`/etc/hosts.equiv`	List of equivalent hosts
`$HOME/.rhosts`	User’s private equivalence list

SEE ALSO

login(1), rlogin(1), inetd(1M), named(1M), gethostent(3N), ruserok(3N), hosts(4), hosts.equiv(4), inetd.conf(4), services(4), environ(5), pty(7).

NAME

rlogind - remote login server

SYNOPSIS

/usr/lbin/rlogind [**-ln**] [**-B** *bannerfile*]

Kerberos V5 Network Authentication environments:

/usr/lbin/rlogind [**-clnKkRr**] [**-B** *bannerfile*]

DESCRIPTION

rlogind is the server for the *rlogin*(1) program. It provides a remote login facility with two kinds of authentication methods:

1. Authentication based on privileged port numbers where the client's source port must be in the range 512 through 1023. In this case **rlogind** assumes it is operating in normal or non-secure environment.
2. Authentication based on Kerberos V5. In this case **rlogind** assumes it is operating in a Kerberos V5 Network Authentication, i.e., secure environment.

The *inetd* daemon invokes rlogind if a service request is received at ports indicated by the **login** or **klogin** services specified in **/etc/services** (see *inetd*(1M) and *services*(4)). Service requests arriving at the **klogin** port assume a secure environment and expect Kerberos authentication to take place.

To start **rlogind** from the *inetd* daemon in a non-secure environment, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
login   stream   tcp   nowait   root   /usr/lbin/rlogind   rlogind
```

In a secure environment, **/etc/inetd.conf** must contain an entry:

```
klogin   stream   tcp   nowait   root   /usr/lbin/rlogind   rlogind -K
```

See *inetd.conf*(4) for more information.

To prevent non-secure access, the entry for **login** should be commented out in **/etc/inetd.conf**. Any non-Kerberos access will be denied since the entry for the port indicated by **login** has now been removed or commented out. In a such a situation, a generic error message,

rcmd: connect <*hostname***> : Connection refused**

is displayed. See DIAGNOSTICS for more details.

Options

rlogind recognizes the following options:

-c Ignore checksum verification. This option is used to achieve interoperability between clients and servers using different checksum calculation methods. For example, the checksum calculation in a application developed with Kerberos V5 Beta 4 API is different from the calculation in a Kerberos V5-1.0 application.

-l Prevents any authentication based on the user's **.rhosts** file unless the user is logging in as super-user.

-B*bannerfile*
Causes the file, *bannerfile*, to be displayed to incoming rlogin requests.

In a secure environment, **rlogind** will recognize the following additional options:

-K Authorization based on Kerberos V5 must succeed or access will be rejected (see *sis*(5) for details on authorization).

-R Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts must succeed. For more information on equivalent accounts, see *hosts.equiv*(4).

-r Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts (see *hosts.equiv*(4)).

r

2. Authorization based on Kerberos V5.

-k Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authorization based on Kerberos V5.
2. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts.

Note: The **-k** option is ignored when used with **-K**, and the **-r** option is ignored when used with **-R**. Also, if no options are specified, the default option is **-K**.

Operation

When a service request is received, the following protocol is initiated by **rlogind**:

1. **rlogind** checks the client's source port. If the port is not in a privileged port, i.e., in the range 512 through 1023, and **rlogind** is operating in a non-secure environment, the connection is terminated. In a secure environment, the action taken depends on the command line options:

   -R The source port must be a privileged port otherwise **rlogind** terminates the connection.

   -r If the source port is not a privileged port then Kerberos authorization must succeed or the connection is terminated.

   -k The source port must be a privileged port if Kerberos authorization fails.

   -K No action is taken.

2. **rlogind** checks the client's source address and requests the corresponding host name (see *gethostent*(3N), *hosts*(4), and *named*(1M)). If it cannot determine the hostname, it uses the Internet dot-notation representation of the host address.
3. **rlogind**, in a secure environment, proceeds with the Kerberos authentication process described in *sis*(5). If authentication succeeds, then the authorization selected by the command line option **-K**, **-R**, **-k**, or **-r** is performed. The authorization selected could be as specified in *hosts.equiv*(4) or Kerberos authorization as specified in *sis*(5).
4. **rlogind** then allocates a STREAMS based pseudo-terminal (see *ptm*(7), *pts*(7)), and manipulates file descriptors so that the slave half of the pseudo-terminal becomes **stdin**, **stdout**, and **stderr** for a login process.
5. This login process is an instance of *login*(1) invoked with the **-f** option if authentication has succeeded. In a non-secure environment, if automatic authentication fails, *login*(1) prompts the user with the normal login sequence. In a secure environment, if authentication fails, **rlogind** generates an error message and quits.

The **rlogind** process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. The protocol described in *ptm*(7) and *pts*(7) is used to enable and disable flow control via Ctrl-S/Ctrl-Q under the direction of the program running on the slave side of the pseudo-terminal, and to flush terminal output in response to interrupt signals. The login process sets the baud rate and **TERM** environment variable to correspond to the client's baud rate and terminal type (see *environ*(5)).

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Errors in establishing a connection cause an error message to be returned with a leading byte of 1 through the socket connection, after which the network connection is closed. Any errors generated by the login process or its descendents are passed through by the server as normal communication.

fork: No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

`Cannot allocate pty on remote host`

The server was unable to obtain a pseudo-terminal for use with the login process. Either all pseudo-terminals were in use, or the pty driver has not been properly set up. Note, the number of slave devices that can be allocated depends on NSTRPTY, a kernel tunable parameter. This can be changed via SAM (see *ptm*(7), *pts*(7)).

Next step: Check the pty configuration of the host where **`rlogind`** executes.

`Permission denied`

The server denied access because the client was not using a reserved port. This should only happen to interlopers trying to break into the system.

`/usr/bin/login: ...`

The login program could not be started via *exec*(2) for the reason indicated.

Next step: Try to correct the condition causing the problem. If this message persists, contact your system administrator.

`rcmd: connect : <hostname>: Connection refused.`

This generic message could be due to a number of reasons. One of the reasons could be because the entry for *login* service is not present in **`/etc/inetd.conf`**. This entry may have been removed or commented out to prevent non-secure access.

Kerberos specific errors are listed in *sis*(5).

WARNINGS

The integrity of each host and the connecting medium is assumed if the "privileged port" authentication procedure is used in a non-secure environment or if the command line options **`-R`** or **`-r`** are used in a secure environment. Although both these methods provide insecure access, they are useful in an "open" environment. This is insecure, but is useful in an “open” environment.

Note also that all information, including any passwords, are passed unencrypted between the two hosts when **`rlogind`** is invoked in a non-secure environment.

AUTHOR

`rlogind` was developed by the University of California, Berkeley.

FILES

`/etc/hosts.equiv`	List of equivalent hosts
`$HOME/.rhosts`	User's private equivalence list

SEE ALSO

login(1), rlogin(1), inetd(1M), named(1M), gethostent(3N), ruserok(3N), hosts(4), hosts.equiv(4), inetd.conf(4), services(4), environ(5), pty(7), sis(5).

NAME

rlp - send LP line printer request to a remote system

SYNOPSIS

/usr/sbin/rlp -I*id* [**-C** *class*] [**-J** *job*] [**-T** *title*] [**-i**[*numcols*]] [-*kfont*] [**-w** *num*]
[**-cdfghlnptv**] *file*

DESCRIPTION

rlp transfers a spooling request to a remote system to be printed. **rlp** communicates with a spooling daemon on a remote system to transfer the spooling request. Options can be set only on the original system. Transfers of a remote request use only the **-I** option and the file.

This command is intended to be used only by the spool system in response to the **lp** command and should not be invoked directly (see *lp*(1)).

Options

rlp recognizes the following options and command-line arguments:

-I*id*	The argument *id* is the request ID.
-C *class*	Take the *class* argument as a job classification for use on the banner page.
-J *job*	Take the *job* argument as the job name to print on the banner page. Normally, the first file's name is used.
-T *title*	Use the *title* argument as the title used by **pr** instead of the file name (see *pr*(1)). **-T** is ignored unless the **-p** option is specified.
-h	Suppress the printing of the banner page.
-i[*numcols*]	Cause the output to be indented. If the next argument is numeric, it is used as the number of blanks to be printed before each line; otherwise, 8 characters are printed.
-*kfont*	Specify a *font* to be mounted on font position *k*, where *k* is from **1** through **4**.
-w*num*	Use the *num* argument number as the page width for **pr**.

The following single-letter options are used to notify the line printer spooler that the files are not standard text files. The spooling system uses the appropriate filters (if the option is supported) to print the data accordingly. These options are mutually exclusive.

-c	The files are assumed to contain data produced by *cifplot*.
-d	The files are assumed to contain data from *tex* (DVI format).
-f	Use a filter that interprets the first character of each line as a standard FORTRAN carriage control character.
-g	The files are assumed to contain standard plot data as produced by the **plot** routines.
-l	Use a filter that suppresses page breaks.
-n	The files are assumed to contain data from **ditroff** (device-independent **troff**).
-p	Use **pr** to format the files.
-t	The files are assumed to contain data from **troff** (cat phototypesetter commands).
-v	The files are assumed to contain a raster image for devices such as the Benson Varian.

WARNINGS

Some remote line printer models may not support all of these options. Options not supported are silently ignored.

When **rlp** is transferring a request that originated on another system, only the **-I** option and the file is used. This saves **rlp** from having to set the various options multiple times. Specifying unused options does not produce an error.

AUTHOR

rlp was developed by the University of California, Berkeley and HP.

FILES

/etc/passwd
/usr/sbin/rlpdaemon
/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

accept(1M), enable(1), lp(1), lpadmin(1M), lpsched(1M), lpstat(1), rcancel(1M), rlpdaemon(1M), rlpstat(1M).

NAME

rlpdaemon - remote spooling line printer daemon, message write daemon

SYNOPSIS

/usr/sbin/rlpdaemon [**-i**] [**-l**] [**-L** *logfile*]

DESCRIPTION

rlpdaemon is a line printer daemon (spool area handler) for remote spool requests. **rlpdaemon** is normally invoked at boot time from the **/sbin/rc** file or started by *inetd*(1M), when necessary. **rlpdaemon** runs on a system that receives requests to be printed. **rlpdaemon** transfers files to the spooling area, displays the queue, or removes jobs from the queue.

rlpdaemon is also used as a server process to write a message on the user's terminal, upon receiving a request from a remote system.

Options

-i Prevent **rlpdaemon** from remaining after a request is processed. This is required if **rlpdaemon** is started from *inetd*(1M).

-l Cause **rlpdaemon** to log error messages and valid requests received from the network to the file **/var/adm/lp/lpd.log**. This can be useful for debugging.

-L *logfile* Change the file used for writing error conditions from the file **/var/adm/lp/lpd.log** to *logfile*.

When **rlpdaemon** is started by *inetd*(1M), access control is provided via the file **/var/adm/inetd.sec** to allow or prevent a host from making requests. When **rlpdaemon** is not started by *inetd*(1M), all requests must come from one of the machines listed in the file **/etc/hosts.equiv** or **/var/spool/lp/.rhosts**. When **/var/spool/lp/.rhosts** is used for access, the user name should be **lp**.

The following entry should exist in **/etc/services** for remote spooling:

```
printer      515/tcp      spooler
```

EXAMPLES

To start **rlpdaemon** from **/sbin/rc**, invoke the command:

```
/usr/sbin/rlpdaemon
```

To start **rlpdaemon** from **inetd**, the following line should be included in the file **/etc/inetd.conf**:

```
printer stream tcp nowait root /usr/sbin/rlpdaemon rlpdaemon -i
```

WARNINGS

If the remote system is the same as the local system and **rlpdaemon** was not started by *inetd*(1M), the local system name *must* be included in file **/etc/hosts.equiv**.

AUTHOR

rlpdaemon was developed by the University of California, Berkeley and HP.

FILES

/etc/hosts.equiv
/etc/services
/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*
/var/adm/inetd.sec

SEE ALSO

accept(1M), enable(1), lp(1), inetd(1M), lpadmin(1M), lpsched(1M), lpstat(1), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M). hosts.equiv(4), inetd.conf(4), inetd.sec(4), services(4).

HP-UX System Administrator manuals

NAME

rlpstat - print status of LP spooler requests on a remote system

SYNOPSIS

/usr/sbin/rlpstat [**-d** *printer*] [**-u** *user*] [*id* ...]

DESCRIPTION

rlpstat reports the status of the specified jobs or all requests associated with a user. If no arguments are specified, **rlpstat** reports on any requests currently in the queue.

For each request submitted (i.e., each invocation of **lp** — see *lp*(1)) **rlpstat** reports the request ID, user's name, total size of the request, date of the request, and, if it is being transferred, the device.

This command is intended to be used only by the spool system in response to the **lpstat** command and should not be invoked directly (see *lpstat*(1M)).

Options

rlpstat recognizes the following options and command-line arguments:

-d *printer*	Specify a particular printer. Otherwise, the default line printer is used (or the value of the **LPDEST** environment variable).
-u *user*	Status is requested on all requests for the user who executed the **rlpstat** command on the specified printer (see the **-d** option).
id	Status is requested on the specified request IDs (as returned by **lp**). All the request IDs must be for the same printer.

AUTHOR

rlpstat was developed by the University of California, Berkeley, and HP.

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

enable(1), lp(1), lpadmin(1M), lpsched(1M), lpstat(1), rcancel(1M), rlp(1M), rlpdaemon(1M).

NAME

rmsf - remove a special (device) file

SYNOPSIS

/sbin/rmsf [**-a**|**-k**] [**-D** *directory*] [**-q**|**-v**] *special_file* ...

/sbin/rmsf [**-C** *class* | **-d** *driver*] [**-D** *directory*] **-H** *hw_path* [**-k**] [**-q**|**-v**]

DESCRIPTION

The **rmsf** command removes one or more special files from the current directory, and potentially removes information about the associated device or devices from the system.

If no options are specified, **rmsf** removes only the *special_files* specified on the command line. The **-k** option causes **rmsf** to remove the definition of the device from the system without removing any special files. The **-a** option causes **rmsf** to remove the device definition, and all special files that map to it from the **/dev** directory (or the directory specified with the **-D** option). By default, **rmsf** only removes *special_file* as given on the command line, however, when the **-a** option is used and *special_file* is an absolute path name *special_file* will be removed even if it does not reside in the **/dev** directory (or the directory specified with the **-D** option).

If a **-H** *hw_path* is specified alone, all special files mapping to devices at that hardware path and the system definition of those devices are removed. The **-C** and **-d** options remove only those special files that are associated with the given device driver or that belong to the given device class, respectively. This is useful when there is more than one type of special file mapped to a single hardware path. If the **-k** option is specified, the definition of all devices at that hardware path are removed from the system, again without removing any special files.

Normally, **rmsf** displays a message as the special files are deleted for each driver. The **-q** (quiet) option suppresses the deletion message. The **-v** (verbose) option displays the deletion message and the name of each special file as it is deleted.

Note that most drivers do not support the ability to be removed from the system.

If the device being removed from the system uses a dynamically assigned major number, that number will be freed up for future allocation.

Options

rmsf recognizes the following options:

-a Remove the definition of the device from the system along with all special files that refer to the device. This option cannot be used with **-k**.

-C *class* Match devices that belong to a given device class, *class*. Device classes can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in files in the directory **/usr/conf/master.d**. This option cannot be used with **-d**.

-d *driver* Match devices that are controlled by the specified device driver, *driver*. Device drivers can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in files in the directory **/usr/conf/master.d**. This option cannot be used with **-C**.

-D *directory* Override the default device installation directory **/dev** and remove the special files from *directory* instead. *directory* must exist; otherwise, **rmsf** displays an error message and exits. See WARNINGS.

-H *hw_path* Match devices at a given hardware path, *hw-path*. Hardware paths can be listed with the **ioscan** command (see *ioscan*(1M)). A hardware path specifies the addresses of the hardware components leading to a device. It consists of a string of numbers separated by periods (**.**), such as **52** (a card), **52.3** (a target address), and **52.3.0** (a device). If a hardware component is a bus converter, the following period, if any, is replaced by a slash (**/**) as in **2**, **2/3**, and **2/3.0**.

If the specified path contains fewer numbers than are necessary to reach a device, special files are made for all devices at addresses that extend the given path. If the specified path is **56**, then special files are made for the devices at addresses **56.0**, **56.1**, **56.2**, etc.

-k Remove the definition of the device from the system, but not any special files. This option cannot be used with **-a**.

-q Quiet option. Normally, **rmsf** displays a message as each driver is removed. This option suppresses the driver message, but not error messages. See the **-v** option.

-v Verbose option. In addition to the normal processing message, display the name of each sepecial file as it is removed. See the **-q** option. Print the names of the files as **rmsf** is removing them.

RETURN VALUE

rmsf exits with one of the following values:

0 Successful completion, including warning diagnostics.
1 Failure. An error occurred.

DIAGNOSTICS

Most of the diagnostic messages from **rmsf** are self-explanatory. Listed below are some messages deserving further clarification. Errors cause **rmsf** to halt immediately. Warnings allow the program to continue.

Errors

No such device in the system

No device in the system matched the options specified. Use **ioscan** to list the devices in the system (see *ioscan*(1M)).

special_file **is not a special file**

The file is not associated with an I/O device.

Warnings

Cannot remove *driver* **at** *hw_path*

The definition of the device located at *hw_path* and controlled by *driver* cannot be removed from the kernel. That is *driver* does not support the **unbind** function.

No device associated with *special_file*

The special file does not map to a device in the system; the file is removed unless the **-k** option was specified.

EXAMPLES

Remove the special file **mux0** from the current directory:

```
rmsf ./mux0
```

Remove the system definition of the device associated with **/dev/lp0** along with all special files that refer to the device:

```
rmsf -a /dev/lp0
```

Remove the system definitions for all devices associated with hardware path 52.6.0:

```
rmsf -k -H 52.6.0
```

WARNINGS

Most commands and subsystems assume their device files are in **/dev**, therefore the use of the **-D** option is discouraged.

Most device drivers do not support the *unbind* operation necessary to remove the device from the system.

AUTHOR

rmsf was developed by HP.

FILES

/dev/config
/etc/ioconfig
/usr/conf/master.d/*

SEE ALSO

rm(1), insf(1M), ioscan(1M), lsdev(1M), lssf(1M), mksf(1M), ioconfig(4).

NAME

rmt - remote magnetic-tape protocol module

SYNOPSIS

/usr/sbin/rmt

DESCRIPTION

rmt is a program used by the remote dump and restore programs for manipulating a magnetic tape drive through an interprocess communication (IPC) connection. The **fbackup** and **frecover** commands also use **rmt** to achieve remote backup capability (see *fbackup*(1M) and *frecover*(1M)). **rmt** is normally started up with an **rexec()** or **rcmd()** call (see *rexec*(3C) and *rcmd*(3C)).

rmt accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. DDS devices that emulate magnetic tapes are also supported. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number***\n**

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number***\n***error-message***\n**

where *error-number* is one of the possible error numbers described in *errno*(2) and *error-message* is the corresponding error string as printed from a call to **perror()** (see *perror*(3C)). The protocol is comprised of the following commands (a space is present between each token):

O *device mode* Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to **open()** (see *open*(2)). If a device is already open, it is closed before a new open is performed.

o *device mode* Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of an octal number suitable for passing to **open()**. If a device is already open, it is closed before a new open is performed.

C *device* Close the currently open device. The *device* specified is ignored.

L *whence offset* Perform an **lseek()** operation using the specified parameters (see *lseek*(2)). The response value is that returned from by **lseek()**.

W *count* Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from by **write()** (see *write*(2)).

R *count* Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 Kbytes), it is truncated to the data buffer size. **rmt** then performs the requested **read()** and responds with **A***count-read***\n** if the read was successful. Otherwise an error is returned in the standard format. If the read was successful, the data read is then sent.

I *operation count* Perform a **MTIOCOP ioctl()** command using the specified parameters. Parameters are interpreted as ASCII representations of the decimal values to be placed in the **mt_op** and **mt_count** fields of the structure used in the **ioctl()** call. The return value is the *count* parameter when the operation is successful.

S Return the status of the open device, as obtained with a **MTIOCGET ioctl()** call. If the operation was successful, an ACK is sent with the size of the status buffer, then the status buffer is sent (in binary).

s Return the status of the open device, as obtained with a **fstat()** call. If the operation was successful, an ACK is sent with the size of the status buffer, then the status buffer is sent (in binary). **f** Return the status of the open device, as obtained with a **fstat()** call. If the operation was successful, an ACK is sent with the size of the status buffer, then the status buffer is sent in the following ASCII format:

machine<blank>*value*<newline>
stat_struct_member_name<blank>*value*<newline>

r

The end of the data is indicated by an ASCII NULL character. See **/usr/include/sys/stat.h** for the **struct stat** definition. In addition to the struct stat information, there is an entry in the buffer describing the machine type as returned from a **uname()** call (see *uname*(2)). In the above format "machine" is a key word. All fields except **st_spare4** of the **struct stat** are returned.

m Return the status of the open device, as obtained with a **MTIOCGET ioctl()** call. If the operation was successful, an ack is sent with the size of the status buffer, then the status buffer is sent in the following ASCII format:

machine<blank>*value*<newline>
mtget_struct_member_name<blank>*value*<newline>

The end of the data is indicated by an ASCII NULL character. See **/usr/include/sys/mtio.h** for the **struct mtget** definition. In addition to the struct mtget information there is an entry in the buffer describing the machine type as returned from a **uname()** call. In the above format "machine" is a keyword.

Any other command causes **rmt** to exit.

RETURN VALUE

Device status is returned in the field **mt_gstat**. **/usr/include/sys/mtio.h** contains defined macros for checking the status bits.

DIAGNOSTICS

All responses are of the form described above.

WARNINGS

Use of this command for remote file access protocol is discouraged.

AUTHOR

rmt was developed by the University of California, Berkeley.

SEE ALSO

ftio(1), fbackup(1M), frecover(1M), dump(1M), restore(1M), rcmd(3C), rexec(3C).

NAME

route - manually manipulate the routing tables

SYNOPSIS

/usr/sbin/route [**-f**] [**-n**] [**-p** *pmtu*] **add** [**net**|**host**] *destination* [**netmask** *mask*] *gateway* [*count*]

/usr/sbin/route [**-f**] [**-n**] **delete** [**net**|**host**] *destination* [**netmask** *mask*] *gateway* [*count*]

/usr/sbin/route -f [**-n**]

DESCRIPTION

The **route** command manipulates the network routing tables manually. You must have appropriate privileges.

Subcommands

The following subcommands are supported.

add Add the specified host or network route to the network routing table. If the route already exists, a message is printed and nothing changes.

delete Delete the specified host or network route from the network routing table.

Options and Arguments

route recognizes the following options and arguments.

-f Delete all route table entries that specify a remote host for a gateway. If this is used with one of the subcommands, the entries are deleted before the subcommand is processed.

-n Print any host and network addresses in Internet dot notation, except for the default network address, which is printed as **default**.

-p *pmtu* Specifies a path maximum transmission unit (MTU) value for a static route. The minimum value allowed is 68 bytes; the maximum is the MTU of the outgoing interface for this route. This option can be applied to both host and network routes.

net or **host** The type of *destination* address. If this argument is omitted, routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If the *destination* has a local address part of **INADDR_ANY(0)**, the route is assumed to be to a network; otherwise, it is treated as a route to a host.

destination The destination host system where the packets will be routed. *destination* can be one of the following:

- A host name (the official name or an alias, see *gethostent*(3N)).
- A network name (the official name or an alias, see *getnetent*(3N)).
- An Internet address in dot notation (see *inet*(3N)).
- The keyword **default**, which signifies the wildcard gateway route (see *routing*(7)).

netmask *mask* The mask that will be bit-wise ANDed with *destination* to yield a net address where the packets will be routed. *mask* can be specified as a single hexadecimal number with a leading **0x**, with a dot-notation Internet address, or with a pseudo-network name listed in the network table (see *networks*(4)). The length of the *mask*, which is the number of contiguous 1's starting from the leftmost bit position of the 32-bit field, can be shorter than the default network mask for the *destination* address. (see *routing*(7)). If the **netmask** option is not given, *mask* for the route will be derived from the *netmasks* associated with the local interfaces. (see *ifconfig*(1M)). *mask* will be defaulted to the longest *netmask* of those local interfaces that have the same network address. If there is not any local interface that has the same network address, then *mask* will be defaulted to the default network mask of *destination*.

gateway The gateway through which the destination is reached. *gateway* can be one of the following:

- A host name (the official name or an alias, see *gethostent*(3N)).
- An Internet address in dot notation (see *inet*(3N)).

count An integer that indicates whether the gateway is a remote host or the local host. If the route leads to a destination through a remote gateway, *count* should be a number greater than 0. If the route leads to *destination* and the gateway is the local host, *count* should be 0. The default for *count* is zero. The result is not defined if *count* is negative.

Operation

All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname()**; if the host name is not found, the *destination* is searched for as a network name using **getnetbyname()**. *destination* and *gateway* can be in dot notation (see *inet*(3N)).

If the **-n** option is not specified, any host and network addresses are displayed symbolically according to the name returned by **gethostbyaddr()** and **getnetbyaddr()**, respectively, except for the default network address (printed as **default**) and addresses that have unknown names. Addresses with unknown names are printed in Internet dot notation (see *inet*(3N)).

If the **-n** option is specified, any host and network addresses are printed in Internet dot notation except for the default network address which is printed as **default**.

If the **-f** option is specified, **route** deletes all route table entries that specify a remote host for a gateway. If it is used with one of the subcommands described above, the entries are deleted before the subcommand is processed.

Path MTU Discovery is a technique for discovering the maximum size of an IP datagram that can be sent on an internet path without causing datagram fragmentation in the intermediate routers. In essence, a source host that utilizes this technique initially sends out datagrams up to the the size of the outgoing interface. The Don't Fragment (DF) bit in the IP datagram header is set. As an intermediate router that supports Path MTU Discovery receives a datagram that is too large to be forwarded in one piece to the next-hop router and the DF bit is set, the router will discard the datagram and send an ICMP Destination Unreachable message with a code meaning "fragmentation needed and DF set". The ICMP message will also contain the MTU of the next-hop router. When the source host receives the ICMP message, it reduces the path MTU of the route to the MTU in the ICMP message. With this technique, the host route in the source host for this path will contain the proper MTU.

The **-p** *pmtu* option is useful only if you know the network environment well enough to enter an appropriate *pmtu* for a host or network route. IP will fragment a datagram to the *pmtu* specified for the route on the local host before sending the datagram out to the remote. It will avoid fragmentation by routers along the path, if the *pmtu* specified in the **route** command is correct.

ping can be used to find the *pmtu* information for the route to a remote host. The *pmtu* information in the routing table can be displayed with the **netstat -r** command (see *netstat*(1)).

The loopback interface **(lo0)** is automatically configured when the system boots with the TCP/IP software. The default IP address and netmask of the loopback interface are 127.0.0.1 and 255.0.0.0, respectively.

The 127.0.0.0 loopback route is set up automatically when **lo0** is configured so that packets for any 127.*.*.* address will loop back to the local host. Users cannot add or delete any 127.*.*.* loopback routes.

Output

add *destination*: **gateway** *gateway*

The specified route is being added to the tables.

delete *destination*: **gateway** *gateway*

The specified route is being deleted from the tables.

Flags

The values of the *count* and *destination* type fields in the **route** command determine the presence of the **G** and **H** flags in the **netstat -r** display and thus the route type, as shown in the following table.

Count	Destination Type	Flags	Route Type
=0	network	**U**	Route to a network directly from the local host
>0	network	**UG**	Route to a network through a remote host gateway
=0	host	**UH**	Route to a remote host directly from the local host
>0	host	**UGH**	Route to a remote host through a remote host gateway
=0	**default**	**U**	Wildcard route directly from the local host
>0	**default**	**UG**	Wildcard route through a remote host gateway

DIAGNOSTICS

The following error diagnostics can be displayed:

add a route that already exists

The specified entry is already in the routing table.

delete a route that does not exist

The specified route was not in the routing table.

cannot update loopback route

Routes for any 127.*.*.* loopback destination cannot be added or deleted.

WARNINGS

Reciprocal **route** commands must be executed on the local host, the destination host, and all intermediate hosts if routing is to succeed in the cases of virtual circuit connections or bidirectional datagram transfers.

The HP-UX implementation of **route** does not presently support a **change** subcommand.

AUTHOR

route was developed by the University of California, Berkeley.

FILES

/etc/networks
/etc/hosts

SEE ALSO

netstat(1), ifconfig(1M), ping(1M), ndd(1M), getsockopt(2), recv(2), send(2), gethostent(3N), getnetent(3N), inet(3N), routing(7).

NAME

rpc.nisd, rpc.nisd_resolv, nisd, nisd_resolv - NIS+ service daemon

SYNOPSIS

/usr/sbin/rpc.nisd [**-ACDFhlv**] [**-Y** [**-B** [**-t** *netid*]]] [**-d** *dictionary*] [**-L** *load*] [**-S** *level*]

rpc.nisd_resolv

DESCRIPTION

The **rpc.nisd** daemon is an RPC service that implements the NIS+ service. This daemon must be running on all machines that serve a portion of the NIS+ namespace.

rpc.nisd is usually started from a system startup script.

rpc.nisd_resolv is an auxillary process that is started by **rpc.nisd** when it is invoked with **-B** option. Note that **rpc.nisd_resolv** should not be started independently.

Options

-A Authentication verbose mode. The daemon logs all the authentication related activities to *syslogd*(1M) with **LOG_INFO** priority.

-B Provide ypserv compatible DNS forwarding for NIS host requests. The DNS resolving process, **rpc.nisd_resolv**, is started and controlled by **rpc.nisd**. This option requires that the **/etc/resolv.conf** file be set up for communication with a DNS nameserver. The **nslookup** utility can be used to verify communication with a DNS nameserver. See *resolver*(4) and *nslookup*(1).

-C Open diagnostic channel on **/dev/console**.

-D Debug mode (don't fork).

-F Force the server to do a checkpoint of the database when it starts up. Forced checkpoints may be required when the server is low on disk space. This option removes updates from the transaction log that have propagated to all of the replicas.

-L *number*
Specify the "load" the NIS+ service is allowed to place on the server. The load is specified in terms of the *number* of child processes that the server may spawn. This *number must* be at least 1 for the callback functions to work correctly. The default is 128.

-S *level* Set the authorization security level of the service. The argument is a number between 0 and 2. By default, the daemon runs at security level 2.

- **0** Security level 0 is designed to be used for testing and initial setup of the NIS+ namespace. When running at level 0, the daemon does not enforce any access controls. Any client is allowed to perform any operation, including updates and deletions.
- **1** At security level 1, the daemon accepts both **AUTH_SYS** and **AUTH_DES** credentials for authenticating clients and authorizing them to perform NIS+ operations. This is not a secure mode of operation since **AUTH_SYS** credentials are easily forged. It should not be used on networks in which any untrusted users may potentially have access.
- **2** At security level 2, the daemon accepts only **AUTH_DES** credentials for authentication and authorization. This is the highest level of security currently provided by the NIS+ service. This is the default security level if the **-S** option is not used.

-Y Put the server into NIS (YP) compatibility mode. When operating in this mode, the NIS+ server will respond to NIS Version 2 requests using the version 2 protocol. Because the YP protocol is not authenticated, only those items that have read access to nobody (the unauthenticated request) will be visible through the V2 protocol. It supports only the standard Version 2 maps in this mode (see **-B** option and **NOTES** in *ypfiles*(4)).

-d *dictionary*
Specify an alternate dictionary for the NIS+ database. The primary use of this option is for testing. Note that the string is not interpreted, rather it is simply passed to the **db_initialize** function. See *nis_db*(3N).

-h Print list of options.

r

-t *netid* Use *netid* as the transport for communication between **rpc.nisd** and **rpc.nisd_resolv**. The default transport is **tcp**.

-v Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon (see *syslogd*(1M)) at **LOG_INFO** priority. This option is most useful for debugging problems with the service (see also **-A** option).

EXAMPLES

The following example sets up the NIS+ service.

```
rpc.nisd
```

The following example sets up the NIS+ service, emulating YP with DNS forwarding.

```
rpc.nisd -YB
```

EXTERNAL INFLUENCES

Environment Variables

NETPATH The transports that the NIS+ service will use can be limited by setting this environment variable (see *netconfig*(4)).

FILES

/var/nis/parent.object
This file contains an XDR encoded NIS+ object that describes the namespace above a root server. This parent namespace may be another NIS+ namespace or a foreign namespace such as one served by the Domain Name Service. It is only present on servers that are serving the root of the namespace.

/var/nis/root.object
This file contains an XDR encoded NIS+ object that describes the root of the namespace. It is only present on servers that are serving the root of the namespace.

/etc/rc.config.d/namesvrs
initialization script for NIS+

AUTHOR

rpc.nisd and **rpc.nisd_resolv** were developed by Sun Microsystems, Inc.

SEE ALSO

nis_cachemgr(1M), nisinit(1M), nissetup(1M), nslookup(1), syslogd(1M), nis_db(3N), netconfig(4), nisfiles(4), resolver(4), ypfiles(4).

NAME

rpc.nispasswdd, nispasswdd - NIS+ password update daemon

SYNOPSIS

/usr/sbin/rpc.nispasswdd [**-a** *attempts*] [**-c** *minutes*] [**-D**] [**-g**] [**-v**]

DESCRIPTION

rpc.nispasswdd daemon is an **ONC+ RPC** service that services password update requests from *nispasswd*(1) and *yppasswd*(1). It updates password entries in the **NIS+ passwd** table.

rpc.nispasswdd is normally started from a system startup script after the NIS+ server (*rpc.nisd*(1M)) has been started. **rpc.nispasswdd** will determine whether it is running on a machine that is a master server for one or more NIS+ directories. If it discovers that the host is not a master server, then it will promptly exit. It will also determine if *rpc.nisd*(1M) is running in NIS(YP) compatibility mode (the **-Y** option) and will register as **yppasswdd** for NIS(YP) clients as well.

rpc.nispasswdd will send to syslog all failed password update attempts, which will allow an administrator to determine whether someone was trying to "crack" the passwords.

rpc.nispasswdd has to be run by a superuser.

Options

-a *attempts*	Set the maximum number of attempts allowed to authenticate the caller within a password update request session. Failed attempts are processed by *syslogd*(1M) and the request is cached by the daemon. After the maximum number of allowed attempts the daemon severs the connection to the client. The default value is set to **3**.
-c *minutes*	Set the number of minutes a failed password update request should be cached by the daemon. This is the time during which if the daemon receives further password update requests for the same user and authentication of the caller fails, then the daemon will simply not respond. The default value is set to **30** minutes.
-D	Debug. Run in debugging mode.
-g	Generate **DES** credential. By default the DES credential is not generated for the user if they do not have one. By specifying this option, if the user does not have a credential, then one will be generated for them and stored in the NIS+ cred table.
-v	Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon. This option is useful for debugging problems.

RETURN VALUE

0	Success.
1	An error has occurred.

FILES

/etc/rc.config.d/namesvrs Initialization script for NIS+.

SEE ALSO

nispasswd(1), passwd(1), yppasswd(1), rpc.nisd(1M), syslogd(1M), nsswitch.conf(4).

NAME

rpcbind - universal addresses to RPC program number mapper

SYNOPSIS

rpcbind [**-d**] [**-w**]

DESCRIPTION

rpcbind is a server that converts RPC program numbers into universal addresses. It must be running on the host to be able to make RPC calls on a server on that machine.

When an RPC service is started, it tells **rpcbind** the address at which it is listening, and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts **rpcbind** on the server machine to determine the address where RPC requests should be sent.

rpcbind should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so **rpcbind** must be started before port monitors are invoked.

When **rpcbind** is started, it checks that certain name-to-address translation calls function correctly. If they fail, the network configuration databases may be corrupt. Since RPC services cannot function correctly in this situation, **rpcbind** reports the condition and terminates.

rpcbind can only be started by the super-user.

Options

rpcbind recognizes the following options:

-d Run in debug mode. In this mode, **rpcbind** will not fork when it starts, will print additional information during operation, and will abort on certain errors. With this option, the name-to-address translation consistency checks are shown in detail.

-w Do a warm start. If **rpcbind** aborts or terminates on **SIGINT** or **SIGTERM**, it will write the current list of registered services to **/tmp/portmap.file** and **/tmp/rpcbind.file**. Starting **rpcbind** with the **-w** option instructs it to look for these files and start operation with the registrations found in them. This allows **rpcbind** to resume operation without requiring all RPC services to be restarted.

WARNINGS

Terminating **rpcbind** with **SIGKILL** will prevent the warm-start files from being written.

All RPC servers must be restarted if the following occurs: **rpcbind** crashes (or is killed with **SIGKILL**) and is unable to to write the warm-start files; **rpcbind** is started without the **-w** option after a graceful termination; or, the warm-start files are not found by **rpcbind**.

AUTHOR

rpcbind was developed by Sun Microsystems, Inc.

FILES

/tmp/portmap.file

/tmp/rpcbind.file

SEE ALSO

rpcinfo(1M), rpcbind(3N).

NAME

rpcinfo - report RPC information

SYNOPSIS

rpcinfo [**-m**] [**-s**] [*host*]

rpcinfo -p [*host*]

rpcinfo -T *transport host prognum* [*versnum*]

rpcinfo -l [**-T** *transport*] *host prognum* [*versnum*]

rpcinfo [**-n** *portnum*] **-u** *host prognum* [*versnum*]

rpcinfo [**-n** *portnum*] **-t** *host prognum* [*versnum*]

rpcinfo -a *serv_address* **-T** *transport prognum* [*versnum*]

rpcinfo -b [**-T** *transport*] *prognum versnum*

rpcinfo -d [**-T** *transport*] *prognum versnum*

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

In the first synopsis, **rpcinfo** lists all the registered RPC services with **rpcbind** on *host*. If *host* is not specified, the local host is the default. If **-s** is used, the information is displayed in a concise format.

In the second synopsis, **rpcinfo** lists all the RPC services registered with **rpcbind**, version 2. Also note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the **rpcbind** protocol).

The third synopsis makes an RPC call to procedure **0** of *prognum* and *versnum* on the specified *host* and reports whether a response was received. *transport* is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote **rpcbind**.

The *prognum* argument is a number that represents an RPC program number (see *rpc*(4)).

If a *versnum* is specified, **rpcinfo** attempts to call that version of the specified *prognum*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *prognum* by calling version 0, which is presumed not to exist; if it does exist, **rpcinfo** attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note that the version number is required for **-b** and **-d** options.

The other ways of using **rpcinfo** are described in the *EXAMPLES* section.

r

Options

-T *transport* Specify the transport on which the service is required. If this option is not specified, **rpcinfo** uses the transport specified in the **NETPATH** environment variable, or if that is unset or null, the transport in the *netconfig*(4) database is used. This is a generic option, and can be used in conjunction with other options as shown in the *SYNOPSIS*.

-a *serv_address*
Use *serv_address* as the (universal) address for the service on *transport* to ping procedure **0** of the specified *prognum* and report whether a response was received. The **-T** option is required with the **-a** option.

If *versnum* is not specified, **rpcinfo** tries to ping all available version numbers for that program number. This option avoids calls to remote **rpcbind** to find the address of the service. The *serv_address* is specified in universal address format of the given transport.

-b Make an RPC broadcast to procedure **0** of the specified *prognum* and *versnum* and report all hosts that respond. If *transport* is specified, it broadcasts its request only on the specified transport. If broadcasting is not supported by any transport, an error message is printed. Use of broadcasting should be limited because of the potential for adverse effect on other systems.

-d Delete registration for the RPC service of the specified *prognum* and *versnum*. If *transport* is specified, unregister the service on only that transport, otherwise unregister the service on all the transports on which it was registered. Only the owner of a service can delete a registration, except the super-user who can delete any service.

-l Display a list of entries with a given *prognum* and *versnum* on the specified *host*. Entries are returned for all transports in the same protocol family as that used to contact the remote **rpcbind**.

-m Display a table of statistics of **rpcbind** operations on the given *host*. The table shows statistics for each version of **rpcbind** (versions 2, 3 and 4), giving the number of times each procedure was requested and successfully serviced, the number and type of remote call requests that were made, and information about RPC address lookups that were handled. This is useful for monitoring RPC activities on *host*.

-n *portnum* Use *portnum* as the port number for the **-t** and **-u** options instead of the port number given by **rpcbind**. Use of this option avoids a call to the remote **rpcbind** to find out the address of the service. This option is made obsolete by the **-a** option.

-p Probe **rpcbind** on *host* using version 2 of the **rpcbind** protocol, and display a list of all registered RPC programs. If *host* is not specified, it defaults to the local host. Note that version 2 of the **rpcbind** protocol was previously known as the portmapper protocol.

-s Display a concise list of all registered RPC programs on *host*. If *host* is not specified, it defaults to the local host.

-t Make an RPC call to procedure 0 of *prognum* on the specified *host* using TCP, and report whether a response was received. This option is made obsolete by the **-T** option as shown in the third synopsis.

-u Make an RPC call to procedure 0 of *prognum* on the specified *host* using UDP, and report whether a response was received. This option is made obsolete by the **-T** option as shown in the third synopsis.

EXAMPLES

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo
```

To show all of the RPC services registered with **rpcbind** on the machine named **klaxon** use:

```
example% rpcinfo klaxon
```

To show whether the RPC service with program number *prognum* and version *versnum* is registered on the machine named **klaxon** for the transport TCP use:

```
example% rpcinfo -T tcp klaxon prognum versnum
```

To show all RPC services registered with version 2 of the **rpcbind** protocol on the local machine use:

```
example% rpcinfo -p
```

To delete the registration for version 1 of the **walld** (program number **100008**) service for all transports use:

```
example# rpcinfo -d 100008 1
```

or

```
example# rpcinfo -d walld 1
```

AUTHOR

rpcinfo was developed by Sun Microsystems, Inc.

SEE ALSO

rpcbind(1M), rpc(3N), netconfig(4), rpc(4).

NAME

rpr - repair parity information in an HP SCSI disk array LUN

SYNOPSIS

rpr -b *block device_file*

DESCRIPTION

rpr repairs the data parity information on a LUN in an HP SCSI disk array when the LUN is configured in a data redundant RAID-level (RAID_1, RAID_3, or RAID_5). *block* is the logical address of the data block corresponding to the parity block needing repair. *block* is specified using the **-b** parameter. *device_file* is the name of the device file for the LUN.

Use **scn** (see *scn*(1M)) to identify data blocks that do not have correct parity blocks.

RETURN VALUE

rpr returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **rpr**
- SCSI (device level) communications
- system calls

Error messages generated by rpr:

usage: rpr -b block> <special>
rpr encountered an error in command syntax. Re-enter the command with all required arguments, in the order shown.

rpr: LUN does not exist
The addressed LUN is not configured, and is not known to the array controller.

rpr: LUN # too big
The LUN number, derived from the device file name, is out of range.

rpr: Not a raw file
rpr must be able to open the device file for raw access.

rpr: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

rpr: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

rpr uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, **unlink()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **rpr** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To repair block 12345 of the LUN **/dev/rdsk/c2t6d0** on a series 800:

```
rpr -b 12345 /dev/rdsk/c2t6d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

rpr was developed by HP.

SEE ALSO

scn(1M).

NAME

rquotad - remote quota server

SYNOPSIS

/usr/sbin/rpc.rquotad

DESCRIPTION

rquotad is an RPC server that returns quotas for a user of a local file system currently mounted by a remote machine by means of NFS (see *rpc*(3C)). The results are used by **quota** to display user quotas for remote file systems (see *quota*(1)). **rquotad** is normally invoked by **inetd** (see *inetd*(1M)).

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

directory/**quotas**	Quota statistics static storage for a file system, where *directory* is the root of the file system.

SEE ALSO

inetd(1M), rpc(3C), services(4), quota(5), nfs(7).

NAME

rstatd - kernel statistics server

SYNOPSIS

/usr/lib/netsvc/rstat/rpc.rstatd [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

rstatd is an RPC server that returns performance statistics obtained from the kernel. The **rup** utility prints this information (see *rup*(1)).

inetd invokes **rstatd** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

rstatd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file because enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if

- **portmap** dies (see *portmap*(1M)),
- another **rpc.rstatd** registers with **portmap**, or
- **rpc.rstatd** becomes unregistered with *portmap*.

The **-n** option is more efficient since a new process is not launched for each RPC request. Note, this option is the default.

AUTHOR

rstatd was developed by Sun Microsystems, Inc.

SEE ALSO

rup(1), inetd(1M), portmap(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

runacct - run daily accounting

SYNOPSIS

/usr/sbin/acct/runacct [*mmdd* [*state*]]

DESCRIPTION

runacct is the main daily accounting shell procedure. It is normally initiated via *cron*(1M). *runacct* processes connect, fee, disk, and process accounting files. It also prepares summary files for *prdaily* or billing purposes.

runacct takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into **active**. When an error is detected, a message is written to **/dev/console**, mail (see *mail*(1), *mailx*(1), or *elm*(1)) is sent to **root** and **adm**, and *runacct* terminates. *runacct* uses a series of lock files to protect against re-invocation. The files **lock** and **lock1** are used to prevent simultaneous invocation, and **lastdate** is used to prevent more than one invocation per day.

runacct breaks its processing into separate, restartable *states* using **statefile** to remember the last *state* completed. It accomplishes this by writing the *state* name into **statefile**. *runacct* then looks in **statefile** to see what it has done and to determine what to process next. *states* are executed in the following order:

SETUP	Move active accounting files into working files.
WTMPFIX	Verify integrity of **wtmp** file, correcting date changes if necessary.
CONNECT	Produce connect session records in **tacct.h** format.
PROCESS	Convert process accounting records into **tacct.h** format.
MERGE	Merge the connect and process accounting records.
FEES	Convert output of *chargefee* into **tacct.h** format and merge with connect and process accounting records.
DISK	Merge disk accounting records with connect, process, and fee accounting records.
MERGETACCT	Merge the daily total accounting records in **daytacct** with the summary total accounting records in **/var/adm/acct/sum/tacct**.
CMS	Produce command summaries.
USEREXIT	Any installation-dependent accounting programs can be included here.
CLEANUP	Cleanup temporary files and exit.

r

To restart *runacct* after a failure, first check the **active** file for diagnostics, then fix up any corrupted data files such as **pacct** or **wtmp**. The **lock** files and **lastdate** file must be removed before *runacct* can be restarted. The argument *mmdd* is necessary if *runacct* is being restarted, and specifies the month and day for which *runacct* will rerun the accounting. Entry point for processing is based on the contents of **statefile**; to override this, include the desired *state* on the command line to designate where processing should begin.

EXAMPLES

To start *runacct*.

```
nohup runacct 2> /var/adm/acct/nite/fd2log &
```

To restart *runacct*.

```
nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &
```

To restart *runacct* at a specific *state*.

```
nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &
```

WARNINGS

Normally it is not a good idea to restart *runacct* in its **SETUP** *state*. Run **SETUP** manually, then restart via:

runacct *mmdd* **WTMPFIX**

If *runacct* failed in its **PROCESS** *state*, remove the last **ptacct** file because it will not be complete.

FILES

/var/adm/acct/nite/active
/var/adm/acct/nite/daytacct
/var/adm/acct/nite/lastdate
/var/adm/acct/nite/lock
/var/adm/acct/nite/lock1
/var/adm/pacct∗
/var/adm/acct/nite/ptacct∗**.***mmdd*
/var/adm/acct/nite/statefile
/var/adm/wtmp

SEE ALSO

mail(1), acct(1M), acctcms(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), cron(1M), fwtmp(1M), acct(2), acct(4), utmp(4)

STANDARDS CONFORMANCE

runacct: SVID2, SVID3

NAME

rusersd - network username server

SYNOPSIS

/usr/lib/netsvc/rusers/rpc.rusersd [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

rusersd is an RPC server that returns a list of users on the network. The **rusers** command prints this information (see *rusers*(1)).

inetd invokes **rusersd** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

rusersd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file since enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if

- **portmap** dies (see *portmap*(1M)),
- another **rpc.rusersd** registers with **portmap**, or
- **rpc.rusersd** becomes unregistered with **portmap**.

The **-n** option is more efficient because a new process is not launched for each RPC request. This option is the default.

AUTHOR

rusersd was developed by Sun Microsystems, Inc.

SEE ALSO

rusers(1), inetd(1M), portmap(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

rwall - write to all users over a network

SYNOPSIS

/usr/sbin/rwall *hostname* ...

/usr/sbin/rwall -n *netgroup* ...

/usr/sbin/rwall -h *host* **-n** *netgroup*

DESCRIPTION

rwall reads a message from standard input until EOF, then sends the message, preceded by the line **Broadcast Message** ..., to all users logged in on the specified host machines. With the **-n** option, **rwall** sends the message to the specified network hosts defined in **/etc/netgroup** (see *netgroup*(4)).

A machine can only receive such a message if it is running **rwalld**, which is normally started from **/etc/inetd.conf** by the **inetd** daemon (see *inetd*(1M)).

WARNINGS

The timeout is kept fairly short so that the message can be sent to a large group of machines (some of which may be down) in a reasonable amount of time. Thus, the message may not get through to a heavily loaded machine.

AUTHOR

rwall was developed by Sun Microsystems, Inc.

FILES

/etc/inetd.conf

SEE ALSO

rwalld(1M), shutdown(1M), wall(1M), netgroup(4).

NAME

rwalld - network rwall server

SYNOPSIS

/usr/lib/netsvc/rwall/rpc.rwalld [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

rwalld is an RPC server that handles **rwall** requests (see *rwall*(1)). **rwalld** calls **wall** to send a message to all users logged into the host on which **rwalld** is running (see *wall*(1)).

inetd invokes **rwalld** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

rwalld recognizes the following options and command-line options:

-l *log_file* Log any errors to *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the log file includes date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file because enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if:

- **portmap** dies (see *portmap*(1M)),
- another **rpc.rwalld** registers with **portmap**, or
- **rpc.rwalld** becomes unregistered with **portmap**.

The **-n** option is more efficient because a new process is not launched for each RPC request. Note, this option is the default.

AUTHOR

rwalld was developed by Sun Microsystems, Inc.

SEE ALSO

inetd(1M), portmap(1M), rwall(1M), wall(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

rwhod - system status server

SYNOPSIS

/usr/lbin/rwhod [**-s**] [**-r**]

DESCRIPTION

rwhod is the server that maintains the database used by **rwho** and **ruptime** (see *rwho*(1) and *ruptime*(1)). **rwhod** sends status information to and receives status information from other nodes on the local network that are running **rwhod**.

rwhod is started at system boot time if the RWHOD variable is set to 1 in the file **/etc/rc.config.d/netdaemons**.

As an information sender, it periodically queries the state of the system and constructs status messages that are broadcast on a network.

As an information receiver, it listens for other **rwhod** servers' status messages, validates them, then records them in a collection of files located in the **/var/spool/rwho** directory.

By default, **rwhod** both sends and receives information. **rwhod** also supports the following options:

-s Configures server to be an information sender only.

-r Configures server to be an information receiver only.

Status messages are generated approximately once every three minutes. **rwhod** transmits and receives messages at the port indicated in the **who** service specification (see *services*(4)). The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];            /* tty name */
        char    out_name[8];            /* user id */
        long    out_time;               /* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order before transmission. System load averages are calculated from the number of jobs in the run queue over the last 1-, 5- and 15-minute intervals. The host name included is the one returned by the **gethostname()** system call (see *gethostname*(2)). The array at the end of the message contains information about the users logged in on the sending machine. This information includes the contents of the **utmp** entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line (see *utmp*(4)).

rwhod discards received messages if they did **not** originate at a **rwho** server's port, or if the host's name, as specified in the message, contains any unprintable ASCII characters.

Valid messages received by **rwhod** are placed in files named **whod.***hostname* in the **/var/spool/rwho** directory. These files contain only the most recent message in the format described above.

WARNINGS

rwhod does not relay status information between networks. Users often incorrectly interpret the server dying as a machine going down.

AUTHOR

rwhod was developed by the University of California, Berkeley.

FILES

/var/spool/rwho/whod.*	Information about other machines.

SEE ALSO

rwho(1), ruptime(1).

NAME

sa1, sa2, sadc - system activity report package

SYNOPSIS

/usr/lbin/sa/sa1 [*t n*]

/usr/lbin/sa/sa2 [**-ubdycwaqvmA**] [**-s** *time*] [**-e** *time*] [**-i** *sec*]

/usr/lbin/sa/sadc [*t n*] [*ofile*]

DESCRIPTION

System activity data can be accessed at the special request of a user (see *sar*(1)) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include CPU utilization counters, buffer usage counters, disk and tape I/O activity counters, tty device activity counters, switching and system-call counters, file-access counters, queue activity counters, and counters for inter-process communications.

sadc and shell procedures **sa1** and **sa2** are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *ofile* or to standard output. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time to mark the time at which the counters restart from zero. Executing the following command in a system startup script:

```
/usr/lbin/sa/sadc /var/adm/sa/sa‘date +%d‘
```

writes the special record to the daily data file to mark the system restart. Instructions for creating system startup scripts may be found in the 10.0 File System Layout White Paper, which is online in file **/usr/share/doc/filesys.ps**.

The shell script **sa1**, a variant of **sadc**, is used to collect and store data in binary file **/var/adm/sa/sa***dd* where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries, if placed in **crontab**, produce records every 20 minutes during working hours and hourly otherwise (see *cron*(1M)):

```
0 *    * * 0,6  /usr/lbin/sa/sa1
0 8-17 * * 1-5  /usr/lbin/sa/sa1 1200 3
0 18-7 * * 1-5  /usr/lbin/sa/sa1
```

The shell script **sa2**, a variant of **sar**, writes a daily report in file **/var/adm/sa/sar***dd*. The options are explained in *sar*(1). The following **crontab** entry reports important activities hourly during the working day:

```
5 18 * * 1-5  /usr/lbin/sa/sa2 -s 8:00 -e 18:01 -i 3600 -A
```

The structure of the binary daily data file is:

```
struct sa {
    struct sysinfo si;      /* see /usr/include/sys/sysinfo.h  */
    int  sztext;            /* current entries of text table   */
    int  szinode;           /* current entries of inode table  */
    int  szfile;            /* current entries of file table   */
    int  szproc;            /* current entries of proc table   */
    int  msztext;           /* size of text table  */
    int  mszinode;          /* size of inode table */
    int  mszfile;           /* size of file table  */
    int  mszproc;           /* size of proc table  */
    long  textovf;          /* cumul. overflows of text table  */
    long  inodeovf;         /* cumul. overflows of inode table */
    long  fileovf;          /* cumul. overflows of file table  */
    long  procovf;          /* cumul. overflows of proc table  */
    time_t  ts;             /* time stamp, seconds  */
    long  devio[NDEVS][4];  /* device info for up to NDEVS units */
#define IO_OPS  0           /* cumul. I/O requests  */
#define IO_BCNT 1           /* cumul. blocks transferred */
#define IO_ACT  2           /* cumul. drive busy time in ticks  */
#define IO_RESP 3           /* cumul. I/O resp time in ticks  */
};
```

S

FILES

`/tmp/sa.`*adrfl*	address file
`/var/adm/sa/sa`*dd*	daily data file
`/var/adm/sa/sar`*dd*	daily report file

SEE ALSO

cron(1M), sar(1), timex(1).

STANDARDS CONFORMANCE

`sa1`: SVID2, SVID3

`sa2`: SVID2, SVID3

`sadc`: SVID2, SVID3

NAME

sam - system administration manager

SYNOPSIS

/usr/sbin/sam [**-display** *display*] [**-f** *login*] [**-r**]

DESCRIPTION

The **sam** command starts a menu-driven System Administration Manager program (SAM) that makes it easy to perform system administration tasks with only limited, specialized knowledge of the HP-UX operating system. SAM discovers most aspects of a system's configuration through automated inquiries and tests. Help menus describe how to use SAM and perform the various management tasks. Context-sensitive help on the currently highlighted field is always available by pressing the **F1** function key. Status messages and a log file monitor keep the user informed of what SAM is doing.

Running SAM

SAM has been tuned to run in the Motif environment, but it can be run on text terminals as well. To run SAM in the Motif environment, be sure that Motif has been installed on your system, and that the **DISPLAY** environment variable is set to the system name on which the SAM screens should be displayed (or use the **-display** command line option).

Generally, SAM requires superuser (user **root**) privileges to execute successfully. However, SAM can be configured (through the use of "Restricted SAM"; see below) to allow subsets of its capabilities to be used by non-**root** users. When Restricted SAM is used, non-**root** users are promoted to **root** when necessary to enable them to execute successfully.

Options

sam recognizes the following options.

-display *display*	Set the **DISPLAY** value for the duration of the SAM session.
-f *login*	Execute SAM with the privileges associated with the specified *login*. When used in conjunction with **-r**, the Restricted SAM Builder is invoked and initialized with the privileges associated with the specified *login*. You must be a superuser to use this option. See "Restricted SAM" below for more information.
-r	Invoke the Restricted SAM Builder. This enables the system administrator to provide limited nonsuperuser access to SAM functionality. You must be a superuser to use this option. See "Restricted SAM" below for more information.

SAM Functional Areas

SAM performs system administration tasks in the following areas:

Auditing and Security (Trusted Systems)

- Set global system security policies
  - Maximum account inactivity period
  - Password generation policies
  - Null password usage and use of password restriction rules
  - Password aging
  - Maximum unsuccessful login attempts
  - Single-user boot authorization
  - Terminal security policies
- Turn the Auditing system on or off
- Set the parameters for the Audit Logs and Size Monitor
- View all or selected parts of the audit logs
- Modify (or view) which users, events, and/or system calls get audited
- Convert your system to a Trusted System
- Convert your system to a non-Trusted System

Backup and Recovery

- Interactively back up files to a valid backup device (cartridge tape, cartridge tape autochanger, magnetic tape, DAT, magneto-optical disk, or magneto-optical disk autochanger). The SAM interface is suspended so that you can read and/or respond to the interactive messages produced by **`fbackup`** (see *fbackup*(1M)).
- Recover files online from a valid backup device. The SAM interface is suspended so that you can read/respond to the interactive messages produced by **`frecover`** (see *frecover*(1M)).
- Add to, delete from, or view the automated backup schedule.
- Obtain a list of files from a backup tape.
- View various backup and recovery log files.

Disk and File Systems Management

- Add, configure, or unconfigure disk devices. This includes hard drives, floppy drives, CD-ROMs, magneto-optical devices, and disk arrays.
- Add, modify, or remove local file systems, or convert them to long file names.
- Configure HFS or VxFS file systems.
- Remote (NFS) file systems configuration, including:
  - Add, modify, or remove remote (NFS) file systems.
  - Allow or disallow access by remote systems to local file systems.
  - Modify RPC (Remote Procedure Call) services' security.
- Add, remove, or modify device or file system swap.
- Change the primary swap device.
- Add, modify, or remove dump devices.
- Examine, create, extend, or reduce a volume-group pool of disks.
- Create, extend or change number of mirrored copies of a logical volume and associated file system.
- Remove a logical volume or increase its size.
- Split or merge mirrored copies of a logical volume.
- Share or unshare volume groups (only on ServiceGuard clusters running MC/LockManager distributed lock-manager software).

Kernel and Device Configuration

- Change the configuration for I/O device and pseudo drivers.
- Modify operating system parameters.
- Modify dump device configuration in the kernel.
- Minimize kernel and system configuration to reduce memory usage (Series 700 only).
- Add or remove optional subsystems such as NFS, LAN, NS, CD-ROM, etc.
- Generate a new kernel.

Networks/Communications

- Configure one or more LAN cards.
- Configure ARPA services.
- Configure the Network File System (NFS).
- Configure X.25 card or cards and PAD (Packet Assembler/Disassembler) services (if X.25 has been purchased).

Peripheral Devices Management

- Administer the LP spooler or Distributed Print Services and associated printers and plotters (see "Printer and Plotter Management" below).

- Add, modify, or remove the configuration of disk devices.
- Add or remove terminals and modems.
- Configure terminal security policies (Trusted Systems only).
- Lock and unlock terminals (Trusted Systems only).
- Add or remove tape drives.
- Add or remove hardware interface cards and HP-IB instruments.
- View current configuration of peripherals and disk space information.

Printer and Plotter Management
SAM supports two methods for managing printers and plotters:

LP Spooler

- Add and remove local, remote, and networked printers and plotters to/from the LP spooler.
- Enable and disable printers and plotters from printing requests accepted by the LP spooler.
- Accept and reject requests for printers, plotters, and print classes.
- Modify the fence priority of printers and plotters.
- Set the system default print destination.
- Start and stop the LP scheduler.

HP Distributed Print Service (HPDPS)

- Add and remove physical printers (parallel, serial, or network interface and remote printers), logical printers, print queues, spoolers, and supervisors.
- Enable and disable logical printers, print queues, and physical printers to accept print jobs.
- Pause and resume print queues, physical printers, and print jobs.
- Start and stop spoolers and supervisors
- Modify attributes of physical printers, logical printers, print queues, spoolers, and supervisors.
- Remove a single print job or all print jobs assigned to a physical printer, logical printer, print queue, spooler or supervisor.

Process Management

- Kill, stop or continue processes.
- Change the nice priority of processes.
- View the current status of processes.
- Schedule periodic tasks via cron.
- View current periodic (cron) tasks.
- Run performance monitors.
- Display system properties such as: machine model and ID; number of installed processors, their version and speed; operating-system release version; swap statistics, real, physical, and virtual memory statistics; network connection information.

Remote Administration

- Configure remote systems for remote administration.
- Execute SAM on systems configured for remote administration.

Routine Tasks

- Shut down the system.
- View and remove large files. Specify size and time-since-accessed of large files to display or remove.
- View and remove unowned files. Specify size and time-since-accessed of unowned files to display or remove.

- View and remove core files.
- View and trim ASCII or non-ASCII log files. Add or remove files from the list of files to monitor. Set recommended size for trimming.

User and Group Account Management

- Add, remove, view, and modify user accounts.
- Remove or reassign ownership of files belonging to removed or modified user accounts.
- Modify a user account's group membership.
- Set up password aging for a user account.
- Add, remove, view, and modify groups.
- Customize adding and removing users by specifying steps to be performed before and/or after SAM does its processing for the task. The **`Task Customization`** action items in SAM Users and Groups leads you through this capability. See "Customizing SAM Tasks" below for more information.
- Deactivate and reactivate user accounts.
- Manage trusted system security policies on a per-user basis. The policies that can be managed include:
  - Account lifetime
  - Maximum account inactivity period
  - Password generation policies
  - Null password usage and use of password restriction rules
  - Maximum password length
  - Password aging
  - Maximum unsuccessful login attempts
  - Generation of admin numbers for new or reactivated accounts
  - Single-user boot authorization
  - Authorized login times

Adding New Functionality to SAM

You can easily add stand-alone commands, programs, and scripts to SAM. SAM is suspended while the executable program is running. When it finishes, the SAM interface is restored. You can also write your own help screen for each menu item you create. To add functionality to SAM, select the "Add Custom Menu Item" or "Add Custom Menu Group" action items from the SAM Areas menu. (Note that the new item is added to the hierarchy that is currently displayed, so you need to navigate to the desired hierarchy before adding the item.)

File System Protection When Removing Users

When removing users or files from a system, there is always the unfortunate possibility that the wrong user may be removed or that files belonging to a user who is removed are deleted inadvertently during the removal process. For example, user **`bin`** is the owner of (from the operating system's perspective) the majority of the executable commands on the system. Removing this user would obviously be disastrous. On the other hand, suppose user **`joe`** owns all of the files comprising the test suite for a project. It may be appropriate to remove **`joe`**, but the test suite should be left intact and assigned to a new owner. SAM provides two features to help protect against inadvertent removal of users or files when removing users:

- When prompting for the name of a user to remove from the system, SAM checks the name given against a list of names specified in the file **`/etc/sam/rmuser.excl`**. If the name matches one within the file, SAM does not remove the user.
- When SAM removes a user, all files (or a subset thereof) for that user are also removed, unless the ownership is given to another user. Before removing a file belonging to the user, SAM checks to see if the file resides in a path that has been excluded from removal. SAM uses the file **`/etc/sam/rmfiles.excl`** to determine which paths have been excluded from removal. So, for example, if the path **`/users/joe/test`** is named in the file, SAM will not remove any files residing beneath that directory. SAM logs a list of all files it removes in the file **`/var/tmp/sam_remove.log`**.
- SAM does not remove or reassign any files if the user being removed has the same user ID as another user on the system.

Files **/etc/sam/rmuser.excl** and **/etc/sam/rmfiles.excl** can be edited to contain users and directories that you want to exclude from removal by SAM.

Customizing SAM Tasks

You can customize the following SAM tasks:

- Add a New User Account to the System
- Remove a User Account from the System

For each of these tasks, you can specify steps you want performed before and/or after SAM does its processing for the task. Before SAM performs one of the tasks, it checks to see if a pretask step (executable file) was defined. If so, SAM invokes the executable, passes it a set of parameters (see below), and waits for its completion. You can halt SAM's processing of a task by exiting from your executable with a nonzero value (for example if an error occurs during execution of your executable).

After SAM has finished processing, it checks for a posttask step, performing the same type of actions as for the pretask step.

The executable file must have these characteristics:

- Must be owned by root.
- Must be executable only by root, and if writable, only by root.
- Must reside in a directory path where all the directories are writable only by owner.
- The full path name of the executable file must be given in the SAM data entry form.

The same parameters are passed from SAM to your program for both the pretask and posttask steps. Here are the parameters passed for each task:

- **Add a New User Account to the System**

  -l *login_name*
  -v *user_id*
  -h *home_directory*
  -g *group*
  -s *shell*
  -p *password*
  -R *real_name*
  -L *office_location*
  -H *home_phone*
  -O *office_phone*

  The file **/usr/sam/lib/ct_adduser.ex** contains an example of how to process these parameters.

- **Remove a User Account From the System**

  There can be one of three possible parameters, depending on the option selected in the SAM data entry form. The parameter can be *one* of these three:

-f *user_name*	Option supplied when all of *user_name*'s files are being removed.
-h *user_name*	Option supplied when *user_name*'s home directory and files below it are being removed.
-n *new_owner user_name*	Option supplied when all of *user_name*'s files are being assigned to *new_owner*.

  The file **/usr/sam/lib/ct_rmuser.ex** contains an example of how to process these parameters.

Restricted SAM

SAM can be configured to provide a subset of its functionality to certain users or groups of users. It can also be used to build a template file for assigning SAM access restrictions on multiple systems. This is done through the Restricted SAM Builder. System administrators access the Restricted SAM Builder by invoking SAM with the **-r** option (see "Options" above). In the Builder, system administrators may assign subsets of SAM functionality on a per-user or per-group basis. Once set up, the **-f** option (see "Options" above) can then be used by system administrators to verify that the appropriate SAM functional areas, and only those areas, are available to the specified user.

A nonroot user that has been given Restricted SAM privileges simply executes **/usr/sbin/sam** and sees only those areas the user is privileged to access. For security reasons, the "List" and "Shell Escape" choices are not provided. (Note that some SAM functional areas require the user to be promoted to root in order to execute successfully. SAM does this automatically as needed.)

SAM provides a default set of SAM functional areas that the system administrator can assign to other users. Of course, system administrators are able to assign custom lists of SAM functional areas to users as necessary.

SAM Logging

All actions taken by SAM are logged into the SAM log file **/var/sam/log/samlog**. The log entries in this file can be viewed via the SAM utility **samlog_viewer** (see *samlog_viewer*(1M)). **samlog_viewer** can filter the log file by user name, by time of log entry creation, and by level of detail.

The "Options" menu in the SAM Areas Menu enables you to start a log file viewer and to control certain logging options. These options include whether or not SAM should automatically start a log file viewer whenever SAM is executed, whether or not SAM should trim the log file automatically, and what maximum log file size should be enforced if automatic log file trimming is selected.

VT320 Terminal Support

Because the VT320 terminal has predefined local functions for keys labeled as **F1**, **F2**, **F3** and **F4**, users should use following mapping when they desire to use function keys:

HP or Wyse60	**VT320 or HP 700/60 in VT320 mode**
F1	**PF2** *(1)*
F2	**PF1** *(1)*
F3	**spacebar**
F4	**PF3** *(1)*
F5	**F10**, [EXIT], **F5** *(2)*
F6	none
F7	**F18**, first unlabeled key to right of **Pause/Break** *(2)*
F8	**F19**, second unlabeled key to right of **Pause/Break** *(2)*

(1) See the "Configuration: HP 700/60 in DEC mode, or DEC terminals with PC-AT-type keyboard" subsection below.

(2) When using PC-AT keyboard with HP 700/60 in VT320 mode.

Since DEC terminals do not support the softkey menu, that menu is not displayed on those terminals.

Many applications use **TAB** for forward navigation (moving from one field to another) and **shift-TAB** for backward navigation. Users having DEC terminals or using terminals in DEC emulation modes such as VT100 or VT320 may note that these terminals/emulators may produce the same character for **TAB** and **shift-TAB**. As such, it is impossible for an application to distinguish between the two and both of them are treated as if the **TAB** key was pressed. This presents an inconvenience to users if they want to go backward. In most cases, they should complete rest of the input fields and get back to the desired field later.

S

VT100 Terminal Support

VT100 does not allow the **F1**–**F8** function keys to be configured. Therefore, the following keyboard mappings apply to VT100 terminals:

HP or Wyse60	**VT100 or HP 700/60 in VT100 mode**
F1	**PF2** *(1)*
F2	**PF1** *(1)*
F3	**spacebar**
F4	**PF3**, **spacebar** or **PF3**, **=** *(1)*
F5	**Return**
F6	none
F7	none
F8	none

(1) See the "Configuration: HP 700/60 in DEC mode, or DEC terminals with PC-AT-type keyboard" subsection below.

See the comments on softkeys and **TAB** keys in the "VT320 Terminal Support" subsection above.

Configuration: HP 700/60 Terminal in DEC Mode, or DEC Terminal with PC-AT-Type Keyboard

Customers using the following configuration may want to be aware of the following keyboard difference.

It may be possible for a user with the "HP 700/60 terminal in DEC mode, or DEC terminal with PC-AT-type keyboard" configuration to be told to press function key **F1** through **F4** to achieve some desired result. For an HP 700/60 terminal in DEC mode or DEC terminals, these functions keys may be mapped onto **PF1**–**PF4** keys. However, the PC-AT-type keyboard does not provide **PF1**–**PF4** keys, as does the DEC/ANSI keyboard.

Key	**Maps to**
Num Lock	**PF1**
/	**PF2**
*****	**PF3**
–	**PF4**

The **Num Lock**, **/**, *****, and – keys are located on the keyboard, in a row above the number pad on the right side of the keyboard. Please note that although this keyboard is called a PC-AT-type keyboard, it is supplied by HP. A PC-AT-type keyboard can be recognized by location of ESC key at the left-top of the keyboard.

Wyse60 Terminal Support

On Wyse60, use the **DEL** key (located next to **Backspace**) to backspace. On an HP 700/60 with a PC-AT-type keyboard in Wyse60 mode, the **DEL** key is located in the bottom row on the number pad.

Wyse60 terminals provide a single line to display softkey labels unlike HP terminals which provide two lines. Sometimes this may result in truncated softkey labels. For example, the `Help on Context` label for **F1** may appear as `Help on C`. Some standard labels for screen-oriented applications, such as SAM and `swinstall` are as follows:

The SAM label:	**May appear on the Wyse60 as:**
`Help On Context`	`Help On C`
`Select/Deselect`	`Select/D`
`Menubar on/off`	`Menubar`

DEPENDENCIES

SAM runs in an X Window environment as well as on the following kinds of terminals or terminal emulators:

- HP-compatible terminal with programmable function keys and on-screen display of function key labels.
- VT-100 and VT-320
- WY30 and WY60

Depending on what other applications are running concurrently with SAM, more swap space may be required. SAM requires the following amounts of internal memory:

8 MB	If using terminal based version of SAM.
16 MB	If using Motif X Window version of SAM.

For more detailed information about how to use SAM on a terminal, see the *Managing Systems and Workgroups* manual.

AUTHOR

`sam` was developed by HP.

FILES

`/etc/sam/custom`	Directory where SAM stores user privileges.
`/etc/sam/rmfiles.excl`	File containing a list of files and directories that are excluded from removal by SAM.
`/etc/sam/rmuser.excl`	File containing a list of users that are excluded from removal by SAM.

S

`/usr/sam/bin`	Directory containing executable files, which can be used outside of any SAM session.
`/usr/sam/help/$LANG`	Directory containing SAM language specific online help files.
`/usr/sam/lbin`	Directory containing SAM executables, which are intended only for use by SAM and are not supported in any other context.
`/usr/sam/lib`	Directory for internal configuration files.
`/var/sam`	Directory for working space, including lock files (if a SAM session dies, it may leave behind a spurious lock file), preferences, logging, and temporary files.
`/var/sam/log/samlog`	File containing unformatted SAM logging messages. This file should not be modified by users. Use **`samlog_viewer`** to view the contents of this file (see *samlog_viewer*(1M)).
`/var/sam/log/samlog.old`	Previous SAM log file. This file is created by SAM when **`/var/sam/log/samlog`** is larger than the user specified limit. Use **`samlog_viewer`** with its **`-f`** option to view the contents of this file (see *samlog_viewer*(1M)).

SEE ALSO

samlog_viewer(1M).

Managing Systems and Workgroups
Installing and Administering ARPA Services
Installing and Administering LAN/9000
Installing and Administering NFS Services
Installing and Administering Network Services
Installing and Administering X.25/9000

S

NAME

sar - system activity reporter

SYNOPSIS

sar [**-ubdycwaqvmAMS**] [**-o** *file*] *t* [*n*]

sar [**-ubdycwaqvmAMS**] [**-s** *time*] [**-e** *time*] [**-i** *sec*] [**-f** *file*]

DESCRIPTION

In the first form above, **sar** samples cumulative activity counters in the operating system at *n* intervals of *t* seconds. If the **-o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second form, with no sampling interval specified, **sar** extracts data from a previously recorded *file*, either the one specified by **-f** option or, by default, the standard system activity daily data file **/var/adm/sa/sa***dd* for the current day *dd*. The starting and ending times of the report can be bounded via the **-s** and **-e** *time* arguments of the form *hh*[:*mm*[:*ss*]]. The **-i** option selects records at *sec*-second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by option:

-u Report CPU utilization (the default); portion of time running in one of several modes. On a multi-processor system, if the **-M** option is used together with the **-u** option, per-CPU utilization as well as the average CPU utilization of all the processors are reported. If the **-M** option is not used, only the average CPU utilization of all the processors is reported:

cpu	cpu number (only on a multi-processor system with the **-M** option);
%usr	user mode;
%sys	system mode;
%wio	idle with some process waiting for I/O (only block I/O, raw I/O, or VM pageins/swapins indicated);
%idle	otherwise idle.

-b Report buffer activity:

bread/s	Number of physical reads per second from the disk (or other block devices) to the buffer cache;
bwrit/s	Number of physical writes per second from the buffer cache to the disk (or other block device);
lread/s	Number of reads per second from buffer cache;
lwrit/s	Number of writes per second to buffer cache;
%rcache	Buffer cache hit ratio for read requests e.g., 1 − bread/lread;
%wcache	Buffer cache hit ratio for write requests e.g., 1 − bwrit/lwrit;
pread/s	Number of reads per second from character device using the **physio()** (raw I/O) mechanism;
pwrit/s	Number of writes per second to character device using the **physio()** (i.e., raw I/O) mechanism; mechanism.

-d Report activity for each block device, e.g., disk or tape drive. One line is printed for each device that had activity during the last interval. If no devices were active, a blank line is printed. Each line contains the following data:

device Logical name of the device and its corresponding instance. Devices are categorized into the following four device types:

disk1 – HP-IB disks (CS/80)
disk2 – CIO HP-FL disks (CS/80)
disk3 – SCSI and NIO FL disks
sdisk – SCSI disks;

%busy Portion of time device was busy servicing a request;

avque Average number of requests outstanding for the device;

S

r+w/s	Number of data transfers per second (read and writes) from and to the device;
blks/s	Number of bytes transferred (in 512-byte units) from and to the device;
avwait	Average time (in milliseconds) that transfer requests waited idly on queue for the device;
avserv	Average time (in milliseconds) to service each transfer request (includes seek, rotational latency, and data transfer times) for the device.

-y Report tty device activity:

rawch/s	Raw input characters per second;
canch/s	Input characters per second processed by **canon()**;
outch/s	Output characters per second;
rcvin/s	Receive incoming character interrupts per second;
xmtin/s	Transmit outgoing character interrupts per second;
mdmin/s	Modem interrupt rate (not supported; always 0).

-c Report system calls:

scall/s	Number of system calls of all types per second;
sread/s	Number of **read()** and/or **readv()** system calls per second;
swrit/s	Number of **write()** and/or **writev()** system calls per second;
fork/s	Number of **fork()** and/or **vfork()** system calls per second;
exec/s	Number of **exec()** system calls per second;
rchar/s	Number of characters transferred by read system calls block devices only) per second;
wchar/s	Number of characters transferred by write system calls (block devices only) per second.

-w Report system swapping and switching activity:

swpin/s	Number of process swapins per second;
swpot/s	Number of process swapouts per second;
bswin/s	Number of 512-byte units transferred for swapins per second;
bswot/s	Number of 512-byte units transferred for swapouts per second;
pswch/s	Number of process context switches per second.

-a Report use of file access system routines:

iget/s	Number of file system **iget()** calls per second;
namei/s	Number of file system **lookuppn()** (pathname translation) calls per second;
dirblk/s	Number of file system blocks read per second doing directory lookup.

-q Report average queue length while occupied, and percent of time occupied. On a multi-processor machine, if the **-M** option is used together with the **-q** option, the per-CPU run queue as well as the average run queue of all the processors are reported. If the **-M** option is not used, only the average run queue information of all the processors is reported:

cpu	cpu number (only on a multi-processor system and used with the **-M** option)
runq-sz	Average length of the run queue(s) of processes (in memory and runnable);
%runocc	The percentage of time the run queue(s) were occupied by processes (in memory and runnable);
swpq-sz	Average length of the swap queue of runnable processes (processes swapped out but ready to run);

%swpocc The percentage of time the swap queue of runnable processes (processes swapped out but ready to run) was occupied.

-v Report status of text, process, inode and file tables:

text-sz (Not Applicable);

proc-sz The current-size and maximum-size of the process table;

inod-sz The current-size and maximum-size of the inode table (inode cache);

file-sz The current-size and maximum-size of the system file table;

text-ov (Not Applicable);

proc-ov The number of times the process table overflowed (number of times the kernel could not find any available process table entries) between sample points;

inod-ov The number of times the inode table (inode cache) overflowed (number of times the kernel could not find any available inode table entries) between sample points;

file-ov The number of times the system file table overflowed (number of times the kernel could not find any available file table entries) between sample points.

-m Report message and semaphore activities:

msg/s Number of System V **msgrcv()** calls per second;

sema/s Number of System V **semop()** calls per second;

select/s Number of System V **select()** calls per second. This value will only be reported if the "-S" option is also explicitly specified.

-A Report all data. Equivalent to **-udqbwcayvm**.

-M Report the per-processor data on a multi-processor system when used with **-q** and/or **-u** options. If the **-M** option is not used on a multi-processor system, the output format of the **-u** and **-q** options is the same as the uni-processor output format and the data reported is the average value of all the processors.

EXAMPLES

Watch CPU activity evolve for 5 seconds:

```
sar 1 5
```

Watch CPU activity evolve for 10 minutes and save data:

```
sar -o temp 60 10
```

Review disk and tape activity from that period later:

```
sar -d -f temp
```

Review cpu utilization on a multi-processor system later:

```
sar -u -M  -f temp
```

WARNINGS

Users of **sar** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

FILES

/var/adm/sa/sa*dd* daily data file, where *dd* is two digits representing the day of the month.

SEE ALSO

sa1(1M).

STANDARDS CONFORMANCE

sar: SVID2, SVID3

NAME

savecrash - save a crash dump of the operating system

SYNOPSIS

/sbin/savecrash [**-cflPrvzZ**] [**-D** *dumpdevice* **-O** *offset*] [**-m** *minfree*]
[**-s** *chunksize*] [**-w** **NOSWAP**|**SWAPEACH**|**SWAPEND**] [*dirname*]

DESCRIPTION

savecrash saves the crash dump information of the system (assuming one was made when the system crashed) and writes a reboot message in the shutdown log file.

dirname is the name of the existing directory in which to store the crash dump; the default is **/var/adm/crash**.

savecrash saves the crash image and related files in the directory *dirname***/crash.***n*. The trailing *n* in the directory name is a number that increases by one every time **savecrash** is run with the same *dirname*. This number is kept in the file *dirname***/bounds**, which is created if it does not already exist.

Usually, **savecrash** creates the **INDEX** file in the crash directory from the crash dump header, copies all kernel modules that were loaded in memory at the time of the crash, and copies all dump device contents into crash image files.

When **savecrash** writes out a crash dump directory, it checks the space available on the file system containing *dirname*. **savecrash** will not use that portion of the file system space which is reserved for the superuser. Additional space on the file system can be reserved for other uses with **-m** *minfree*, where *minfree* is the amount of additional space to reserve. This option is useful for ensuring enough file system space for normal system activities after a panic.

If there is insufficient space in the file system for the portions of the crash dump that need to be saved, **savecrash** will save as much as will fit in the available space. (Priority is given to the index file, then to the kernel module files, and then to the physical memory image.) The dump will be considered saved, and **savecrash** will not attempt to save it again, unless there was insufficient space for *any* of the physical memory image. (See the description of option **-r**.)

savecrash also writes a reboot message in the shutdown log file (**/etc/shutdownlog**), if one exists. (If a shutdown log file does not exist, **savecrash** does not create one.) If the system crashes as a result of a kernel panic, **savecrash** also records the panic string in the shutdown log.

By default, when the primary paging device is not used as one of the dump devices or after the crash image on the primary paging device has been saved, **savecrash** runs in the background. This reduces system boot-up time by allowing the system to be run with only the primary paging device.

It is possible for dump devices to be used also as paging devices. If **savecrash** determines that a dump device is already enabled for paging, and that paging activity has already taken place on that device, a warning message will indicate that the dump may be invalid. If a dump device has not already been enabled for paging, **savecrash** prevents paging from being enabled to the device by creating the file **/etc/savecore.LCK**. **swapon** does not enable the device for paging if the device is locked in **/etc/savecore.LCK** (see *swapon*(1M) for more details). As **savecrash** finishes saving the image from each dump device, it updates the **/etc/savecore.LCK** file and optionally executes **swapon** to enable paging on the device.

Options

-c Mark the dump in the dump device as saved, without performing any other action. The **-c** option is useful for manually inhibiting dump actions called by **/sbin/init.d/savecrash**.

-f Run **savecrash** in the foreground only. By default, **savecrash** runs in the background when the primary paging device does not contain an unsaved portion of the crash image. Turning this option on increases system boot-up time, but guarantees that the dump has been saved when control returns to the caller.

-l Logs the panic information to **/etc/shutdownlog** as described above, but does not actually save the dump. The dump is marked as saved so that future invocations of **savecrash** do not create duplicate log entries.

-P Only preserves swap-endangered dump device contents into crash image files. Swap-endangered dump devices are those devices that are also configured as swap devices by the system. If all dump devices are configured as swap devices, the entire dump will be preserved in the crash directory. If no swap devices are used as dump devices (dedicated dump devices), only the **INDEX** file and kernel modules will be copied into the crash directory.

-r Resaves a dump that a previous invocation of **savecrash** has marked as already saved. This is useful if the first invocation did ran out of space, and enough space has since been freed to try again.

-v Enables additional progress messages and diagnostics.

-z **savecrash** will compress all physical memory image files and kernel module files in the dump directory.

-Z **savecrash** will not compress any files in the dump directory.

If neither **-z** nor **-Z** is specified, **savecrash** will determine whether or not to compress files based on the file sizes involved and the amount of available file system space. Compression may take place in the background after the image has been saved or in the foreground while saving the image, depending on available disk space.

-D *dumpdevice*
dumpdevice is the name of the device containing the header of the raw crash image. The console messages from the time of the panic will identify the major and minor numbers of this device. This option, in combination with **-O**, can be used to tell **savecrash** where to find the dump in the rare instances that **savecrash** doesn't know where to look.

-O *offset*
offset is the offset in kBytes, relative to the beginning of the device specified with **-D** above, of the header of the raw crash image. The console messages from the time of the panic will identify this offset. This option, in combination with **-D**, can be used to tell **savecrash** where to find the dump in the rare instances that **savecrash** doesn't know where to look.

-m *minfree*
minfree is the amount of free space (in kBytes) that must remain free for ordinary user files after **savecrash** completes, in addition to space reserved for the superuser. If necessary, only part of the dump will be saved to achieve this requirement. *minfree* may be specified in bytes (**b**), kilobytes (**k**), megabytes (**m**), or gigabytes (**g**). The default *minfree* value is zero, and the default unit is kilobytes.

-s *chunksize*
chunksize is the size (default kBytes) of a single physical memory image file before compression. The kByte value must be a multiple of page size (divisible by 4) and between 64 and 1048576. *chunksize* may be specified in units of bytes (**b**), kilobytes (**k),** megabytes (**m**), or gigabytes (**g**). Larger numbers increase compression efficiency at the expense of both **savecrash** time and debugging time. If **-s** is not specified, a default is chosen based on the physical memory size and the amount of available file system space.

-w *opt* Defines the interaction between **savecrash** and **swapon**. *opt* can be one of the following values:

NOSWAP Do not run **swapon** from **savecrash**.

SWAPEACH (default) Call **swapon** each time **savecrash** finishes saving the image from each dump device. This option provides the most efficient use of paging space.

SWAPEND Only call **swapon** when **savecrash** finishes saving the image file from *all* dump devices. If this option is used, no additional paging space other than the primary paging space is available until the complete crash dump image is saved. This option provides a second chance to retrieve the crash image if **savecrash** fails on first attempt.

For compatibility with earlier **savecore(1M)** syntax, the values of **0**, **1** and **2** can be used in plase of **NOSWAP**, **SWAPEACH**, and **SWAPEND**, respectively. This usage is obsolescent.

RETURN VALUE

Upon exit, **savecrash** returns the following values:

0 A crash dump was found and saved, *or* **savecrash** has preserved dump information from the primary swap device and is continuing to run in the background to complete its tasks.
1 A crash dump could not be saved due to an error.
2 No crash dump was found to save.

3 A partial crash dump was saved, but there was insufficient space to preserve the complete dump.
4 The *savecrash* process continued in the background, see the **INDEX** file for actual results.

WARNINGS

savecrash relies on the expectation that device numbers have the same meaning (point to the same devices) at the time the system dumps and at the time the dump is saved. If, after a crash, the system was booted from a different boot device in order to run **savecrash**, it is possible that this expectation will not be met. If so, **savecrash** may save an incomplete or incorrect dump or may fail to save a dump at all. Such cases cannot be reliably detected, so there may be no warning or error message.

If **savecrash** encounters an error while running in the background (such as running out of space), it will not be easily detectable by the caller. If the caller must ensure that the **savecrash** operation was successful, for example before writing to a dump device, the caller should specify **-f** to force **savecrash** to run in the foreground, and should then examine the exit status of the **savecrash** process when it finishes.

AUTHOR

savecrash was developed by HP and the University of California, Berkeley.

FILES

/etc/shutdownlog	shutdown log
/etc/rc.config.d/savecrash	savecrash startup configuration file
/sbin/init.d/savecrash	savecrash startup file
dirname/**bounds**	crash dump number
/stand/vmunix	default kernel image saved by savecrash

SEE ALSO

adb(1), crashutil(1M), swapon(1M).

NAME

scn - scan an HP SCSI disk array LUN for parity consistency

SYNOPSIS

scn -i *number_of_initiators device_file*

DESCRIPTION

scn scans the disks of the LUN in an HP SCSI disk array identified by the character device file *device_file*. The parity information for any block reporting inconsistent parity is corrected by an immediate call to **rpr**.

RETURN VALUE

scn returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **scn**
- SCSI (device level) communications
- system calls

Error messages generated by scn:

usage: scn -i <num initiators> <special>
scn encountered an error in command syntax. Enter the command again with all required arguments, in the order shown.

scn: LUN # too big
The LUN number, derived from the device file name, is out of range.

scn: Not a raw file
Utilities must be able to open the device file for raw access. That is, you must specify a character device file rather than a block device file.

scn: LUN does not exist
The addressed LUN is not configured, and is not known to the array controller.

scn: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

scn uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **scn** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To scan the LUN at **/dev/rdsk/c2t0d0** on a Series 800 computer with two hosts (initiators) attached:

```
scn -i 2 /dev/rdsk/c2t0d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

scn was developed by HP.

SEE ALSO

rpr(1M).

NAME

scsictl - control a SCSI device

SYNOPSIS

scsictl [**-akq**] [**-c** *command*]... [**-m** *mode*[**=***value*]]... *device*

DESCRIPTION

The **scsictl** command provides a mechanism for controlling a SCSI device. It can be used to query mode parameters, set configurable mode parameters, and perform SCSI commands. The operations are performed in the same order as they appear on the command line.

device specifies the character special file to use.

Options

scsictl recognizes the following options.

-a Display the status of all mode parameters available, separated by semicolon-blank (**;**) or newline.

-c *command*
Cause the device to perform the specified command. *command* can be one of the following:

erase For magneto-optical devices that support write without erase, this command can be used to pre-erase the whole surface to increase data throughput on subsequent write operations. This command maintains exclusive access to the surface during the pre-erasure.

sync_cache
For devices that have an internal write cache, this command causes the device to flush its cache to the physical medium.

-k Continue processing arguments even after an error is detected. The default behavior is to exit immediately when an error is detected.

Command line syntax is always verified for correctness, regardless of the **-k** option. Improper command line syntax causes **scsictl** to exit without performing any operations on the device.

-m *mode* Display the status of the specified *mode* parameter. *mode* can be one of the following:

immediate_report
For devices that support immediate reporting, this mode controls how the device responds to write requests. If immediate report is enabled (**1**), write requests can be acknowledged before the data is physically transferred to the media. If immediate report is disabled (**0**), the device is forced to await the completion of any write request before reporting its status.

ir Equivalent to **immediate_report**.

queue_depth
For devices that support a queue depth greater than the system default, this mode controls how many I/Os the driver will attempt to queue to the device at any one time. Valid values are (**1**–**255**). Some disk devices will not support the maximum queue depth settable by this command. Setting the queue depth in software to a value larger than the disk can handle will result in I/Os being held off once a QUEUE FULL condition exists on the disk.

-m *mode***=***value*
Set the mode parameter *mode* to *value*. The available mode parameters and values are listed above.

Mode parameters that take only a binary value (**1** or **0**) can also be specified as either **on** or **off**, respectively.

-q Suppress the labels that are normally printed when mode parameters are displayed. Mode parameter values are printed in the same order as they appear on the command line, separated by semicolon-blank (**;**) or newline.

Mode parameters and commands need only be specified up to a unique prefix. When abbreviating a mode parameter or command, at least the first three characters must be supplied.

DIAGNOSTICS

Diagnostic messages are generally self-explanatory.

EXAMPLES

To display all the mode parameters, turn **immediate_report** on, and redisplay the value of **immediate_report**:

```
scsictl -a -m ir=1 -m ir /dev/rdsk/c0t6d0
```

producing the following output:

```
immediate_report = 0; queue_depth = 8; immediate_report = 1
```

The same operation with labels suppressed:

```
scsictl -aq -m ir=1 -m ir /dev/rdsk/c0t6d0
```

produces the following output:

```
0; 8; 1
```

WARNINGS

Not all devices support all mode parameters and commands listed above. Changing a mode parameter may have no effect on such a device.

Issuing a command that is not supported by a device can cause an error message to be generated.

scsictl is not supported on sequential-access devices using the tape driver.

The **immediate_report** mode applies to the entire device; the section number of the *device* argument is ignored.

To aid recovery, immediate reporting is not used for writes of file system data structures that are maintained by the operating system, writes to a hard disk (but not a magneto-optical device) through the character-device interface, or writes to regular files that the user has made synchronous with **O_SYNC** or **O_DSYNC** (see *open*(2) and *fcntl*(2)).

DEPENDENCIES

disc3

When the system is rebooted, the **disc3** driver always resets the value of the **immediate_report** mode parameter to **off**. If **ioctl()** or **scsictl** is used to change the setting of immediate reporting on a SCSI device, the new value becomes the default setting upon subsequent configuration (e.g., opens) of this device and retains its value across system or device powerfail recovery. However, on the next system reboot, the **immediate-report** mode parameter is again reset to the value of the tunable system parameter, **default_disk_ir**. This is set in the *system_file* used to create the HP-UX system by the **config** command (see *config*(1M)).

S

sdisk

If **ioctl()** or **scsictl** is used to change the setting of immediate reporting on a SCSI device, the new value becomes the default setting upon subsequent configuration (e.g., opens) of this device until the "last close" of the device, that is, when neither the system nor any application has the device open (for example, unmounting a file system via **umount** and then mounting it again via **mount** (see *mount*(1M)). On the next "first open", the **immediate-report** mode parameter is again reset to the value of the tunable system parameter, **default_disk_ir**. This is set in the *system_file* used to create the HP-UX system by the **config** command (see *config*(1M)).

SEE ALSO

config(1M), diskinfo(1M), fcntl(2), open(2).

NAME

see - access bytes in the HP SCSI disk array controller EEPROM

SYNOPSIS

see -d special

see -b *byte_number* **-h** *hex_byte device_file*

DESCRIPTION

see displays, or changes bytes in the controller EEPROM of the HP SCSI disk array associated with device file *device_file*. A 64-byte area in the EEPROM is accessible to the user. Although the command is directed to a single LUN, the EEPROM settings affect all the LUNs of the device.

Options

-d Display only. Displays the current values of the bytes in the accessible portion of the EEPROM.

-b *byte_number* **-v** *hex_byte*
Loads the hexadecimal value **hex_byte** into the decimal byte **byte_number** of the user accessible 64-byte region in the EEPROM.

BYTE DESCRIPTION

The following list of user accessible bytes in the EEPROM, and their default values is provided for informational purposes only. Changing the values can result in "incorrect" controller behavior with respect to HP SCSI disk array utilities, and other support software. See WARNINGS.

byte	meaning	C2425/7 value	C2430 value
0	enable synchronous negotiation	0x00	0x00
1	enable wide negotiation	0x00	0x00
2	spin-up algorithm	0x01	0x01
3	spin-up delay	0x32	0x1e
4	ready timeout	0x0a	0x17
5	host command delay at power on	0x00	0x00
6	firmware drive cmd timeout value	0x64	0x64
7	default RAID level	0x00	0x05
8	option control bits MSB	0x00	0x00
9	option control bits LSB	0x27	0x53
10	sense key for drive failures	0x06	0x06
11	inquiry data byte 7	0x12	0x32
12	ROM sequence control bits	0x01	0x01
13	synchronization control bits	0x02	0x02
14	inquiry revision level format	0x00	0x00
15	diagnostic self-test options	0x01	0x01
16	host command delay for bus reset	0x00	0x00
17	inquiry unconfigured device type	0x20	0x20
18	software command timeout value	0x14	0x14
19	software command timeout actions	0x07	0x07
20	drive bus reset to ready wait	0x08	0x08
21	host delay after data phase	0x00	0x00
22	drive scan disabled channel (MSB)	0x00	0x00
23	drive scan disabled channel (LSB)	0x00	0x00
24	time to asynchronous event	0x00	0x00
25	fan polling interval	0x00	0x00
26	power supply polling interval	0x00	0x00
27	reserved	0x00	0x00
28	Error Reporting Options (MSB)	0x01	0x01
29	Error Reporting Options (LSB)	0x00	0x00
30-63	reserved	0x00	0x00

RETURN VALUE

see returns the following values:

0 Successful completion.

-1 Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **see**
- SCSI (device level) communications
- system calls

Error messages generated by see:

usage: see <-d | -b <byteno> -v <hex byte>> <special>
An error in command syntax has occurred. Re-enter command with the required arguments, in the order shown.

see: Arg out of range
One of the arguments has exceeded its maximum or minimum size, or is incorrect in form. Check the size and form of each argument.

see: device busy
To ensure that **see** does not modify a disk array that is being used by another process, **see** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

see: LUN # too big
The LUN number, which is derived from the device special file name, is out of range.

see: LUN does not exist
The addressed LUN is not configured, and thus is not known to the array controller.

see: Not a raw file
Utilities must be able to open the device file for raw access.

see: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

see: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

see uses the following system calls:

stat(), **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **see** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To display the values of the accessible EEPROM bytes on HP SCSI disk array **/dev/rdsk/c2t6d0** on a Series 700:

```
see -d /dev/rdsk/c2t6d0
```

WARNING

Changing the values of EEPROM bytes can result in incorrect controller behavior with respect to utilities and support software that may not be immediately obvious. Also, the EEPROM can only be written to a finite number of times, and if its write count is exceeded, it must be replaced.

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

see was developed by HP.

NAME

sendmail - send mail over the Internet

SYNOPSIS

/usr/sbin/sendmail [*flags*] [*address* ...]

newaliases

mailq

DESCRIPTION

sendmail sends a message to one or more *recipients*, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; **sendmail** is used only to deliver pre-formatted messages.

With no flags, **sendmail** reads its standard input up to an end-of-file or a line consisting only of a single dot and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, e.g., if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

Parameters

-B*file*	Set the body type to *type*. Current legal values are 7BIT or 8BITMIME.
-ba	Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.
-bd	Run as a daemon. **sendmail** will fork and run in background listening on socket 25 for incoming SMTP connections.
-bi	Initialize the alias database.
-bm	Deliver mail in the usual way (default).
-bp	Print a listing of the queue.
-bs	Use the SMTP protocol as described in RFC821 on standard input and output. This flag implies all the operations of the *ba* flag that are compatible with SMTP.
-bt	Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
-bv	Verify names only - do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.
-C*file*	Use alternate configuration file. **sendmail** refuses to run as root if an alternate configuration file is specified.
-d*X*	Set debugging value to **X**.
-F*fullname*	Set the full name of the sender.
-f*name*	Sets the name of the "from" person (i.e., the sender of the mail). **-f** can only be used by "trusted" users (normally *root*, *daemon*, and *network*) or if the person you are trying to become is the same as the person you are.
-h*N*	Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, "Received:" lines in the message are counted.
-n	Don't do aliasing.
-o*xvalue*	Set option *x* to the specified *value*. Options are described below.
-p*protocol*	Set the name of the protocol used to receive the message. This can be a simple protocol name such as "UUCP" or a protocol and hostname, such as "UUCP:ucbvax".

S

-q*time*	Processed saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with **s** being seconds, **m** being minutes, **h** being hours, **d** being days, and **w** being weeks. For example, **-q1h30m** or **-q90m** would both set the timeout to one hour thirty minutes. If *time* is specified, **sendmail** will run in background. This option can be used safely with **bd**.
-qI*substr*	Limit processed jobs to those containing *substr* as a substring of the queue id.
-qR*substr*	Limit processed jobs to those containing *substr* as a substring of one of the recipients.
-qS*substr*	Limit processed jobs to those containing *substr* as a substring of the sender.
-r*name*	An alternate and obsolete form of the **f** flag.
-t	Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed, that is, they will *not* receive copies even if listed in the message header.
-v	Go into verbose mode. Alias expansions will be announced, etc.
-X*logfile*	Log all traffic in and out of mailers in the indicated log file. This should only be used as a last resort for debugging mailer bugs. It will log a lot of data very quickly.

Options

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **-o** flag or in the configuration file. The options are:

A*file*	Use alternate alias file.
b*nblocks*	The minimum number of free blocks needed on the spool filesystem.
c	On mailers that are considered "expensive" to connect to, don't initiate immediate connection. This requires queueing.
C*N*	Checkpoint the queue file after every *N* successful deliveries (default 10). This avoids excessive duplicate deliveries when sending to long mailing lists interrupted by system crashes.

d*x*	Set the delivery mode to *x*. Delivery modes are
	i interactive (synchronous) delivery;
	b background (asynchronous) delivery;
	q queue only; i.e., expect the messages to be delivered next time the queue is run.

D	Try to automatically rebuild the alias database if necessary.
e*x*	Set error processing to mode *x*. Valid modes are **m** to mail back the error message, **w** to "write" back the error message (or mail it back if the sender is not logged in), **p** to print the errors on the terminal (default), **q** to throw away error messages (only exit status is returned), and **e** to do special processing for the BerkNet. If the text of the message is not mailed back by modes **m** or **w** and if the sender is local to this machine, a copy of the message is appended to the file **dead.letter** in the sender's home directory.
f	Save UNIX -style From lines at the front of messages.
G	Match local mail names against the GECOS portion of the password file.
g*N*	The default group id to use when calling mailers.
H*file*	The SMTP help file.
h*N*	The maximum number of times a message is allowed to "hop" before we decide it is in a loop.

i	Do not take dots on a line by themselves as a message terminator.
j	Send error messages in MIME format.
K*timeout*	Set connection cache timeout.
k*N*	Set connection cache size.
L*n*	The log level.
l	Pay attention to the Errors-To: header.
m	Send to "me" (the sender) also if I am in an alias expansion.
n	Validate the right hand side of aliases during a *newaliases*(1M) command.
o	If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.
Q*queuedir*	Select the directory in which to queue messages.
S*file*	Save statistics in the named file.
s	Always instantiate the queue file, even under circumstances where it is not strictly necessary. This provides safety against system crashes during delivery.
T*time*	Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (e.g., because of a host being down) for this amount of time, failed messages will be returned to the sender. The default is three days.
t*stz***,***dtz*	Set the name of the time zone.
U*userdatabase*	If set, a user database is consulted to get forwarding information. You can consider this an adjunct to the aliasing mechanism, except that the database is intended to be distributed; aliases are local to a particular host.
u*N*	Set the default user id for mailers.
Y	Fork each job during queue runs. May be convenient on memory-poor machines.
7	Strip incoming messages to seven bits.

In aliases, the first character of a name may be a vertical bar to cause interpretation of the rest of the name as a command to pipe the mail to. It may be necessary to quote the name to keep **sendmail** from suppressing the blanks from between arguments. For example, a common alias is:

```
msgs: "|/usr/bin/msgs -s"
```

Aliases may also have the syntax "*:include: filename*" to ask **sendmail** to read the named file for a list of recipients. For example, an alias such as:

```
poets: ":include:/usr/local/lib/poets.list"
```

would read **/usr/local/lib/poets.list** for the list of addresses making up the group.

sendmail returns an exit status describing what it did. The codes are defined in <**sysexits.h**>:

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork".
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as **newaliases**, **sendmail** will rebuild the alias database. If invoked as **mailq**, **sendmail** will print the contents of the mail queue.

FILES

Except for the file **/etc/mail/sendmail.cf** itself, the following pathnames are all specified in **/etc/mail/sendmail.cf**. Thus, these values are only approximations.

/etc/mail/aliases	raw data for alias names
/etc/mail/aliases.db	data base of alias names
/etc/mail/sendmail.cf	configuration file
/usr/share/lib/sendmail.hf	help file
/var/log/sendmail.st	collected statistics
/var/spool/mqueue/*	temp files
/etc/mail/sendmail.pid	The process id of the daemon
/etc/mail/sendmail.cw	The list of all hostnames that are recognized as local, which causes sendmail to accept mail for these hosts and attempt local delivery
/etc/mail/service.switch	The fallback mechanism for hostname and alias lookups

SEE ALSO

elm(1), expand_alias(1), idlookup(1), mail(1), mailq(1), mailstats(1), mailx(1), praliases(1), convert_awk(1M), identd(1M), killsm(1M), mtail(1M), newaliases(1M), smrsh(1M), aliases(5).

HISTORY

The **sendmail** command appeared in BSD 4.2. This version of HP-UX sendmail was originally based on sendmail 8.7.1 and includes only minor changes.

NAME

service.switch - indicate lookup sources and fallback mechanism

SYNOPSIS

/etc/mail/service.switch

DESCRIPTION

/etc/mail/service.switch is a *sendmail*(1M) service switch similar to **/etc/nsswitch.conf** (see *switch*(5)) that indicates the lookup source for hostnames and aliases. It consists of two lines, one for hosts and one for aliases. The lookup sources are listed after the 'hosts' or 'aliases' name. For hosts, one or more of the following can be listed: files (for **/etc/hosts**), dns, nis, or nisplus. For aliases, one or more of the following can be listed: files (for **/etc/mail/aliases**), nis, or nisplus.

Sample Configurations

1. The default configuration for service.switch is to use dns for hostname lookups and the aliases file for aliases. (Note that due to a bug, the hostname lookup will never fallback to a file lookup, so anything listed after dns will be ignored.)

```
hosts    dns files
aliases  files
```

2. To work with a non-dns environment that uses file lookups (**/etc/hosts**), the following service.switch can be used:

```
hosts    files
aliases  files
```

3. To work with a NIS environment that does not use DNS, the following service.switch can be used:

```
hosts    nis files
aliases  nis files
```

4. To work with a NISPLUS environment that does not use DNS, the following service.switch can be used:

```
hosts    nisplus files
aliases  nisplus files
```

Modifying SENDMAIL.CF

The **sendmail.cf** file must be modified to request the usage of the **service.switch** file. Otherwise, the default for **sendmail.cf** is to use DNS for host name lookups, and files for alias lookups. To use NIS, NISPLUS, or files, the following line must be uncommented in **sendmail.cf**:

```
#O ServiceSwitchFile=/etc/mail/service.switch
```

SEE ALSO

sendmail(1M)

S

HISTORY

The **service.switch** file appeared in sendmail V8.

NAME

setboot - display and modify variables in the stable storage

SYNOPSIS

/usr/sbin/setboot

/usr/sbin/setboot [**-p** *primary_path*] [**-a** *alternate_path*] [**-s on**|**off**] [**-b on**|**off**]

DESCRIPTION

The **setboot** command sets the appropriate variable in the stable storage to the **setboot** arguments.

If no options are specified, **setboot** displays the current values of variables in the stable storage.

Only the superuser can write to the stable storage.

Options

The **setboot** command supports the following options:

-p *primary_path*	Set the primary boot path variable to *primary_path*.
-a *alternate_path*	Set the alternate boot path variable to *alternate_path*.
-s on | **off**	Enable or disable the autosearch sequence.
-b on | **off**	Enable or disable the autoboot sequence.

RETURN VALUE

The **setboot** command returns 0 upon successful completion or 1 if the command failed.

ERRORS

The following conditions cause errors:

Invalid option specified.

/dev/kepd file not found.

Incorrect arguments.

Must be superuser to write to the stable storage.

Boot path not in correct format.

Stable storage read/write failure.

EXAMPLES

Set the primary path to **2/4.1.0** and enable the autoboot sequence:

```
setboot -p 2/4.1.0 -b on
```

WARNINGS

The **setboot** command fails under the following circumstances:

- The number of writes to the stable storage exceeds the number allowed by the architecture implementation.
- Hardware failure.
- The implementation does not have memory for the alternate boot path, in which case, this variable is neither readable nor writable.

AUTHOR

setboot was developed by HP.

FILES

/dev/kepd	Special device file used by the **setboot** command.

SEE ALSO

Managing SwitchOver/UX Manual.

NAME

setext (vxfs) - set extent attributes

SYNOPSIS

/usr/sbin/setext [**-F vxfs**] [**-V**] [**-e** *extent_size*] [**-r** *reservation*] [**-f** *flag*] *file*

DESCRIPTION

The **setext** command allows space to be reserved for a file, and a fixed extent size to be specified for a file. The file must already exist.

Options

-F vxfs	Specifies the VxFS file system type.
-V	Echoes the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options. This option allows the user to verify the command line.
-e *extent_size*	Specify a fixed extent size. The extent size is specified in file system blocks.
-r *reservation*	Preallocate space for *file*. The reservation is specified in file system blocks.
-f *flag*	The available allocation flags are

-f align
Specify that all extents must be aligned on *extent_size* boundaries relative to the start of allocation units.

-f chgsize
Specify that the reservation is to be immediately incorporated into the file. The file's on-disk inode is updated with size increased to include the reserved space. Unlike an **fcntl F_FREESP** operation that "truncates-up" [see *fcntl*(2)], the space included in the file is not initialized. Only users with appropriate privileges can use the **-f chgsize** flag.

-f contig
Specify that the reservation must be allocated contiguously.

-f noextend
Specify that the file may not be extended once the preallocated space has been exhausted.

-f noreserve
Specify that the reservation is not a persistent attribute of the file. Instead, the space is allocated until the final close of the file, at which time any space not used by the file is freed. The temporary reservation is not visible to the user (via *getext*(1M) or the **VX_GETEXT** ioctl, for instance).

-f trim
Specify that the reservation is trimmed to the current file size upon last close by all processes that have the file open.

NOTES

setext is only available with the Advanced VxFS package.

Multiple flags may be specified by specifying multiple instances of **-f** in a command line.

The allocation flags must be specified with either the **-e** or **-r** option.

Only the **align** and **noextend** allocation flags are persistent attributes of the file and therefore visible via *getext*(1M) or the **VX_GETEXT** ioctl. Other allocation flags may have persistent effects, but are not visible as allocation flags.

Under certain circumstances, **fsadm** may reorganize the extent map of a file in such a way as to make it less contiguous. However, it will not change the geometry of a file that has a fixed extent size.

SEE ALSO

getext(1M), vxfsio(7) (particularly the section on **VX_SETEXT**).

NAME

setmnt - establish the file-system mount table, **/etc/mnttab**

SYNOPSIS

/usr/sbin/setmnt

DESCRIPTION

The **setmnt** command creates the **/etc/mnttab** table (see *mnttab*(4)), which is needed by both the **mount** and **umount** commands (see *mount*(1M)). **setmnt** reads the standard input and creates an entry in **/etc/mnttab** for each line of input. Input lines have the format:

filesys node

where *filesys* is the name of the device special file associated with the file system (such as **/dev/dsk/c0t5d0**) and *node* is the root name of that file system. Thus *filesys* and *node* become the first two strings in the mount table entry.

WARNINGS

The **mount** and **umount** commands rewrite the **/etc/mnttab** file whenever a file system is mounted or unmounted if **/etc/mnttab** is found to be out of date with the mounted file system table maintained internally by the HP-UX kernel. The **syncer** command also updates **/etc/mnttab** if it is out of date (see *syncer*(1M)).

/etc/mnttab should never be manually edited. Use of this command to write invalid information into **/etc/mnttab** is strongly discouraged.

The **setmnt** command is not intented to be run interactively; input should be directed to it from a file (for example, **setmnt < /tmp/file.mnt**). If run interactively, terminate input with a **ctrl-D**.

setmnt silently enforces an upper limit on the maximum number of **/etc/mnttab** entries.

It is unwise to use **setmnt** to create false entries for **mount** and **umount**.

This command is obsolete and it may not be available for future releases.

FILES

/etc/mnttab	Mounted file system table

SEE ALSO

devnm(1M), mount(1M), syncer(1M), mnttab(4).

STANDARDS CONFORMANCE

setmnt: SVID2, SVID3

NAME

setprivgrp - set special privileges for groups

SYNOPSIS

setprivgrp *groupname* [*privileges*]

setprivgrp -g [*privileges*]

setprivgrp -n [*privileges*]

setprivgrp -f *file*

DESCRIPTION

The **setprivgrp** command associates a group with a list of privileges, thus providing access to certain system capabilities for members of a particular group or groups. The privileges can be displayed with the **getprivgrp** command (see *getprivgrp*(1)).

Privileges can be granted to individual groups, as defined in the **/etc/group** file, and globally for all groups.

Only a superuser can use the **setprivgrp** command.

Options and Arguments

setprivgrp recognizes the following options and arguments:

privileges	One or more of the keywords described below in "Privileged Capabilities".
groupname	The name of a group defined in the file named **/etc/group**. The current privileges for *groupname*, if any, are replaced by the specified *privileges*. To retain prior privileges, they must be respecified.
-g	Specify global privileges that apply to all groups. The current privileges, if any, are replaced by the specified *privileges*, To retain prior privileges, they must be respecified.
-n	If no *privileges* are specified, delete all privileges for all groups, including global privileges. If one or more *privileges* are specified, delete the specified privileges from the current privilege lists of all groups, including the global privilege list, but do not delete unspecified privileges.
-f *file*	Set the privileges according to entries in the file *file*. This file is usually **/etc/privgroup**. The entry formats are described below in "Group Privileges File Format".

S

Privileged Capabilities

The following system capabilities can be granted to groups:

CHOWN	Can use **chown()** to change file ownerships (see *chown*(2)).
LOCKRDONLY	Can use **lockf()** to set locks on files that are open for reading only (see *lockf*(2)).
MLOCK	Can use **plock()** to lock process text and data into memory, and the **shmctl() SHM_LOCK** function to lock shared memory segments (see *plock*(2) and *shmctl*(2)).
RTPRIO	Can use **rtprio()** to set real-time priorities (see *rtprio*(2)).
RTSCHED	Can use **sched_setparam()** and **sched_setscheduler()** to set POSIX.4 real-time priorities (see *rtsched*(2)).
SERIALIZE	Can use **serialize()** to force the target process to run serially with other processes that are also marked by this system call (see *serialize*(2)).
SETRUGID	Can use **setuid()** and **setgid()** to change, respectively, the real user ID and real group ID of a process (see *setuid*(2) and *setgid*(2)).

Group Privileges File Format

The file specified with the **-f** option should contain one or more lines in the following formats:

groupname [*privileges*]

-g [*privileges*]

-n [*privileges*]

They are described above in "Options and Arguments".

RETURN VALUE

setprivgrp exits with one of the following values:

0	Successful completion.
>0	Failure.

AUTHOR

setprivgrp was developed by HP.

FILES

/etc/group
/etc/privgroup

SEE ALSO

getprivgrp(1), chown(2), getprivgrp(2), lockf(2), plock(2), rtprio(2), rtsched(2), serialize(2), setgid(2), setuid(2), shmctl(2), privgrp(4).

NAME

setuname - change machine information

SYNOPSIS

setuname [**-s** *name*] [**-n** *node*] [**-t**]

DESCRIPTION

The **setuname** command is used to modify the value for system name and/or the node name by using the appropriate option(s).

The **setuname** command attempts to change the parameter values in both the running kernel and the system configuration to cross reboots. A temporary change affects only the running kernel.

Options

The **setuname** command supports the following options:

-s *name*	Changes the system name (e.g., HP-UX) in the **sysname** field of the **utsname** structure where *name* is the new system name and consists of alphanumeric characters and the special characters dash, underbar, and dollar sign.
-n *node*	Changes the name in the **nodename** field of the **utsname** structure where *node* specifies the new node name and consists of alphanumeric characters and the special characters dash, underbar, and dollar sign.
-t	Signifies a temporary change. The change will not survive a reboot.

Either or both of the **-s** or **-n** options must be given when invoking **setuname**.

The size of the *name* and *node* is limited to **UTSLEN**−1 characters. **UTSLEN** is defined in **<sys/utsname.h>**. Only users having appropriate privileges can use this command.

EXAMPLES

To permanently change the system name to **HP-UX** and the node name to **the-node**, issue the following command:

```
setuname -s HP-UX -n the-node
```

To temporarily change the system name to **SYSTEM** and the node name to **new-node**, issue the following command:

```
setuname -s SYSTEM -n new-node -t
```

SEE ALSO

uname(1), uname(2).

NAME

showmount - show all remote mounts

SYNOPSIS

/usr/sbin/showmount [**-a**] [**-d**] [**-e**] [*host*]

DESCRIPTION

showmount lists all clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd** server on *host* (see *mountd*(1M)). The default value for *host* is the value returned by **hostname** (see *hostname*(1)).

Options

-a Print all remote mounts in the format

name:*directory*

where *hostname* is the name of the client, and *directory* is the directory or root of the file system that was mounted.

-d List directories that have been remotely mounted by clients.

-e Print the list of exported file systems.

WARNINGS

If a client crashes, executing **showmount** on the server will show that the client still has a file system mounted. In other words, the client's entry is not removed from **/etc/rmtab** until the client reboots and executes:

```
umount -a
```

Also, if a client mounts the same remote directory twice, only one entry appears in **/etc/rmtab**. Doing a **umount** of one of these directories removes the single entry and **showmount** no longer indicates that the remote directory is mounted.

AUTHOR

showmount was developed by Sun Microsystems, Inc.

SEE ALSO

hostname(1), exportfs(1M), mountd(1M), exports(4), rmtab(4).

NAME

shutdown - terminate all processing

SYNOPSIS

/sbin/shutdown [**-h**|**-r**] [**-y**] [**-o**] [*grace*]

DESCRIPTION

The **shutdown** command is part of the HP-UX system operation procedures. Its primary function is to terminate all currently running processes in an orderly and cautious manner. **shutdown** can be used to put the system in single-user mode for administrative purposes such as backup or file system consistency checks (see *fsck*(1M)), and to halt or reboot the system. By default, **shutdown** is an interactive program.

Options and Arguments

shutdown recognizes the following options and arguments.

-h	Shut down the system and halt.
-r	Shut down the system and reboot automatically.
-y	Do not require any interactive responses from the user. (Respond **yes** or **no** as appropriate to all questions, such that the user does not interact with the shutdown process.)
-o	When executed on the cluster **server** in a diskless cluster environment, shutdown the **server** only and do not reboot **clients**. If this argument is not entered the default behavior is to reboot all **clients** when the **server** is shutdown.
grace	Either a decimal integer that specifies the duration in seconds of a grace period for users to log off before the system shuts down, or the word **now**. The default is 60. If *grace* is either 0 or **now**, **shutdown** runs more quickly, giving users very little time to log out.

If neither **-r** (reboot) nor **-h** (halt) is specified, **standalone** and **server** systems are placed in single-user state. Either **-r** (reboot) or **-h** (halt) must be specified for a **client**; shutdown to single-user state is not allowed for a **client**. See *dcnodes*(1M), *init*(1M).

Shutdown Procedure

shutdown goes through the following steps:

- The **PATH** environment variable is reset to **/usr/bin:/usr/sbin:/sbin**.
- The **IFS** environment variable is reset to space, tab, newline.
- The user is checked for authorization to execute the **shutdown** command. Only authorized users can execute the **shutdown** command. See FILES for more information on the **/etc/shutdown.allow** authorization file.
- The current working directory is changed to the root directory (**/**).
- All file systems' super blocks are updated; see *sync*(1M). This must be done before rebooting the system to ensure file system integrity.
- The real user ID is set to that of the superuser.
- A broadcast message is sent to all users currently logged in on the system telling them to log out. The administrator can specify a message at this time; otherwise, a standard warning message is displayed.
- The next step depends on whether a system is **standalone**, a **server**, or a **client**.
  - If the system is **standalone**, **/sbin/rc** is executed to shut down subsystems, unmount file systems, and perform other tasks to bring the system to run level 0.
  - If the system is a **server**, the optional **-o** argument is used to determine if all **clients** in the cluster should also be rebooted. The default behavior (command line parameter **-o** is not entered) is to reboot all **clients** using **/sbin/reboot;** entering **-o** results in the **server** only being rebooted and the **clients** being left alone. Then **/sbin/rc** is executed to shut down subsystems, unmount file systems, and perform other tasks to bring the system to run level 0.
  - If the system is a **client**, **/sbin/rc** is executed to bring the system down to run-level 2, and then **/sbin/reboot** is executed. Shutdown to the single-user state is not an allowed option for **clients.**

S

- The system is rebooted or halted by executing **/sbin/reboot** if the **-h** or **-r** option was chosen. If the system was not a cluster client and the system was being brought down to single-user state, a signal is sent to the **init** process to change states (see *init*(1M)).

DIAGNOSTICS

device busy

This is the most commonly encountered error diagnostic, and happens when a particular file system could not be unmounted; see *mount*(1M).

user not allowed to shut down this system

User is not authorized to shut down the system. User and system must both be included in the authorization file **/etc/shutdown.allow**.

EXAMPLES

Immediately reboot the system and run HP-UX again:

```
shutdown -r 0
```

Halt the system in 5 minutes (300 seconds) with no interactive questions and answers:

```
shutdown -h -y 300
```

Go to run-level **s** in 10 minutes:

```
shutdown 600
```

FILES

/etc/shutdown.allow

Authorization file.

The file contains lines that consist of a system host name and the login name of a user who is authorized to reboot or halt the system. A superuser's login name must be included in this file in order to execute **shutdown**. However, if the file is missing or of zero length, the **root** user can run the **shutdown** program to bring the system down.

This file does not affect authorization to bring the system down to single-user state for maintenance purposes; that operation is permitted only when invoked by a superuser.

A comment character, **#**, at the beginning of a line causes the rest of the line to be ignored (comments cannot span multiple lines without additional comment characters). Blank lines are also ignored.

The wildcard character **+** can be used in place of a host name or a user name to specify all hosts or all users, respectively (see *hosts.equiv*(4)).

For example:

```
# user1 can shut down systemA and systemB
systemA user1
systemB user1
# root can shut down any system
+ root
# Any user can shut down systemC
systemC  +
```

WARNINGS

The user name compared with the entry in the **shutdown.allow** file is obtained using **getlogin()** or, if that fails, using **getpwuid()** (see *getlogin*(3C) and *getpwent*(3C)).

The hostname in **/etc/shutdown.allow** is compared with the hostname obtained using **gethostbyname()** (see *gethostent*(3N)).

shutdown must be executed from a directory on the root volume, such as the **/** directory.

The maximum broadcast message that can be sent is approximately 970 characters.

When executing **shutdown** on an NFS diskless cluster server and the **-o** option is not entered, clients of the server will be rebooted. No clients should be individually rebooted or shutdown while the cluster is being shutdown.

SEE ALSO

dcnodes(1M), fsck(1M), init(1M), killall(1M), mount(1M), reboot(1M), sync(1M), dcnodes(3X), gethostent(3N), getpwent(3C), hosts.equiv(4).

NAME

sig_named - send signals to the domain name server

SYNOPSIS

sig_named [**-v**] [**debug** [**+**] *debug-level* | **dump** | **kill** | **restart** | **stats** | **trace**]

DESCRIPTION

sig_named sends the appropriate signal to the domain name server **/usr/sbin/named**. The process ID is obtained from **/var/run/named.pid** or from *ps*(1) if **/var/run/named.pid** does not exist.

Options

sig_named recognizes the following options and command-line arguments:

-v Verify that the name server is running before sending the signal. The verification is done using **ps** (see *ps*(1)).

debug [**+**]*debug-level*
Set the debugging output sent to **/var/tmp/named.run** to *debug-level*. If debugging is already on, it is turned off before the debug level is set. If **+** precedes *debug-level*, the current debugging level is raised by the amount indicated. If *debug-level* is zero, debugging is turned off.

dump Signal the name server to dump its database. The database is dumped to **/var/tmp/named_dump.db**.

kill Kill the name server process.

restart Signal the name server to reload its database.

stats Remove the old statistics file, **/var/tmp/named.stats**. Signal the name server to dump its statistics. Show the statistics file on the standard output.

trace Toggles tracing of incoming queries to **/var/adm/syslog/syslog.log.**

AUTHOR

sig_named was developed by HP.

FILES

/var/run/named.pid	Process ID
/var/tmp/named.run	Debug output
/var/tmp/named_dump.db	Dump of the name server database
/var/tmp/named.stats	Nameserver statistics data

SEE ALSO

kill(1), named(1M).

NAME

smrsh - restricted shell for sendmail

SYNOPSIS

smrsh -c *command*

DESCRIPTION

The **smrsh** program is intended as a replacement for **sh** for use in the **prog** mailer in *sendmail*(1M) configuration files. It sharply limits the commands that can be run using the **|program** syntax of **sendmail** in order to improve the overall security of your system. Briefly, even if a "bad guy" can get **sendmail** to run a program without going through an alias or forward file, **smrsh** limits the set of programs that he or she can execute.

Briefly, **smrsh** limits programs to be in the directory **/var/adm/sm.bin**, allowing the system administrator to choose the set of acceptable commands. It also rejects any commands with the characters \, **<**, **>**, **|**, **;**, **&**, **$**, **(**, **)**, **\r** (carriage return), and **\n** (newline) on the command line to prevent "end run" attacks.

Initial pathnames on programs are stripped, so forwarding to **/usr/ucb/vacation**, **/usr/bin/vacation**, **/home/server/mydir/bin/vacation**, and **vacation** all actually forward to **/var/adm/sm.bin/vacation**.

System administrators should be conservative about populating **/var/adm/sm.bin**. Reasonable additions are *vacation*(1), *rmail*(1), and the like. No matter how brow-beaten you may be, never include any shell or shell-like program (such as *perl*(1)) in the **sm.bin** directory. Note that this does not restrict the use of shell or perl scripts in the **sm.bin** directory (using the **#!** syntax); it simply disallows execution of arbitrary programs.

FILES

/var/adm/sm.bin	Directory for restricted programs

SEE ALSO

sendmail(1M).

S

NAME

snmpd, snmpdm - Simple Network Management Protocol (SNMP) Daemon

SYNOPSIS

/usr/sbin/snmpd [**-a**] [**-authfail**] [**-C** *contact*] [**-Contact** *contact*] [**-h**] [**-help**]
[**-L** *location*] [**-Location** *location*] [**-l** *logfile*] [**-logfile** *logfile*] [**-m** *logmask*]
[**-mask** *logmask*] [**-n**] [**-P** *portnum*] [**-Port** *portnum*] [**-sys** *description*]
[**-sysDescr** *description*]

/usr/sbin/snmpd [**-e** *extendFile*]

/usr/sbin/snmpdm [**-a**] [**-authfail**] [**-C** *contact*] [**-Contact** *contact*] [**-h**] [**-help**]
[**-L** *location*] [**-Location** *location*] [**-l** *logfile*] [**-logfile** *logfile*] [**-m** *logmask*]
[**-mask** *logmask*] [**-n**] [**-P** *portnum*] [**-Port** *portnum*] [**-sys** *description*]
[**-sysDescr** *description*]

DESCRIPTION

The Master SNMP Agent (**/usr/sbin/snmpdm**) and the collection of subAgents (**/usr/sbin/mib2agt**, **/usr/sbin/hp_unixagt**, ...) that would attach to the Master Agent collectively form a single SNMP Agent. The SNMP Agent accepts SNMP Get, GetNext and Set requests from an SNMP Manager which cause it to read or write the Management Information Base (**MIB**). The **MIB** objects are instrumented by the subAgents.

The Master Agent can bind to three kinds of subAgents, namely,

- Loosely coupled subAgents or separate process subAgents which open IPC communication channels to communicate with the Master Agent,
- Shared library subAgents which are dynamically linkable libraries,
- Remotely coupled subagents which could run on a different processor or operating system and communicate with the Master Agent using TCP.

Options

The Master agent **/usr/sbin/snmpdm** and the script **/usr/sbin/snmpd** recognize the following command line options:

-authfail
-a Suppress sending authenticationFailure traps.

-Contact *contact*
-C *contact* Specify the contact person responsible for the network management agent. This option overrides the contact person specified in the Master Agent configuration file **/etc/SnmpAgent.d/snmpd.conf**. It does not alter the value specified in the file. By default, the agent's contact is a blank string. To configure the agent's contact, add the contact after the word contact: in the configuration file **/etc/SnmpAgent.d/snmpd.conf** or use the **-C** option.

-e *extendFile* This option is provided for backward compatibility with the pre-emanate snmpd.ea extensible SNMP agent. It is applicable only to the script **/usr/sbin/snmpd**, and only if the EMANATE extensible agent is installed. It is installed if the file **/usr/sbin/extsubagt** exists. This option causes the extsubagt to use the command line specified extendFile instead of the default file **/etc/SnmpAgent.d/snmpd.extend** to add user defined MIB objects to the SNMP agent.

-help
-h Display command line options and log mask values.

-Location *location*
-L *location* Specify the location of the agent. This option overrides the location specified in **/etc/SnmpAgent.d/snmpd.conf**. It does not alter the value in **/etc/SnmpAgent.d/snmpd.conf**. By default, the agent's location is a blank string. To configure the agent's location, add the location to **/etc/SnmpAgent.d/snmpd.conf** or use the **-L** option.

-logfile *logfile*

S

-l *logfile* Use the *logfile* for logging rather than the default logfile, **/var/adm/snmpd.log**. A value of **-** will direct logging to **stdout**.

-mask *logmask*
-m *logmask* Sets the initial logging mask to *logmask*. The logmask option may not be used in the agent start up scripts. This option should be used only while debugging the agent. See the **SNMP Agent Logging** section for valid values of logmask and for other details.

-n Normally **snmpdm** puts itself into the background as if the command was terminated with an ampersand(&). This option inhibits that behavior and makes the agent run in the foreground.

-Port *portnum*
-P *portnum* Specify the UDP port number that the agent will listen on for SNMP requests. The default is port 161. The value can also be specified in **/etc/services**. Only the super-user can start **snmpdm** and only one **snmpdm** can execute on a particular UDP port.

-sysDescr *description*
-sys *description*
Allows the user to specify the value for the **system.sysDescr MIB** object. The format is a text string enclosed in quotes. This option overrides the **sysDescr** specified in **/etc/SnmpAgent.d/snmpd.conf**. For example,

```
snmpdm -sys "nsmd1, test system"
```

SNMPv1 Security

An **SNMP** Manager application can request to read a **MIB** value available at an agent by issuing an **SNMP** GetRequest, or a manager application may request to alter a **MIB** value by issuing an **SNMP** SetRequest. Each **SNMP** request is accompanied by a community name, which is essentially a password that enables **SNMP** access to **MIB** values on an agent.

Note, the agent does not respond to any **SNMP** requests, including GetRequests, if no community name is configured in **/etc/SnmpAgent.d/snmpd.conf**. To configure the agent to respond to GetRequests accompanied by a specific community name, add the community name as a **get-community-name** to the configuration file. By default the **get-community-name** is set to **public** in the file. For details on this configuration file see the *snmpd.conf*(4) manual page.

By default, the agent does not allow managers to alter **MIB** values (it returns errors for **SNMP** SetRequests). To configure the agent to respond to **SNMP** SetRequests (AND GetRequests), add a **set-community-name** to the file **/etc/SnmpAgent.d/snmpd.conf**.

SNMPv2c

Simple Network Management Protocol Community based Version 2 (**SNMPv2**) is supported in this version of the **SNMP** Agent.

Traps

The agent also sends information to a manager without an explicit request from the manager. Such an operation is called a **trap**. By default, **SNMP** traps are not sent to any destination. To configure the agent to send traps to one or more specific destinations, add the trap destinations to **/etc/SnmpAgent.d/snmpd.conf**.

Then Master Agent (**snmpdm**) and the MIB-2 subAgent (**mib2agt**) collaborate to send the following **SNMP** traps:

coldStart Sends a coldStart trap when the SNMP Agent is invoked.

linkDown Sends a linkDown trap when an interface goes down.

linkUp Sends a linkUp trap when an interface comes up.

authenticationFailure
Sends an authenticationFailure trap when an **SNMP** request is sent to the agent with a community name that does not match any community names the agent is configured to work with.

SNMP Agent Logging

The SNMP Agent provides the capability to log various types of errors and events. There are three types of logging: Errors, Warnings and Traces.

Log Masks

Log masks enable the user to specify the particular classes of messages that should be logged to **/var/adm/snmpd.log** or *logfile*. There are three different ways in which you can specify the logmask that you want. They are: (1) decimal number, (2) hex number, or (3) text string. The three may not be used in combination.

To select multiple output types do the following. For decimal or hex format simply add the individual log-mask values together and enter that number. When entering strings, place multiple strings on the same line, space separated, without quotes.

```
==================================================================
                                        LOG MASK VALUES
  FUNCTION                       Decimal   Hex        String
==================================================================
Log factory trace messages      8388608  0x00800000  FACTORY_TRACE
Log factory warning messages  268435456  0x10000000  FACTORY_WARN
Log factory error messages    536870912  0x20000000  FACTORY_ERROR
```

For example, to turn on error log messages:

```
decimal format: snmpdm -m 536870912
hex     format: snmpdm -m 0x20000000
string  format: snmpdm -m FACTORY_ERROR
```

Using **-m** or **-mask** logmask command line options might cause the master agent to run in the foreground and the agent would not daemonize. This could potentially cause system hang during boot times if any of these options were added to the start up scripts, since the boot up environment might wait indefinitely for the agent to daemonize. So it is adviced not to add these command line options to the start up scripts but use these options only during an agent debug session.

Supported MIB Objects

The Management Information Base (**MIB**) is a conceptual database of values **MIB** (**objects**) on the agent. The Master SNMP Agent implements a small number of **MIB** objects but most **MIB** objects are implemented by subAgents that attach to the Master Agent. See **/opt/OV/snmp_mibs/** on systems with OpenView products installed for definitions of particular **MIB** objects.

This version of the SNMP Agent includes three subAgents, **/usr/sbin/mib2agt**, and **/usr/sbin/hp_unixagt** which implement the MIB-2 and hp-unix MIBs respectively, and the third **/usr/sbin/trapdestagt** which is used in configuring destinations for the agent's traps. The MIBs for the subagents mib2agt and hp_unixagt are described in **/opt/OV/snmp_mibs/rfc1213-MIB-II** and **/opt/OV/snmp_mibs/hp-unix** on systems with OpenView products installed.

The MIB-2 subAgent supports most of the objects in **RFC1213**. The **EGP** group is not supported. The hp-unix subAgent supports most of the objects in the hp-unix MIB.

S

DEPRECATED MIBS

The ieee8023Mac **MIB** group corresponding to the following OID is no longer supported:

private(4).enterprises(1).hp(11).nm(2).interface(4).ieee8023Mac(1)

This **MIB** group is replaced with the "Ether-Like" **MIB** group (RFC1398) which corresponds to OID:

mgmt(2).mib-2(1).transmission(10).dot3(7)

SNMP Agent Startup

The SNMP Agent startup mechanism is built upon the System V.4 file system paradigm. The startup script, **/etc/netmanrc**, which was used in previous releases of the SNMP Agent is no longer used.

Automatic Startup

As installed, the SNMP Master Agent and all subAgents should startup automatically each time the system re-boots or any time the system transitions from run level 1 to run level 2. When the system enters run level 2 the operating system will execute **/sbin/init.d/SnmpMaster** which will startup the Master Agent. Similarly, the operating system invoked **/sbin/init.d/SnmpMib2**, **/sbin/init.d/SnmpHpunix** and **/sbin/init.d/SnmpTrpDst** will startup the MIB2, HP Unix and Trap Dest subAgents respectively immediately after the Master Agent is started.

Prior to executing these startup scripts the system will examine all scripts in **/etc/rc.config.d** for environment variables which could potentially influence the startup of the Master Agent and each subAgent. See the particular startup script or configuration file for details on supported environment variables. The user should never modify scripts in **/sbin/init.d**. Instead the startup behavior should be controlled by adjusting values in the appropriate configuration script in **/etc/rc.config.d**.

Manual Startup

There are two ways to start the SNMP Agent manually. The first way is to execute **snmpdm** and then start each subAgent. Separate process subAgents are started by invoking the particular subAgent executables.

The second and simplest way to start the SNMP Agent manually is to execute the **snmpd** startup script which will invoke the Master Agent and all subAgents who have been installed and designed to operate in this paradigm. The **snmpd** script is layered upon the V.4 startup paradigm and so makes use of the component startup scripts in **/sbin/init.d** and configuration scripts in **/etc/rc.config.d**. When **snmpd** is invoked it starts **/usr/sbin/snmpdm**, passes all its command line arguments to it and then executes each script (S*) found in **/sbin/SnmpAgtStart.d**.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages appear. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **snmpdm** behaves as if all internationalization variables are set to "C." See *environ*(5). Many SNMP Agent log messages are only available in English.

International Code Set Support

Supports single-byte character code sets.

AUTHOR

snmpd was developed by HP, Massachusetts Institute of Technology and SNMP Research.

FILES

/usr/sbin/snmpd
/usr/sbin/snmpdm
/usr/sbin/mib2agt
/usr/sbin/hp_unixagt
/usr/sbin/trapdestagt
/etc/SnmpAgent.d/snmpd.conf
/var/adm/snmpd.log
/opt/OV/snmp_mibs/
/sbin/SnmpAgtStart.d/

SEE ALSO

snmpd.conf(4).

RFC 1155, RFC 1157, RFC 1212, RFC 1213, RFC 1231, RFC 1398.

NAME

softpower - determine if softpower hardware is installed on the system

SYNOPSIS

`/sbin/softpower`

DESCRIPTION

The **`softpower`** command determines whether a software controlled power switch is installed on the system.

RETURN VALUE

`softpower` returns the following values:

0 Softpower hardware detected on the system.
1 Softpower hardware was not detected on the system.
2 The command failed because it is being run on an earlier version of HP-UX that does not support the appropriate **`sysconf`** call.

AUTHOR

`softpower` was developed by HP.

SEE ALSO

sysconf(2).

NAME

spd - set physical drive parameters on an HP SCSI disk array

SYNOPSIS

spd [**-a**] [**-c**] [**-d**] [**-f**] [**-r**] [**-x**] [**-M**] *drive device_file*

DESCRIPTION

spd changes the status of a drive on an HP SCSI disk array associated with device file *device_file*.

Options

-a	Add drive. Adds a drive to the set of drives known by the controller.
-c	Clear the warning, or "failed disk" error status. Parity checking via **scn** must be performed immediately following this operation. See WARNING.
-d	Delete drive. Deletes a drive from the set of drives known by the controller.
-f	Fail drive. Marks the drive as failed, and may place the LUN(s) residing on it in a dead or degraded state, depending on RAID level, and the presence of other failed drives in the LUN.
-r	Replace drive. Marks the drive as replaced, which instructs the controller to start reconstructing the LUN(s) associated with this replaced drive.
-x	Replace and format drive. Marks a drive as replaced, and instructs the controller to physically format the drive before starting LUN reconstruction.
-M	Force the selected option. By altering the state of individual disk mechanisms contained within a disk array, the state of configured LUN devices may also be altered. For example, if the disk array has a LUN configured into a RAID 5 configuration, and one of the disk mechanisms used to create the LUN is in a FAILED state, the disk array will be operating in a DEGRADED mode. If the user selected the FAIL DRIVE option on one of the functioning disk mechanisms of the LUN, the state of the LUN would be changed from DEGRADED to DEAD. **spd** will warn the user of any significant change in LUN state resulting from their selected option. The **-M** option suppresses these warnings and performs the selected operation.
drive	Specified in the form **c***X***i***Y* , where *X* (a decimal number) represents the SCSI channel number, and *Y* (a decimal number) represents the SCSI-ID number of the disk drive.

RETURN VALUE

spd returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

S

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **spd**
- SCSI (device level) communications
- system calls

Error messages generated by spd:

usage: spd <-a | -c | -d | -f | -r | -x> <-M> <cXiY> <special>

An error in command syntax has occurred. Enter command again with all required arguments, in the order shown.

spd: device busy

To ensure that **spd** does not modify a disk array that is being used by another process, **spd** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a “**device busy**” error message is returned by the driver. To eliminate the “**device busy**” condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

spd: Arg out of range

One of the arguments has exceeded its maximum or minimum size, or is incorrect in form. Check the size and form of each argument.

`spd: LUN does not exist`
The addressed LUN is not configured, and thus is not known to the array controller.

`spd: LUN # too big`
The LUN number, which is derived from the device file name, is out of range.

`spd: Not a raw file`
Utilities must be able to open the device file for raw access.

`spd: Not an HP SCSI disk array`
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:
Sense data associated with the failed operation is printed.

`spd: Transfer length error`
The amount of data actually sent to or received from the device was not the expected amount.

Error messages generated by system calls:
spd uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **spd** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

The following command clears the FAILED state of the drive at ID 3 on channel 2 on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700. The **-M** option must be selected to suppress warning messages. The **scn** operation must be performed immediately to ensure accurate data parity information.

```
spd -c -M c2i3 /dev/rdsk/c2t6d0
scn /dev/rdsk/c2t6d0
```

To add the drive at ID 3 on channel 2 to the set of drives the array controller knows about on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700:

```
spd -a c2i3 /dev/rdsk/c2t6d0
```

To delete the drive at ID 3 on channel 2 from the set of drives the array controller knows about on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 800:

```
spd -d c2i3 /dev/rdsk/c2t6d0
```

To mark the drive at ID 3 on channel 2 as failed, thus rendering any redundant RAID mode LUN (s) residing on it as dead or degraded, on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700:

```
spd -f -M c2i3 /dev/rdsk/c2t6d0
```

To mark the drive at ID 3 on channel 2 as replaced, thus initiating the reconstruction of any redundant RAID mode LUN (s) residing on it, on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700:

```
spd -r b2a3 /dev/rdsk/c2t6d0
```

NOTE

Failing a drive on a RAID_0 LUN will leave it with an "optimal" LUN status, even though the controller will no longer access the failed drive and its data.

WARNING

Clearing a "failed" disk status might leave the array with inconsistent parity. This can lead to corrupted data if the array LUN ever operates in "degraded" state. Parity scan and repair must be performed immediately after clearing the "failed" state of a disk array.

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

spd was developed by HP.

NAME

spray - spray packets

SYNOPSIS

/usr/sbin/spray *host* [**-c** *count*] [**-l** *length*]

DESCRIPTION

spray sends a one-way stream of packets to *host* using RPC, then reports how many were received by *host* and what the transfer rate was. The host name can be either a name or an internet address.

Options

spray recognizes the following options and command-line arguments:

-c *count*	Specifies how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100 000 bytes.
-l *length*	The number of bytes in the Ethernet packet holding the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32-bit quantities, not all values of *length* are possible. The *spray* command rounds up to the nearest possible value. When *length* is greater than the size of an Ethernet packet, the system breaks the datagram into multiple Ethernet packets. The default value of *length* is 86 bytes (the size of the RPC and UDP headers).

AUTHOR

spray was developed by Sun Microsystems, Inc.

SEE ALSO

ping(1M), sprayd(1M).

NAME

sprayd - spray server

SYNOPSIS

/usr/lib/netsvc/spray/rpc.sprayd [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

sprayd is an RPC server that records the packets sent by **spray** from another system (see *spray*(1M)).

inetd invokes **sprayd** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

sprayd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, host name, process id and name of the function generating the error, and the error message. Note that different services can share a single log file since enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if

- **portmap** dies (see *portmap*(1M)),
- Another **rpc.sprayd** registers with **portmap**, or
- **rpc.sprayd** becomes unregistered with *portmap*.

The **-n** option is more efficient because a new process is not launched for each RPC request. **-n** is the default.

AUTHOR

sprayd was developed by Sun Microsystems, Inc.

SEE ALSO

inetd(1M), spray(1M), portmap(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

sss - set spindle sync state of drives in an HP SCSI disk array

SYNOPSIS

sss -d [*drive_list*] *device_file*

sss -on [**-s**] [*drive_list*] *device_file*

sss -off [*drive_list*] *device_file*

DESCRIPTION

sss displays or changes the spindle synchronization state of the disk drives in the HP SCSI disk array associated with device file *device_file*. Though *device_file* is the name of a device file corresponding to a LUN, **sss** operates (by default) on all disk drives physically connected to the array controller, without regard to the drives' LUN ownership. Even if multiple LUNs (or sub-LUNs) are present, **sss** should be directed to only one of them (that is, specify the name of the device file for only one of the LUNs in the **sss** command line). To affect a subset of the physical drives in the array, specify which drives to affect in *drive_list*.

Options

-d Display only. Displays the current spindle synchronization status. This has two components: the drive's master or slave status and its state of spindle synchronization (on or off).

-on Sync on. Enables spindle synchronization; one drive is designated master and the rest are designated slaves, unless the **-s** "slave only" tag is present, in which case all designated drives will be slaves. If only one drive is designated, it will be a master.

-off Sync off. Disables spindle synchronization.

-s Slave only. Only used with the **-on** option. Make all designated drives slaves. This is useful when replacing a drive in a set of drives which already have spindle synchronization enabled. If you have replaced the master drive, use the **-on** option without **-s**, and specify the new drive only.

drive_list A list of drives used to specify which drives in the array will be affected by the synchronization operation. *drive_list* is in the form **c***X***i***Y* , where *X* (a decimal number) represents the SCSI channel number, and *Y* (a decimal number) represents the SCSI-ID number of the desired drive. Drives names in *drive_list* are separated by commas. If no *drive_list* is present, **sss** defaults to all physical drives attached to the array controller, regardless of which LUNs they belong to.

RETURN VALUE

sss returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **sss**
- SCSI (device level) communications
- system calls

Error messages generated by sss:

usage: sss <-d | -on [-s] | -off> [cXiY,...] <special>

sss encountered an error in command syntax. Enter the command again with the required arguments, in the order shown.

sss: Arg out of range

One of the arguments has exceeded its maximum or minimum size, or is incorrect in form. Check the size and form of each argument.

sss: device busy

To ensure that **sss** does not modify a disk array that is being used by another process, **sss** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the spindle sync state of the drives in the array (see *vgchange*(1M)).

sss: LUN # too big
The LUN number, which is derived from the device file name, is out of range.

sss: Not a raw file
sss must be able to open the device file for raw access.

sss: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

sss: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

SCSI (device level) communication errors:
Sense data associated with the failed operation is printed.

Error messages generated by system calls:
sss uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **sss** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To display the spindle synchronization status of drives on HP SCSI disk array **/dev/rdsk/c22d0s2** on a Series 800:

```
sss -d /dev/rdsk/c22d0s2
```

To enable spindle synchronization on all drives of the HP SCSI disk array **/dev/rdsk/c410d3l3s0** on a Series 700:

```
sss -on /dev/rdsk/c410d3l3s0
```

The drive on SCSI channel 3 at SCSI ID **0** of the HP SCSI disk array **/dev/rdsk/c410d3l3s0** has just been replaced. The other drives in the array are synchronized, and the replaced one was a slave. To enable spindle synchronization on the new drive on a Series 700:

```
sss -on -s c3i0 /dev/rdsk/c410d3l3s0
```

If, in the replacement scenario above, the replaced drive was the master, to enable spindle synchronization and make the new drive a master:

```
sss -on c3i0 /dev/rdsk/c410d3l3s0
```

or, alternatively, enable the whole set again:

```
sss -on /dev/rdsk/c410d3l3s0
```

DEPENDENCIES

This utility is currently supported only on HP C2425, HP C2427, and HP C2430 disk arrays.

AUTHOR

sss was developed by HP.

NAME

statd - network status monitor

SYNOPSIS

/usr/sbin/rpc.statd [**-l** *log_file*]

DESCRIPTION

statd is an RPC server. It interacts with **lockd** to provide crash and recovery functions for the locking services on NFS (see *lockd*(1M)).

Options

statd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, host name, process id and name of the function generating the error, and the error message.

FILES

/var/statmon/sm/*
/var/statmon/sm.bak/*
/var/statmon/state

WARNINGS

Changes in status of a site are detected only upon startup of a new status monitor and lock daemon.

AUTHOR

statd was developed by Sun Microsystems, Inc.

SEE ALSO

fcntl(2), lockf(2), signal(2), lockd(1M), sm(4).

NAME

strace - write STREAMS event trace messages to standard output

SYNOPSIS

strace [*mod sub pri* **] ...**

DESCRIPTION

strace gets STREAMS event trace messages from STREAMS drivers and modules via the STREAMS log driver (**strlog(7)**), and writes these messages to standard output. By default, **strace** without arguments writes all STREAMS trace messages from all drivers and modules. **strace** with command-line arguments limits the trace messages received.

The arguments, which must be specified in groups of three, are:

mod	Specifies the STREAMS module identification number from the streamtab entry.
sub	Specifies a subidentification number (often corresponding to a minor device).
pri	Specifies a tracing priority level. **strace** gets messages of a level equal to or less than the value specified by *pri*. Only positive integer values are allowed.

The value **all** can be used for any argument in the **strace** command line to indicate that there are no restrictions for that argument.

Multiple sets of the three arguments can be specified to obtain the messages from more than one driver or module.

Only one **strace** process can open the STREAMS log driver at a time.

When **strace** is invoked, the log driver compares the sets of command line arguments with actual trace messages, returning only messages that satisfy the specified criteria.

STREAMS event trace messages have the following format:

seq time tick pri ind mod sub text

Components are interpreted as follows:

seq Trace event sequence number.

time Time the message was sent expressed in *hh:mm:ss*.

tick Time the message was sent expressed in machine ticks since the last boot.

pri Tracing priority level as defined by the STREAMS driver or module that originates the messages.

ind Can be any combination of the following three message indicators:

E	The message has also been saved in the error log.
F	The message signaled a fatal error.
N	The message has also been mailed to the system administrator.

mod Module identification number of the trace message source.

sub Subidentification number of the trace message source.

text Trace message text.

strace runs until terminated by the user.

EXAMPLES

Display all trace messages received from the driver or module identified by *mod* **28**:

```
strace 28 all all
```

Display trace messages of any tracing priority level from the driver or module identified by *mod* **28** and its minor devices identified by the *sub* **2**, **3**, or **4**:

```
strace  28 2 all  28 3 all  28 4 all
```

Display the trace messages from the same driver or module and *subs* but limit the priority levels to 0 for *subs* 2 and 3; 1 for *sub* 4, driver or module **28**:

S

```
        strace  28 2 0  28 3 0  28 4 1
```

WARNINGS

Running **strace** with several sets of arguments can impair STREAMS performance, particularly for those modules and drivers that are sending the messages.

Also be aware that **strace** may not be able to handle a large number of messages. If drivers and modules return messages to **strace** too quickly, some may be lost.

FILES

/usr/lib/nls/msg/C/strace.cat	NLS catalog for **strace**.

SEE ALSO

strclean(1M), strerr(1M), strlog(7).

NAME

strchg, strconf - change or query stream configuration

SYNOPSIS

strchg -h *module1*[**,** *module2*]...

strchg -p [**-a**|**-u** *module*]

strchg -f *file*

strconf

strconf -t

strconf -m *module*

DESCRIPTION

The **strchg** and **strconf** commands are used to change or query the configuration of the stream associated with the user's standard input. The **strchg** command pushes modules on and/or pops modules off the stream. The **strconf** command queries the configuration of the stream. Only the superuser or owner of a STREAMS device may alter the configuration of that stream.

strchg Options

The **strchg** command uses the following options:

-h *module1*[,*module2*] ...
: **strchg** pushes modules onto a stream. The modules are pushable STREAMS modules as defined by *module1*, *module2*, and so on. The modules are pushed in order. That is, *module1* is pushed first, *module2* is pushed second, etc.

-p
: With the **-p** option alone, **strchg** pops the topmost module from the stream.

-a
: With the **-p** and **-a** options, all the modules above the topmost driver are popped.

-u *module*
: With the **-p** and **-u** *module* options, all modules above but not including module are popped off the stream.

The **-a** and **-u** options are mutually exclusive.

-f *file*
: The user can specify a *file* that contains a list of modules representing the desired configuration of the stream. Each module name must appear on a separate line where the first name represents the topmost module and the last name represents the module that should be closest to the driver. The **strchg** command will determine the current configuration of the stream and pop and push the necessary modules in order to end up with the desired configuration.

The **-h**, **-f**, and **-p** options are mutually exclusive.

strconf Options

Invoked without any arguments, **strconf** prints a list of all the modules in the stream as well as the topmost driver. The list is printed in one name per line where the first name printed is the topmost module on the stream (if one exists) and the last item printed is the name of the driver.

The **strconf** command uses the following options:

-t
: Only the topmost module (if one exists) is printed.

-m *module*
: **strconf** checks if the named *module* is present on the stream. If so, **strconf** prints the message, **yes**, and returns zero. If not, **strconf** prints the message, **no**, and returns a non-zero value.

The **-t** and **-m** options are mutually exclusive.

Notes

If the user is neither the owner of the stream nor the superuser, the **strchg** command will fail. If the user does not have read permissions on the stream and is not the superuser, the **strconf** command will fail.

If modules are pushed in the wrong order, one could end up with a stream that does not function as expected. For ttys, if the line discipline module is not pushed in the correct place, one could have a terminal that does not respond to any commands.

DIAGNOSTICS

strchg returns zero on success. It prints an error message and returns non-zero status for various error conditions, including usage error, bad module name, too many modules to push, failure of an **ioctl** on the stream, or failure to open file from the **-f** option.

strconf returns zero on success (for the **-m** or **-t** option, "success" means the named or topmost module is present). It returns a non-zero status if invoked with the **-m** or **-t** option and the module is not present. It prints an error message and returns non-zero status for various error conditions, including usage error or failure of an **ioctl** on the stream.

EXAMPLES

The following command pushes the module **ldterm** on the stream associated with the user's standard input:

```
strchg -h ldterm
```

The following command pops the topmost module from the stream associated with **/dev/term/24**. The user must be the owner of this device or be superuser.

```
strchg -p < /dev/term/24
```

If the file, **fileconf**, contains the following:

```
compat
ldterm
ptem
```

then the command

```
strchg -f fileconf
```

will configure the user's standard input stream so that the module **ptem** is pushed over the driver, followed by **ldterm** and **compat** closest to the stream head.

The **strconf** command with no arguments lists the modules and topmost driver on the stream. For a stream that only has the module **ldterm** pushed above the ports driver, it would produce the following output:

```
ldterm
ports
```

The following command asks if **ldterm** is on the stream:

```
strconf -m ldterm
```

and produces the following output while returning an exit status of 0:

```
yes
```

FILES

/usr/lib/nls/msg/C/strchg.cat	NLS catalogs
/usr/lib/nls/msg/C/strconf.cat	NLS catalogs

SEE ALSO

streamio(7).

NAME

strclean - remove outdated STREAMS error log files

SYNOPSIS

strclean [**-d** *logdir*] [**-a** *age*]

DESCRIPTION

strclean cleans the STREAMS error logger directory of log files (**error.** *mm-dd*) that contain error messages sent by the STREAMS log driver, *strlog*(7). If the **-d** option is not used to specify another directory, **strclean** removes error log files in the **/var/adm/streams** directory. If the **-a** option is not used to specify another age, **strclean** removes error log files that have not been modified in three days.

Options

strclean recognizes the following options and command-line arguments:

-d *logdir* Specifies a directory for the location of the STREAMS error log files to be removed if this is not the default directory **/var/adm/streams**.

-a *age* Specifies a maximum age in days for the STREAMS error log files if this not the default age of 3. The value of *age* must be an integer greater than or less than 3.

EXAMPLES

Remove day-old error log files from a directory called **/tmp/streams**:

```
strclean -d /tmp/streams -a 1
```

FILES

/var/adm/streams/error.mm-dd	One or more error log file or files on which **strclean** operates. The *mm-dd* in the filename indicates the month and day of the messages contained in the file.
/usr/lib/nls/msg/C/strclean.cat	NLS catalog for **strclean**.

SEE ALSO

strerr(1M), strlog(7).

NAME

strdb - STREAMS debugging tool

SYNOPSIS

strdb [*system*]

DESCRIPTION

strdb symbolically displays the contents of various STREAMS data structures. The argument *system* allows substitutes for the default **/stand/vmunix**. **strdb** can only handle a 32-bit kernel actually running on a system. For crash dumps and for 64-bit kernels, use the **q4** debugger.

strdb runs in two modes, STREAMS subsystem and primary. STREAMS subsystem commands report the status of open streams. Primary commands display STREAMS data structures.

In a typical **strdb** session, you will do the following:

- Run **strdb**. When **strdb** starts up, you are in primary mode.
- Execute the **:S** command to enter STREAMS subsystem mode.
- Enter STREAMS subsystem commands such as **s**, **d**, and **la** to find the open stream you want to examine.
- Enter the **qh** command to select a stream and display the stream head read queue. This command returns you to primary mode.
- Enter primary mode navigation keys to display fields in the stream head read queue, and traverse the rest of the stream's queues.

The following commands are available in primary mode.

:?	Display a help menu for primary mode.
^D	Exit from **strdb**.
:q	Exit from **strdb**.
^K	If logging is enable, dump current screen to log file
^L	Refresh the screen.
:u	Disable data structure stacking. By default data structure stacking is turned on. When stacking is on, **strdb** pushes each structure it displays onto a stack so that it can be reviewed later. See **ˆP** and other stack commands described below.
:s	Re-enable data structure stacking. By default data structure stacking is turned on. The **:u** command turns it off. When stacking is on, **strdb** pushes each structure it displays onto a stack so that it can be reviewed later. See **ˆP** and other stack commands described below.
:l *name* **o**|**c**	If the **o** option is specified, open a log file, *name*, and start logging. Alternatively if the **c** option is specified, close a log file, *name*, and stop logging.
:S	Enter STREAMS subsystem mode.
navigation key	Display the field specified by *navigation key* in the currently displayed data structure. **strdb** provides different navigation keys for each STREAMS data structure. Each key indicates a particular field in the data structure to display. The navigation keys for STREAMS data structures are described after the STREAMS subsystem commands below.
?	Display a help menu for the displayed data structure's navigation keys.
^R	Update the displayed data structure with new values from **/dev/kmem** on a running system.
^P	Pop the displayed data structure off the data structure stack, and display the data structure now at the top of the stack.
:m	Mark the displayed data structure. Later the data structure stack can be popped back to this structure using **ˆU** as described below.
^U	Pop the data structure stack back to a structure marked with **:m**, and display this structure.

^T	Transpose the top two data structure stack entries. Unlike **^P**, this command allows the data structure on the top of the stack to be saved for later viewing.
:b *addr* [*len*]	Display *len* bytes of binary data at address *addr*. The default for *len* is 256.
:x *name addr*	Display structure *name* located at address *addr*.
:x ?	Display the structure names accepted by **:x**.

The following commands are available in STREAMS subsystem mode.

?	Display a help menu for STREAMS subsystem mode.
h	Display a help menu for STREAMS subsystem mode.
q	Exit from STREAMS subsystem mode to primary mode.
v	Print the version of STREAMS data structures displayed.
s d | **m**	If the **d** option is specified, list the STREAMS drivers included in the kernel **S800** or **dfile** file. Alternatively if the **m** option is specified, list the included modules.
la *name*	List all open streams for device *name*. The *name* is one of those shown by the **s** command.
lm *name minor*	List all modules pushed on the stream for device *name* and minor *minor*.
ll *name minor*	List all drivers linked under the multiplexor *name* with minor *minor*.
lp *name minor*	List all drivers persistently linked under the multiplexor *name* with minor *minor*.
qc *name file*	Write the **q_count** values for the driver *name* into file, *file*.
qh *name minor*	Display the streams head read queue for the STREAMS driver *name* and minor *minor*. This command returns the user to primary mode.

strdb provides different navigation keys for each STREAMS data structure it displays. Each key indicates a particular data structure field to display. The navigation keys for each STREAMS data structure are described in the following paragraphs.

The navigation keys for the STREAMS **queue** structure are:

i	Displays the **q_init** structure pointed to by the **q_qinfo** field.
m	Displays the **msgb** structure pointed to by the **q_first** field.
z	Displays the **msgb** structure pointed to by the **q_last** field.
n	Displays the **queue** structure pointed to by the **q_next** field.
l	Displays the **queue** structure pointed to by the **q_link** field.
b	Displays the **qband** structure pointed to by the **q_bandp** field.
o	Displays the **queue** structure pointed to by the **q_other** field.

The navigation keys for the STREAMS **qinit** structure are:

i	Displays the **module_info** structure pointed to by the **qi_minfo** field.
s	Displays the **module_stat** field pointed to by the **qi_mstat** field.

The navigation keys for the STREAMS **msgb** structure are:

n	Displays the **msgb** structure pointed to by the **b_next** field.
p	Displays the **msgb** structure pointed to by the **b_prev** field.
m	Displays the data pointed to by the **b_rptr** field.
c	Displays the **msgb** structure pointed to by the **b_cont** field.
d	Displays the **datab** structure pointed to by the **b_datap** field.

The navigation keys for the STREAMS **datab** structure are:

d	Displays the **a__datab** structure pointed to by the **db_f** field.

The navigation keys for the STREAMS **qband** structure are:

n	Displays the **qband** structure pointed to by the **qb_next** field.
f	Displays the **msgb** structure pointed to by the **qb_first** field.
l	Displays the **msgb** structure pointed to by the **qb_last** field.

AUTHOR

strdb was developed by HP.

SEE ALSO

STREAMS/UX for HP9000 Reference Manual.

NAME

strerr - receive error messages from the STREAMS log driver

SYNOPSIS

strerr [**-a** *sys_admin_mail_name*] [**-d** *logdir*]

DESCRIPTION

The **strerr** daemon receives error messages from the STREAMS log driver (*strlog*(7)) for addition to the STREAMS error log files (**error.***mm-dd*) in the STREAMS error logger directory (**/var/adm/streams** by default). When first called, **strerr** creates the log file **error.***mm-dd*. This is a daily log file, where *mm* indicates the month and *dd* indicates the day of the logged messages. **strerr** then appends error messages to the log file as they are received from the STREAMS log driver.

STREAMS error log messages have the following format:

seq time tick pri ind mod sub text

Components are interpreted as follows:

seq	Error event sequence number.
time	Time the message was sent expressed in *hh:mm:ss*.
tick	Time the message was sent expressed in machine ticks since the last boot.
pri	Error priority level as defined by the STREAMS driver or module that originates the messages.
ind	Can be any combination of the following three message indicators:
	T The message has also been saved in the trace log.
	F The message signaled a fatal error.
	N The message has also been mailed to the system administrator.
mod	Module identification number of the error message source.
sub	Subidentification number of the error message source.
text	Error message text.

strerr runs continuously until terminated by the user.

Options

strerr recognizes the following options and command-line arguments:

-a *sys_admin_mail_name*	Specify the user's mail name for sending mail messages. Mail is sent to the system administrator by default.
-d *logdir*	Specify the directory to contain the error log file. Default is **/var/adm/streams**.

S

WARNINGS

Only one **strerr** process can open the STREAMS log driver at a time. This restriction is intended to maximize performance.

The STREAMS error logging mechanism works best when it is not overused. **strerr** can degrade STREAMS performance by affecting the response, throughput, and other behaviors of the drivers and modules that invoke it. **strerr** also fails to capture messages if drivers and modules generate messages at a higher rate than its optimum read rate. If there are missing sequence numbers among the messages in a log file, messages have been lost.

FILES

/usr/lib/nls/msg/C/strerr.cat	NLS catalog for **strerr**.
/var/adm/streams/error.mm-dd	error log file or files on which **strerr** operates

SEE ALSO

strace(1M), strlog(7).

NAME

strvf - STREAMS verification tool

SYNOPSIS

strvf [**-v**]

DESCRIPTION

strvf executes a series of subcommands that verify whether or not STREAMS is currently installed and configured on your system. All output is sent to **stdout**. Verbose output is always sent to the logfile **/var/adm/streams/strvf.log**.

These subcommands make sure that the STREAMS kernel daemons are running and that **open()**, **putmsg()**, **getmsg()**, **ioctl()**, and **close()** can be performed on **/dev/echo**.

Options

-v	Specifies verbose output to be displayed

EXAMPLES

strvf	Verify STREAMS is working. Brief summary of status is displayed on screen. Verbose description of each subcommand and its status is copied to the logfile.
strvf -v	Verify STREAMS is working. Verbose description of each subcommand and its status is displayed on the screen and copied to the logfile. This option is useful in troubleshooting **strvf** failures.

FILES

/var/adm/streams/strvf.log	Logfile containing a verbose description and status of all subcommands.
/dev/echo	Loopback STREAMS driver used by **strvf**.

SEE ALSO

open(2), close(2), getmsg(2), putmsg(2), streamio(7).

NAME

swacl - view or modify the Access Control Lists (ACLs) which protect software products

SYNOPSIS

swacl -l *level* [**-D** *acl_entry* | **-F** *acl_file* | **-M** *acl_entry*] [**-f** *software_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*][@ *target_selections*]

Remarks

- SD-UX commands are included with the HP-UX Operating System and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The **swacl** command displays or modifies the Access Control Lists (ACLs) which:

- Protect the specified *target_selections* (hosts, software depots or root filesystems).
- Protect the specified *software_selections* on each of the specified *target_selections* (software depots only).

All root filesystems, software depots, and products in software depots are protected by ACLs. The SD commands permit or prevent specific operations based on whether the ACLs on these objects permit the operation. The **swacl** command is used to view, edit, and manage these ACLs. The ACL must exist and the user must have the appropriate permission (granted by the ACL itself) in order to modify it.

ACLs offer a greater degree of selectivity than standard file permissions. ACLs allow an object's owner (i.e. the user who created the object) or the local superuser to define specific read, write, or modify permissions to a specific list of users, groups, or combinations thereof.

Some operations allowed by ACLs are run as local superuser. Because files are loaded and scripts are run as superuser, granting a user write permission on a root filesystem or insert permission on a host effectively gives that user superuser privileges.

Protected Objects

The following objects are protected by ACLs:

- Each host system on which software is being managed by SD,
- Each root filesystem on a host (including alternate roots),
- Each software depot on a host,
- Each software product contained within a depot.

Options

If the **-D**, **-F**, or **-M** option is not specified, **swacl** prints the requested ACL(s) to the standard output.

The **swacl** command supports the following options:

-D *acl_entry* Deletes an existing entry from the ACL associated with the specified object(s). For this option, the permission field of the ACL entry is not required. Multiple **-D** options can be specified.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-F *acl_file* Assigns the ACL contained in *acl_file* to the object. All existing entries are removed and replaced by the entries in the file. Only the ACL's entries are replaced; none of the information contained in the comment portion (lines with the prefix "#") of an ACL listing is modified with this option. The *acl_file* is usually the edited output of a **swacl** list operation.

If the replacement ACL contains no syntax errors and the user has **control** permission on the ACL (or is the local superuser), the replacement succeeds.

-l *level* Defines which level of SD ACLs to view/modify.

The supported *levels* of depot, host, root, and product objects that can be protected are:

depot View/modify the ACL protecting the software depot(s) identified by the *target_selections*.

host View/modify the ACL protecting the host system(s) identified by the *target_selections*.

root View/modify the ACL protecting the root filesystem(s) identified by the *target_selections*.

product View/modify the ACL protecting the software product identified by the *software_selection*. Applies only to products in depots, not installed products in roots.

The supported *levels* of templates are:

global_soc_template
View/modify the template ACL used to initialize the ACL(s) of future software depot(s) or root filesystem(s) added to the host(s) identified by the *target_selections*. Additionally, **swacl** can be used to set templates to be used when new ACLs are created.

global_product_template
View/modify the template ACL used to initialize the **product_template** ACL(s) of future software depot(s) added to the host(s) identified by the *target_selections*.

product_template
View/modify the template ACL used to initialize the ACL(s) of future product(s) added to the software depot(s) identified by the *target_selections*.

-M *acl_entry* Adds a new ACL entry or changes the permissions of an existing entry. Multiple **-M** options can be specified.

-t *target_file* Read the list of *target_selections* from *file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Only one of the **-D**, **-F**, or **-M** options can be specified for an invocation of **swacl**.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The **swacl** command supports the following syntax for each *software_selection*:

product[**,** *version*]

The *version* component usually has the following form:

[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*]

- The <*op*> (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches. Shell patterns are not allowed with these operators.

- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.

- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings.

- No space or tab characters are allowed in a software selection.

- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products in the depot when used with **-l** *product*.

Target Selections

The SD commands support this syntax for each *target_selection*.

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

The following PC information applies only to HP OpenView Software Distributor.

S

The **swacl** command supports the following syntax for specifying PCs:

[*pc_controller*]

EXTERNAL INFLUENCES

Defaults Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values,
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swacl** command. If a default value exists, it is listed after the "=".

distribution_target_directory=/var/spool/sw
Defines the default location of the target depot.

level= Defines the level of SD ACLS to view/modify. The supported levels are: **host**, **depot**, **root**, **product**, **product_template**, **global_soc_template**, or **global_product_template**.

See the discussion of the **-l** option above for more information.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

select_local=true
If no *target_selections* are specified, select the default **target_directory** of the local host as the *target_selection* for the command.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

verbose=1
Controls the verbosity of the output (stdout). A value of
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

Environment Variables

SD programs are affected by external environment variables, set environment variables for use by the control scripts, and use other environment variables that affect command behavior.

The external environment variable that affects the **swacl** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

Note: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

OPERATION

Each entry in an ACL has the following form:

entry_type[**:** *key*]**:** *permissions*

For example: **user:steve@newdist:crwit**

An ACL can contain multiple entries.

List Output Format

The output of a list operation is in the following format:

```
# swacl   Object_type      Access Control List
#
# For   depot | host:[host]:[/directory]
#
# Date:  date_stamp
#
# Object Ownership:  User=   user_name
#                    Group=  group_name
#                    Realm=  host_name
#
# default_realm = host_name
entry_type:[key:]permissions
entry_type:[key:]permissions
entry_type:[key:]permissions
```

This output can be saved into a file, modified, and then used as input to a **swacl** modify operation (see the **-F** option above).

PC Controller ACLs

The following applies only to HP OpenView Software Distributor.

When listing an ACL at a PC controller, this additional information is listed:

```
# Locally Configured SD Controller Access:
#       user: user_name@hostname:permissions
#       group: group_name@hostname:permissions
```

This output describes the user and group granted all SD access permissions to all objects at the PC controller.

Object Ownership

An *owner* is also associated with every SD object, as defined by the user name, group and hostname. The owner is the user who created the object. When using **swacl** to view an ACL, the owner is printed as a comment in the header.

S

Default Realm

An ACL defines a default *realm* for an object. The realm is currently defined as the name of the host system on which the object resides. When using **swacl** to view an ACL, the default realm is printed as a comment in the header.

Entry Types

The following *entry_types* are supported:

any_other Permissions for all other users and hosts that do not match a more specific entry in the ACL. (Example: **any_other:-r--t**.)

group Permissions for a named group. This type of ACL entry must include a key that identifies that group. The format can be: **group:***group_name***:***permissions* or **group:***group_name*@*hostname***:***permissions*. (Example: **group:adm:crwit**.)

host Permissions for an SD agent from the specified host system. SD agents require *product level* read access via either a **host**, **other**, or **any_other** entry type in order to copy or install products from depots. This type of ACL entry must include a key containing a hostname or number (in Internet dot notation) of a system. (Example: **host:newdist@fc.hp.com:-r--t**.)

object_owner
Permissions for the object's owner, whose identity is listed in the comment header. (Example: **object_owner:crwit**.)

object_group
Permissions for members of the object's group, whose identity is listed in the comment header. (Example: **object_group:crwit**.)

other Permissions for others who are not otherwise named by a more specific entry type. The format for **other** can be: **other:***permissions* for others on the local host (only one such entry allowed) or **other:@***hostname***:***permissions* for others at remote hosts (Only one such entry per remote host allowed). (Example: **other:@newdist:-r--t**.)

user Permissions for a named user. This type of ACL entry must include a key that identifies that user. The format for **user** can be: **user:***user_name***:***permissions* or **user:***user_name@hostname***:***permissions*. (Example: **user:rml:crwit**.)

Keys

Expressions (patterns) are **not** permitted in keys.

A key is required for **user**, **group** and **host** entry types. A key is optional for **other** entry types, and specifies the hostname to which the entry applies. Only one **other** entry type may exist *without* a key, and this entry applies to users at the default realm (host) of the ACL.

A hostname in a key will be listed in its Internet address format (dot notation) if **swacl** cannot resolve the address using the local lookup mechanism (DNS, NIS, or */etc/hosts*). A hostname within an ACL entry must be resolvable when used with the **-D** and **-M** options. Unresolvable hostname values are accepted in files provided with the **-F** option.

Permissions

Permissions are represented as the single character abbreviations indicated below. Some permissions either apply only to, or have different meaning for, certain types of objects, as detailed below. The following permissions may be granted:

r *ead* Grants permission to read the object. On **host**, **depot**, or **root** objects, read permission allows **swlist** operations. On products within depots, read permission allows product files to be installed or copied with **swinstall**or**swcopy**.

w *rite* Grants permission to modify the object itself.

• On a **root** object (e.g. installed root filesystem), this also grants permission to modify the products installed (contained) within it.

• On a **depot** object, it does **not** grant permission to modify the products contained within it. Write access on products is required to modify products in a depot.

• On a **host** container, write permission grants permission to unregister depots. It does **not** grant permission to modify the depots or roots contained within it.

i *nsert* On a **host** object, grants permission to create (insert) a new software depot or root filesystem object, and to register roots and depots. On a **depot** object, grants permission to create (insert) a new product object into the **depot**.

c *ontrol* Grants permission to modify the ACL using **swacl**.

t *est* Grants permission to perform access checks and to list the ACL.

a *ll* A wildcard which grants all of the above permissions. It is expanded by **swacl** to **crwit**.

RETURN VALUE

The **swacl** command returns:

0 The *software_selections* and/or *target_selections* were successfully displayed or modified.
1 The display/modify operation failed on all *target_selections*.
2 The modify/modify operation failed on some *target_selections*.

S

DIAGNOSTICS

The **swacl** command writes to stdout, stderr, and to the daemon logfile.

Standard Output

The **swacl** command prints ACL information to stdout when the user requests an ACL listing.

Standard Error

The **swacl** command writes messages for all WARNING and ERROR conditions to stderr. A report that the *software_selections* do not exist is also given if the user has **no** access permissions to the object.

Logging

The **swacl** command does not log summary events. It logs events about each ACL which is modified to the **swagentd** logfile associated with each *target_selection*.

EXAMPLES

To list the ACLs for the **COBOL** and **FORTRAN** products in depot **/var/spool/swtest**:

```
swacl -l product COBOL FORTRAN  @ /var/spool/swtest
```

The ACL listed to the standard output is similar to this example ACL:

```
#
# swacl   Product Access Control Lists
#
# For depot: newdist:/var/spool/swtest
#
# Date:  Wed May 26 11:14:31 1993
#
#
# For product:  COBOL,r=3.2
#
#
# Object Ownership:  User=  robason
#                    Group= swadm
#                    Realm= newdist.fc.hp.com
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
group:swadm:crwit
any_other:-r--t
#
# For product:  FORTRAN,r=9.4
#
#
# Object Ownership:  User=  robason
#                    Group= swadm
#                    Realm= newdist.fc.hp.com
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
user:rob@ lehi.fc.hp.com:-r--t
user:barb:-r--t
user:ramon:-r--t
group:swadm:crwit
other:-r--t
host:lehi.fc.hp.com:-r--t
```

To list the product template ACL on host **newdist**:

```
swacl -l global_product_template  @ newdist
```

To list the host ACL on the local system:

```
swacl -l host
```

To read, edit, then replace the ACL protecting the default depot **/var/spool/sw**:

```
swacl -l depot > new_acl_file
vi new_acl_file
swacl -l depot -F new_acl_file
```

To add an entry for user **george** on host **newdist** to the ACL protecting **COBOL** in the default depot at host **lehi**:

```
swacl -l product -M user:george@newdist:crwit COBOL @ lehi:
```

To deny all access to the users **steve** and **george** for the depot **/var/spool/sw** at host **newdist**:

```
swacl -l depot -M user:steve:- -M user:george:-
@ newdist:/var/spool/sw
```

To delete entries for local user **rick** from all products in the default local depot:

```
swacl -l product -D user:rick \*
```

WARNINGS

It is possible to edit an ACL in such a way as to render it inaccessible. Be careful not to remove all **control** permissions on an ACL. As a safeguard, the local super-user may always edit SD ACLs, regardless of permissions

Operations are allowed by ACLs using local superuser. Because files are loaded and scripts are run as superuser, granting write permission on a root filesystem or insert permission on a host effectively gives that user superuser privileges.

swacl is not a general purpose ACL editor, it works only on ACLs protecting SD objects.

The following line applies only to HP OpenView Software Distributor.

For PC controllers, the user defined as the Locally Configured SD Controller Access may always edit SD ACLs.

The SD-UX version of **swacl** does not support the viewing and modification of ACLs on remote targets.

The following limitation applies only to HP OpenView Software Distributor.

The **root** ACLs do not apply to PC controllers. When installing to PC targets, the **depot** and **product** ACLs on the PC controller apply, since the install is enacted by first copying the PC products into the PC depot.

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/adm/sw/security/
The directory which contains ACLs for the system itself, template ACLS, and the secrets file used to authenticate remote requests.

/var/spool/sw/
The default location of a source and target software depot.

PC FILES

The following files apply only to HP OpenView Software Distributor.

`...\SD\DATA\`
The directory which contains all of the configurable and non-configurable data for SD.

`...\SD\DATA\DEPOT\`
The default location of the source and target PC depot.

`...\SD\DATA\SECURITY\`
The directory which contains ACLs for the system itself, template ACLS, and the secrets file used to authenticate remote requests.

`<WINDOWS>\SWAGENTD.INI`
Contains the configurable options for the SD PC controller, including the user and group granted all SD access.

AUTHOR

`swacl` was developed by the Hewlett-Packard Company.

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swagentd, swagent, SWAGENTD.EXE - serve local or remote SD software management tasks

SYNOPSIS

swagent executed by **swagentd** only.

swagentd [**-k**] [**-n**] [**-r**] [**-x** *option=value*] [**-X** *option_file*]

SWAGENTD.EXE (*HP OpenView Software Distributor only.*)

Remarks

- SD-UX commands are included with the HP-UX operating system and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The roles of UNIX target and source systems require two processes known as the **daemon** and **agent**. For most purposes, the distinction between these two processes is invisible to the user and they can be viewed as a single process.

Each SD command interacts with the daemon and agent to perform its requested tasks.

The **swagentd** daemon process must be scheduled before a UNIX system is available as a target or source system. This can be done either manually or in the system start-up script. The **swagent** agent process is executed by **swagentd** to perform specific software management tasks. The **swagent** agent is never invoked by the user.

The following paragraph applies only to HP OpenView Software Distributor.

The roles of PC target and source systems require a single Windows application, **SWAGENTD.EXE**, which combines the **swagent** and **swagentd** functionality. Each PC running the **SWAGENTD.EXE** is a PC controller. When distributing PC software, it acts as a fanout server to PC targets. (These targets run SD PC agent programs to perform the actual software installation tasks.)

Options

The **swagentd** command supports the following options to control its behavior:

-k The *kill* option stops the currently running daemon. Stopping the daemon will not stop any agent processes currently performing management tasks (such as installing or removing software), but will cause any subsequent management requests to this host to be refused. This option is equivalent to sending a SIGTERM to the daemon that is running.

-n The *no fork* option runs the daemon as a synchronous process rather than the default behavior of forking to run it asynchronously. This is intended for running the daemon from other utilities that schedule processes, such as **init**.

-r The *restart* option stops the currently running daemon and restarts a new daemon. This operation is required whenever modifying default options that apply to the daemon since defaults are only processed on startup.

-x *option=value*
Set the *option* to *value* and override the default value (or a value in an *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *options_file*.

S

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swagentd** command. If a default value exists, it is listed after the "=".

Daemon options

These options apply only to the daemon, **swagentd**. After changing daemon options, the daemon must be restarted in order for these options to be recognized (see the **-r** option).

agent=/usr/lbin/swagent
The location of the agent program invoked by the daemon.

logfile=/var/adm/sw/swagentd.log
This is the default log file for the **swagentd** daemon.

max_agents=-1
The maximum number of agents that are permitted to run simultaneously. The value of -1 means that there is no limit.

minimum_job_polling_interval=1
(Applies only to HP OpenView Software Distributor.) Defines how often, in minutes, the daemon will wake up and scan the job queue to determine if any scheduled jobs need to be initiated or if any active jobs need their remote target status cached locally (see **swinstall.job_polling_interval**). If set to 0, no scheduled jobs will be initiated, and no caching of active jobs will occur.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

Agent options

These options apply only to the agent, **swagent**.

alternate_source=
If the **swinstall** or **swcopy** controller has set **use_alternate_source=true**, the target agent will consult and use the configured value of its own **alternate_source** option to determine the source that it will use in the install or copy.

The agent's value for **alternate_source** is specified using the **host:path** syntax. If the host portion is not specified, the local host is used. If the path portion is not specified, the path sent by the command is used. If there is no configured value at all for **alternate_source**, the agent will apply the controller-supplied path to its own local host.

compress_cmd=/usr/contrib/bin/gzip
Defines the command called by the source agent to compress files before transmission. If the **compression_type** is set to other than **gzip** or **compress**, this path must be changed.

compression_type=gzip
Defines the default **compression_type** used by the agent when it compresses files during or after transmission. If **uncompress_files** is set to false, the **compression_type** is recorded for each file compressed so that the correct uncompression can later be applied during a **swinstall**, or a **swcopy** with **uncompress_files** set to true. The **compress_cmd** specified must produce files with the **compression_type** specified. The **uncompress_cmd** must be able to process files of the **compression_type** specified unless the format is **gzip**, which is uncompressed by the internal uncompressor (**funzip**). The only supported compression types are **compress** and **gzip**.

config_cleanup_cmd=/usr/lbin/sw/config_clean
Defines the script called by the agent to perform release-specific configure cleanup steps.

install_cleanup_cmd=/usr/lbin/sw/install_clean
Defines the script called by the agent to perform release-specific install cleanup steps immediately after the last postinstall script has been run. For an OS update, this script should at least remove commands that were saved by the **install_setup** script. This script is executed after all filesets have been installed, just before the reboot to the new operating system.

install_setup_cmd=/usr/lbin/sw/install_setup
Defines the script called by the agent to perform release-specific install preparation. For an OS update, this script should at least copy commands needed for the checkinstall, preinstall, and postinstall scripts to a path where they can be accessed while the real commands are being updated. This script is executed before any kernel filesets are loaded.

kernel_build_cmd=/usr/sbin/mk_kernel
Defines the script called by the agent for kernel building after kernel filesets have been loaded.

kernel_path=/stand/vmunix
Defines the path to the system's bootable kernel. This path is passed to the **kernel_build_cmd** via the **SW_KERNEL_PATH** environment variable.

mount_cmd=/sbin/mount
Defines the command called by the agent to mount all filesystems.

reboot_cmd=/sbin/reboot
Defines the command called by the agent to reboot the system after all filesets have been loaded, if any of the filesets required reboot.

remove_setup_cmd=/usr/lbin/sw/remove_setup
Defines the script called by the agent to perform release-specific remove preparation. For an OS update, this script will invoke the **tlink** command when a fileset is removed.

rpc_binding_info_alt_source=ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) used when the agent attempts to contact an alternate source depot specified by the **alternate_source** option. HP-UX supports both the **udp**(**ncadg_ip_udp:[2121]**) and **tcp**(**ncacn_ip_tcp:[2121]**) protocol sequence/endpoint. SD on SunOS only supports udp (**ncadg_ip_udp:[2121]**). By default **udp** is used.

source_depot_audit=true
If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can set this option to track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. (A user running **swinstall/swcopy** from a target machine cannot set this option; only the administrator of the source depot machine can set it.)

When **swagent.source_depot_audit** is set to **true**, a **swaudit.log** file is created on the source depot (for writable directory depots) or in **/var/tmp** (for *tar* images, CD-ROMs, or other nonwritable depots).

Users can invoke the **swlist** interactive user interface (using **swlist -i -d**) to view, print, or save the audit information on a remote or local depot. Users can view audit information based on language preference, as long as the system has the corresponding SD message catalog files on it. For example, a user can view the source audit information in

S

Japanese during one invocation of **swlist**, then view the same information in English at the next invocation.

system_file_path=/stand/system
Defines the path to the kernel's template file. This path is passed to the **system_prep_cmd** via the **SW_SYSTEM_FILE_PATH** environment variable.

system_prep_cmd=/usr/lbin/sysadm/system_prep
Defines the kernel build preparation script called by the agent. This script must do any necessary preparation so that control scripts can correctly configure the kernel about to be built. This script is called before any kernel filesets have been loaded.

uncompress_cmd=
Defines the command called by the target agent to uncompress files after transmission. This command processes files which were stored on the media in a compressed format. If the *compression_type* stored with the file is **gzip**, the internal uncompression (**funzip**) is used instead of the external **uncompress_cmd**. The default value for HP-UX is undefined.

Session File

swagentd and **swagent** do not use a session file.

Environment Variables

The environment variable that affects the **swagentd** and **swagent** commands is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

Note: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Signals

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots. Requests to start new sessions are refused during this wait.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

S

Locking

The **swagentd** ensures that only one copy of itself is running on the system.

Each copy of **swagent** that is invoked uses appropriate access control for the operation it is performing and the object it is operating on.

The following section on the Initialization File applies only to HP OpenView Software Distributor.

Windows Initialization File

The **SWAGENTD.EXE** supports the following configuration values in the file:

<WINDOWS>\SWAGENTD.INI - the initialization file for the PC controller.

Values must be specified in the initialization file using this syntax:

[*section* **]**

*keyword***=***value*

(These values are usually modified using the **SWAGENTD.EXE**'s Configure dialog.)

Environment Section

These keywords are defined in the **Environment** section of the Windows initialization file.

`DataDirectory=...\sd\data`
The data directory set at installation, which contains the PC depot and other PC controller data files.

`ViewCommand=sdview.exe`
The command used to display the PC controller logfile when the View Log... menu item is selected.

Note that **`SWAGENTD.EXE`** does not allow the configuration of the **`rpc_binding_info`** option, as described for **`swagentd`** and **`swagent`** above. To allow correct execution on a variety of TCP/IP stacks, the **`SWAGENTD.EXE`** always uses the value **`ncacn_ip_tcp:[2121]`**.

Security Section

These keywords are defined in the **`Security`** section of the Windows initialization file.

`user=root`
The UNIX user at the specified UNIX host (below) who is granted all SD access permissions to the PC controller. (This remote user has permission to perform any SD action at this PC controller.)

`group=swadm`
The UNIX group at the specified UNIX host (below) which is granted all SD access permissions to the PC controller. (This remote group has permission to perform any SD action at this PC controller.)

`hostname=`*hostname*
The UNIX host from which the specified user and group (above) is granted all SD access permissions.

`secret=-sdu-`
The secret password used in SD's internal form of authentication. (The **`-sdu-`** default value matches the default value defined for the SD commands.)

RETURN VALUES

When the **`-n`** option is not specified, the **`swagentd`** returns:

`0`	When the daemon is successfully initialized and is now running in the background.
`non-zero`	When initialization failed and the daemon terminated.

When the **`-n`** option is specified, the **`swagentd`** returns:

`0`	When the daemon successfully initialized and then successfully shutdown.
`non-zero`	When initialization failed or the daemon unsuccessfully terminated.

DIAGNOSTICS

The **`swagentd`** and **`swagent`** commands log events to their specific logfiles.

The **`swagent`** (target) log files cannot be relocated. They always exist relative to the root or depot target path (e.g. **`/var/adm/sw/swagent.log`** for the root **`/`** and **`/var/spool/sw/swagent.log`** for the depot **`/var/spool/sw`**).

The following line applies only to HP OpenView Software Distributor.

The target log files may be viewed using the **`swjob`** command.

Daemon Log
The daemon logs all events to **`/var/adm/sw/swagentd.log`**. (The user can specify a different logfile by modifying the **`logfile`** option.)

Agent Log
When operating on (alternate) root filesystems, the **`swagent`** logs messages to the file **`var/adm/sw/swagent.log`** beneath the root directory (e.g. **`/`** or an alternate root directory).

Source Depot Audit Log
If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. Refer to the **`swagent.source_depot_audit`** option for more information.

S

When operating on software depots, the **swagent** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**). When accessing a read-only software depot (e.g. as a source), the **swagent** logs messages to the file **/tmp/swagent.log**.

EXAMPLES

To start the daemon:

```
/usr/sbin/swagentd
```

To restart the daemon:

```
/usr/sbin/swagentd -r
```

To stop the daemon:

```
/usr/sbin/swagentd -k
```

FILES

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/var/adm/sw/
The directory which contains all configurable and non-configurable data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/host_object
The file which stores the list of depots registered at the local host.

The following on PC files applies only to HP OpenView Software Distributor.

PC FILES

...\SD\DATA\
The directory which contains all of the configurable and non-configurable data for SD.

<WINDOWS>\SWAGENTD.INI
Contains the configurable options for the SD PC controller.

AUTHOR

swagentd was developed by the Hewlett-Packard Company. **swagent** was developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swapinfo - system paging space information

SYNOPSIS

/usr/sbin/swapinfo [**-mtadfnrMqw**]

DESCRIPTION

swapinfo prints information about device and file system paging space. (Note: the term 'swap' refers to an obsolete implementation of virtual memory; HP-UX actually implements virtual memory by way of paging rather than swapping. This command and others retain names derived from 'swap' for historical reasons.)

By default, **swapinfo** prints to standard output a two line header as shown here, followed by one line per paging area:

```
            Kb       Kb       Kb       PCT      START/   Kb
  TYPE      AVAIL    USED     FREE     USED     LIMIT    RESERVE  PRI      NAME
```

The fields are:

TYPE One of:

dev Paging space residing on a mass storage device, either taking up the entire device or, if the device contains a file system, taking up the space between the end of the file system and the end of the device. This space is exclusively reserved for paging, and even if it is not being used for paging, it cannot be used for any other purpose. Device paging areas typically provide the fastest paging.

fs Dynamic paging space available from a file system. When this space is needed, the system creates files in the file system and uses them as paging space. File system paging is typically slower than device paging, but allows the space to be used for other things (user files) when not needed for paging.

localfs File system paging space (see **fs** above) on a file system residing on a local disk.

network File system paging space (see **fs** above) on a file system residing on another machine. This file system would have been mounted on the local machine via NFS.

reserve Paging space on reserve. This is the amount of paging space that could be needed by processes that are currently running, but that has not yet been allocated from one of the above paging areas. See "Paging Allocation" below.

memory Memory paging area (also known as pseudo-swap). This is the amount of system memory that can be used to hold pages in the event that all of the above paging areas are used up. See "Paging Allocation" below. This line appears only if memory paging is enabled.

Kb AVAIL The total available space from the paging area, in blocks of 1024 bytes (rounded to nearest whole block if necessary), including any paging space already in use.

For file system paging areas the value is not necessarily constant. It is the current space allocated for paging (even if not currently used), plus the free blocks available on the file system to ordinary users, minus RESERVE (but never less than zero). AVAIL is never more than LIMIT if LIMIT is non-zero. Since paging space is allocated in large chunks, AVAIL is rounded down to the nearest full allocation chunk.

For the memory paging area this value is also not necessarily constant, because it reflects allocation of memory by the kernel as well as by processes that might need to be paged.

Kb USED The current number of 1-Kbyte blocks used for paging in the paging area. For the memory paging area, this count also includes memory used for other purposes and thus unavailable for paging.

Kb FREE The amount of space that can be used for future paging. Usually this is the difference between Kb AVAIL and Kb USED. There could be a difference if some portion of a device paging area is unusable, perhaps because the size of the paging area is not a multiple of the allocation chunk size, or because the tunable parameter **maxswapchunks** is not set high enough.

PCT USED The percentage of capacity in use, based on Kb USED divided by Kb AVAIL; 100% if Kb AVAIL is zero.

START/LIMIT
For device paging areas, START is the block address on the mass storage device of the start of the paging area. The value is normally 0 for devices dedicated to paging, or the end of the file system for devices containing both a file system and paging space.

For file system paging areas, LIMIT is the maximum number of 1-Kbyte blocks that will be used for paging, the same as the *limit* value given to **swapon**. A file system LIMIT value of **none** means there is no fixed limit; all space is available except that used for files, less the blocks represented by **minfree** (see *fs*(4)) plus RESERVE.

RESERVE For device paging areas, this value is always "—". For file system paging areas, this value is the number of 1-Kbyte blocks reserved for file system use by ordinary users, the same as the *reserve* value given to **swapon**.

PRI The same as the *priority* value given to **swapon**. This value indicates the order in which space is taken from the devices and file systems used for paging. Space is taken from areas with lower priority values first. *priority* can have a value between 0 and 10. See "Paging Allocation" below.

NAME For device paging areas, the block special file name whose major and minor numbers match the device's ID. The **swapinfo** command searches the **/dev** tree to find device names. If no matching block special file is found, **swapinfo** prints the device ID (major and minor values), for example, **28,0x15000**.

For file system swap areas, NAME is the name of a directory on the file system in which the paging files are stored.

Paging Allocation

Paging areas are enabled at boot time (for device paging areas configured into the kernel) or by the **swapon** command (see *swapon*(1M)), often invoked by **/sbin/init.d/swap_start** during system initialization based on the contents of **/etc/fstab**. When a paging area is enabled, some portion of that area is allocated for paging space. For device paging areas, the entire device is allocated, less any leftover fraction of an allocation chunk. (The size of an allocation chunk is controlled by the tunable parameter **swchunk**, and is typically 2 MB.) For file system paging areas, the *minimum* value given to **swapon** (rounded up to the nearest allocation chunk) is allocated.

When a process is created, or requests additional space, space is reserved for it by increasing the space shown on the **reserve** line above. When paging activity actually occurs, space is used in one of the paging areas (the one with the lowest priority number that has free space available, already allocated), and that space will be shown as used in that area.

The sum of the space used in all of the paging areas, plus the amount of space reserved, can never exceed the total amount allocated in all of the paging areas. If a request for more memory occurs which would cause this to happen, the system tries several options:

1. The system tries to increase the total space available by allocating more space in file system paging areas.
2. If all file system paging areas are completely allocated and the request is still not satisfied, the system will try to use memory paging as described on the **memory** line above. (Memory paging is controlled by the tunable parameter **swapmem_on**, which defaults to 1 (on). If this parameter is turned off, the **memory** line will not appear.)
3. If memory paging also cannot satisfy the request, because it is full or turned off, the request is denied.

Several implications of this procedure are noteworthy for understanding the output of **swapinfo**:

- Paging space will not be allocated in a file system paging area (except for the *minimum* specified when the area is first enabled) until all device paging space has been reserved, even if the file system paging area has a lower priority value.
- When paging space is allocated to a file system paging area, that space becomes unavailable for user files, even if there is no paging activity to it.
- Requests for more paging space will fail when they cannot be satisfied by reserving device, file system, or memory paging, even if some of the reserved paging space is not yet in use. Thus it is possible for requests for more paging space to be denied when some, or even all, of the paging areas show zero usage — space in those areas is completely reserved.

- System available memory is shared between the paging subsystem and kernel memory allocators. Thus, the system may show memory paging usage before all available disk paging space is completely reserved or fully allocated.

Options

`swapinfo` recognizes the following options:

`-m` Display the AVAIL, USED, FREE, LIMIT, and RESERVE values in Mbytes instead of Kbytes, rounding off to the nearest whole Mbyte (multiples of 1024^2). The output header format changes from `Kb` to `Mb` accordingly.

`-t` Add a totals line with a TYPE of `total`. This line totals only the paging information displayed above it, not all paging areas; this line might be misleading if a subset of `-dfrM` is specified.

`-a` Show all device paging areas, including those configured into the kernel but currently disabled. (These are normally omitted.) The word `disabled` appears after the NAME, and the Kb AVAIL, Kb USED, and Kb FREE values are 0. The `-a` option is ignored unless the `-d` option is present or is true by default.

`-d` Print information about device paging areas only. This modifies the output header appropriately.

`-f` Print information about file system paging areas only. This modifies the output header appropriately.

`-n` Categorize file system paging area information into `localfs` areas and `network` areas, instead of calling them both `fs` areas.

`-r` Print information about reserved paging space only.

`-M` Print information about memory paging space only.

The `-d`, `-f`, `-n`, `-r` and `-M` options can be combined. The default is `-dfnrM`.

`-q` Quiet mode. Print only a total "Kb AVAIL" value (with the `-m` option, Mb AVAIL); that is, the total paging space available on the system (device, file system, reserve, or memory paging space only if `-d`, `-f`, `-r`, or `-M` is specified), for possible use by programs that want a quick total. If `-q` is specified, the `-t` and `-a` options are ignored.

`-w` Print a warning about each device paging area that contains wasted space; that is, any device paging area whose allocated size is less than its total size. This option is effective only if `-d` is also specified or true by default.

RETURN VALUE

`swapinfo` returns 0 if it completes successfully (including if any warnings are issued), or 1 if it reports any errors.

DIAGNOSTICS

`swapinfo` prints messages to standard error if it has any problems.

EXAMPLES

List all file system paging areas with a totals line:

```
swapinfo -ft
```

WARNINGS

`swapinfo` needs kernel access for some information. If the user does not have appropriate privileges for kernel access, `swapinfo` will print a warning and assume that the defaults for that information have not been changed.

Users of `swapinfo` must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

The information in this manual page about paging allocation and other implementation details may change without warning; users should not rely on the accuracy of this information.

AUTHOR

`swapinfo` was developed by HP.

SEE ALSO

swapon(1M), swapon(2), fstab(4), fs(4).

NAME

swapon - enable device or file system for paging

SYNOPSIS

Preferred Forms:

/usr/sbin/swapon -a [**-u**] [**-t** *type*]...

/usr/sbin/swapon [**-e** |**-f**] [**-p** *priority*] [**-u**] *device* ...

/usr/sbin/swapon [**-m** *min*] [**-l** *limit*] [**-r** *reserve*] [**-p** *priority*] *directory* ...

Obsolescent Form:

/usr/sbin/swapon *directory* [*min limit reserve priority*]

DESCRIPTION

The **swapon** command enables devices or file systems on which paging is to take place. (**NOTE:** the term 'swap' refers to an obsolete implementation of virtual memory; HP-UX actually implements virtual memory by way of paging rather than swapping. This command and others retain names derived from 'swap' for historical reasons.)

By enabling a **device** for paging, the device can be accessed directly (without going through the file system) during paging activity. When a **file system** is enabled for paging, the device(s) on which the file system resides are accessed indirectly through the file system. There are advantages and disadvantages to both type of paging. Keep the following tradeoffs in mind when enabling devices or file systems for paging.

Paging directly to a **device** is significantly faster than doing so through the file system. However, the space on the device that is allocated to paging cannot be used for anything else, even if it is not being actively used for paging.

Paging through a **file system**, while slower, provides a more efficient use of the space on the device. Space that is not being used for paging in this case can be used by the file system. Paging across a network to a remote machine is always file system paging.

The system begins by paging on only a single device so that only one disk is required at bootstrap time. Calls to **swapon** normally occur in the system startup script **/sbin/init.d/swap_start** making all paging space available so that the paging activity is interleaved across several disks.

Normally, the **-a** argument is given, causing all devices marked as **swap** and all file systems marked as **swapfs** in the file **/etc/fstab** to be made available to the paging system. By using the fields in **/etc/fstab** (*special_file_name* or *directory*; see *fstab*(5)), the system determines which block device or file system to use. The *special_file_name* specified for each **swap** entry must specify a block special file. The *directory* specified for each **swapfs** entry must specify a directory within the file system to be enabled.

The second form of **swapon** enables individual block devices to be used for paging. The *device* name must specify a block special file. If more than one device is given, any options specified will be applied to all devices. If a file system exists on the specified block device and neither an **-e** nor **-f** option is specified, **swapon** fails and an error message is given. This prevents a file system from being inadvertently destroyed. To request paging in the space between the end of the file system and the end of the device, use **-e**. To force paging to a device containing a file system (destroying the file system), the **-f** option can be used. Use this with extreme caution!

In either of the previous forms, an attempt to enable paging to a device will fail and a warning message will be issued if **swapon** determines that the device is being used by the **savecore** command to retrieve system dump information (see *savecore*(1M)). The **-u** option can be used to forcibly enable paging to devices being used by **savecore**; however, this may overwrite system dump information contained on the device.

The last two forms of **swapon** provide two different methods for enabling file systems for paging. The third form is the preferred method, with the fourth being provided only for backward compatibility. The *directory* name specifies a directory on the file system that is to be enabled for paging. A directory named **/paging** is created at the root of the specified file system (unless the file system's name ends with **/paging**). All paging files are created within this directory. The optional arguments to the fourth form have the same meaning as the arguments to the options in the third form. Note that, in the fourth form, if any of the optional arguments are specified, all must be specified. In the third form, if more than one directory is given, any options specified will be applied to all directories.

After a file system has been enabled for paging, the optional arguments can be modified by subsequent **swapon** commands.

Options

swapon recognizes the following options and arguments:

-a Cause all devices marked as **swap** and all file systems marked as **swapfs** in the file **/etc/fstab** to be made available to the paging system. The *options* field in **/etc/fstab** entries is read by **swapon**, and must contain elements formatted as follows:

min=*min*	See the **-m** option for the value of *min*.
lim=*limit*	See the **-l** option for the value of *limit*. (File system paging areas only.)
res=*reserve*	See the **-r** option for the value of *reserve*. (File system paging areas only.)
pri=*priority*	See the **-p** option for the value of *priority*. (File system paging areas only.)
end	See the **-e** option for the meaning of this option. (Device paging areas only.)

See *fstab*(4) for an example entry.

-e Use space after the end of the file system on the block device for paging. An error message is returned if no file system is found on the device. This option cannot be used with the **-f** option. Do not confuse this with paging to a file system. This option is for use with a disk that has *both* a file system and dedicated paging space on it.

-f Force the *device* to be enabled, which will destroy the file system on it. Use with extreme caution. Normally, if a file system exists on the *device* to be enabled, **swapon** fails and displays an error message. This option cannot be used with the **-e** option.

-l *limit* *limit* specifies the maximum space the paging system is allowed to take from the disk, provided space is available that is not reserved for exclusive use by the file system. The value of *limit* is rounded up so that it is a multiple of the paging allocation chunk size, which is set with the kernel tunable parameter **swchunk** (see *config*(1M) and *swapinfo*(1M)). See WARNINGS. The default value for *limit* is 0, indicating there is no limit to the amount of file system space the paging system can use.

limit can be specified in decimal (no prefix), octal (**0** prefix), or hexadecimal (**0x** prefix). It may be specified in units of kilobytes (**k** suffix), megabytes (**M** suffix), or file system blocks (no suffix). (A kilobyte is 1024 bytes; a megabyte is 1024 kilobytes; the size of a file system block is determined by the administrator when the file system is created.)

-m *min* *min* indicates the space the paging system will initially take from the file system. The value of *min* is rounded up so that it is a multiple of the paging allocation chunk size, which is set with the kernel tunable parameter **swchunk** (see *config*(1M) and *swapinfo*(1M)). The default value for *min* is 0, indicating no paging space is to be allocated initially. *min* can be specified in the same forms as *limit*, above.

-p *priority* *priority* indicates the order in which space is taken from the file systems and devices used for paging. Space is taken from the systems with lower priority numbers first. Under most circumstances, space is taken from device paging areas before file system paging areas, regardless of priority. See "Paging Allocation" in *swapinfo*(1M) for more information. *priority* can have a value from 0 to 10 and has a default value of 1.

-r *reserve* *reserve* specifies the space, in addition to the space currently occupied by the file system, that is reserved for file system use only, making it unavailable to the paging system. This reserved space is in addition to the minimum free space specified by the administrator when the file system was created. See WARNINGS. The default value for *reserve* is 0 indicating that no file system space is reserved for file system use only. *reserve* can be specified in the same forms as *limit*, above.

-t *type* Restrict the type of the paging area. If the **-t** option is omitted, all of the paging areas defined in **/etc/fstab** are made available. *type* can have one of the following values:

dev	Device paging areas.
fs	File system paging areas.
local	Paging areas defined on the local system.
remote	Paging areas defined on remote systems.

-u Unlock block device files which are being used by the **savecore** command. Normally, **swapon** will not enable paging on a device if it is being used by **savecore** to retrieve system dump information. The list of devices in use is maintained in the file **/etc/savecore.LCK**. This option forces the device to be enabled, which may overwrite any system dump information contained on the device. This option should be used with extreme caution.

RETURN VALUE

swapon returns one of the following values:

0	Successful completion.
>0	An error condition occurred.

EXAMPLES

The first two examples enable paging to the file system containing the **/paging** directory. The maximum number of file system blocks available to the paging system is set to 5000, the number of file system blocks reserved for file system use only is set to 10000, and the priority is set to 2. The number of file system blocks initially taken by the paging system defaults to 0 in the first example, and is set to 0 in the second example. On a file system with the default 8kB block size, these examples allocate approximately 40MB of file system paging.

```
/usr/sbin/swapon -l 5000 -r 10000 -p 2 /paging
/usr/sbin/swapon /paging 0 5000 10000 2
```

This example enables paging to two block devices and sets the priority of both devices to 0.

```
/usr/sbin/swapon -p 0 /dev/dsk/c10t0d0 /dev/dsk/c13t0d0
```

This example enables paging to a block device, using the space after the end of the file system for paging and letting the priority default to 1.

```
/usr/sbin/swapon -e /dev/dsk/c4t0d0
```

This example enables paging to a block device, forcing paging even if a file system exists on the device.

```
/usr/sbin/swapon -f /dev/dsk/c12t0d0
```

WARNINGS

Once file system blocks have been allocated for paging space, the file system cannot be unmounted unless the system is rebooted.

If any paging area becomes unavailable while the system is running, for example if a network failure occurs while paging to a remote system, the system will immediately halt.

The file system block size used by the **-l**, **-m**, and **-r** options varies between file systems, and is defined by the system administrator at the time the file system is created. The **dumpfs** command can be used to determine the block size for a particular file system (see *dumpfs*(1M)).

When using the **-l** and **-r** options, the reserve space specified by the **-r** option takes precedence over the **-l** option. Thus, if:

D	= Total disk space available to ordinary users
R	= Reserve space specified by the **-r** option
limit	= Paging space limit specified by the **-l** option
L	= Space currently available to the paging system
F	= Space currently occupied by the file system

the following relationships hold:

$F + R + limit < D$	In normal operation
$L = 0$	If $F + R >= D$

S

$0 <= L <= limit$ If $F + R + limit >= D$

FILES

/dev/dsk/c*card***t***target***d***device*	Normal paging devices
/etc/fstab	File system table
/etc/savecore.LCK	List of devices being used by **savecore**

AUTHOR

swapon was developed by HP and the University of California, Berkeley.

SEE ALSO

config(1M), savecore(1M), swapinfo(1M), swapon(2), fstab(4).

NAME

swask - ask for user response

SYNOPSIS

swask [**-v**] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *options_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- SD-UX commands are included with the HP-UX Operating System and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The **swask** command runs interactive software **request** scripts for the software objects selected (and for HP OpenView Software Distributor) to one or more targets specified by *target_selections*. These scripts store the responses in a **response** file (named **response**) for later use by the **swinstall** and **swconfig** commands. The **swinstall** and **swconfig** commands can also run the interactive request scripts directly, using the **ask** option.

If the **-s** option is specified, software is selected from the distribution source. If the **-s** option is not specified, software installed on the target systems is selected. For each selected software that has a request script, executing that script generates a response file. By specifying the **-c** *catalog* option, **swask** stores a copy of the response file to that catalog for later use by **swinstall** or **swconfig**.

Options

The **swask** command supports the following options:

-v Turns on verbose output to stdout.

-c *catalog* Specifies the pathname of an exported catalog which stores the response files created by the request script. **swask** creates the catalog if it does not already exist.

If the -c *catalog* option is omitted and the source is local, **swask** copies the response files into the source depot, <*distribution.path*>**/***catalog*.

-C *session_file*
Saves the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Reads the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-s *source* Specifies the source depot (or tape) from which software is selected for the ask operation.

-S *session_file*
Executes **swask** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information from a command-line session with the **-C** *session_file* option.

-t *targetfile* Specifies a default set of *targets* for **swask.**

-x *option=value*
Sets the session *option* to *value* and overrides the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Reads the session options and behaviors from *option_file*.

S

Operands

swask supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swask supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:
[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,l=***location*][**,fr** *<op> revision*]
[**,fa** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields. For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=AA.12**, **r<AA.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=**, is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/ .**

Target Selections

swask supports the following syntax for each *target_selection*.

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

S

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swask** commands. If a default value exists, it is listed after the "=".

ask=true
Executes the request script, if one is associated with the selected software, and stores the user response in a file named **response**.

If **ask=as_needed**, the **swask** command first determines if a response file already exists in the catalog and executes the request script only when a response file is absent.

autoselect_dependencies=true
Controls the automatic selection of prerequisite and corequisite software that is not explicitly selected by the user. When set to **true**, requisite software will be automatically selected for configuration. When set to **false**, requisite software which is not explicitly selected will not be automatically selected for configuration.

autoselect_patches=true
Automatically selects the latest patches (based on superseding and ancestor attributes) for a software object that a user selects. The **patch_filter** option can be used in conjunction with **autoselect_patches** to limit which patches will be selected. Requires patches that are in an enhanced SD format. Patches not in enhanced format will not respond to **autoselect_patches**.

enforce_scripts=true
Stops the request process if a request script fails. If set to **false**, the request process proceeds even when a request script fails.

log_msgid=0
Controls the log level for the events logged to the command log file, the target agent log file, and the source agent log file by prepending identification numbers to log file messages:
0 No such identifiers are prepended (default).
1 Applies to ERROR messages only.
2 Applies to ERROR and WARNING messages.
3 Applies to ERROR, WARNING, and NOTE messages.
4 Applies to ERROR, WARNING, NOTE, and certain other log file messages.

logdetail=false
Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

See **loglevel** below and the *sd*(5) manual page, by typing **man 5 sd**, for more information.

logfile=/var/adm/sw/swask.log
Defines the default log file for swask.

loglevel=1
Controls the log level for the events logged to the command logfile and the target agent logfile. A value of
0 provides no information to the logfile.
1 enables verbose logging of key events to the log files.
2 enables very verbose logging, including per-file messages, to the log files.

patch_filter=*.*
Used in conjunction with the **autoselect_patches** or **patch_match_target options to filter the available** criteria specified by the filter. A key use is to allow filtering by the "category" attribute. Requires patches that are in an enhanced SD patch format.

verbose=1
Controls the verbosity of the output (stdout):
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

Session Files

Each invocation of **swask** defines a task session. The invocation options, source information, software selections, and target hosts are saved before the task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swask.last**. This file is overwritten by each invocation of **swask**.

To save session information in a different location, execute **swask** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for a session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session, specify the session file as the argument for the **-S** *session__file* option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swask** take precedence over the values in the session file.

Software and Target Lists

You can use files containing software and target selections as input to the **swask** command. See the **-f** and **-t** options for more information.

Environment Variables

The environment variable that affects the **swlist** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See *lang*(5) for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Environment variables that affect scripts:

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to SW_LOCATION to locate the product's installed files. The configure script is only run when SW_ROOT_DIRECTORY is "/".

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a request script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

RETURN VALUES

swask returns one of these codes:

0	Command successful on all targets
1	Command failed on all targets
2	Command failed on some targets

DIAGNOSTICS

The **swask** command writes to stdout, stderr, and to the **swask** logfile.

Standard Output

An interactive **swask** session does not write to stdout. A non-interactive **swask** session writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

An interactive **swask** session does not write to stderr. A non-interactive **swask** session writes messages for all WARNING and ERROR conditions to stderr.

Logging

Both interactive and non-interactive **swask** sessions log summary events at the host where the command was invoked. They log detailed events to the **swask.log** logfile associated with each *target_selection*.

Command Log
The **swask** command logs all stdout and stderr messages to the the logfile **/var/adm/sw/swask.log**. Similar messages are logged by an interactive **swask** session. You can specify a different logfile by modifying the **logfile** option.

EXAMPLES

Run all request scripts from the default depot (**/var/spool/sw**) depot and write the response file (named **response**) back to the same depot:

```
swask -s /var/spool/sw \*
```

Run the request script for **Product1** from depot **/tmp/sample.depot.1** on remote host **swposix**, create the catalog **/tmp/test1.depot** on the local controller machine, and place the response file (named **response**) in the catalog:

```
swask -s swposix:/tmp/sample.depot.1 -c /tmp/test1.depot Product1
```

Run request scripts from remote depot **/tmp/sample.depot.1** on host **swposix** only when a response file is absent, create the catalog **/tmp/test1.depot** on the local controller machine, and place the response file (named **response**) in the catalog:

S

```
swask -s swposix:/tmp/sample.depot.1 -c /tmp/test1.depot
-x ask=as_needed \*
```

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **$HOME/.sw/defaults**.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options, with their default values, for documentation purposes only.

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of log files.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/adm/sw/swask.log
Contains all stdout and stderr messages generated by **swask**.

AUTHOR

swask was developed by the Hewlett-Packard Company.

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(5), swconfig(1M), and swinstall(1M).

NAME

swconfig - configure, unconfigure, or reconfigure installed software

SYNOPSIS

swconfig [**-p**] [**-u**] [**-v**] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- SD-UX commands are included with the HP-UX Operating System and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The **swconfig** command configures, unconfigures, and reconfigures installed and linkinstalled software products for execution on the specified targets. The **swconfig** command transitions software between INSTALLED and CONFIGURED states.

Software is automatically configured and unconfigured as part of the **swinstall** and **swremove** commands (respectively). The user can defer configuration when software is installed. The **swconfig** command can (un)configure software independent of **swinstall** and **swremove**, e.g. to configure (unconfigure) hosts that share software from a server host where the software is actually installed. The **swconfig** command must also be executed when the initial configuration by **swinstall** failed, was deferred, or needs to be changed.

Configuration primarily involves the execution of vendor-supplied configure scripts. These scripts perform configuration tasks which enable the use of the software on the target hosts. The **swconfig** command also allows software to unconfigure the hosts on which it no longer will be run. A vendor can supply unconfigure scripts to "undo" the configuration performed by the configure script.

The configure scripts are not run by **swinstall** and **swremove** when an alternate root directory is specified. Instead, the **swconfig** command must be run after that software has been made available to client hosts, to configure those hosts. Similarly, **swconfig** must be used on client hosts to unconfigure those hosts. Configuration can also be deferred on software installed to the root directory **/**, for example when multiple configured versions have been allowed, by using the **defer_configure** option with **swinstall**.

Other features of **swconfig** include:

- The **swconfig** command supports only configuration of compatible software by default, controllable through the **allow_incompatible** option.
- If a fileset specifies a prerequisite on other software, that software must be in a "configured" state before the software specifying the dependency will be configured.
- The **swconfig** command will configure multiple versions of a product if the user has set **allow_multiple_versions=true**. The vendor must therefore detect and prevent multiple configured versions in their configure scripts, if that is necessary.
- A vendor's configure script is as useful for operations required for software updates as for new installs. The scripts must also be designed to handle reinstall.
- The ability to ask for a user response by running a **request** script. See the **ask** default option for more information.

Options

swconfig supports the following options:

-c *catalog* Specifies the pathname of an exported catalog which stores copies of the response file or files created by a request script (if **-x ask=true** or **-x ask=as_needed**). Response files are also stored in the **Installed Products Database**.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* (Applies only to HP OpenView Software Distributor.) Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-p Previews a configuration task by running the session through the analysis phase only.

-Q *date* (Applies only to HP OpenView Software Distributor.) Schedules the job for this date. The date's format can be changed by modifying the file **/var/adm/sw/getdate.templ**.

-S *session_file*
Execute **swconfig** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-u Causes **swconfig** to unconfigure the software instead of configuring it.

-v Turns on verbose output to stdout. (The **swconfig** logfile is not affected by this option.) Verbose output is enabled by default; see the **verbose** option below.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

S

Software Selections

The **swconfig** command supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:
[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,l=***location*][**,fr** <*op*> *revision*]
[**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.

- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.

- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings.

- No space or tab characters are allowed in a software selection.

- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

Target Selections

swconfig supports this syntax for each *target_selection*.

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swlist** commands. If a default value exists, it is listed after the "=".

The policy options that apply to **swconfig** are:

agent_auto_exit=true

Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when -p (preview) is used. This enhances network reliability and performance. The default is **true** - the target agent will automatically exit when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

S

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

allow_incompatible=false

Requires that the software products which are being configured be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

allow_multiple_versions=false

Prevents the configuration of another, independent version of a product when a version already is configured at the target.

If set to **true**, another version of an existing product can be configured in its new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

ask=false

When **ask=true**, executes a **request script**, which asks for a user response. If **ask=as_needed**, the **swask** command first determines if a response file already exists in the control directory and executes the **request** script only when a response file is absent.

If set to **ask=true**, or **ask=as_needed**, you can use the **-c** *catalog* option to specify the pathname of an exported catalog to store copies of the response file or files created by the **request** script.

See *swask*(1M) for more information on **request** scripts.

autoremove_job = false

(Applies only to HP OpenView Software Distributor.) Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or controller/agent logfiles) cannot be queried with **swjob**.

autoselect_dependencies=true

Controls the automatic selection of prerequisite and corequisite software that is not explicitly selected by the user. This option does not apply to **swconfig -u**. The default is: **true.** The requisite software will be automatically selected for configuration. Specifying **false** causes requisite software, which is not explicitly selected, to not be automatically selected for configuration.

autoselect_dependents=false

Controls the automatic selection of dependent software that is not explicitly selected by the user. A dependent is the opposite of a requisite. A dependent fileset has established either a prerequisite or a corequisite on the fileset under discussion. Specifying **true** causes dependent software to be automatically selected for unconfiguration. The default, **false** causes dependent software, which is not explicitly selected, to not be automatically selected for unconfiguration.

S

controller_source

Location of a depot for the controller to access to resolve selections. This has no effect on which sources the target uses. Specify this as host, /path, or host:/path. Useful for reducing network traffic between controller and target.

enforce_dependencies=true

Requires that all dependencies specified by the *software_selections* be resolved at the *target_selections*.

The **swconfig**, command will not proceed unless the dependencies have also been selected or already exist at the target in the correct state (INSTALLED or CONFIGURED). This prevents unusable software from being configured on the system.

If set to **false**, dependencies will still be checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite dependencies, if not enforced, may cause the configuration to fail.

job_title=
(Applies only to HP OpenView Software Distributor.) This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

log_msgid=0
Controls whether numeric identification numbers are prepended to logfile messages produced by SD. A value of 0 (default) indicates no such identifiers are attached. Values of 1-4 indicate that identifiers are attached to messages:
1 applies to ERROR messages only
2 applies to ERROR and WARNING messages
3 applies to ERROR, WARNING, and NOTE messages
4 applies to ERROR, WARNING, NOTE, and certain other logfile messages.

logdetail=false
Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

See **loglevel** below and the *sd*(5) manual page, by typing **man 5 sd** , for more information.

logfile=/var/adm/sw/swconfig.log

This is the default command log file for the **swconfig** command.

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. (See **logdetail** above and the *sd*(5) manual page, by typing **man 5 sd** , for more information.) A value of

0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging to the logfiles.

mount_all_filesystems=true
By default, the **swconfig** command attempts to automatically mount all filesystems in the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

reconfigure=false
Prevents software which is already in the CONFIGURED state from being reconfigured. If set to **true**, CONFIGURED software can be reconfigured.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and on which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running the **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence.

select_local=true
If no *target_selections* are specified, select the local host as the target of the command.

`software=`
: Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

`targets=`
: Defines the default *target_selections*. There is no supplied default (see **`select_local`** above). If there is more than one target selection, they must be separated by spaces.

`verbose=1`
: Controls the verbosity of the output (stdout). A value of
  - **`0`** disables output to stdout. (Error and warning messages are always written to stderr).
  - **`1`** enables verbose messaging to stdout.

`write_remote_files=false`
: Prevents the configuring of files on a target which exists on a remote (NFS) filesystem. All files on a remote filesystem will be skipped.

  If set to **`true`** and if the superuser has write permission on the remote filesystem, the remote files will not be skipped, but will be configured.

Session File

Each invocation of the **`swconfig`** command defines a configuration session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is automatically saved to the file **`$HOME/.sw/sessions/swremove.last`**. This file is overwritten by each invocation of **`swconfig`**.

You can also save session information to a specific file by executing **`swconfig`** with the **`-C`** *session__file* option.

A session file uses the same syntax as the defaults files. If you do not specify a specific path for the session file, the default location is **`$HOME/.sw/sessions/`**.

To re-execute a session file, specify the session file as the argument for the **`-S`** *session__file* option of **`swconfig`**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **`swconfig`** take precedence over the values in the session file.

Environment Variables

The environment variable that affects the **`swconfig`** command is:

`LANG` Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **`C`** is used. See *lang*(5) for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **`/etc/rc.config.d/LANG`**. For example, **`/etc/rc.config.d/LANG`**, must be set to **`LANG=ja_JP.SJIS`** or **`LANG=ja_JP.eucJP`** to make the agent and daemon log messages display in Japanese.

Environment variables that affect scripts are:

`SW_CONTROL_DIRECTORY`
: Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

`SW_LOCATION`
: Defines the location of the product, which may have been changed from the default product directory. When combined with the **`SW_ROOT_DIRECTORY`**, this variable tells scripts where the product files are located.

`SW_PATH`
: A PATH variable which defines a minimum set of commands available for use in a control script (e.g. **`/sbin:/usr/bin`**).

S

SW_ROOT_DIRECTORY

Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to SW_LOCATION to locate the product's installed files. The configure script is only run when SW_ROOT_DIRECTORY is "/".

SW_SESSION_OPTIONS

Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a **request** script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC

This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Signals

The **swconfig** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swconfig** prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

Each agent will complete the configuration task (if the execution phase has already started) before it wraps up. This avoids leaving software in a corrupt state.

RETURN VALUES

The **swconfig** command returns:

0 The *software_selections* were successfully configured.
1 The configure operation failed on all *target_selections*.
2 The configure operation failed on some *target_selections*.

DIAGNOSTICS

The **swconfig** command writes to stdout, stderr, and to specific logfiles.

Standard Output

The **swconfig** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

The **swconfig** command also writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swconfig** command logs summary events at the host where the command was invoked. It logs detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log

The **swconfig** command logs all stdout and stderr messages to the the logfile **/var/adm/sw/swconfig.log**. (The user can specify a different logfile by modifying the **logfile** option.)

Target Log

A **swagent** process performs the actual configure operation at each *target_selection*. The **swagent** logs events to the file **/var/adm/sw/swagent.log**.

The following line applies only to HP OpenView Software Distributor.

Command and target log files can be viewed using the **swjob** command.

S

EXAMPLES

Configure the C and Pascal products on the local host:

```
swconfig cc pascal
```

Configure *Product1*, use any associated response files generated by a request script, and save response files under **/tmp/resp1**:

```
swconfig -x ask=true -c /tmp/resp1 Product1
```

Reconfigure the HP Omniback product:

```
swconfig -x reconfigure=true Omniback
```

Configure the version of HP Omniback that was installed at **/opt/Omniback_v2.0**:

```
swconfig Omniback,l=/opt/Omniback_v2.0
```

Unconfigure the *software_selections* listed in the file **/tmp/install.products** on the hosts listed in the file **/tmp/install.hosts**:

```
swconfig -u -f /tmp/install.products -t /tmp/install.hosts
```

The following example applies only to HP OpenView Software Distributor

Configure the C and Pascal products on remote hosts:

```
swconfig cc pascal @  hostA hostB hostC
```

LIMITATIONS

The SD-UX version of **swconfig** does not support the configuration, unconfiguration, or reconfiguration of installed software on remote targets.

The following paragraph applies only to HP OpenView Software Distributor.

The **swconfig** command does not apply to PC controllers or PC targets. For PC targets, configuration operations are packaged into the PC product as one or more of its actions, and then executed when the PC product is installed.

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD software management command options.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD software management commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all configurable and non-configurable data for SD software management commands. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD software management command options.

/var/adm/sw/getdate.templ
Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

AUTHOR

swconfig was developed by the Hewlett-Packard Company.

SEE ALSO

Managing HP-UX Software with SD-UX manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swgettools - utility for retrieving the SD product from new SD media in preparation for an OS update

SYNOPSIS

swgettools [**-s** *source*][**-t** *temp_directory_path*]

DESCRIPTION

The **swgettools** command updates or reinstalls the latest SD commands (SW-DIST product) to your system from media or a depot. The new SD commands are needed to install updated releases of HP-UX.

Prerequisites

- The **swgettools** script needs a temporary directory with at least 2 MB of free space. If there is not enough space in the temporary directory, **swgettools** will fail.

  By default, **swgettools** uses the **/var/tmp** directory. Use the **bdf /var/tmp** command to determine if **/var/tmp** has adequate space.

  If you do not have 2 MB free in **/var/tmp**, use the **-t** *temp_dir_location* option to specify a different temporary directory.

- The **SW-GETTOOLS** product loaded by **swgettools** requires enough space to install the following files:

/var/adm/sw/lbin	*(1 MB)*
/var/adm/sw/sbin	*(3 MB)*
/var/adm/sw/lib	*(6 MB)*
/usr/lbin/sw/bin	*(5 MB)*

  These files are automatically removed when you reboot.

- Your system needs at least 32 MB of RAM to successfully update to HP-UX version 10.30 or greater.

Procedure

To update SD, you must first load the **swgettools** command onto your system. You can then run **swgettools** to get the new SW-DIST product, which contains the new SD-UX commands.

- **swgettools** is shipped in the **catalog/SW-GETTOOLS/pfiles** directory. Depending on whether the HP-UX software is on CD, tape, or a remote system in a software depot, use **cp**, **tar**, or **rcp** to load **swgettools** onto your system. (HP recommends that you use the default directory, **/var/tmp**.)
- Run **swgettools**. This creates the **SW-GETTOOLS** product, which updates **SW-DIST**.
- Use the new version of **swinstall** to update the rest of the operating system. See *swinstall*(1M) for more information.

  CAUTION: You **MUST** use the latest version of **swinstall** to update your system to the latest version of HP-UX. If you use a previous version of **swinstall**, the update will fail.

- Reboot the system.

For complete instructions regarding updating HP-UX, see *Installing HP-UX 11.0 and Updating HP-UX 10.x to 11.0*.

CAUTION: Ensure that the booted kernel is **/stand/vmunix** before you install any kernel software. The **swinstall** process assumes that the system has booted using the kernel at **/stand/vmunix**. Installing kernel software or performing an operating system update with any other kernel might result in loss of data or other errors.

Options

The **swgettools** command supports the following options:

-s *source* Specifies the path for the source media. Possible locations are: a local directory that is an SD depot, a character-special tape device file connected to a tape drive that has an SD media tape loaded, a CD-ROM mount point that has an SD media CD-ROM loaded, or a remote machine and depot combination. The default source type is *directory*. The syntax is:

[*host*][**:**][**/***directory*]

The remote host can be specified by its host name, domain name, or Internet address. The absolute path to the remote depot follows, separated by a colon with no spaces. The colon is required when both a host and directory are specified. The directory path must be absolute. For example, to specify a remote machine and depot combination:

```
swperf:/var/spool/sw
```

-t *temp_directory_path*
Specifies a temporary directory to use during the **swgettools** process. An absolute pathname is required. By default **/var/tmp** is used. The temporary directory must exist and must have at least 2 MB of free disk space for **swgettools** to succeed. (You can use the **bdf** *temp_directory_path* command to determine if the directory has adequate space.)

RETURN VALUES

The **swgettools** command returns:

0 Successful completion
1 Usage incorrect
2 Error during execution

EXAMPLES

To install the new SW-DIST product from CD-ROM media at **/mnt/cdrom_depot**:

```
cp /mnt/cdrom_depot/catalog/SW-GETTOOLS/pfiles/swgettools /var/tmp
/var/tmp/swgettools -s /mnt/cdrom_depot
```

To install the new SW-DIST product from tape media at **/dev/rmt/0m**:

```
cd /var/tmp
tar -xvf /dev/rmt/0m catalog/SW-GETTOOLS/pfiles/swgettools
cp /var/tmp/catalog/SW-GETTOOLS/pfiles/swgettools /var/tmp/swgettools
rm -rf /var/tmp/catalog
/var/tmp/swgettools -s /dev/rmt/0m
```

To install the new SW-DIST from a remote depot on *swperf* at **/var/spool/sw**:

```
rcp swperf:/var/spool/sw/catalog/SW-GETTOOLS/pfiles/swgettools /var/tmp
/var/tmp/swgettools -s swperf:/var/spool/sw
```

AUTHOR

swgettools was developed by the Hewlett-Packard Company.

FILES

The **swgettools** command installs the following supporting files into four directories. These files are removed when the system is rebooted after the installation or update is complete.

/var/adm/sw/lbin	˜ 1 MB
/var/adm/sw/sbin	˜ 3 MB
/var/adm/sw/lib	˜ 6 MB
/usr/lbin/sw/bin	˜ 5 MB

SEE ALSO

Managing HP-UX Software with SD-UX, Installing HP-UX 11.0 and Updating HP-UX 10.x to 11.0, sd(5), swinstall(1M).

S

NAME

swinstall, swcopy - install and configure software products; copy software products for subsequent installation or distribution

SYNOPSIS

swinstall [*XToolkit Options*] [**-i**] [**-p**] [**-r**] [**-v**] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

swcopy [*XToolkit Options*] [**-i**] [**-p**] [**-v**] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- **swinstall** and **swcopy** have an interactive user interface. You can invoke it by typing **swinstall**, **swcopy**, or by including the **-i** option on the command line.
- SD-UX commands are included with the HP-UX operating system and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar the following:

  The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The **swinstall** command installs the *software_selections* from a software *source* to either the local host or, in the case of the HP OpenView Software Distributor product, to one or more *target_selections* (root filesystems). By default, the software is configured for use on the target after it is installed. (The software is not configured when installed into an alternate root directory.)

The **swcopy** command copies or merges *software_selections* from a software *source* to one or more software depot *target_selections* These depots can then be accessed as a software source by the **swinstall** command.

Updating the Operating System

To perform an OS update with **swinstall** (or to reinstall SD from media), you must use first use the **swgettools** command to get the newest version of **swinstall**.

CAUTION: You **MUST** use the latest version of **swinstall** to update your system to the latest version of HP-UX. If you use a previous version of **swinstall**, the update will fail.

The **os_name** and **os_release** options let you specify the desired OS name and release during an HP-UX update. (These options should only be specified from the command line.) The SD **readme** file lists correct syntax for these options. You can display the **readme** file by entering:

```
swlist -a readme -l product SW-DIST
```

The **match_target** option, if set to **true**, selects software by locating filesets on the source that match the target system's installed filesets.

Refer to the **Default Options** section of this manual page, *swgettools*(1M), and *Installing HP-UX 11.0 and Updating HP-UX 10.x to 11.0* for more information.

Installing Kernel Software

In HP-UX, the kernel installation process requires that the system boots using the kernel at **/stand/vmunix**. Make sure that your system is booted to the **/stand/vmunix** kernel before you install any kernel software or perform an operating system update.

Installing PC Software

The following paragraph applies only to HP OpenView Software Distributor.

For PC software installation, the **swinstall** command first copies or merges *software_selections* from a software *source* to one or more PC *target_selections* (PC controllers). Each PC controller is a fanout server, providing the *software_selections* (copied to it) to PC targets. At each PC target an SD PC agent process performs the actual installation.

Features and Differences between swinstall and swcopy

The key difference between **swinstall** and **swcopy** is that **swinstall** installs software for actual (or eventual) use, while **swcopy** copies software into a depot, making it available as a source for installation by **swinstall**.

NOTE: To copy to a tape, see the *swpackage*(1M) manpage.

Other features (differences) include:

- The **swinstall** command executes several vendor-supplied scripts during the installation and configuration of the *software_selections*. The **swcopy** command does not execute these scripts. The **swinstall** command supports the following scripts:

  request a script that asks the user questions and stores responses in a **response** file. The response file can then be used by configuration or other scripts.

  checkinstall a script executed during the analysis of a **target_selection**, it checks that the installation can be attempted. If this check fails, the software product is not installed.

  preinstall a script executed immediately before the software's files are installed.

  postinstall a script executed immediately after the software's files are installed.

  configure a script executed during the configuration of a *target_selection*, it configures the target for the software (and the software for the target). The **preinstall** and **postinstall** scripts are not intended to be used for configuration tasks. They are to be used for simple file management needs such as removing obsolete files from the previous revision (which was just updated).

  unpreinstall a script executed immediately after the software's actual files are restored if the software install will fail and the **autorecover_product** option is set to **true**. The script undoes the steps performed by **preinstall** script.

  unpostinstall a script executed immediately before the software's actual files are restored if the software install failed and the **autorecover_product** option is set to **true**. The script undoes the steps performed by **postinstall** script.

- When a depot is created or modified using **swcopy**, **catalog files** are built that describe the depot (as opposed to the **Installed Products Database** (IPD) files that are built by the **swinstall** command).
- By default, the **swinstall** command only allows the selection of compatible software from the *source*. This constraint ensures that the architecture of the software matches that of the *target_selections*. No compatibility checks are performed by the **swcopy** command. (A depot can be a repository of software targeted for a variety of architectures and operating systems.)
- By default, **swinstall** supports updates to higher revisions of software. If a *software_selection* of the same revision is already installed, **swinstall** will not reinstall it. If a *software_selection* has a lower revision than the same software which is already installed, **swinstall** will not reinstall it. (The user can override these behaviors with control options.)
- The **swinstall** command creates hard links and symbolic links as specified for the software. If it encounters a symbolic link where it expected a regular file, **swinstall** follows the symbolic link and updates the file to which it points.
- The **swinstall** command does not remove a product's current files before installing the new ones. A fileset's install scripts can do that, if necessary. Files being replaced are overwritten unless they are in use. If in use, they are unlinked or moved to #<file>. If the *autorecover_product* option is set to **true**; all files are saved to #<file>, and restored if the install fails.
- The **swinstall** command supports kernel building scripts and rebooting. Before or after software that modifies the kernel is installed or updated, **swinstall** executes system-specific scripts to prepare for or build the new version of the kernel. The remaining *software_selections* are then installed. These scripts are defined in **swagent** options and include: **install_setup_cmd**, **system_prep_cmd**, **kernel_build_cmd**, and

S

install_cleanup_cmd.

After software that requires a system reboot is installed or updated, **swinstall** automatically reboots the system. The reboot command is defined by the **swagent** option: **reboot_cmd**.

When updating the operating system, you must use first use the **swgettools** command to get the newest version of **swinstall**. (See *swgettools*(1M) for more information.) Then you should install kernel software first to ensure that a new kernel can be generated before the rest of the operating system is updated. After all the *software_selections* are updated or installed, **swinstall** reboots using the new kernel, then executes the configure scripts for each *software_selection*. After these scripts complete, it reboots the system again to restore it to its normal state.

- No kernel building or system reboots are performed by **swcopy**.
- Both the **swinstall** and **swcopy** commands perform various checks prior to installing or copying the *software_selections*, for example disk space analysis.

Options

swinstall and **swcopy** support the following options:

XToolKit Options
: The **swinstall** and **swcopy** commands support a subset of the standard X Toolkit options to control the appearance of the GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X*(1) manual page by typing **man X** for a definition of these options.

-i
: Runs the command in interactive mode by invoking the Graphical User Interface (GUI). [Note: The GUI is only supported on HP-UX]. The SD-UX **swinstall**, **swcopy**, and **swremove** commands also support an interactive terminal user interface (text based) in which screen navigation is done with the keyboard (no mouse).

-l
: (*Applies only to* **HP-UX 10.X**.) Runs the command in *linkinstall* mode which makes software installed under a server's *shared root* available to a diskless client's *private root* (HP-UX only).

When run in the *linkinstall* mode, **swinstall**:

- Creates NFS mounts to the software to make it accessible from the target. This may involve delayed mounting for alternate roots.
- Modifies the target's *fstab* file.
- Modifies the source's **exports** file to add mount permission for the target.

Mounts are created by examining the *share_link* product attribute. Not all products support **linkinstall**. Some products may be visible without creating a new mount if they reside under an old one.

S

-p
: Previews an install task by running the session through the analysis phase only.

-r
: (Optional) Causes the command to operate on *target_selections* that are alternate root directories (root filesystems other than **/**).

Note that you cannot use this option to relocate software during installation. You must use the **l=location** syntax in the software selection component.

-v
: Turns on verbose output to stdout. (The **swinstall** or **swcopy** logfile is not affected by this option.) Verbose output is enabled by default; see the **verbose** option below.

-c *catalog*
: Specifies the pathname of an exported catalog which stores copies of the response file or files created by a request script (if **-x ask=true** or **-x ask=as_needed**). The response files are also stored in the **Installed Products Database** after the installation process is complete.

-C *session_file*
: Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* *(Applies only to HP OpenView Software Distributor.)*

Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-Q *date* *(Applies only to HP OpenView Software Distributor.)*

Schedules the job for this date. The date's format can be changed by modifying the file **/var/adm/sw/getdate.templ**.

-s *source* Specifies the source depot (or tape) from which software is installed or copied. The default source type is *directory*. The syntax is:

[*host*][**:**][**/***directory*] A host may be specified by its host name, domain name, or internet address. A directory must be specified by an absolute path.

-S *session_file*
Execute **swinstall** or **swcopy** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information from a command-line session with the **-C** *session_file* option.

-t *target_file* (*Applies only to HP OpenView Software Distributor*.)

Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

The **swinstall** and **swcopy** commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swinstall and **swcopy** support the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,l=***location*][**,fr** *<op> revision*]
[**,fa** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

S

[ ], *****, **?**, **!**

For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

Target Selection

The **swinstall** and **swcopy** commands support the following syntax for each *target_selection*. The **:** (colon) is required if both a host and directory are specified.

[*host*][**:**][**/***directory*]

A host may be specified by its host name, domain name, or internet address. A directory must be specified by an absolute path.

For *linkinstall,* on HP-UX 10.* systems: if the [*directory*] part of the selection is a relative path, then the value of *default.shared_root=true* is pre-pended for sources and the value of *default.private_root=true* is pre-pended for targets. These are normally **/export/shared_roots** and **/export/private_roots**, respectively.

PC Targets

The following applies only to HP OpenView Software Distributor.

The **swcopy** command supports the syntax:

[*pc_controller*]

and the **swinstall** command supports the syntax:

[*pc_controller*][**::**][*pc_target*]

This syntax applies only to PCs. The PC controller is a fanout server. The PC target may be a PC machine, user, or group name. Valid targets for a PC controller can be listed using **swlist -l machine|user|group**. PC targets can be further qualified for whether they refer to a PC machine, user, or group type with the following syntax:

name[**,t=***type*][**,k=***address*]

The *type* only needs to be specified if a name applies to more than one **machine**, **user**, or **group**. (The *address* is used internally for machines and is generally not needed on the command line.) The keyword ***** can be substituted for **pc_target**, specifying an installation to all target machines:

@ *pc_controller***::***

S

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swinstall** and **swcopy** commands. If a default value exists, it is listed after the "=".

agent_auto_exit=true
Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when -p (preview) is used. This enhances network reliability and performance. The default is **true** - the target agent automatically exits when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000
Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI is used. The default of 10000 is slightly less than 7 days.

allow_downdate=false
(*Applies only to* **swinstall**.) Prevents the installation of an older revision of fileset that already exists at the target(s). (Many software products do not support "downdating".) If set to **true**, the older revision can be installed.

allow_incompatible=false
(*Applies only to* **swinstall**.) Requires that the software products which are being installed be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

allow_multiple_versions=false
(*Applies only to* **swinstall**.) Prevents the installation of another, independent version of a product when a version already is already installed at the target.

If set to **true**, another version of an existing product can be installed into a new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

ask=false
(*Applies only to* **swinstall**.) When **ask=true**, executes a request script which asks for a user response. If **ask=as_needed**, the **swinstall** command first determines if a response file already exists in the catalog specified in the **-c** option or source depot and executes the request script only when a response file is absent.

If set to **ask=true**, or **ask=as_needed**, you can use the **-c** *catalog* option to specify the pathname of an exported catalog to store copies of the response file or files created by the request script.

See *swask*(1M) for more information on request scripts.

autoreboot=false
(*Applies only to* **swinstall**.) Prevents the installation of software requiring a reboot from the non-interactive interface. If set to **true**, this software can be installed and the target system(s) will be automatically rebooted.

An interactive session always asks for confirmation before software requiring a reboot is installed.

autorecover_product=false
(*Applies only to* **swinstall**.) Causes **swinstall** to remove the original files as they are updated. If an error occurs during the installation (e.g. network failure), then the original files are lost, and the installation must be re-tried.

If set to **true**, all files are saved as backup copies until all filesets in the current product loading are complete; then they are removed. At the cost of a temporary increase in disk space and slower performance, this allows for automatic recovery of the original filesets in that product if the load fails.

The following option applies only to HP OpenView Software Distributor.

autoremove_job=false

Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or target logfiles) cannot be queried with **swjob**.

Install jobs to PCs cannot be automatically removed. They should not be removed until the job completes on all PC targets.

autoselect_dependencies=true

Automatically select dependencies when software is being selected. When set to **true**, and any software which has dependencies is selected for install, **swinstall** or **swcopy** makes sure that the dependencies are met. If they are not already met, they are automatically selected for you. If set to **false**, automatic selections are not made to resolve requisites.

autoselect_patches=true

Automatically selects the latest patches (based on superseding and ancestor attributes) for a software object that a user selects for a **swinstall** or **swcopy** operation. When set to **false**, the patches corresponding to the selected object are not automatically selected.

The **patch_filter=** option can be used in conjunction with **autoselect_patches**.

autoselect_reference_bundles=true

If **true**, bundles that are **sticky** are automatically installed or copied, along with the software it is made up of. If **false**, the software can be installed, or copied, without automatically including sticky bundles that contain it.

codeword=

Provides the "codeword" needed to unlock protected HP CD-ROM software.

Some HP software products are shipped on CD-ROM as "protected" products. That is, they cannot be installed or copied unless a "codeword" and "customer ID" are provided. The codeword is found on the CD-ROM certificate which you received from HP. You may use this default specification on the command line or the SD-UX Interactive User Interface to enter the codeword.

This default stores the codeword for future reference, and you need to enter the codeword only once. If you purchase a new HP product and a previous codeword has already been entered for that CD-ROM, just enter the new codeword as usual and the codewords will be merged internally.

NOTE: For HP-UX B.10.10 and later systems, SD searches the **.codewords** file on the server that is providing protected software to other hosts. It looks for valid customer_id/codeword pairs. In doing so, SD eliminates the need to enter codewords and customer_ids on every host that is "pulling" the software.

S

To properly store the customer_id/codeword for a CD-ROM, run **swinstall -p** or **swcopy -p** on the host serving the CD-ROM. After the codeword has been stored, clients installing or copying software using that host and CD-ROM as a source will no longer need a codeword or customer_id.

controller_source=

Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location: [*host*][**:**][**/***directory*]

The **controller_source_option** supports the same syntax as the **-s** *source* option. This option has no effect on which sources the target uses and is ignored when used with the Interactive User Interface.

create_target_path=true

Causes the agent to create the target directory if it does not already exist. If set to **false**, a new target directory is not created. This option can prevent the erroneous creation of new target depots or new alternate root directories.

compress_files=false

(*Applies only to* **swcopy**.) If set to **true**, files not already compressed are compressed before transfer from a source. This enhances performance on slower networks for **swinstall** and **swcopy**, and results in smaller depots for **swcopy**, unless **uncompress_files** is also set to **true**.

customer_id=

This number, also printed on the Software Certificate, is used to "unlock" protected software and restrict its installation to a specific site or owner. It is entered using the **-x** *customer_id=* option or by using the Interactive User Interface. The *customer_id* can be used on any HP-UX 10.0X compatible HP9000 system.

defer_configure=false

(*Applies only to* **swinstall**.) Causes **swinstall** to automatically configure the *software_selections* after they are installed. When an alternate root directory is specified, **swinstall** never performs the configuration task, since only hosts using the software should be configured. If set to **true**, this option allows configuration to be deferred even when the root directory is **/**.

An additional version of a product will not be configured if another version is already configured. The **swconfig** command must be run separately.

distribution_source_directory=/var/spool/sw

Defines the default location of the source depot. This syntax can be *host***:***path*. The **-s** option overrides this value.

distribution_target_directory=/var/spool/sw

(*Applies only to* **swcopy**.) Defines the default location of the target depot.

enforce_dependencies=true

Requires that all dependencies specified by the *software_selections* be resolved either in the specified source, or at the *target_selections* themselves.

The **swinstall** and **swcopy** commands will not proceed unless the dependencies have also been selected or already exist at the target in the correct state (INSTALLED or AVAILABLE). This prevents unusable software from being installed on the system. It also ensures that depots contain usable sets of software.

If set to **false**, dependencies are still checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite dependencies, if not enforced, may cause the installation or configuration to fail.

enforce_dsa=true

Prevents the command from proceeding past the analysis phase if the disk space required is beyond the available free space of the impacted filesystem(s). If set to **false**, the install or copy operation uses the filesystems' minfree space and may fail because it reaches the filesystem's absolute limit.

enforce_kernbld_failure=true

(*Applies only to* **swinstall**.) Prevents **swinstall** from proceeding past the kernel build phase if the kernel build processes fail. If set to **false**, the install operation continues (without suspension if in the interactive mode) despite failure or warnings from either the system preparation process or the kernel build process.

enforce_scripts=true

(*Applies only to* **swinstall**.) By default, if a fileset **checkinstall** script fails (i.e. returns with an exit code 1), that fileset is not installed. If a product **checkinstall** script fails, no filesets in that product are installed. If set to **false**, the install proceeds even if a **checkinstall** script fails.

job_polling_interval=30

The following option applies only to HP OpenView Software Distributor

(Applies only to **swinstall**.*)* Defines the polling interval, in minutes, used by the daemon. It specifies how often a PC install job is polled to cache the progress of remote targets on the controller.

job_title=

The following option applies only to HP OpenView Software Distributor

This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked. The default value is to have no title. If a title is specified, it should be enclosed in quotes.

layout_version=1.0
(Applies only to **swcopy***.)* Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd*(5) for more information.

logdetail=false
Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

See **loglevel=1** and the *sd*(5) manual page by typing **man 5 sd** for more information.

logfile=/var/adm/sw/swremove.log
This is the default command log file for the **swinstall** command.

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. (See **logdetail=false** and the *sd*(5) manual page for more information.) A value of:

0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging, including per-file messages, to the logfiles.

log_msgid=0
Controls whether numeric identification numbers are prepended to logfile messages produced by SD:

0 (default) No identifiers are attached to messages.
1 Applies to ERROR messages only.
2 Applies to ERROR and WARNING messages.
3 Applies to ERROR, WARNING, and NOTE messages.
4 Applies to ERROR, WARNING, NOTE, and certain other logfile messages.

match_target=false
(*Applies only to* **swinstall**.) If set to **true**, software selection is done by locating filesets on the source that match the target system's installed filesets. If multiple targets are specified, the first in the list is used as the basis for selections.

mount_all_filesystems=true
Attempt to mount all filesystems in the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

os_name
(*Applies only to* **swinstall**.) This option can be used in conjunction with **os_release** to specify the desired OS name during an HP-UX update. The **os_name** option should only be specified from the command line. Refer to the SD **readme** file for correct syntax. You can display the **readme** file by entering:

swlist -a readme -l product SW-DIST

os_release
(*Applies only to* **swinstall**.) This option can be used in conjunction with **os_name** to specify the desired OS release during an HP-UX update. The **os_release** option should only be specified from the command line. Refer to the SD **readme** file for correct syntax. You can display the **readme** file by entering:

swlist -a readme -l product SW-DIST

patch_filter=software_specification
This option can be used in conjunction with the **autoselect_patches** or **patch_match_target** options to filter the selected patches to meet the criteria specified by *software_specification*. The default value of this option is ***.***.

patch_match_target=false
If set to **true**, this option selects the latest patches (software identified by the *is_patch=true* attribute) that correspond to software on the target root or depot.

The **patch_filter=** option can be used in conjunction with **patch_match_target**.

patch_save_files=true
(Applies only to **swinstall***)* Saves the original versions of files modified by patches, which permits the future rollback of a patch. Patched files are saved to **/var/adm/sw/save**. When set to **false**, patches cannot be rolled back (removed) unless the base software modified by the patch is removed at the same time.

To commit a patch by removing the corresponding saved files, use the **swmodify** command's **patch_commit** option.

polling_interval=2
Defines the polling interval, in seconds, used by the interactive GUI or TUI of the controller. It specifies how often each target agent is polled to obtain status information about the task being performed. When operating across wide-area networks, the polling interval can be increased to reduce network overhead.

recopy=false
(*Applies only to* **swcopy**.) Do not copy a fileset that is already available on the target at the same version. If **recopy=true**, copy the fileset in any case.

register_new_depot=true
(*Applies only to* **swcopy**.) Causes **swcopy** to register a newly created depot with the local **swagentd**. This action allows other SD commands to automatically "see" this depot. If set to **false**, a new depot is not automatically registered. It can be registered later with the **swreg** command.

register_new_root=true
(*Applies only to* **swinstall**.) Causes alternate roots to be registered during **swinstall**. These can be listed with **swlist**.

reinstall=false
When re-installing or re-copying an existing revision of a fileset, this option causes that fileset to be skipped, i.e. not re-installed. If set to **true**, the fileset is re-installed or re-copied.

reinstall_files=true
Causes all the files in a fileset to always be reinstalled or recopied, even when the file already exists at the target and is identical to the new file. If set to **false**, files that have the same *checksum* (see next option), size and timestamp are not re-installed. This check enhances performance on slow networks or slow disks.

reinstall_files_use_cksum=true
This option affects the operation when the **reinstall_files** option is set to false. It causes the checksums of the new and old file to be computed and compared to determine if the new file should replace the old one. (The checksum is slower, but is a more robust way to check for files being equivalent.) If set to **false**, the checksums are not computed, and files are reinstalled or not based only on their size and timestamp.

remove_obsolete_filesets=false
(*Applies to* **swcopy**only**)** Controls whether **swcopy** automatically removes obsolete

filesets from target products in the target depot. If set to **true**, **swcopy** removes obsolete filesets from the target products that were written to during the copy process. Removal occurs after the copy is complete. Filesets are defined as obsolete if they were not part of the most recent packaging of the product residing on the source depot.

retry_rpc=1
Defines the number of times a lost source connection is retried during file transfers in **swinstall** or **swcopy**. A lost connection is one that has timed out. When used in conjunction with the **rpc_timeout** option, the success of installing over slow or busy networks can be increased. If set to zero, any **rpc_timeout** to the source causes the task to abort. If set from 1 to 9, the install of each fileset is attempted that number of times. The **reinstall_files** option should also be set to false to avoid installing files within the fileset that were successfully installed.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the *sd*(5) man page by typing **man 5 sd** for more information.

rpc_timeout=5.
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

select_local=true
If no *target_selections* are specified, select the default root directory **/** (**swinstall**), or the default **target_directory** (**swcopy**), at the local host as the target of the command.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

software_view=all_bundles
Indicates the software view to be used as the default level for the software listing in the GUI. It can be set to **all_bundles**, **products**, or a bundle category tag (to indicate to show only bundles of that category). For HP OpenView Software Distributor the default value is **products**.

S

source_cdrom=/SD_CDROM
Defines the default location of the source CD-ROM. This syntax can be *host***:***path*.

source_tape=/dev/rmt/0m
Defines the default location of the source tape, usually the character-special file of a local tape device. If the **host:path** syntax is used, the host must match the local host. The **-s** option overrides this value.

source_type=directory
Defines the default source type: **cdrom**, **directory**, or **tape**. The source type derived from the **-s** option overrides this value.

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

uncompress_files=false
(*Applies only to* **swcopy**.) If set to **true**, files being transferred from a source are uncompressed before **swcopy** store them on the target depot.

use_alternate_source=false
Empowers each target agent to use its own, configured alternate source, instead of the one specified by the user. If **false**, each target agent uses the same source (the source

specified by the user and validated by the command). If **true**, each target agent uses its own configured value for the source.

verbose=1
Controls the verbosity of the output (stdout). A value of
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

write_remote_files=false
Prevents the installation or copying of files to a target which exists on a remote filesystem. All files destined for a remote filesystem are skipped.

If set to **true** and if the superuser has write permission on the remote filesystem, the remote files are installed or copied.

Session File

Each invocation of the **swinstall** or **swcopy** command defines an installation or copy session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swinstall{swcopy}.last**. This file is overwritten by each invocation of **swinstall** or **swcopy**.

You can also save session information from interactive or command-line sessions. From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu. From a command-line session, you can save session information by executing **swinstall** or **swcopy** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for a session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu. To re-execute a session from a command-line, specify the session file as the argument for the **-S** *session__file* option of **swinstall** or **swcopy**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swinstall** or **swcopy** take precedence over the values in the session file.

Software and Target Lists

The **swinstall** and **swcopy** commands support software selections, target selections, and patch filter selections from separate input files.

You can specify software and target selection lists with the **-f** and **-t** options. Software and targets specified in these files are selected for operation instead of (or in addition to) files listed in the command line. (See the **-f** and **-t** options for more information.)

Additionally, the **swinstall** and **swcopy** interactive user interfaces read a default list of hosts on which to operate. The list is stored in:

/var/adm/sw/defaults.hosts	the system-wide default list of hosts
$HOME/.swdefaults.hosts	the user-specific default list of hosts

For each interactive command, target hosts containing roots, depots, and hosts serving as PC controllers are specified in separate lists (**hosts**, **hosts_with_depots**, and **pc_controllers** respectively). The list of hosts are enclosed in {} braces and separated by white space (blank, tab and newline). For example:

```
swinstall.hosts={hostA hostB hostC hostD hostE hostF}
swinstall.pc_controllers={pc1 pc2}   (HP OpenView Software Distributor only.)
swcopy.hosts_with_depots={hostS}
swcopy.pc_controllers={pc1 pc2}      (HP OpenView Software Distributor only.)
```

The **swinstall** and **swcopy** interactive user interfaces read a default list of patch filters that you can use as selection criteria for patch software. The list is stored in:

/var/adm/sw/defaults.patchfilters
the system-wide default list of patch filters.

S

$HOME/.sw/defaults.patchfilters
the user-specific default list of patch filters.

The list of patch filters is enclosed in braces {} and separated by white space (blank, tab, or newline). For example:

```
swinstall.patch_filter_choices={
*.*,c=enhancement
*.*,c=critical
}
swcopy.patch_filter_choices={
Product.Fileset,c=halts_system
}
```

The following paragraph applies only to HP OpenView Software Distributor.

For PC software installation, the interactive interface generates PC target lists by querying the PC controller (and it's associated fileserver). All users, groups, and machines returned from this query are included in the default list from which to choose. Additionally, all machines returned from this query are automatically selected for installation when the user selects a PC controller.

Environment Variables

The environment variable that affects the **swinstall** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Environment variables that affect scripts:

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other **control** scripts for the software are located (e.g. subscripts).

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available to for use in a **control** script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells **control** scripts the root directory in which the products are installed. A script must use this directory as a prefix to SW_LOCATION to locate the product's installed files. The configure script is only run when SW_ROOT_DIRECTORY is "/".

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a request script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

S

Additional environment variables that affect scripts for **swinstall**:

SW_DEFERRED_KERNBLD
This variable is normally unset. If it is set, the actions necessary for preparing the system file **/stand/system** cannot be accomplished from within the *postinstall* scripts, but instead must be accomplished by the *configure*scripts. This occurs whenever software is installed to a directory other than /, such as for a cluster client system. This variable should be read only by the *configure* and *postinstall* scripts of a kernel fileset. The **swinstall** command sets these environment variables for use by the kernel preparation and build scripts.

SW_INITIAL_INSTALL
This variable is normally unset. If it is set, the **swinstall** session is being run as the back end of an initial system software installation ("cold" install).

SW_KERNEL_PATH
The path to the kernel. The default value is **/stand/vmunix**, defined by the **swagent** option or **kernel_path**.

SW_SESSION_IS_KERNEL
Indicates whether a kernel build is scheduled for the current install/remove session. A **TRUE** value indicates that the selected kernel fileset is scheduled for a kernel build and that changes to **/stand/system** are required. A null value indicates that a kernel build is not scheduled and that changes to **/stand/system** are not required.

The value of this variable is always equal to the value of **SW_SESSION_IS_REBOOT**.

SW_SESSION_IS_REBOOT
Indicates whether a reboot is scheduled for a fileset selected for removal. Because all HP-UX kernel filesets are also reboot filesets, the values of this variables is always equal to the value of **SW_SESSION_IS_KERNEL**.

SW_SYSTEM_FILE_PATH
The path to the kernel's system file. The default value is **/stand/system**.

Signals

The **swinstall** and **swcopy** commands catch the signals SIGQUIT and SIGINT. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

Each agent completes the install or copy task (if the execution phase has already started) before it wraps up. This avoids leaving software in a corrupt state.

Locking

SD commands use a common locking mechanism for reading and modifying the Installed Products Database (IPD) and software depots. This mechanism allows multiple readers but only one writer on an IPD or depot:

S

Write Locks

swinstall commands that modify the IPD are restricted from simultaneous modification using *fcntl*(2) locking on the file <*IPD location*>**/swlock** (e.g. **/var/adm/sw/products/swlock**).

swcopy commands that modify a software depot are restricted from simultaneous modification using *fcntl*(2) locking on the file <*depot directory*>**/catalog/swlock** (e.g. **/var/spool/sw/catalog/swlock**).

Read Locks

Both **swinstall** and **swcopy** commands set *fcntl*(2) read locks on source depots using the **swlock** file mentioned above. When a read lock is set, it prevents all SD commands from performing modifications (i.e. from setting write locks).

Terminal Support

For in-depth information about terminal support refer to:

- The *Managing HP-UX Software with SD-UX* manual
- Start the GUI or TUI, select the *Help* menu, then select the *Keyboard...* option to access the *Keyboard Reference Guide*.

RETURN VALUES

An interactive **swinstall** or **swcopy** session always returns 0. A non-interactive **swinstall** or **swcopy** session returns:

0 The *software_selections* were successfully installed/copied.
1 The install/copy operation failed on all *target_selections*.
2 The install/copy operation failed on some *target_selections*.

DIAGNOSTICS

The **swinstall** and **swcopy** commands write to stdout, stderr, and to specific logfiles.

Standard Output

An interactive **swinstall** or **swcopy** session does not write to stdout. A non-interactive **swinstall** or **swcopy** session writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

An interactive **swinstall** or **swcopy** session does not write to stderr. A non-interactive **swinstall** or **swcopy** session writes messages for all WARNING and ERROR conditions to stderr.

Logging

Both interactive and non-interactive **swinstall** and **swcopy** sessions log summary events at the host where the command was invoked. They log detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log

The **swinstall** and **swcopy** commands log all stdout and stderr messages to the the logfile **/var/adm/sw/swinstall.log** (**/var/adm/sw/swcopy.log**). Similar messages are logged by an interactive **swinstall** and **swcopy** session. The user can specify a different logfile by modifying the **logfile** option.

Target Log

A **swagent** process performs the actual install or copy operation at each *target_selection*. For install tasks, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). For copy tasks, the **swagent** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**).

The following line applies only to HP OpenView Software Distributor.

Command and target log files can be viewed using the **swjob** command.

Source Depot Audit Log

If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. (Note that a user running **swinstall/swcopy** from a target machine cannot set this option; only the administrator of the source depot machine can set it. See the *source_depot_audit* option in the *swagent*(1M) man page.)

S

EXAMPLES

swinstall

To invoke an interactive session of **swinstall**:

```
swinstall
```

Select the C and Pascal products from the network source software server (sw_server) and start an interactive session:

```
swinstall -i -s sw_server cc pascal
```

The following example applies only to HP OpenView Software Distributor

Install the C and Pascal products to a set of remote hosts:

```
swinstall -s sw_server cc pascal @ hostA hostB hostC
```

Update the HP Omniback product from a CD-ROM mounted at /cd :

```
swinstall -s /cd/swmedia Omniback
```

Install an incompatible version of HP Omniback into the directory **/exports**:

```
swinstall -x allow_incompatible=true -s/products Omniback,a=arch \
     @ /exports
```

Install all products from the cartridge tape **/dev/rmt/0**:

```
swinstall -s /dev/rmt/0 \*
```

Reinstall the *software_selections* listed in the file **/tmp/install.products** on the hosts listed in the file **tmp/install.hosts:**

```
swinstall -x reinstall=true -f/tmp/install.products \
     -t/tmp/install.hosts
```

Execute **swinstall** interactively using the session file **/tmp/case.selections** as a basis:

```
swinstall -i -S /tmp/case.selections
```

Install all the software from local depot **/tmp/sample.depot.1**, using any response files generated by request scripts:

```
swinstall -s /tmp/sample.depot.1 -x ask=true \*
```

Install **Product1** from remote depot **/tmp/sample.depot.1** on host **swposix** and use an existing response file (previously generated by the **swask** command) located in **/tmp/bar.depot**:

```
swinstall -s swposix:/tmp/sample.depot.1 -c /tmp/bar.depot Product1
```

Install all products in remote depot **/tmp/sample.depot.1** on host **swposix ,** use any response files generated by request scripts, create catalog **/tmp/bar.depot** and copy all response files to the new catalog:

```
swinstall -s swposix:/tmp/sample.depot.1 -c /tmp/bar.depot \
     -x ask=true \*
```

Install all products in remote depot **/tmp/sample.depot.1 on host swposix ,** use response files, run request scripts only when a response file is absent, create catalog **/tmp/bar.depot** and copy all response files to the new catalog:

```
swinstall -s swposix:/tmp/sample.depot.1 -c swposix:/tmp/bar.depot \
     -x ask=as_needed \*
```

Install all patches in the depot that correspond to currently installed software and are of the **critical** category:

```
swinstall -s /tmp/sample.depot.1 -x patch_match_target=true \
-x patch_filter=\"*.*, c=critical\"
```

The following example applies to HP-UX 10. only.*

To *linkinstall* the product TEST to the clients **clientA, clientB** from the server:

```
swinstall -l -r -s :OS_700 TEST @ clientA clientB
```

The following example applies to HP-UX 10. only.*

To *linkinstall* product TEST2 to your own "/" directory from an application server on "serve":

```
swinstall -l -s serve TEST2
```

The following example applies only to HP OpenView Software Distributor.

Install the C product to a set of PC end targets:

```
swinstall -s sw_serve cc @ pc_controller::PC1 pc_controller::PC2
```

To schedule the above installation to run at the indicated time:

```
swinstall -Q 12/01,11:00 -s sw_serve cc @ \
     pc_controller::PC1 pc_controller::PC2
```

swcopy

Invoke an interactive session of **swcopy**:

```
swcopy
```

Invoke an interactive session, using default depot at hostX as the source:

```
swcopy -i -s hostX
```

Copy all products from the cartridge tape **/dev/rmt/0m** to the default depot on the local host:

```
swcopy -s /dev/rmt/0m \*
```

Load the *software_selections* listed in the file **/tmp/load.products** using the default source/depot:

```
swcopy -f /tmp/load.products
```

The following example applies only to HP OpenView Software Distributor.

Copy the C and Pascal products to some local and remote depots:

```
swcopy -s sw_server cc pascal @ /var/spool/sw hostA:/tmp/sw hostB
```

LIMITATIONS

- The SD-UX versions of **swinstall** and **swcopy** do not support the installation and configuration of software products on remote targets. The TUI is supported only on SD-UX.

  The following PC information applies only to HP OpenView Software Distributor.

- When copying software to a PC controller, the **swcopy** command only supports a single PC depot (configured on the PC controller).
- For PC software installation, the **swinstall** command first copies software to the PC depot, where it is then accessed by the SD PC agent at each PC target. Options that apply to **swcopy** only apply when installing PC software to the PC controller.

FILES

$HOME/.swdefaults
: Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **$HOME/.swdefaults.hosts**.

$HOME/.sw/defaults.hosts
: Contains the user-specific default list of hosts to manage.

$HOME/.sw/defaults.patchfilters
: Contains the user-specific default list of patch filters.

$HOME/.sw/sessions/
: Contains session files automatically saved by the SD commands or explicitly saved by the user.

/usr/lib/sw/sys.defaults
: Contains the master list of current SD options with their default values.

/var/adm/sw/
: The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
: Contains the active system-wide default values for some or all SD options.

/var/adm/sw/defaults.hosts
: Contains the system-wide default list of hosts to manage.

/var/adm/sw/defaults.patchfilters
: Contains the system-wide default list of patch filters.

/var/adm/sw/getdate.templ
: Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
: The Installed Products Database (IPD), a catalog of all products installed on a system.

The following applies only to HP OpenView Software Distributor.

/var/adm/sw/queue/
: The directory which contains the information about all active and complete install jobs, copy jobs, and other jobs initiated by the SD commands.

/var/spool/sw/
The default location of a source and target software depot.

PC FILES

The following applies only to HP OpenView Software Distributor.

...\SD\DATA\
The directory which contains all of the configurable and non-configurable data for SD.

...\SD\DATA\DEPOT\
The default location of a source and target PC depot.

AUTHOR

swinstall and **swcopy** were developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, *Installing HP-UX 11.0 and Updating HP-UX 10.x to 11.0*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swjob, sd - display and monitor job information and create and remove jobs

SYNOPSIS

swjob [**-i**] [**-R**] [**-u**] [**-v**] [**-a** *attribute*] [**-C** *session_file*] [**-f** *jobid_file*] [**-S** *session_file*]
[**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*]
[*jobid(s)*] [**@** *target_selections*]

sd [*XToolkit Options*] [**-x** *option=value*] [**-X** *option_file*]

Remarks

- This command applies only to the HP OpenView Software Distributor product. It is not part of the SD-UX command set shipped with the HP-UX operating system.
- For a description of the SD objects, attributes and data formats, see the *sd(4)* man page by typing **man 4 sd**.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd**.

DESCRIPTION

The **swjob** command displays job information and removes jobs. It supports these features:

- Display the current install jobs, copy jobs, and other SD jobs initiated by the SD commands.
- Specify a specific job to list or remove.
- Display the command logfile for a specific job.
- Display the target logfile for a specific target.

The **sd** command invokes the Graphical User Interface (GUI) to create, monitor and remove job status and logs. It provides an interactive interface to the same functionality that **swjob** provides. In addition, it can be used to initiate new install, copy, and remove jobs.

Options

When no options or operands are specified, **swjob** lists the jobs that exist on the local host. These jobs may be pending, active, in the background or completed. The **swjob** command supports the following options:

XToolKit Options
The **sd** command supports a subset of the standard XToolkit options to control the appearance of the system GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X*(1) man page by typing **man X** for a definition of these options.

-i Runs the command in interactive mode (invokes the GUI.) This invocation, "swjob -i", is an alias for the command **sd**.

-R Applies to target lists as a shorthand for **@ *::***.

-u Causes **swjob** to remove the specified job(s).

-v Causes **swjob** to list all available attributes, one per line. The option applies only to the default list.

-a *attribute* Each job has its own set of attributes. These attributes include such things as job title, schedule date, or results. The **-a** option selects a specific attribute to display. You can specify multiple **-a** options to display multiple attributes. See also *sd*(4) for details on these attributes. This option applies only to the default list.

The logfiles summarizing a job or detailing target actions can be displayed using **-a log**, if **-a log** is specified and no other attribute is specified (i.e. no other attribute may be specified).

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *jobid_file* Read the list of *jobids* from *jobid_file* instead of (or in addition to) the command line.

S

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-S *session_file*
Execute **swjob** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

The **swjob** command supports two types of operands: *jobid* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "jobid at targets".

- jobid The **swjob** command supports the following syntax for each job id:

  jobid

- target selections The **swjob** command supports the following syntax for each target selection:

  [*host*][**:**][*directory*]

Additionally, the **swjob** command supports the syntax:

[*pc_controller*][**::**][*pc_target*]

- This syntax only applies to PCs.
- The PC controller is a fanout server.
- The PC target may be a PC machine, user, or group name.
- Valid targets for a PC controller can be listed using:

  swlist -l machine|user|group

PC targets can be further qualified for whether they refer to a PC machine, user, or group type with the following syntax:

name[**,t=***type*][**,k=***address*]

- The *type* must be specified when a name applies to more than one **machine**, **user**, or **group**. (The *address* is used internally for machines and is generally not needed on the command line.)
- The keyword ***** can be substituted for **pc_targets**, specifying an installation to all target machines:

  @ pc_controller::*

- The ***** cannot be used when retrieving logfiles using **-a log**.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

S

command **-X** *option_file*

The following section lists all of the keywords supported by the **swjob** command. If a default value exists, it is listed after the "=".

The policy options that apply to **swjob** are:

agent_timeout_minutes=10000
Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

force_job_removal=false
By default, the master job is removed from the controller only after a removal of the job information stored on each of its targets succeeds. If the job should be removed regardless of the success of the removal of job information from its targets, set this option to **true**.

one_liner={jobid operation state progress results title}
Defines the attributes which will be listed for each job when no **-a** option is specified. Each attribute included in the **one_liner** definition is separated by <tab> or <space>. Any attributes, except **log** may be included in the **one_liner** definition. If a particular attribute does not exist for an object, that attribute is silently ignored.

patch_one_liner=title patch_state
Specifies the attributes displayed for each object listed when the **-l patch** option is invoked and when no **-a** or **-v** option is specified. The default display attributes are **title** and **patch_state**.

poll_now=false
The status information displayed for a job is as recent as the last time the daemon polled remote targets for information (the **swinstall** option **job_polling_interval**). If the most recent status is wanted set this option to **true**.

remove_fanout_depot=true
When a job is removed the depot software associated with that job is automatically removed. If the software that is part of this job is the same software being used by another job, then be sure to not delete the software as part of the job removal. If the depot software should be retained, set this option to **false**.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the *sd(5)* man page by typing **man 5 sd** for more information.

S

rpc_timeout=5.
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

targets=
Defines the default *target_selections*. There is no supplied default. If there is more than one target selection, they must be separated by spaces.

verbose=0
Controls the verbosity of the output (stdout). A value of

0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

Session File

Each invocation of the **swjob** command defines a job display session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is automatically saved to the file **$HOME/.sw/sessions/swjob.last.** This file is overwritten by each invocation of **swjob**.

You can also save session information to a specific file by executing **swjob** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swjob**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swjob** take precedence over the values in the session file.

Environment Variables

SD programs are affected by external environment variables.

SD programs that execute control scripts set environment variables for use by the control scripts. **swjob** does not set environmental variables, but it uses them.

Environment variables that affect the SD commands:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Signals

The **swjob** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swjob** prints a message, sends a Remote Procedure Call (RPC) to the daemons to wrap up, and then exits.

Each agent will complete the list task before it wraps up.

OPERATION

Different views of the job information are available. The types of listings that can be selected are given below.

- Default Listing
- Target Listing
- Logfile Listing

Default Listing

If **swjob** is invoked with no options or operands, it lists all jobs that are on the local host. This listing contains one line for each job. The line includes the job tag attribute and all other attributes selected via the **one_liner** option.

Listing jobs on a remote controller is not supported. If a *jobid* is given, information for only that job is displayed.

Status Listing

If a *-R* or @ *target_specification* is given, the targets for that job and their status are displayed. By default the status information includes Type, State, Progress and Results.

Logfile Listing

One of the attributes "log" encompasses a variety of logfile types. The type of logfile returned when the **-a** *log attribute* is given depends on the operands given. The types of logfiles:

No target_selections	Show the controller logfile (default).

@ target Show the agent or PC controller logfile.

@ pc_controller::pc_target
Show the PC target logfile.

RETURN VALUES

The **swjob** command returns:

0 The job information was successfully listed or the job was successfully removed.
1 The list /remove operation failed for all *jobids*.
2 The list /remove operation failed for some *jobids*.

DIAGNOSTICS

The **swjob** command writes to stdout, stderr, and to the agent logfile.

Standard Output

All listings are printed to stdout.

Standard Error

The **swjob** command writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swjob** command does not log summary events. It logs events about each read task to the **swagent** logfile associated with each *target_selection*.

EXAMPLES

To list all of the jobs that exist on the local host:

```
swjob
```

To show the scheduled date for job hostA-0001:

```
swjob -a schedule hostA-0001
```

For job hostA-0001 list the targets and their status:

```
swjob -R hostA-0001
```
or
```
swjob hostA-0001 @ *
```

For job hostA-0001 give the status for pc_controller1 and all its PC targets

```
swjob hostA-0001  @  pc_controller1::*
```

For job hostA-0001 list the controller log:

```
swjob -a log hostA-0001
```

S

For job hostA-0001 list the log for pc_controller1:

```
swjob -a log hostA-0001 @  pc_controller1
```

For job hostA-0001 list the log for PC target pc1 on pc_controller1:

```
swjob -a log hostA-0001 @  pc_controller1::pc1
```

LIMITATIONS

The **swjob** command only runs on HP-UX. Any **PC controller** target selections apply only to PCs.

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/queue/
The directory which contains the information about all active and complete install jobs, copy jobs, and other jobs initiated by the SD commands.

AUTHOR

swjob was developed by the Hewlett-Packard Company.

SEE ALSO

HP OpenView Software Distributor Administrator's Guide, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swlist - display information about software products

SYNOPSIS

swlist [**-d**|**-r**] [**-i**] [**-R**] [**-v**] [**-a** *attribute*] [**-C** *session_file*] [**-f** *software_file*] [**-l** *level*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- **swlist** has an interactive user interface that you can invoke by typing **swlist -i**.
- SD-UX commands are included with the HP-UX Operating System and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only

DESCRIPTION

The **swlist** command displays information about software products installed at or available from the specified *target_selections*. It supports these features:

- Specify bundles, products, subproducts, and/or filesets to list.
- Display the files contained in each fileset.
- Display a table of contents from a software source.
- Specify the attributes to display for each software object.
- Display all attributes for bundles, products, subproducts, filesets and/or files.
- Display the full **software_spec** to be used with software selections.
- Display the **readme** file for products.
- Display the depots on a specified host.
- Create a list of products, subproducts, and/or filesets to use as input to the other commands.
- List the categories of available or applied patches.
- List applied patches and their state (applied or committed).

Previewing Product and OS Update Information

To preview information about new software in the depot, you can use **swlist** to view the **readme** file for each product, including OS update information contained in the SD (SW-DIST product) **readme**. For example, to display the latest OS update information:

```
swlist -d -a readme -l product SW-DIST @ hostA:/depot11
```

Options

When no options or operands are specified, **swlist** lists the software bundles (and products which are not part of a bundle) that are installed at the local host. **swlist** supports the following options:

-d List software available from a depot (instead of software installed on a root filesystem).

-i Invoke the **swlist** interactive user interface. The interactive interface lets you browse SD software objects. Invoking **swlist -i -d** lets you browse depot software.

-r List products installed on an alternate root filesystem (instead of software installed on **/**). Use of **-r** is optional.

-R Shorthand for *-l bundle -l product -l subproduct -l fileset*.

S

-v If no **-a** options are specified, then list all the attributes for an object, one attribute per line. The attributes are listed in the format:

keyword value

If one or more **-a** options are specified, then list the selected attributes in the above format.

-a *attribute*
Each object has its own set of attributes. These attributes include such things as revision, description, vendor information, size, and many others. The **-a** option selects a specific *attribute* to display. You can specify multiple **-a** options to display multiple attributes.

Note that the **tag** attribute (i.e. the identifier) is always displayed for product, subproduct, and fileset objects. The **path** attribute (i.e. the filename) is always displayed for file objects.

The full set of attributes for a given software object can be obtained using the **-v** option. See also the *sd(4)* man page for details on these attributes.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **/.sw/sessions/**. You can recall a session file with the **-S** option. (Note that session management does not apply to the **swlist** interactive user interface invoked by the **-i** option.)

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-l *level*
List all objects down to the specified *level*. Both the specified level(s) and the depth of the specified *software_selections* control the depth of the **swlist** output.

-s *source*
Specifies the software source to list. This is an alternate way to list a source depot. Sources can also be specified as target depots and listed using the **-d** option.

-S *session_file*
Execute **swlist** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option. (Note that session management does not apply to the **swlist** interactive user interface invoked by the **-i** option.)

-t *target_file*
Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file*
Read the session options and behaviors from *option_file*.

S

Operands

swlist supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swlist supports the following syntax for each *software_selection*:

bundle[**.** *product*[**.** *subproduct*][**.** *fileset*]][**,** *version*]

product[**.** *subproduct*][**.** *fileset*][**,** *version*]

The **version** component has the form:

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,l=***location*][**,fr** *<op> revision*]
[**,fa** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

Target Selections

swlist supports this syntax for each *target_selection*.

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

The following PC information applies only to HP OpenView Software Distributor

The command also supports the syntax:

[*pc_controller*]

S

This syntax only applies to the PC controllers and PC depots on PC controllers.

Valid targets for a PC controller can be listed using:

```
swlist -l machine|user|group
```

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swlist** commands. If a default value exists, it is listed after the "=".

The policy options that apply to **swlist** are:

agent_timeout_minutes=10000
Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

codeword=
Provides the "codeword" needed to unlock protected HP CD-ROM software.

Some HP software products are shipped on CD-ROM as "protected" products. That is, they cannot be installed or copied unless a "codeword" and "customer ID" are provided. The codeword is found on the CD-ROM certificate which you received from HP. You may use this default specification on the command line or the SD-UX interactive user interface to enter the codeword.

This default stores the codeword for future reference; it needs to be entered only once. If a new HP product is purchased and a previous codeword has already been entered for that CD-ROM, just enter the new codeword as usual and the codewords will be merged internally.

NOTE: For HP-UX B.10.10 and later systems, SD searches the **.codewords** file on the server that is providing protected software to other hosts. It looks for valid customer_id/codeword pairs. In doing so, SD eliminates the need to enter codewords and customer_ids on every host that is "pulling" the software.

To properly store the customer_id/codeword for a CD-ROM, run **swinstall -p** or **swcopy -p** on the host serving the CD-ROM. After the codeword has been stored, clients installing or copying software using that host and CD-ROM as a source will no longer require a codeword or customer_id.

customer_id=
This number, also printed on the Software Certificate, is used to "unlock" protected software and restrict its installation to a specific site or owner. It is entered using the **-x** *customer_id=* option or by using the interactive user interface. The *customer_id* can be used on any HP-UX 10.0X compatible HP9000 system.

distribution_target_directory=/var/spool/sw
Defines the default location of the target depot.

layout_version=1.0
Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd(5)* for more information.

level= Specify the *level* of the object to list.

The supported software levels are:

bundle	Show all objects down to the bundle level.
product	Show all objects down to the product level. Also use *-l bundle -l product* to show bundles.
subproduct	Show all objects down to the subproduct level.
fileset	Show all objects down to the fileset level. Also use *-l fileset -l subproduct* to show subproducts.

S

file	Show all objects down to the file level (i.e. depots, products, filesets, and files).
category	Show all categories of available patches.
patch	Show all applied patches.

The supported depot and root levels are:

depot	Show only the depot level (i.e. depots which exist at the specified target hosts).
root	List all alternate roots.
shroot	List all registered shared roots (HP-UX 10.X only).
prroot	List all registered private roots (HP-UX 10.X only).

The machine, user, and group levels apply only to HP OpenView Software Distributor PC targets:

machine	Show the *machines* known to a PC controller.
user	Show the *users* known to a PC controller.
group	Show the *groups* known to a PC controller.

one_liner=revision title

Defines the attributes which will be listed for each object when no **-a** or **-v** options are specified. Each attribute included in the **one_liner** definition is separated by <tab> or <space>. Any attributes may be included in the **one_liner** definition. If a particular attribute does not exist for an object, that attribute is silently ignored. For example, the **description** attribute is valid for products, subproducts, and filesets, but the **architecture** attribute is only valid for products.

patch_one_liner=title patch_state

Specifies the attributes displayed for each object listed when the **-l patch** option is invoked and when no **-a** or **-v** option is specified. The default display attributes are **title** and **patch_state**.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the **sd.5** man page by typing **man 5 sd** for more information.

rpc_timeout=5.

Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

select_local=true

If no *target_selections* are specified, select the default **target_directory** of the local host as the *target_selection* for the command.

software=

Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

software_view=all_bundles

Indicates the software view to be used as the default level for the software listing in the GUI. It can be set to **all_bundles**, **products**, or a bundle category tag (to indicate to show only bundles of that category). For HP OpenView Software Distributor the default value is **products**.

targets=

Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

verbose=0

Controls how attribute values are displayed. A value of

0 displays only the attribute value.

1 displays both the attribute keyword and value. (See the **-v** option above.)

Session File

Each invocation of **swlist** defines a task session. The command automatically saves options, source information, software selections, and target selections before the task actually commences. This lets you re-execute the command even if the session ends before the task is complete. You can also save session information from interactive or command-line sessions.

Session information is saved to the file **$HOME/.sw/sessions/swlist.last.** This file is overwritten by each invocation of the command. The file uses the same syntax as the defaults files.

From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu.

From a command-line session, you can save session information by executing the command with the **-C***session__file* option. You can specify an absolute path for a session file. If you do not specify a directory, the default location is **$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu.

To re-execute a session from a command-line, specify the session file as the argument for the **-S** option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command-line options and parameters take precedence over the values in the session file.

Environment Variables

The environment variable that affects the **swlist** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See *lang*(5) for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Signals

The **swlist** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swlist** prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

Each agent will complete the list task (if the execution phase has already started) before it wraps up.

OPERATION

The output from **swlist** follows this rule with all options: only the lowest level listed (product, subproduct, fileset or file) will be uncommented. Among other things, this allows the output from **swlist** to be used as input to other commands. The one exception is the list that contains files; file-level output is not accepted by other commands.

The types of listings that can be selected are given below. Some of these listings are not exclusive choices, but rather ways to view the objects while controlling the amount of output.

- Default Listing
- Software Listing
- Root Listing
- Depot Listing
- Multiple Targets Listing
- Verbose Listing
- Fanout Listing (*HP OpenView Software Distributor only*)

Default Listing

If **swlist** is invoked with no *software_selections* and no *target_selections*, a listing of all installed products on the local host is produced. This listing contains one line for each product. The line includes the product tag attributes and all other attributes selected via the **one_liner** option.

If *target_selections* (i.e. target hosts) are specified, this same format listing is produced for the installed software at each of the specified hosts.

Software Listing

A listing of software objects is controlled by the specified *software_selections*, and also by the **-l** option (**swlist.level=**). **swlist** lists the contents of each software object specified in the *software_selections*. For example, if you specify product selections, the subproducts and/or filesets contained immediately below each product will be listed. If you specify fileset selections, the files contained in each fileset will be listed.

The depth of objects listed is controlled with the **-l** option. This option can expand or restrict the depth in concert with the specified software selections. By default, the contents of a specified software selection are always listed (as described above). The **-l** option can defeat this listing by specifying a level equivalent to the level of objects in the *software_selections*. For example, if you want to list specific product selections but not their contents, use **-l product**. If you want to list specific fileset selections but not their contained files, use **-l fileset**. The *software_selection* options only apply if the level is bundle, product, subproduct, fileset, file, or patch.

Depot Listing

Another class of objects that **swlist** can display are software depots. For example, the user can list all registered depots on a given host. A combination of the **-l depot** option and *target_selections* operands can produce a variety of depot listings.

Multiple Targets Listing

Multiple *target_selections* (i.e. root filesystems, alternate roots, or depots) are listed sequentially: list all the requested objects and attributes from the first *target_selection*, followed by the second *target_selection*, etc.

Verbose Listing

The **-v** option causes a verbose listing to be generated. A verbose listing includes all attributes defined for an object. The **swlist** command prints the keyword and value for each attribute. The attributes are listed one per line. The user can post-process (filter) the output with *grep*(1), *awk*(1), and/or *sed*(1) to get the fields of interest.

The depot's attributes are displayed if **swlist** is called with the **-v** and **-l depot** options, and a specific depot *target_selection*.

Attributes for a particular software level (**product/subproduct/fileset/file**) are displayed based on the depth of the specified *software_selections*. For example, *swlist -v product1.fileset1* will give all fileset attributes for *fileset1*. If the **-v** option is used with the **-l option**, the different listing are:

- To display attributes for all products, use **swlist -v -l product**
- To display attributes for all products and subproducts, use **swlist -v -l subproduct**
- To display attributes for all products and filesets, use **swlist -v -l fileset**
- To display attributes for all products, filesets, and files, use **swlist -v -l file**

Fanout Listing

The following applies only to HP OpenView Software Distributor

The **swlist** command can also list the users, groups, or machines associated with a PC controller; i.e. the valid PC targets which are available for installation through the PC controller. To list these PC targets, use the **-l user**, **-l group**, or **-l machine** options.

RETURN VALUE

The **swlist** command returns:

0 The *software_selections* and/or *target_selections* were successfully listed.
1 The list operation failed on all *target_selections*.
2 The list operation failed on some *target_selections*.

DIAGNOSTICS

The **swlist** command writes to stdout, stderr, and to the agent logfile.

Standard Output

All listings are printed to stdout.

Standard Error

The **swlist** command writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swlist** command does not log summary events. It logs events about each read task to the **swagent** logfile associated with each *target_selection*.

You can use the **swlist** interactive interface (**swlist -i -d**) to view the **swaudit.log** file.

EXAMPLES

Run the **swlist** interactive interface:

```
swlist -i @ host1
```

Use interactive **swlist** to view a depot:

```
swlist -i -d @ /tmp/depot
```

List all of the products installed on the local host:

```
swlist
```

Generate a comprehensive listing that includes all filesets for the product NETWORKING:

```
swlist -v -l fileset NETWORKING
```

List all the attributes for the ARPA-RUN fileset:

```
swlist -v NETWORKING.ARPA.ARPA-RUN
```

List the C product installed on several remote hosts:

```
swlist cc @ hostA hostB hostC
```

List the FRAME product relocated to directory **/opt** on host1:

```
swlist FRAME,1=/opt @ host1
```

List all the versions of the FRAME product installed on the toolserver host:

```
swlist FRAME @ toolserver
```

List all products in a shared root (HP-UX 10.X only):

```
swlist -r @ /export/shared_roots/OS_700
```

List products in a client's private root (HP-UX 10.X only):

```
swlist -r @  /export/private_roots/client
```

List the contents of the local tape, **/dev/rmt/0m**:

```
swlist -d @ /dev/rmt/0m
```

or, alternatively:

```
swlist -s /dev/rmt/0m
```

List the tag and revision attributes for all products on the local tape **/dev/rmt/0m**:

```
swlist -d -a revision @ /dev/rmt/0m
```

or, alternatively:

```
swlist -a revision -s /dev/rmt/0m @
```

Display the README file for the FRAME product:

```
swlist -a readme FRAME
```

List the products stored in a remote depot:

```
swlist -d @ hostA:/depot
```

List all depots on a host:

```
swlist -l depot @ hostA
```

List the categories defined in the depot mounted at **/CD**.

```
swlist -d -l category @ /CD
```

Output:

```
critical_patch  1.0  Patches to fix system hangs or data corruption
S747_upgrade    2.0  Patches needed to upgrade to an S747
security_patch  2.0  Patches affecting system security
```

List a particular attribute of a category object identified by the tag **critical_patch**.

```
swlist -a description -l category critical_patch
```

Use the **swlist -l** option and **patch** level to display the values of a fileset's *applied_patches* attribute.

```
swlist -l patch BogusProduct
```

Output:

```
BogusProduct          1.0             This is a Bogus Product
BogusProduct.FakeFS   Fake fileset
PHZX-0004.FakeFS      Patch for defect X      superseded
PHZX-3452.FakeFS      Patch for defect Y      applied
```

Another example showing just the patch:

```
swlist -l patch PHZX-0004
```

Output:

```
PHZX-0004             1.0             Patch product
PHZX-0004.FakeFS      Patch for defect X      superseded
```

S

The next two examples apply only to HP OpenView Software Distributor

List all machines known to a PC controller:

```
swlist -l machine @ server
```

List all users known to a PC controller:

```
swlist -l user @ server
```

LIMITATIONS

The following applies only to HP OpenView Software Distributor

For PCs, the **swlist** command only applies to the PC controller and the PC depot on the PC controller.

FILES

`$HOME/.swdefaults`
Contains the user-specific default values for some or all SD options.

`$HOME/.sw/sessions/`
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options (with their default values).

`/var/adm/sw/`
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/host_object`
The file which stores the list of depots registered at the local host.

`/var/adm/sw/products/`
The Installed Products Database (IPD), a catalog of all products installed on a system.

`/var/spool/sw/`
The default location of a source and target software depot.

AUTHOR

`swlist` was developed by the Hewlett-Packard Company and Mark H. Colburn (see pax(1)).

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M) swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swmodify - modify software products in a target root or depot

SYNOPSIS

swmodify [**-d**|**-r**] [**-p**] [**-u**] [**-v**] [**-V**] [**-a** *attribute*=[*value*]] [**-C** *session_file*] [**-f** *software_file*] [**-P** *pathname_file*] [**-s** *product_specification_file*| [**-S** *session_file*] [**-x** *option*=*value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selection*]

Remarks

- SD-UX commands are included with the HP-UX operating system and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following applies only to HP OpenView Software Distributor.

DESCRIPTION

The **swmodify** command modifies the definitions of software objects installed into a primary or alternate root, or available from a software depot. It supports the following features:

- adding new objects - The user can add new bundles, products, subproducts, filesets, control files, and files to existing objects (which will contain them).
- deleting existing objects - The user can delete existing bundles, products, subproducts, filesets, control files, and files from the objects which contain them.
- modifying attribute values - The user can add an attribute, delete an attribute, or change the existing value of an attribute for any existing object. When adding a new object, the user can at the same time define attributes for it.
- committing software patches - The user can remove saved backup files, committing the software patch.

With the exception of control files, **swmodify** does not manipulate the actual files that make up a product (fileset). The command manipulates the catalog information which describes the files. However, **swmodify** can replace the contents of control files.

Common uses of **swmodify** include:

- adding file definitions to the existing list of file definitions in a fileset. Example: If a fileset's control scripts add new files to the installed file system, the scripts can call **swmodify** to "make a record" of those new files.
- changing the values of existing attributes. Example: If a product provides a more complex configuration process (beyond the SD configure script), that script can set the fileset's state to CONFIGURED upon successful execution.
- defining new objects. Example: to "import" the definition of an existing application that was not installed by SD, construct a simple PSF describing the product. Then invoke **swmodify** to load the definition of the existing application into the IPD.

S

Options

swmodify supports the following options:

-d Perform modifications on a depot (not on a primary or alternate root). The given *target_selection* must be a depot.

-p Preview a modify session without modifying anything within the *target_selection*.

-r Perform modifications on an alternate root (and not the primary root, /). The given *target_selection* must be an alternate root.)

-u If no *-a attribute=value* options are specified, then delete the given *software_selections* from within the given *target_selection*. This action deletes the definitions of the software objects from the depot catalog or installed products database.

If *-a attribute* options are specified, then delete these attribute definitions from the given *software_selections* (from within the given *target_selection*).

-v Turn on verbose output to stdout.

-V List all the **SD** layout_versions this command supports.

-a *attribute*[**=***value*]
Add, modify, or delete the *value* of the given *attribute*. If the **-u** option is specified, then delete the *attribute* from the given *software_selections* (or delete the *value* from the set of values currently defined for the *attribute*). Otherwise add/modify the *attribute* for each *software_selection* by setting it to the given *value*.

Multiple **-a** options can be specified. Each attribute modification will be applied to every *software_selection*.

The **-s** and **-a** options are mutually exclusive, the **-s** option cannot be specified when the **-a** option is specified.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-P *pathname_file*
Specify a file containing the pathnames of files being added to or deleted from the IPD instead of having to specify them individually on the command line.

-s *product_specification_file*
The source *Product Specification File* (PSF) describes the product, subproduct, fileset, and/or file definitions which will be added or modified by **swmodify**.

The **-s** and **-u** options are mutually exclusive, the **-s** option cannot be specified when the **-u** option is specified.

-S *session_file*
Execute **swmodify** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *options_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file*
Read the session options and behaviors from *options_file*.

Operands

The **swmodify** command supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swmodify supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,l=***location*][**,fr** *<op> revision*]
[**,fa** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.

- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

If a *product_specification_file* is specified, **swmodify** will select the *software_selections* from the full set defined within the PSF. The software selected from a PSF is then applied to the *target_selection*, with the selected software objects either added to it or modified within it. If a PSF is not specified, **swmodify** will select the *software_selections* from the software defined in the given (or default) *target_selection*.

Target Selection

The **swmodify** command supports the specification of a single, local *target_selection*, using the syntax:

[**@ /***directory*]

When operating on the primary root, no *target_selection* needs to be specified. (The target / is assumed.) When operating on a software depot, the *target_selection* specifies the path to that depot. If the **-d** option is specified and no *target_selection* is specified, the default **distribution_target_directory** is assumed (see below).

S

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following keywords are supported by **swmodify**. If a default value exists, it is listed after the "=". The commands that this option applies to are also specified. The policy options that apply to **swmodify** are:

control_files=
When adding or deleting control file objects, this option lists the tags of those control files. There is no supplied default. If there is more than one tag, they must be separated by white space and surrounded by quotes.

distribution_target_directory=/var/spool/sw
Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

files= When adding or deleting file objects, this option lists the pathnames of those file objects. There is no supplied default. If there is more than one pathname, they must be separated by white space.

layout_version=1.0
Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd*(5) for more information.

logdetail=false
The **logdetail** option controls the amount of detail written to the log file. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option.

logfile=/var/adm/sw/sw<modify>.log
Defines the default log file for **swmodify**.

loglevel=1
Controls the log level for the events logged to the **swmodify** logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. See **logdetail** for more information.
A value of
0 provides no information to the log files.
1 enables verbose logging to the log files.
2 enables very verbose logging to the log files.

patch_commit=false
Commits a patch by removing files saved for patch rollback. When set to **true**, you cannot roll back (remove) a patch unless you remove the associated base software that the patch modified.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

source_file=
Defines the default location of the source product specification file (PSF). The **host:path** syntax is not allowed, only a valid **path** can be specified. The **-s** option overrides this value.

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces. Targets are usually specified in a target input file, as operands on the command line, or in the GUI.

verbose=1
Controls the verbosity of a non-interactive command's output:

0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.
2 for **swmodify**, enables very verbose messaging to stdout.

Session File

Each invocation of the **swmodify** command defines a modify session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is automatically saved to the file **$HOME/.sw/sessions/swmodify.last.** This file is overwritten by each invocation of **swmodify**.

You can also save session information to a specific file by executing **swmodify** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swmodify**. See the *swpackage*(4) by typing **man 4 swpackage** for PSF syntax.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swmodify** take precedence over the values in the session file.

Environment Variables

The environment variable that affects **swmodify** is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Signals

The **swmodify** command ignores SIGHUP, SIGTERM, SIGUSR1, and SIGUSR2. The **swmodify** command catches SIGINT and SIGQUIT. If these signals are received, **swmodify** prints a message and then exits. During the actual database modifications, **swmodify** blocks these signals (to prevent any data base corruption). All other signals result in their default action being performed.

RETURN VALUES

The **swmodify** command returns:

0 The add, modify, or delete operation(s) were successfully performed on the given *software_selections*.
1 An error occurred during the session (e.g. bad syntax in the PSF, invalid *software_selection*, etc.) Review stderr or the logfile for details.

DIAGNOSTICS

The **swmodify** command writes to stdout, stderr, and to specific logfiles.

Standard Output

In verbose mode, the **swmodify** command writes messages for significant events. These include:
- a begin and end session message,
- selection, analysis, and execution task messages.

Standard Error

The **swmodify** command also writes messages for all WARNING and ERROR conditions to stderr.

Logfile

The **swmodify** command logs events to the command logfile and to the **swmodify** logfile associated with each *target_selection*.

Command Log

The **swmodify** command logs all messages to the the logfile **/var/adm/sw/swmodify.log**. (The user can specify a different logfile by modifying the **logfile** option.)

Target Log

When modifying installed software, **swmodify** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). When modifying available software (within a depot), **swmodify** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**).

EXAMPLES

Add additional files to an existing fileset:

```
swmodify -xfiles='/tmp/a /tmp/b /tmp/c'  PRODUCT.FILESET
```

Replace the definitions of existing files in an existing fileset (e.g. to update current values for the files' attributes):

```
chown root /tmp/a /tmp/b
swmodify -x files='/tmp/a /tmp/b'  PRODUCT.FILESET
```

Delete control files from a fileset in an existing depot:

```
swmodify -d -u -x control_files='checkinstall subscript' \
         PRODUCT.FILESET @  /var/spool/sw
```

Create a new fileset definition where the description is contained in the PSF file **new_fileset_definition**:

```
swmodify -s new_fileset_definition
```

Delete an obsolete fileset definition:

```
swmodify -u PRODUCT.FILESET
```

Commit a patch (remove files saved for patch rollback):

```
swmodify -x patch_commit=true PATCH
```

Create some new bundle definitions for products in an existing depot:

```
swmodify -d -s new_bundle_definitions  \*   @  /mfg/master_depot
```

Modify the values of some fileset's attributes:

```
swmodify -a state=installed  PRODUCT.FILESET
```

Modify the attributes of a depot:

```
swmodify -a title='Manufacturing's master depot' \
         -a description=</tmp/mfg.description @  /mfg/master_depot
```

WARNINGS

If the *target_selection* is a software depot and you delete file definitions from the given *software_selections*, the files' contents are not deleted from the depot.

LIMITATIONS

The following applies only to HP OpenView Software Distributor.

The **swmodify** command does not apply to PC software.

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory

is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/products/`
The Installed Products Database (IPD), a catalog of all products installed on a system.

`/var/spool/sw/`
The default location of a target software depot.

AUTHOR

`swmodify` was developed by the Hewlett-Packard Company.

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swpackage - package software products into a target depot or tape

SYNOPSIS

swpackage [**-p**] [**-v**] [**-V**] [**-C** *session_file*] [**-d** *directory* | *device*] [**-f** *software_file*] [**-s** *product_specification_file* | *directory*] [**-S** *session_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selection*]

Remarks

For a description of the Product Specification File (PSF) used as input to the **swpackage** command, see the *swpackage*(4) man page by typing **man 4 swpackage**.

DESCRIPTION

The **swpackage** command is not distributed; it only operates on the local host. It packages software products into:

- a distribution directory (which can be accessed directly or copied onto a CD-ROM),
- a distribution tape (such as DDS, nine-track or cartridge tapes).

A software *product* is organized into a three-level hierarchy: *products*, *subproducts*, and *filesets*. The actual files that make up a product are packaged into filesets. Subproducts can be used to partition or subset the filesets into logical groupings. (Subproducts are optional.) A product, subproduct, and fileset also have attributes associated with them.

Both directory and tape distributions use the same format. The **swpackage** command:

- Organizes the software to be packaged into products, subproducts, and filesets,
- Provides flexible mechanisms to package source files into filesets,
- Modifies existing products in a distribution directory,
- Repackages products in a distribution directory into a distribution tape.

Both the **swpackage** and **swcopy** commands create or modify a target depot. The differences between these commands are:

- The **swcopy** command copies products from an existing depot to another depot. The **swpackage** command creates products based on the user's specification, and packages these products into a depot.
- **swpackage** can be used to re-package *software_selections* from an existing distribution directory to a distribution tape.
- The **swcopy** command can copy from a local or remote source to a set of local or remote targets. The **swpackage** command packages source files from the local filesystem into a product, for insertion into a local distribution directory or tape.
- After creating a target depot, **swcopy** registers that directory with the local **swagentd** so that it can be found by **swlist**, **swinstall**, etc. With **swpackage**, the depot is not registered; the user must explicitly invoke the **swreg** command.

Layout Version

By default, SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older *layout version 0.8*, but you should use the older version only to create distributions readable by older versions of SD.

Which **layout_version** the SD commands write is controlled by the **layout_version** option or by specifying the **layout_version** attribute in the **PSF** file.

See *sd*(4), the description of the **layout_version** option in the following section and in *sd*(5) for more information. See *sd(4)* for more information on **PSF** files.

Options

swpackage supports the following options:

-p Previews a package session without actually creating or modifying the distribution tape.

S

-v Turns on verbose output to stdout. Verbose output is enabled by default, see the **verbose** option below.

-V List the SD data model revision(s) which **swpackage** supports. By default, **swpackage** always packages using the latest SDU data model revision.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-d *directory* | *device*
If creating a distribution directory, this option defines the pathname of the *directory*. If creating a distribution tape, this option defines the *device* file on which to write the distribution. When creating a distribution tape, the tape device (file) must exist, and the **-x media_type=tape** option must be specified (see below).

Note that the **-d** option is obsolete. Use the **@** *target_selection* operand instead.

You can also specify that the **swpackage** output be "piped" to an external command using:

```
swpackage -d "| <command>" -x media_type=tape -s <source>
```

The **|** symbol and command must be quoted because it is interpreted by **swpackage** and not the shell.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-s *product_specification_file* | *directory*
The source PSF describes the product, subproduct, fileset, and file definitions used to build a software product from a set of source files.

The source can also be an existing *directory* depot (which already contains products).

-S *session_file*
Execute **swpackage** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *options_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file*
Read the session options and behaviors from *options_file*.

S

Software Selections

The **swpackage** command supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,l=***location*][**,fr** *<op> revision*]
[**,fa** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.

- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed sofrware, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

If specified, the software selections cause **swpackage** to only (re)package those software selections from the full set defined in the source *product_specification_file*. If no *software_selections* are specified, then **swpackage** will (re)package all the products defined in the source *product_specification_file*.

Target Selections

The **swpackage** command supports the following syntax for a *target_selection*:

@ /*path*

If creating a distribution directory, this option defines the *path* to the directory. If creating a distribution tape, this option defines the *path* to the device file on which to write the distribution. When creating a distribution tape, the tape device (file) must exist, and the **-x media_type=tape** option must be specified (see below).

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by **swpackage** and **swcopy**. If a default value exists, it is listed after the "=". The commands that this option applies to are also specified.

compress_cmd=/usr/contrib/bin/gzip
: Defines the command called to compress files before installing, copying or packaging. If the **compression_type** option is set to other than **gzip** or **compress,** this path must be changed.

compress_files=false
: If set to **true**, files are compressed, if not already compressed, before transfer from a source. This will enhance performance on slower networks for **swcopy** and **swinstall**, and will result in smaller depots for **swcopy** and **swpackage**, unless the **uncompress_files** is also set to **true**.

S

`compression_type=gzip`

Defines the default compression type used by the agent when it compresses files during or after transmission. If **`uncompress_files`** is set to false, the **`compression_type`** is recorded for each file compressed so that the correct uncompression can later be applied during a **`swinstall`**, or a **`swcopy`** with **`uncompress_files`** set to true. The **`compress_cmd`** specified must produce files with the **`compression_type`** specified. The **`uncompress_cmd`** must be able to process files of the **`compression_type`** specified unless the format is **`gzip`**, which is uncompressed by the internal uncompressor (**`funzip`**).

`create_target_acls=true`

If creating a target depot, **`swpackage`** will create Access Control Lists (ACLs) for the depot (if it is new) and all products being packaged into it. If set to **`false`**, and if the user is the superuser, **`swpackage`** will not create ACLs. (The **`swpackage`** command never creates ACLs when software is packaged on to a distribution tape.)

`distribution_source_directory=/var/spool/sw`

Defines the default distribution directory to read as the source (when the **`source_type`** is *directory).* The **`-s`** option overrides this default.

`distribution_target_directory=/var/spool/sw`

Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

`distribution_target_serial=/dev/rmt/0m`

Defines the default location of the target tape device file. The *target_selection* operand overrides this default.

`enforce_dsa=true`

Prevents a command from proceeding past the analysis phase if the disk space required is beyond the available free space of the impacted file systems. If set to **`false`**, then the install, copy, or package operation will use the file systems' minfree space and may fail because it reaches the file system's absolute limit.

`follow_symlinks=false`

Do not follow symbolic links in the package source files, but include the symbolic links in the packaged products. A value of **`true`** for this keyword causes **`swpackage`** to follow symbolic links in the package source files and include the files they reference in the packaged products.

`include_file_revisions=false`

Do not include each source file's revision attribute in the products being packaged. Because this operation is time consuming, by default the revision attributes are not included. If set to **`true`**, **`swpackage`** will execute *what*(1) and possibly *ident*(1) (in that order) to try to determine a file's revision attribute.

S

`layout_version=1.0`

Specifies the POSIX **`layout_version`** to which the SD commands conform when writing distributions and **`swlist`** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **`layout_version=0.8`** only to create distributions readable by older versions of SD.

See the description of the **`layout_version`** option in *sd(5)* for more information.

`logdetail=false`

The **`logdetail`** option controls the amount of detail written to the log file. When set to **`true`**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **`loglevel`** option.

`logfile=/var/adm/sw/sw<package>.log`

Defines the default log file for the swpackage command.

`loglevel=1`

Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the

logdetail option. See **logdetail** for more information.
A value of
0 provides no information to the log files.
1 enables verbose logging to the log files.
2 enables very verbose logging to the log files.

media_capacity=1330
If creating a distribution tape, this keyword specifies the capacity of the tape in Mbytes. This option is required if the media is not a DDS tape or a disk file. Without this option, **swpackage** sets the size to 1330 Mbytes for tape and "free space up to minfree" on a disk file.

media_type=directory
Defines the type of distribution to create. The recognized types are **directory** and **tape**.

package_in_place=false
If set to **true**, **swpackage** does not put the files that make up a product in the target depot. Instead, **swpackage** inserts references to the original source files, saving disk space.

reinstall_files=true
Causes all the files in a fileset to always be re-installed, re-copied, or re-packaged, even when the file already exists at the target and is identical to the new file. If set to **false**, files that have the same *checksum* (see next option), size and time stamp will not be re-installed, re-copied, or re-packaged. This check enhances performance on slow networks or slow disks.

reinstall_files_use_cksum=true
This option affects the operation when the **reinstall_files** option is set to false. It causes the checksums of the new and old file to be computed and compared to determine if the new file should replace the old one. (The checksum is slower, but is a more robust way to check for files being equivalent.) If set to **false**, the checksums are not computed, and files are (not) reinstalled based only on their size and time stamp. For **swpackage**, the default value for this option is **false**.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

source_file=psf
Defines the default location of the source product specification file (PSF). The **host:path** syntax is not allowed, only a valid **path** can be specified. The **-s** option overrides this value.

source_type=directory
Defines the default source type: **cdrom**, **file**, **directory**, or **tape**. The source type derived from the **-s** option overrides this value.

targets=
Defines the default *target_selections*. There is no supplied default. If there is more than one target selection, they must be separated by spaces. Targets are usually specified in a target input file, as operands on the command line, or in the GUI.

uncompress_cmd=
Defines the command to uncompress files when installing, copying, or packaging. This command processes files which were stored on the media in a compressed format. If the **compression_type** of the file is **gzip** then the internal uncompression (**funzip**) is used instead of the external **uncompress_cmd**.

verbose=
Controls the verbosity of a non-interactive command's output:
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.
2 for **swpackage** and **swmodify**, enables very verbose messaging to stdout.

The **-v** option overrides this default if it is set to 0. Applies to all commands.

write_remote_files=false
Prevents the installation, copying, or packaging of files to a target which exists on a remote (NFS) filesystem. Also prevents the removal of files from a remote filesystem. All files destined for (or already on) a remote filesystem will be skipped.

If set to **true** and if the superuser has write permission on the remote filesystem, the remote files will not be skipped, but will be installed, copied, packaged, or removed.

Session File

Each invocation of the **swpackage** command defines a packaging session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swpackage.last**. This file is overwritten by each invocation of **swpackage**.

You can also save session information to a specific file by executing **swpackage** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swpackage**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swpackage** take precedence over the values in the session file.

Environment Variables

The environment variable that affects **swpackage** is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Signals

The **swpackage** command catches the signals SIGQUIT and SIGINT. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

Locking

SD commands use a common locking mechanism for reading and modifying both root directories and software depots. This mechanism allows multiple readers but only one writer on a root or depot.

The SD commands which modify software in an (alternate) root directory are restricted from simultaneous modification using *fcntl*(2) locking on the file

var/adm/sw/products/swlock

relative to the root directory (e.g. **/var/adm/sw/products/swlock**).

The SD commands which modify software in a depot are restricted from simultaneous modification using *fcntl*(2) locking on the file

`catalog/swlock`

relative to the depot directory (e.g. **`/var/spool/sw/catalog/swlock`**).

All commands set *fcntl(2)* read locks on roots and depots using the **`swlock`** file mentioned above. When a read lock is set, it prevents other SD commands from performing modifications (i.e. from setting write locks).

PRODUCT SPECIFICATION FILE

This section summarizes the *product_specification_file* (PSF) which drives the **`swpackage`** session. See *swpackage*(4) for a detailed description of a PSF's syntax and semantics.

A PSF is structured as follows:

```
[depot specification]
    [vendor specification]
    [category specification]
    [bundle specification]
    [product specification]
        [control script specification]
        [subproduct specification]
        [fileset specification]
            [control script specification]
            [file specification]
        [fileset specification]
        ...
    [product specification]
    ...
```

If errors encountered while parsing the PSF result in no valid product definitions, **`swpackage`** terminates. All errors are logged to both stderr and the logfile. In summary, the **`swpackage`** user can:

- Specify one or more products;
- For each product, specify one or more filesets.
- For each fileset, specify one or more files.
- (optional) Specify attributes for the target depot/tape;
- (optional) Specify one or more bundles, defining the bundle contents;
- (optional) Specify vendor information for products and bundles;
- (optional) Specify category information for products, bundles and patches.
- (optional) For each product, specify one or more subproducts, defining the subproduct contents;
- (optional) For each product or fileset, specify one or more control scripts.

RETURN VALUES

The **`swpackage`** command returns:

0 The products specified in the *product_specification_file* were successfully packaged into the target depot/tape.

1 An error occurred during the **`swpackage`** session (e.g. bad syntax in the *product_specification_file*.) Review stderr or the log file for details.

DIAGNOSTICS

The **`swpackage`** command writes to stdout, stderr, and to the logfile.

Standard Output

The **`swpackage`** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, packaging, and tape creation messages.

Standard Error

The **`swpackage`** command writes messages for all WARNING and ERROR conditions to stderr.

Logfile

The **`swpackage`** command logs detailed events to the log file **`/var/adm/sw/swpackage.log`**. The user can specify a different logfile by modifying the **`logfile`** option.

S

EXAMPLES

Package the products defined in the PSF *products* into the default target depot:

```
swpackage -s products
```

Preview the same operation (do not create the target depot), and generate very verbose output:

```
swpackage -p -vv -s products
```

Package the products into the target depot *no_files*, insert references to the source files instead of copying them into the depot:

```
swpackage -s products -x package_in_place=true @  no_files
```

Re-package a specific fileset:

```
swpackage -s products -x package_in_place=true product.fileset
@  no_files
```

Re-package the entire contents of the depot **/var/spool/sw** onto the tape at **/dev/rmt/0m**:

```
swpackage -s /var/spool/sw -x media_type=tape  @  /dev/rmt/0m
```

LIMITATIONS

The **swpackage** command does not apply to HP OpenView Software Distributor PC software. PC software is packaged on the PC controller using the PC console, then copied to a UNIX® depot for subsequent distribution.

FILES

/dev/rmt/0m
The default location of a source and target tape.

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/spool/sw/
The default location of a source and target software depot.

AUTHOR

swpackage was developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(4), swreg(1M), swremove(1M), swverify(1M).

NAME

swreg - register or unregister depots and roots

SYNOPSIS

swreg -l *level* [**-u**] [**-v**] [**-C** *session_file*] [**-f** *object_file*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*objects_to_(un)register*] [**@** *target_selections*]

Remarks

- SD-UX commands are included with the HP-UX operating system and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only."

DESCRIPTION

The **swreg** command controls the visibility of depots and roots to users who are performing software management tasks. It must be used to register depots created by **swpackage**.

By default, the **swcopy** command registers newly created depots. By default, the **swinstall** command registers newly created alternate roots (the root, "/", is not automatically registered). The **swremove** command unregisters a depot, or root, when or if the depot is empty. The user invokes **swreg** to explicitly (un)register a depot when the automatic behaviors of **swcopy**, **swinstall**, **swpackage**, and **swremove** do not suffice. For example:

- making a CD-ROM or other removable media available as a registered depot.
- registering a depot created directly by **swpackage**.
- unregistering a depot without removing it with **swremove**.

Options

The **swreg** command supports the following options:

-l *level* Specify the *level* of the object to register or unregister. Exactly one level must be specified. The supported levels are:

depot	The object to be registered is a depot.
root	The object to be registered is a root.
shroot	The object to register is a shared root (HP-UX 10.X only).
prroot	The object to register is a private root (HP-UX 10.X only).

-u Causes **swreg** to unregister the specified objects instead of registering them.

-v Turns on verbose output to stdout. (The **swreg** logfile is not affected by this option.) Verbose output is enabled by default, see the **verbose** option below.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *object_file* Read the list of depot or root objects to register or unregister from *object_file* instead of (or in addition to) the command line.

-S *session_file*
Execute **swreg** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of target *hosts* on which to register the depot or root objects from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

The **swreg** command supports the following syntax for each *object_to_register*:

path

Each operand specifies an object to be registered or unregistered.

The **swreg** command supports the following syntax for each *target_selection*:

[*host*]

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following list describes keywords supported by the **swreg** command. If a default value exists, it is listed after the "=".

distribution_target_directory=/var/spool/sw
Defines the location of the depot object to register if no objects are specified and the **-l** option is specified.

level= Defines the default level of objects to register or unregister. The valid levels are:

depot	Depots which exist at the specified target hosts.
root	All alternate roots.
shroot	All registered shared roots *(HP-UX 10.X only)*.
prroot	All registered private roots *(HP-UX 10.X only)*.

logfile=/var/adm/sw/swreg.log
This is the default command log file for the **swreg** command.

logdetail=false[true]
The **logdetail** option controls the amount of detail written to the log file. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option. See the *sd*(5) man page for additional information by typing **man 5 sd**.

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. (See also **logdetail**.) A value of

0	provides no information to the logfile.
1	enables verbose logging to the logfiles.
2	enables very verbose logging to the logfiles.

S

log_msgid=0
Controls whether numeric identification numbers are prepended to logfile messages produced by SD. A value of 0 (default) indicates no such identifiers are attached. Values of 1-4 indicate that identifiers are attached to messages:
1 applies to ERROR messages only
2 applies to ERROR and WARNING messages
3 applies to ERROR, WARNING, and NOTE messages
4 applies to ERROR, WARNING, NOTE, and certain other logfile messages.

objects_to_register=
Defines the default objects to register or unregister. There is no supplied default (see **distribution_target_directory** above). If there is more than one object, they must be separated by spaces.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

See the *sd*(5) man page by typing **man 5 sd** for details on specifying this option.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up or are not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

select_local=true
If no *target_selections* are specified, select the default **distribution_target_directory** of the local host as the *target_selection* for the command.

targets=
Defines the default *target* hosts on which to register or unregister the specified root or depot objects. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

verbose=1
Controls the verbosity of the **swreg** output (stdout). A value of
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

S

Session File

Each invocation of the **swreg** command defines a registration session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swreg.last**. This file is overwritten by each invocation of **swreg**.

You can also save session information to a specific file by executing **swreg** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swreg**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swreg** take precedence over the values in the session file.

Environment Variables

SD programs are affected by external environment variables.

SD programs that execute control scripts set environment variables for use by the control scripts.

In addition, **swinstall** sets environment variables for use when updating the HP-UX operating system and modifying the HP-UX configuration.

The environment variable that affects the **swreg** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man pages by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Signals

The **swreg** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swreg** prints a message, sends a Remote Procedure Call (RPC) to the daemons to wrap up, and then exits.

RETURN VALUES

The **swreg** command returns:

0 The *objects_to_register* were successfully (un)registered.
1 The register or unregister operation failed on all *target_selections*.
2 The register or unregister operation failed on some *target_selections*.

DIAGNOSTICS

The **swreg** command writes to stdout, stderr, and to the daemon logfile.

Standard Output

The **swreg** command writes messages for significant events. These include:

- a begin and end session message,
- selection and execution task messages for each *target_selection*.

Standard Error

The **swreg** command writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swreg** command logs summary events at the host where the command was invoked. It logs events about each (un)register operation to the **swagentd** logfile associated with each *target_selection*.

EXAMPLES

Create a new depot with **swpackage**, then register it with **swreg**:

```
swpackage -s psf -d /var/spool/sw
swreg -l depot  /var/spool/sw
```

The following example applies only to HP OpenView Software Distributor

Unregister the default depot at several hosts:

```
swreg -u -l depot /var/spool/sw @ hostA hostB hostC
```

Unregister a specific depot at the local host:

```
swreg -u -l depot /cdrom
```

LIMITATIONS

The SD-UX version of **swreg** does not support the registration or unregistration of software depots on remote targets.

For PCs (HP OpenView Software Distributor), the **swreg** command only operates on a single depot, configured at the PC controller.

FILES

`$HOME/.swdefaults`
Contains the user-specific default values for some or all SD options.

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options with their default values.

`/var/adm/sw/`
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/host_object`
The file which stores the list of depots registered at the local host.

PC FILES

The following applies only to HP OpenView Software Distributor

`...\SD\DATA\HOST_OBJ`
The file which stores the list PC depot registered with the PC Controller.

AUTHOR

`swreg` was developed by the Hewlett-Packard Company.

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swremove(1M), swverify(1M).

NAME

swremove - unconfigure and remove software products

SYNOPSIS

swremove [*XToolkit Options*] [**-d**|**-r**] [**-i**] [**-p**] [**-v**] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- **swremove** has an interactive user interface. You can invoke it by typing **swremove** or by including the **-i** option on the command line.
- SD-UX commands are included with the HP-UX Operating System and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The **swremove** command removes *software_selections* from *target_selections* (e.g. root file systems). When removing installed software, **swremove** also unconfigures the software before it is removed. The software is not unconfigured when removed from an alternate root directory since it was not configured during installation. When removing available software (within a depot), **swremove** also does not perform the unconfiguration task.

NOTE : Selecting a bundle for removal *does not always* remove all filesets in that bundle. If a particular fileset is required by another bundle, that fileset will not be removed. For example, if the bundles **Pascal** and **FORTRAN** both use the fileset *Debugger.Run* and you try to remove **FORTRAN**, the fileset *Debugger.Run* will not be removed because it is also used by the bundle **Pascal**. This prevents the removal of one bundle from inadvertently causing the removal of filesets needed by another bundle.

Removing Patches or Patch Rollback Files

To remove patch software, rollback files corresponding to the patch *must* be available for rollback. You must remove the base software modified by the patch. (Removing the base software also removes the patches associated with that software.)

To commit (make permanent) a patch, use the **swmodify** command's **patch_commit** option to remove the files saved for patch rollback, or use the **swinstall** command's *save_patch_files* option to not save them initially. See *swmodify*(1M) and *swinstall*(1M) for more information.

S

PC Software Removal

The following paragraph applies only to HP OpenView Software Distributor.

For PC software removal, the **swremove** command can remove *software_selections* from one or more target PC depots.

Control Scripts

When removing installed software, the **swremove** command executes several vendor-supplied scripts (if they exist) during the removal of the *software_selections*. The **swremove** command supports the following scripts:

checkremove
: a script executed during the analysis of each *target_selection*, it checks to make sure the removal can be attempted. If this check fails, the software product will not be removed.

preremove
: a script executed immediately before the software files are removed.

postremove
: a script executed immediately after the software files are removed.

unconfigure
: a script executed during the unconfiguration of each *target_selection*, it unconfigures the host for

the software (and the software for the host). The **preremove** and **postremove** scripts are not intended for unconfiguration tasks. They are to be used for simple file management needs such as restoring files moved during install. The **unconfigure** script allows the **swremove** command to unconfigure the hosts on which it has been running before removing the software specified.

Options

The **swremove** supports the following options:

XToolKit Options
The **swremove** command supports a subset of the standard X Toolkit options to control the appearance of the GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X*(1) manual page for a definition of these options.

-d Operate on a depot rather than installed software.

-r (optional) Operate on an alternate root rather than **/**. Unconfigure scripts are not run when removing software from an alternate root directory.

-i Runs **swremove** in interactive mode (invokes the Graphical User Interface). The **swremove** command also supports an interactive terminal user interface (TUI) in which screen navigation is done with the keyboard (no mouse).

-p Previews a remove task by running the session through the analysis phase only.

-v Turns on verbose output to stdout. (The **swremove** log file is not affected by this option.) Verbose output is controlled by the default **verbose=x**.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* *(HP OpenView Software Distributor only)*

Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-Q *date* *(HP OpenView Software Distributor only)*

Schedules the job for the specified date. The date format can be changed by modifying the file **/var/adm/sw/getdate.templ**.

-S *session_file*
Execute **swremove** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

swremove supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swremove supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:
[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,l=***location*][**,fr** <*op*> *revision*]
[**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches. Shell patterns are not allowed with these operators.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

S

Target Selections

swremove supports the following syntax for each *target_selection*:

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

The following PC information applies only to HP OpenView Software Distributor.

To remove software from a PC depot, the **swremove** command also supports the following syntax:

[*pc_controller*]

EXTERNAL INFLUENCES

Defaults File

In addition to the standard options, you can change **swremove** behavior and policy options by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by **swremove**. If a default value exists, it is listed after the "=".

The policy options that apply to **swremove** are:

agent_auto_exit=true
Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive user interface, or when **-p** (preview) is used. This enhances network reliability and performance. The default is **true** - the target agent will automatically exit when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000
Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

auto_kernel_build=true
Normally set to true. Specifies whether the removal of a kernel fileset should rebuild the kernel or not. If the kernel rebuild succeeds, the system automatically reboots. If set to false, the system continues to run the current kernel.

If the **auto_kernel_build** option is set to **true**, the **autoreboot** option must also be set to **true**. If the **auto_kernel_build** option is set to **false**, the value of the **autoreboot** option does not matter.

autoreboot=false
Prevents the removal of software requiring a reboot from the non-interactive interface. If set to **true**, then this software can be removed and the target system(s) will be automatically rebooted.

An interactive session always asks for confirmation before software requiring a reboot is removed.

If the **auto_kernel_build** option is set to **true**, the **autoreboot** option must also be set to **true**. If the **auto_kernel_build** option is set to **false**, the value of the **autoreboot** option does not matter.

autoremove_job=false
Applies only to the HP OpenView Software Distributor product. Controls automatic job removal. If the job is automatically removed, job information (job status or controller/agent log files) cannot be queried with **swjob**.

autoselect_dependents=false
Automatically selects all software that depends on the specified software. When set to **true**, and any software that other software depends on is selected for remove, **swremove** automatically selects that other software. If set to **false**, automatic selections are not made to resolve requisites.

autoselect_reference_bundles=true
If **true**, bundles that have the is_sticky attribute set to **true** will be automatically removed when the last of its contents is removed. If **false**, the sticky bundles will not be automatically removed.

controller_source=
Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location:

[*host*][**:**][*path*]

This option has no effect on which sources the target uses and is ignored when used with the Interactive User Interface.

distribution_target_directory=/var/spool/sw
Defines the default location of the target depot.

enforce_dependencies=true
Requires that all dependencies specified by the *software_selections* be resolved at the *target_selections*. For **swremove**, if a selected fileset has dependents (i.e. other software depends on the fileset) and they are not selected, do not remove the selected filesets. If set to **false**, dependencies will still be checked, but not enforced.

enforce_scripts=true
By default, if a fileset **checkremove** script fails (i.e. returns with exit code 1), that fileset will not be removed. If a product **checkremove** script fails, none of the filesets in that product will be removed . If set to **false**, the remove operation will proceed even when a check script fails.

force_single_target=false
This option applies only to the Interactive User Interface when no SD-OV license is in effect on a system that is a diskless server. It causes **swremove** to run in a single target mode, even though a diskless server normally causes **swremove** to run in multi-target mode.

job_title=

This option applies to HP OpenView Software Distributor only.

Specifies an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

log_msgid=0
Controls whether numeric identification numbers are prepended to log file messages produced by SD:

0 (default) No identifiers are attached to messages.
1 Applies to ERROR messages only.
2 Applies to ERROR and WARNING messages.
3 Applies to ERROR, WARNING, and NOTE messages.
4 Applies to ERROR, WARNING, NOTE, and certain other log file messages.

logdetail=false
Controls the amount of detail written to the log file. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option.

See the **loglevel** option and the *sd(5)* manual page for more information.

logfile=/var/adm/sw/swremove.log
This is the default command log file for the **swremove** command.

loglevel=1

Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option.

0 provides no information to the logfile.
1 enables verbose logging to the log files.
2 enables very verbose logging to the log files.

See the **logdetail** option and the *sd*(5) manual page for more information.

mount_all_filesystems=true

By default, the **swremove** command attempts to automatically mount all filesystems in the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files which may be on mounted filesystems are available to be removed.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

polling_interval=2

Defines the polling interval used by the Interactive UI of the controller. It specifies how often each target agent will be polled to obtain status information about the task being performed. When operating across wide-area networks, the polling interval can be increased to reduce network overhead.

remove_empty_depot=true

Controls whether a depot is removed once the last product/bundle has been removed. Useful to set to false if you want to retain existing depot ACLs for subsequent depot reuse.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

See the *sd*(5) manual page (type **man 5 sd**) for more information.

rpc_timeout=5

Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

software=

Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

software_view=products

Indicates the software view to be used by the Interactive UI of the controller. It can be set to **products**, **all_bundles**, or a bundle category tag to indicate to show only bundles of that category.

targets=

Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

target_shared_root=

This option applies to HP-UX 10. only.*

Defines the default location of the alternate root directory.

verbose=1

Controls the verbosity of the output (stdout). A value of

0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

write_remote_files=false

Prevents the removal of files from a remote (NFS) file system. When set to **false**, files on a remote file system are not removed.

If set to **true** and if the superuser has write permission on the remote file system, the remote files are removed.

Session File

Each invocation of **swremove** defines a task session. The command automatically saves options, source information, software selections, and target selections before the task actually commences. This lets you re-execute the command even if the session ends before the task is complete. You can also save session information from interactive or command-line sessions.

Session information is saved to the file **$HOME/.sw/sessions/swremove.last**. This file is overwritten by each invocation of the command. The file uses the same syntax as the defaults files.

From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu.

From a command-line session, you can save session information by executing the command with the **-C** *session__file* option. You can specify an absolute path for a session file. If you do not specify a directory, the default location is **$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu.

To re-execute a session from a command-line, specify the session file as the argument for the **-S** option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command-line options and parameters take precedence over the values in the session file.

Software and Target Lists

The **swremove** command supports software and target selection from separate input files.

You can specify software and target selection lists with the **-f** and **-t** options. Software and targets specified in these files are selected for operation instead of (or in addition to) files listed in the command line. (See the **-f** and **-t** options for more information.)

S

Additionally, the **swremove** interactive user interface reads a default list of hosts on which to operate. The list is stored in:

/var/adm/sw/defaults.hosts	the system-wide default list of hosts
$HOME/.swdefaults.hosts	the user-specific default list of hosts

For each interactive command, target hosts containing roots or depots are specified in separate lists (**hosts** and **hosts_with_depots** respectively.) The list of hosts are enclosed in { } braces and separated by white space (blank, tab and newline). For example:

```
swremove.hosts={hostA hostB hostC hostD hostE hostF}
swremove.hosts_with_depots={hostS}
```

Environment Variables

The environment variable that affects the **swremove** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Environment variables that affect scripts are:

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to SW_LOCATION to locate the product's installed files. The configure script is only run when SW_ROOT_DIRECTORY is "/".

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a *request* script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Additional environment variables that affect scripts for **swremove** are:

PRE_UNIX95
This variable and the **UNIX95** variable are exported with a value that forces "classic" behavior of **swremove** instead of **UNIX95** behavior. For HP-UX 10.30 and later versions, this variable is set to "1".

SW_SESSION_IS_KERNEL
Indicates whether a kernel build is scheduled for the current install/remove session. A **TRUE** value indicates that the selected kernel fileset is scheduled for a kernel build and that changes to **/stand/system** are required. A null value indicates that a kernel build is not scheduled and that changes to **/stand/system** are not required.

The value of this variable is always equal to the value of **SW_SESSION_IS_REBOOT**.

SW_SESSION_IS_REBOOT
Indicates whether a reboot is scheduled for a fileset selected for removal. Because all HP-UX kernel filesets are also reboot filesets, the value of this variables is always equal to the value of **SW_SESSION_IS_KERNEL**.

UNIX95 This variable, along with the **PRE_U95** variable, is exported with a value that forces "classic" behavior of **swremove** instead of **UNIX95** behavior. For the 10.30 or later release of HP-UX, this variable is cleared.

Signals

The **swremove** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swremove** prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

Each agent will complete the removal task (if the execution phase has already started) before it wraps up. This avoids leaving software in a corrupt state.

Terminal Support

For in-depth information about terminal support refer to:

- The *Managing HP-UX Software with SD-UX* manual
- Start the GUI or TUI, select the *Help* menu, then select the *Keyboard...* option to access the *Keyboard Reference Guide*.

RETURN VALUES

An interactive **swremove** session always returns 0. A non-interactive **swremove** session returns:

0 The *software_selections* were successfully removed.
1 The remove operation failed on all *target_selections*.
2 The remove operation failed on some *target_selections*.

DIAGNOSTICS

The **swremove** command writes to stdout, stderr, and to specific log files.

Standard Output

An interactive **swremove** session does not write to stdout. A non-interactive **swremove** session writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

An interactive **swremove** session does not write to stderr. A non-interactive **swremove** session writes messages for all WARNING and ERROR conditions to stderr.

Logging

Both interactive and non-interactive **swremove** sessions log summary events at the host where the command was invoked. They log detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log

A non-interactive **swremove** session logs all stdout and stderr messages to the the logfile **/var/adm/sw/swremove.log**. Similar messages are logged by an interactive **swremove** session. The user can specify a different logfile by modifying the **logfile** option.

Target Log

A **swagent** process performs the actual remove operation at each *target_selection*. When removing installed software, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). When removing available software (within a depot), the **swagent** logs messages to the file *swagent.log* beneath the depot directory (e.g. **/var/spool/sw**).

The following paragraph applies only to HP OpenView Software Distributor.

You can view command and target log files using the **sd** or **swjob** command.

EXAMPLES

Preview the remove of the C and Pascal products installed at the local host:

```
swremove -p cc pascal
```

The following example applies only to HP OpenView Software Distributor.

Remove the C and Pascal products from several remote hosts:

```
swremove cc pascal @ hostA hostB hostC
```

Remove a particular version of HP Omniback:

```
swremove Omniback,l/opt/Omniback_v2.0
```

Remove the entire contents of a local depot:

```
swremove -d * @ /var/spool/sw
```

LIMITATIONS

- The SD-UX version of **swremove** does not support the unconfiguration and removal of software products on remote targets.
- Only the HP-UX version of **swremove** provides a GUI.
- The **swremove** TUI is supported only on SD-UX.

  The following PC information applies only to HP OpenView Software Distributor.
- When removing software from a PC controller, **swremove** operates only on the available software stored in the PC depot (configured on the PC controller). Software installed on PC targets can be removed by packaging a remove action (using the PC console) and distributing that package to PC targets.

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **$HOME/.sw/defaults**.

$HOME/.sw/defaults.hosts
Contains the user-specific default list of hosts to manage.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of log files.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/defaults.hosts
Contains the system-wide default list of hosts to manage.

/var/adm/sw/getdate.templ
Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/spool/sw/
The default location of a target software depot.

PC FILES

The following applies only to HP OpenView Software Distributor.

...\SD\DATA\
The directory which contains all of the configurable and non-configurable data for SD.

`...\SD\DATA\DEPOT\`
The default location of a source and target PC depot.

AUTHOR

`swremove` was developed by the Hewlett-Packard Company.

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swverify(1M).

NAME

swverify - verify software products

SYNOPSIS

swverify [**-d**|**-r**] [**-v**] [**-C** *session_file*] [**-f** *software_file*] [**-J** *job*id*]* [**-Q** *date*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- SD-UX commands are included with the HP-UX operating system and manage software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

  The following information applies to HP OpenView Software Distributor only."

DESCRIPTION

The **swverify** command verifies the *software_selections* at one or more *target_selections* (e.g. root filesystems). When verifying installed software, **swverify** checks software states, dependency relationships, file existence and integrity, in addition to executing vendor-supplied verification scripts.

The **swverify** command also verifies *software_selections* at one or more target depots. For target depots, **swverify** performs all of the checks listed above, but does not execute verification scripts.

The following sentence applies only to HP OpenView Software Distributor.

For PC software verification, the **swverify** command also verifies *software_selections* at one or more target PC depots.

NOTE: swverify does *not* support operations on a tape depot.

The **swverify** command also supports these features:

- Verifies whether installed or configured software is compatible with the hosts on which that software is installed.
- Verifies that all dependencies (prerequisites, corequisites) are being met (for installed software) or can be met (for available software).
- Executes vendor-specific verify scripts if the software products are configured. The most important information that can be conveyed during verification is that information pertaining to the correctness of the product's configuration - and most of this information must be presented by the vendor in the verify script.
- Reports missing files, check all file attributes (ignoring volatile files). These attributes include permissions, file types, size, checksum, mtime, link source and major/minor attributes.

Options

swverify supports the following options:

-d Operate on a depot rather than installed software.

-r Operate on an alternate root rather than **/**. Verify scripts are not run when verifying software in an alternate root directory. Use of **-r** is optional.

-v Turns on verbose output to stdout. (The **swverify** logfile is not affected by this option.) Verbose output is enabled by default; see the **verbose** option below.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* (HP OpenView Software Distributor only) Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-Q *date* (HP OpenView Software Distributor only) Schedules the job for this date. The date's format can be changed by modifying the file **/var/adm/sw/getdate.templ**.

-S *session_file*
Execute **swverify** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *options_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *options_file*.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swverify supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

The **version** component has the form:

[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,l=***location*][**,fr** <*op*> *revision*]
[**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  ==, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

S

The `\*` software specification selects all products. It is not allowed when removing software from the root directory `/`.

Target Selections

The **swverify** command supports the following syntax for each *target_selection*. The : (colon) is required if both a host and directory are specified.

[*host*][**:**][**/***directory*]

The following PC information applies only to HP OpenView Software Distributor.

The **swverify** command also supports the syntax:

[*pc_controller*]

This syntax applies only to PC controllers. The **pc_controller** is a fanout server, and **swverify** will verify software in its PC depot.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swverify** command. If a default value exists, it is listed after the "=". The commands that this option applies to are also specified.

agent_auto_exit=true

Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when **-p** (preview) is used. This enhances network reliability and performance. The default is **true** means the target agent automatically exits when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

allow_incompatible=false

Requires that the software products which are being installed be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

allow_multiple_versions=false

Prevents the installation or configuration of another, independent version of a product when a version already is installed or configured at the target.

If set to **true**, another version of an existing product can be installed into a new location, or can be configured in its new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

autoremove_job=false

This option applies only to HP OpenView Software Distributor.

Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or target log files) cannot be queried with **swjob**.

Install jobs to PCs can not be automatically removed. They should not be removed until the job completes on all PC targets.

autoselect_dependencies=true
Controls the automatic selection of prerequisite and corequisite software that is not explicitly selected by the user. When set to **true**, the requisite software is automatically selected for configuration. When set to **false**, requisite software which is not explicitly selected is not automatically selected for configuration.

check_contents=true
Causes **swverify** to verify the time stamp, size, and checksum attributes of files. If set to **false**, these attributes are not verified.

check_permissions=true
Causes **swverify** to verify the mode, owner, UID, group, and GID attributes of installed files. If set to **false**, these attributes are not verified.

check_requisites=true
Causes **swverify** to verify that the prerequisite and corequisite dependencies of the software selections are being met. If set to **false**, these checks are not performed.

check_scripts=true
Causes **swverify** to run the fileset/product verify scripts for installed software. If set to **false**, these scripts are not executed.

check_volatile=false
Causes **swverify** to not verify those files marked as volatile (i.e. can be changed). If set to **true**, volatile files are also checked (for installed software).

controller_source=
Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location:

[*host*][**:**][*path*]

This option has no effect on which sources the target uses.

distribution_target_directory=/var/spool/sw
Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

S

enforce_dependencies=true
Requires that all dependencies specified by the *software_selections* be resolved either in the specified source, or at the *target_selections* themselves.

If set to **false**, dependencies will still be checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite dependencies, if not enforced, may cause the installation or configuration to fail.

job_title=

This option applies only to HP OpenView Software Distributor.

This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

logdetail=false[true]
Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

logfile=/var/adm/sw/sw<command>.log
Defines the default log file for each SD command. (The agent log files are always located

relative to the target depot or target root, e.g. **/var/spool/sw/swagent.log** and **/var/adm/sw/swagent.log**.)

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. See **logdetail**, above, and the *sd*(5) manual page (by typing **man 5 sd**) for more information. A value of
0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging to the logfiles.

log_msgid=0
Controls the log level for the events logged to the command log file, the target agent log file, and the source agent log file by prepending identification numbers to log file messages:
0 No such identifiers are prepended (default).
1 Applies to ERROR messages only.
2 Applies to ERROR and WARNING messages.
3 Applies to ERROR, WARNING, and NOTE messages.
4 Applies to ERROR, WARNING, NOTE, and certain other log file messages.

mount_all_filesystems=true
By default, the SD commands attempt to mount all filesystems in the */etc/fstab* file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point, and that the expected files are available for a remove or verify operation.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the *sd*(5) man page by typing **man 5 sd** for more information.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

select_local=true
If no *target_selections* are specified, select the default **target_directory** of the local host as the *target_selection* for the command.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces. Targets can be specified in a target input file or as operands on the command line.

verbose=1
Controls the verbosity of a non-interactive command's output:
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.
2 for **swpackage** and **swmodify**, enables very verbose messaging to stdout.

The **-v** option overrides this default if it is set to 0.

Session File

Each invocation of the **swverify** command defines a verify session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swverify.last**. This file is overwritten by each invocation of **swverify**.

You can also save session information to a specific file by executing **swverify** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swverify**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swverify** take precedence over the values in the session file.

Environment Variables

SD programs that execute control scripts set environment variables for use by the control scripts.

The environment variable that affects the **swverify** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

Environment variables that affect scripts:

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available to for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to SW_LOCATION to locate the product's installed files. The configure script is only run when SW_ROOT_DIRECTORY is "/".

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a *request* script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Signals
The **swverify** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swverify** prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

RETURN VALUES
The **swverify** command returns:

0 The *software_selections* were successfully verified.
1 The verify operation failed on all *target_selections*.
2 The verify operation failed on some *target_selections*.

DIAGNOSTICS
The **swverify** command writes to stdout, stderr, and to specific logfiles.

Standard Output
The **swverify** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error
The **swverify** command also writes messages for all WARNING and ERROR conditions to stderr.

Logging
The **swverify** command logs summary events at the host where the command was invoked. It logs detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log
The **swverify** command logs all stdout and stderr messages to the the logfile **/var/adm/sw/swverify.log**. (The user can specify a different logfile by modifying the **log-file** option.)

Target Log
A **swagent** process performs the actual verify operation at each *target_selection*. When verifying installed software, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). When verifying available software (within a depot), the **swagent** logs messages to the file *swagent.log* beneath the depot directory (e.g. **/var/spool/sw**).

The following line applies only to HP OpenView Software Distributor.

Command and target log files can be viewed using the **swjob** command.

EXAMPLES
Verify the C and Pascal products installed at the local host:

```
swverify cc pascal
```

Verify a particular version of HP Omniback:

```
swverify Omniback,1=/opt/Omniback_v2.0
```

Verify the entire contents of a local depot:

```
swverify -d \* @ /var/spool/sw
```

The following example applies only to HP OpenView Software Distributor.

Verify the C and Pascal products on remote hosts:

```
swverify cc pascal @ hostA hostB hostC
```

LIMITATIONS

The SD-UX version of **`swverify`** does not support the verification of software products on remote targets.

The following PC information applies only to HP OpenView Software Distributor.

When verifying software at a PC controller, the **`swverify`** command operates only on the available software stored in the PC depot (configured on the PC controller). Software installed on PC targets can be verified by packaging verify actions (using the PC console), and distributing that package to PC targets.

FILES

`$HOME/.swdefaults`
Contains the user-specific default values for some or all SD options.

`$HOME/.sw/sessions/`
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options with their default values.

`/var/adm/sw/`
The directory which contains all the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/getdate.templ`
Contains the set of date/time templates used when scheduling jobs.

`/var/adm/sw/products/`
The Installed Products Database (IPD), a catalog of all products installed on a system.

`/var/spool/sw/`
The default location of a target software depot.

AUTHOR

`swverify` was developed by the Hewlett-Packard Company.

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, sd(4), sd(5), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swgettools(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swpackage(4), swreg(1M), swremove(1M).

NAME

sync - synchronize file systems

SYNOPSIS

sync [**-l**]

DESCRIPTION

sync executes the **sync()** system call (see *sync*(2)). If the system is to be stopped, the **sync** command must be called to ensure file system integrity.

sync flushes all previously unwritten system buffers including modified super blocks, modified inodes, and delayed block I/O out to disk. This ensures that all file modifications are properly saved before performing a critical operation such as a system shutdown. For additional protection from power failures or possible system crashes, use **syncer** to execute **sync** automatically at periodic intervals (see *syncer*(1M)).

AUTHOR

sync was developed by AT&T and HP.

SEE ALSO

syncer(1M), sync(2).

STANDARDS CONFORMANCE

sync: SVID2, SVID3

NAME

syncer - periodically sync for file system integrity

SYNOPSIS

/usr/sbin/syncer [*seconds*] [**-s**] [**-d** *directory* ...]

DESCRIPTION

syncer is a program that periodically executes **sync()** at an interval determined by the input argument *seconds* (see *sync*(2)). If *seconds* is not specified, the default interval is every 30 seconds. This ensures that the file system is fairly up-to-date in case of a crash. This command should not be executed directly, but should be executed at system boot time via startup script **/sbin/init.d/syncer**.

syncer also updates the **/etc/mnttab** file if it does not match current kernel mount information.

Options

syncer recognizes the following options:

-s Cause **syncer** to not update the **/etc/mnttab** file. Use of this option is provided for special cases of backward compatilibity only, and is strongly discouraged. This option may be removed in a future release.

-d Open directories for cache benefit. All directories must be specified by their full path name. If the **-d** option is not used, no directories are opened.

AUTHOR

syncer was developed by the University of California, Berkeley and HP.

SEE ALSO

brc(1M), init(1M), sync(1M), sync(2).

NAME

sysdef - display system definition

SYNOPSIS

/usr/sbin/sysdef [*kernel* [*master*]]

DESCRIPTION

The command **sysdef** analyzes the currently running system and reports on its tunable configuration parameters. *kernel* and *master* are not used, but can be specified for standards compliance.

For each configuration parameter, the following information is printed:

NAME	The name and description of the parameter.
VALUE	The current value of the parameter.
BOOT	The value of the parameter at boot time, if different from the current value.
MIN	The minimum allowed value of the parameter, if any.
MAX	The maximum allowed value of the parameter, if any.
UNITS	Where appropriate, the units by which the parameter is measured.
FLAGS	Flags that further describe the parameter. The following flag is defined:

M Parameter can be modified without rebooting.

WARNINGS

Users of **sysdef** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

FILES

/usr/conf/master.d Directory containing master files

NAME

syslogd - log system messages

SYNOPSIS

/usr/sbin/syslogd [**-d**] [**-D**] [**-f** *configfile*] [**-m** *markinterval*]

DESCRIPTION

The **syslogd** command reads and logs messages into a set of files described by the configuration file **/etc/syslog.conf**.

Options

syslogd recognizes the following options:

-d	Turn on debugging.
-D	Prevent the kernel from directly printing its messages on the system console. In this case, **syslogd** is responsible for routing all kernel messages to their proper destination.
-f *configfile*	Use *configfile* instead of **/etc/syslog.conf**.
-m *markinterval*	Wait *markinterval* minutes between mark messages, instead of 20 minutes.

syslogd creates the file **/var/run/syslog.pid**, if possible, containing a single line with its process ID. This can be used to kill or reconfigure **syslogd**.

To kill **syslogd**, send it a terminate signal:

```
kill ‘cat /var/run/syslog.pid‘
```

To make **syslogd**, re-read its configuration file, send it a **HANGUP** signal:

```
kill -HUP ‘cat /var/run/syslog.pid‘
```

syslogd collects messages from the UNIX domain socket **/dev/log.un**, an Internet domain socket specified in **/etc/services**, the named pipe **/dev/log**, and from the kernel log device **/dev/klog**. By default, local programs calling **syslog()** send log messages to the UNIX domain socket (see *syslog*(3C)). If UNIX domain sockets are not configured on the system, they write to the named pipe instead. If INET domain sockets are not configured, **syslogd** does not receive messages forwarded from other hosts, nor does it forward messages (see below).

Each message is one line. A message can contain a priority code, marked by a number in angle braces at the beginning of the line. Priorities are defined in the header file **<syslog.h>**.

syslogd configures itself when it starts up and whenever it receives a hangup signal. Lines in the configuration file consist of a **selector** to determine the message priorities to which the line applies and an **action**. The *action* field is separated from the selector by one or more tabs.

S

Selectors are semicolon separated lists of priority specifiers. Each priority has a **facility** indicating the subsystem that generated the message, a dot, and a **level** indicating the severity of the message. Symbolic names can be used. An asterisk selects all facilities. All messages of the specified level or higher (greater severity) are selected. More than one facility can be selected, using commas to separate them. For example:

```
*.emerg;mail,daemon.crit
```

selects all facilities at the **emerg** level and the **mail** and **daemon** facilities at the **crit** level.

The known facilities and levels recognized by **syslogd** are those listed in *syslog*(3C) converted to lowercase without the leading **LOG_**. The additional facility **mark** has a message at priority **LOG_INFO** sent to it every 20 minutes (this can be changed with the **-m** flag). The **mark** facility is not enabled by a facility field containing an asterisk. The level **none** can be used to disable a particular facility. For example,

```
*.debug;mail.none
```

selects all messages except **mail** messages.

The second part of each line describes where the message is to be logged if this line is selected. There are four forms:

- A file name (beginning with a leading slash). The file is opened in append mode. If the file does not exist, it is created.

- A host name preceded by an @ character. Selected messages are forwarded to the **syslogd** on the named host.
- A comma-separated list of users. Selected messages are written to those users' terminals if they are logged in.
- An asterisk. Selected messages are written to the terminals of all logged-in users.

Blank lines and lines beginning with a # character are ignored.

For example, the configuration file:

```
kern,mark.debug   /dev/console
mail.debug        /var/adm/syslog/mail.log
*.info;mail.none  /var/adm/syslog/syslog.log
*.alert           /dev/console
*.alert           root,eric,kridle
*.emerg           *
*.emerg           @admin
```

logs all kernel messages and 20 minute marks onto the system console, all mail system messages to **/var/adm/syslog/mail.log**, and all messages at **info** and above, except mail messages, to the file **/var/adm/syslog/syslog.log**. Messages at **alert** and above are logged to the console and to the users **root**, **eric**, and **kridle** if they are logged in. **emerg** messages are written to all logged-in users' terminals, and forwarded to the host **admin**.

Only a superuser can invoke **syslogd**.

WARNINGS

A configuration file selector selects all messages at the specified level *or higher*. The configuration lines:

```
user.debug         /tmp/logfile
user.info          /tmp/logfile
```

cause the logfile to get *two* copies of all **user** messages at level **info** and above.

Kernel panic messages are not sent to **syslogd.**

All HP-UX kernel messages are treated as if they had the **crit** priority level.

If **syslogd** is invoked with the **-D** option and **syslogd** terminates abnormally, kernel messages will not appear on the system console. In that case, reinvoke **syslogd** without the **-D** option to enable the kernel to send its messages to the system console.

DEPENDENCIES

Series 700

Kernel logging through the special log device **/dev/klog** is not supported.

The **-D** option is not supported.

AUTHOR

syslogd was developed by the University of California, Berkeley.

FILES

/dev/klog	The kernel log device
/dev/log	The named pipe on which **syslogd** reads log messages
/dev/log.un	The UNIX domain socket on which **syslogd** reads log messages
/etc/syslog.conf	Configuration file
/var/run/syslog.pid	Process ID

SEE ALSO

logger(1), syslog(3C).

NAME

talkd - remote user communication server

SYNOPSIS

talkd

DESCRIPTION

Talkd is the server that notifies a user that someone wants to initiate a conversation. It acts as a repository of invitations, responding to requests by clients wishing to initiate a conversation. To initiate a conversation, the client (the **talk** command) sends a message of type LOOK_UP to the server (see **/usr/include/protocols/talkd.h**). This causes the server to search its invitation table to check if an invitation currently exists for the caller to talk to the callee specified in the message. If the lookup fails, the caller then sends a message of type ANNOUNCE, which causes the server to display an announcement on the callee's terminal requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

To activate the talk service, the following entry must be added to the **/etc/inetd.conf** file:

```
ntalk   dgram   udp   wait   root   /usr/lbin/ntalkd ntalkd
```

SEE ALSO

talk(1), write(1).

NAME

telnetd - TELNET protocol server

SYNOPSIS

/usr/lbin/telnetd [**-b** [*bannerfile*]]

DESCRIPTION

The **telnetd** daemon executes a server that supports the DARPA standard TELNET virtual terminal protocol. The Internet daemon (**inetd**) executes **telnetd** when it receives a service request at the port listed in the services data base for **telnet** using the **tcp** protocol (see *inetd*(1M) and *services*(4)).

telnetd operates by allocating a Telnet pseudo-terminal device (see *tels*(7)) for a client, then creating a login process which has the slave side of the Telnet pseudo-terminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the Telnet pseudo-terminal, implementing the TELNET protocol, and passing characters between the client and login process.

NOTE: **telnetd** no longer uses *pty*(7) devices; instead it uses special devices created for TELNET sessions only. For a full description, see *tels*(7).

When a TELNET session is started up, **telnetd** sends TELNET options to the client side, indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal speed* and *terminal type* information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured as a normal terminal for login, with the exception of echoing characters (see *tty*(7)).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*.

telnetd is willing to have the remote client *do*: *binary*, *flow control*, *terminal speed*, *terminal type*, and *suppress go ahead*.

The flow control option permits applications running on a remote host to toggle the flow control on the local host. To toggle flow control for a **telnet** session programmatically, the application program must first call the **tcgetattr** function to get the current **termios** settings. For example,

```
tcgetattr(filedes, &termios_p)
```

Then, the **c_iflag** of the **termios** structure must have **IXON** set(reset) to enable(disable) flow control.

Finally, the **tcsetattr** function call can implement the change. For example,

```
tcsetattr(filedes, TCSANOW, &termios_p)
```

To toggle the flow control interactively, the user can issue a **stty** command using the input options **-ixon** to disable, or **ixon** to enable flow control. (see *stty*(1)).

The terminal speed option permits applications running on a remote host to obtain the terminal speed of the local host session using either *ioctl* or *stty*.

The **telnet** server also supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID) option, whereby users telneting to two or more consenting hosts may avoid going through a second login sequence.

To start **telnetd** from the Internet daemon, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd
```

To override the standard **telnetd** login banner, specify a file containing a custom banner with the **-b** *bannerfile* option. For example, to use **/etc/issue** as the login banner, have **inetd** start **telnetd** with the following lines in **/etc/inetd.conf** (**** provides line continuation):

```
telnet stream tcp nowait root /usr/lbin/telnetd \
     telnetd -b/etc/issue
```

If *bannerfile* is not specified, **telnetd** does not print a login banner.

The system administrator can enable the TAC User ID option on servers designated as participating hosts by having **inetd** start **telnetd** with the **-t** option in **/etc/inetd.conf**:

```
telnet stream tcp nowait root /usr/lbin/telnetd  telnetd -t
```

In order for the TAC User ID option to work as specified, the system administrator must assign to all authorized users of the option the same login name and unique user ID (UUID) on every participating system to which they are allowed TAC User ID access. These same UUIDs should not be assigned to non-authorized users.

Users cannot use the feature on systems where their local and remote UUIDs differ, but they can always use the normal **telnet** login sequence. Also, there may be a potential security breach where a user with one UUID may be able to gain entry to participating systems and accounts where that UUID is assigned to someone else, unless the above restrictions are followed.

A typical configuration may consist of one or more secure front-end systems and a network of participating hosts. Users who have successfully logged onto the front-end system may **telnet** directly to any participating system without being prompted for another login.

telnet uses the same files as **rlogin** to verify participating systems and authorized users, **hosts.equiv** and **.rhosts**. (See *hosts.equiv*(4) and the *Managing Systems and Workgroups* manual for configuration details.)

DIAGNOSTICS

If any error is encountered by **telnetd** in establishing the connection, an error message is returned through the connection, after which the connection is closed and the server exits. Any errors generated by the login process or its descendents are passed through as ordinary data.

The following diagnostic messages are displayed by **telnetd**:

unable to allocate Telnet device

The server was unable to obtain a Telnet pseudo-terminal for use with the login process. Either all Telnet pseudo-terminals were in use or the **telm** driver has not been properly set up (see *tels*(7)).

Next step: Check the Telnet pseudo driver configuration of the host where **telnetd** is executing.

fork: No more processes

telnetd was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

/usr/bin/login: ...

The login program could not be started via **exec*****()** for the reason indicated (see *exec*(2)).

WARNINGS

The terminal type name received from the remote client is converted to lowercase.

telnetd never sends TELNET *go ahead* commands.

AUTHOR

telnetd was developed by the University of California, Berkeley.

SEE ALSO

login(1), rlogin(1), telnet(1), inetd(1M), inetsvcs_sec(1M), ioctl(2), hosts(4), inetd.conf(4), inetd.sec(4), services(4), tels(7), stty(1), tty(7).

DOD MIL_STD 1782.

RFC 854 for the TELNET protocol specification.

NAME

telnetd - TELNET protocol server

SYNOPSIS

/usr/lbin/telnetd [**-b** [*bannerfile*]] [**-A**] [**-a** *authmode*]

DESCRIPTION

The **telnetd** daemon executes a server that supports the DARPA standard TELNET virtual terminal protocol. The Internet daemon (**inetd**) executes **telnetd** when it receives a service request at the port listed in the services data base for **telnet** using the **tcp** protocol (see *inetd*(1M) and *services*(4)).

telnetd operates by allocating a Telnet pseudo-terminal device (see *tels*(7)) for a client, then creating a login process which has the slave side of the Telnet pseudo-terminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the Telnet pseudo-terminal, implementing the TELNET protocol, and passing characters between the client and login process.

NOTE: **telnetd** no longer uses *pty*(7) devices; instead it uses special devices created for TELNET sessions only. For a full description, see *tels*(7).

When a TELNET session is started up, **telnetd** sends TELNET options to the client side, indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal speed, terminal type,* and *authentication* information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The Telnet pseudo-terminal allocated to the client is configured as a normal terminal with the exception of echoing characters (see *tty*(7)).

> **telnetd** is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*.
>
> **telnetd** is willing to have the remote client *do*: *binary*, *flow control*, *terminal speed*, *terminal type*, *suppress go ahead*, and *authentication*.

The flow control option permits applications running on a remote host to toggle the flow control on the local host. To toggle flow control for a **telnet** session programmatically, the application program must first call the **tcgetattr** function to get the current **termios** settings. For example,

```
    tcgetattr(filedes, &termios_p)
```

Then, the **c_iflag** of the **termios** structure must have **IXON** set(reset) to enable(disable) flow control.

Finally, the **tcsetattr** function call can implement the change. For example,

```
    tcsetattr(filedes, TCSANOW, &termios_p)
```

To toggle the flow control interactively, the user can issue a **stty** command using the input options **-ixon** to disable, or **ixon** to enable flow control. (see *stty(1)).*

The terminal speed option permits applications running on a remote host to obtain the terminal speed of the local host session using either *ioctl* or *stty*.

By default, the **telnet** server provides remote execution facilities with authentication based on Kerberos V5. (See *sis*(5).)

The **telnet** server also supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID) option, whereby users telneting to two or more consenting hosts may avoid going through a second login sequence.

To start **telnetd** from the Internet daemon, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
    telnet stream tcp nowait root /usr/lbin/telnetd telnetd
```

To override the standard **telnetd** login banner, specify a file containing a custom banner with the **-b** *bannerfile* option. For example, to use **/etc/issue** as the login banner, have **inetd** start **telnetd** with the following lines in **/etc/inetd.conf** (\ provides line continuation):

```
        telnet stream tcp nowait root /usr/lbin/telnetd \
             telnetd -b/etc/issue
```

If *bannerfile* is not specified, **telnetd** does not print a login banner.

With this Kerberos version of **telnetd, inetd** can start **telnetd** with the following lines in **/etc/inetd.conf**:

t

```
telnet stream tcp nowait root /usr/lbin/telnetd  telnetd -A
```

or

```
telnet stream tcp nowait root /usr/lbin/telnetd  telnetd -a valid
```

The **-A** option is used to ensure that non-secure systems are denied access to the server. It overrides any value specified with the **-a** option except when *authmode* is **debug**. (See *sis*(5).)

The **-a** *authmode* option specifies what mode is to be used for Kerberos authentication. (See *sis*(5).) Values for *authmode* are:

debug	Activates authentication debugging.
valid	Default value. Only allows connections when the remote user can provide valid Kerberos authentication information and is authorized to access the specified account.
none	Authentication information is not required. If no or insufficient Kerberos authentication information is provided, the *login*(1) program provides the necessary user verification.

The system administrator can enable the TAC User ID option on servers designated as participating hosts by having **inetd** start **telnetd** with the **-t** option in **/etc/inetd.conf**:

```
telnet stream tcp nowait root /usr/lbin/telnetd  telnetd -t
```

In order for the TAC User ID option to work as specified, the system administrator must assign to all authorized users of the option the same login name and unique user ID (UUID) on every participating system to which they are allowed TAC User ID access. These same UUIDs should not be assigned to non-authorized users.

Users cannot use the feature on systems where their local and remote UUIDs differ, but they can always use the normal **telnet** login sequence. Also, there may be a potential security breach where a user with one UUID may be able to gain entry to participating systems and accounts where that UUID is assigned to someone else, unless the above restrictions are followed.

A typical configuration may consist of one or more secure front-end systems and a network of participating hosts. Users who have successfully logged onto the front-end system may **telnet** directly to any participating system without being prompted for another login.

telnet uses the same files as **rlogin** to verify participating systems and authorized users, **hosts.equiv** and **.rhosts**. (See *hosts.equiv*(4) and the *Managing Systems and Workgroups* manual for configuration details.)

DIAGNOSTICS

If any error is encountered by **telnetd** in establishing the connection, an error message is returned through the connection, after which the connection is closed and the server exits. Any errors generated by the login process or its descendents are passed through as ordinary data.

Diagnostic messages displayed by **telnetd** are displayed below. Kerberos specific errors are listed in *sis*(5).

t

unable to allocate Telnet device

The server was unable to obtain a Telnet pseudo-terminal for use with the login process. Either all Telnet pseudo-terminals were in use or the **telm** driver has not been properly set up (see *tels*(7)).

Next step: Check the **tels** and **telm** configuration of the host where **telnetd** is executing.

fork: No more processes

telnetd was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

/usr/bin/login: ...

The login program could not be started via **exec*()** for the reason indicated (see *exec*(2)).

WARNINGS

The terminal type name received from the remote client is converted to lowercase.

`telnetd` never sends TELNET *go ahead* commands.

AUTHOR

`telnetd` was developed by the University of California, Berkeley.

SEE ALSO

login(1), rlogin(1), stty(1), telnet(1), inetd(1M), inetsvcs_sec(1M), ioctl(2), hosts(4), inetd.conf(4), inetd.sec(4), services(4), sis(5), tels(7), tty(7).

Managing Systems and Workgroups.

DOD MIL_STD 1782.

RFC 854 for the TELNET protocol specification.

NAME

tftpd - trivial file transfer protocol server

SYNOPSIS

/usr/lbin/tftpd [**-R** *retran-seconds*] [**-T** *total-seconds*] [*path* ...]

DESCRIPTION

tftpd is a server that supports the Internet Trivial File Transfer Protocol (RFC783). The TFTP server operates at the port indicated in the **tftp** service description (see *services*(4)). The server is normally started by **inetd** using the **/etc/inetd.conf** file (see *inetd*(1M) and *inetd.conf*(4)).

The **-R** option specifies the per-packet retransmission timeout, in seconds. The default value is 5 seconds.

The **-T** option specifies the total retransmission timeout, in seconds. The default value is 25 seconds.

The *path* parameter has the following effects:

- **tftpd** operates in either of two modes or their combination. The first mode requires a defined home directory for the pseudo-user **tftp**, and looks for files relative to that path. The second mode requires one or more *paths* be specified on the command line, and allows access only to files whose paths match or begin with one of the command line specifications. The first mode is backward-compatible with previous releases of HP-UX and supports somewhat tighter security. The second mode is compatible with other vendors' implementations of **tftpd** and allows greater flexibility in accessing files.
- If no *path* is specified on the command line, **tftpd** requires an entry in the **/etc/passwd** database (see *passwd*(4)) for an account (pseudo-user) named **tftp**. The password field should be *****, the group membership should be **guest**, and the login shell should be **/usr/bin/false**. For example:

  ```
  tftp:*:510:guest:tftp server:/home/tftpdir:/usr/bin/false
  ```

  tftpd uses a call to **chroot()** to change its root directory to be the same as the home directory of the pseudo-user **tftp**. This restricts access by **tftp** clients to only those files found below the **tftp** home directory (see *chroot*(2)). Furthermore, **tftp** clients can only read files in that directory if they are readable by the pseudo-user **tftp**, and **tftp** clients can only write files in that directory if they exist and are writable by the pseudo-user **tftp**.
- If any *path* is specified on the command line, **tftpd** does not require that a pseudo-user named **tftp** exist in **/etc/passwd**. The specified *path*s control access to files by **tftp** clients. Each *path* is treated as being relative to **/** (not the **tftp** home directory), and can be either a directory or a file. **tftpd** disallows a client access to any file that does not match entirely or in its initial components one of the restriction *path*s. It also disallows access to any file path containing "**..**". However, an accessed file can be a symbolic link that points outside the set of restricted paths.
- If any *path* is specified on the command line and the **tftp** home directory is defined and is not **/**, **tftpd** first looks for a file relative to (under) the home directory. If the file is not found there, then **tftpd** looks for the file relative to **/** with path restrictions applied. Thus if two files with the same name can be found in both locations, **tftpd** accesses the one under **tftp**'s home directory.

Note that **inetd** allows continuation of command lines in **inetd.conf** by ending continued lines with a backlash.

Defining the **tftp** pseudo-user is strongly recommended even when *path*s are specified, because client access is further restricted to files that can be read and/or written by this pseudo-user. It is safe to set the **tftp** pseudo-user's home directory to **/** in this case.

DIAGNOSTICS

The following diagnostics are logged to the **syslogd** facility at the **err** log level (see *syslogd*(1M)).

No security mechanism exists

The pseudo-user **tftp** was not found in the password database (**/etc/passwd**), and **tftpd** was invoked without any *path* arguments.

Add or correct the entry for the pseudo-user **tftp** in the password database **/etc/passwd**. Or, add an access list (*path* arguments) to the **tftpd** arguments in the **inetd** configuration file **/etc/inetd.conf**. Reconfigure **inetd** with the command **inetd -c**.

`Unknown option` *option* **`ignored`**

An invalid option was specified in the **`tftpd`** arguments in the **`inetd`** configuration file **`/etc/inetd.conf`**.

Remove or correct the option. Restart **`inetd`** with the command **`inetd -c`**.

`Invalid total timeout` *value*

The value given for the **`-T`** option was not a number or was a negative number.

Correct the value given for the **`-T`** option. Reconfigure **`inetd`** with the command **`inetd -c`**.

`Invalid retransmission timeout` *value*

The value given for the **`-R`** option was not a number or was a negative number.

Correct the value given for the **`-R`** option. Reconfigure **`inetd`** with the command **`inetd -c`**.

system call`:`

The named *system call* failed. See the corresponding manual entry for a description of the system call. The reason for the failure is explained in the error message appended to the system call.

WARNINGS

When invoked with no *path* arguments, **`tftpd`** cannot follow symbolic links that refer to paths outside of the home directory of the pseudo-user **`tftp`**, because it performs a **`chroot()`**.

AUTHOR

`tftpd` was developed by the University of California, Berkeley, and Hewlett-Packard.

SEE ALSO

tftp(1), inetd(1M), syslogd(1M), chroot(2), inetd.conf(4), passwd(4).

NAME

tic - terminfo compiler

SYNOPSIS

tic [**-v** [*n*]] [**-c**] *file* ...

DESCRIPTION

tic translates terminfo files from source format into the compiled format. Results are placed in the directory **/usr/share/lib/terminfo**.

-vn Specifies that (verbose) output be written to standard error trace information showing tic's progress. The optional integer **n** is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If **n** is omitted, the default level is 1. If **n** is specified and greater than 1, the level of detail is increased.

-c Specifies to check only file for errors. Errors in **use=links** are not detected.

tic compiles all terminfo descriptions in the given files. When a **use=** field is discovered, **tic** searches first the current file, then the master file which is **./terminfo.src**.

If the environment variable **TERMINFO** is set, the results are placed in the location specified by **TERMINFO** instead of in **/usr/share/lib/terminfo**.

Limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

FILES

/usr/share/lib/terminfo/?/* compiled terminal capability data base

SEE ALSO

untic(1M), curses(3X), terminfo(4), captoinfo(1M).

BUGS

Instead of searching **./terminfo.src**, **tic** should check for an existing compiled entry.

STANDARDS CONFORMANCE

tic: SVID2, SVID3

NAME

tsm.lpadmin - add or remove a printer for use with *tsm*(1)

SYNOPSIS

/usr/tsm/bin/tsm.lpadmin -p *printer* **-m** *model*

/usr/tsm/bin/tsm.lpadmin -x *printer*

DESCRIPTION

tsm.lpadmin is used to add (or remove) a printer to the LP spooling system when the printer is connected to the system through a terminal running the Terminal Session Manager (see *tsm*(1)). **tsm.lpadmin** is a shell script that uses **lpadmin** in the normal way but also creates a named pipe to which LP output is directed (see *lpadmin*(1)). This named pipe is opened by TSM and data flowing from it is sent to the printer through the terminal.

Options

tsm.lpadmin recognizes the following options:

-p *printer*	Names a printer to be created with an associated pipe. If **-p** is used, **-m** must also be specified.
-m *model*	Selects a model interface program for *printer*. *model* is one of the model interface names supplied with the LP software (see the Models topic in the *lpadmin*(1)) manual entry. If **-m** is used, **-p** must also be specified.
-x *printer*	Removes *printer* from the LP system. No other options are allowed with **-x**.

Restrictions

To use **tsm.lpadmin** you must be user **lp** or **root**.

AUTHOR

tsm.lpadmin was developed by HP.

FILES

/var/spool/lp/tsm.pipes/*

SEE ALSO

lpadmin(1M), tsm(1).

NAME

ttsyncd - Daemon to maintain the **nis+** password table in sync with the **nis+** trusted table

SYNOPSIS

/usr/lbin/ttsyncd [**-D**] [**-v**] [**-t** *synchour*] [**-i** *interval*]

DESCRIPTION

ttsyncd checks that each login name in the **nis+** password (**passwd**) table appears in the **nis+** trusted table. It will create a user entry in the trusted table for every user that exists in the password table and NOT in the trusted table. Each **nis+** user can potentially log in to an HP trusted system; thus, **ttsyncd** aids trusted systems by creating an entry in the trusted table before the **nis+** user logs in. In turn, the system administrator can modify global security attributes for each **nis+** user before they log in to a trusted system that is part of the **nis+** namespace.

Options

ttsyncd recognizes the following options:

-D Used for debugging **ttsyncd**. All output will go to the screen as well as the syslog.

-v Verbose mode used for debugging **ttsyncd**.

-t Initial daily sync time. The time is in military hour (0 through 23). For example,

-t10	sync time is 10 a.m
-t23	sync time is 11 p.m
-t0	midnight
default	sync once next sync at midnight

-i Sync interval. This value is allowed from 0 through 23. This is only in hours. For example,

-i0	sync continuously (see note below)
-i4	sync every 4 hours
-i20	sync every 20 hours
default	sync 24 hours

DIAGNOSTICS

If it is desired to administer Trusted mode centrally within a **nis+** namespace, **ttsyncd** should be running on every HP **nis+** server. Furthermore, **ttsyncd** can be started automatically at boot time by setting TTSYNCD=1 in **/etc/rc.config.d/comsec**. Additionally, **ttsyncd** will run only on a **nis+** server (root or master) and only if **nis+** is up and running.

EXAMPLES

Command format

Initial sync time at midnight and sync every 2 hours.

```
/usr/lbin/ttsyncd -t0 -i2
```

Initial sync time at 10 a.m and sync every 2 hours.

```
/usr/lbin/ttsyncd -t10 -i2
```

Initial sync time at midnight and sync every 4 hours.

```
/usr/lbin/ttsyncd -i4
```

Sync only at midnight.

```
/usr/lbin/ttsyncd
```

Continuously sync starting at midnight.

```
/usr/lbin/ttsyncd -i0
```

NOTE

If interval **-i0** is given, **ttsyncd** will keep checking the passwd table for change. If there is any change in the passwd table then it will do a sync. This is useful when trusted table needs to be sync'ed as soon passwd table is modified. The drawback is it will consume CPU time for checking the passwd table.

DEPENDENCIES

NIS+ (Network Information Name Service)

ttsyncd requires NIS+ to be configured and running. It should be run only on an **nis+** server. Moreover, **ttsyncd** will self-terminate if the password table does not exist.

AUTHOR

ttsyncd was developed by Hewlett Packard.

FILES

/etc/rc.config.d/comsec
/etc/sbin/init.d/comsec

SEE ALSO

getprpwent(3).

NAME

tunefs - tune up an existing HFS file system

SYNOPSIS

/usr/sbin/tunefs [**-A**] [**-v**] [**-a** *maxcontig*] [**-d** *rotdelay*] [**-e** *maxbpg*] [**-m** *minfree*] *special-device*

DESCRIPTION

The **tunefs** command is used to alter dynamic parameters that affect HFS file system layout policies. Parameters to be altered are specified by the options and arguments provided on the command line as described below.

tunefs affects how the file system blocks are laid out on the disk. The default *rotdelay* value set by the **newfs** and **mkfs** commands (see *newfs*(1M) and *mkfs*(1M)) is 0 milliseconds, causing file system blocks to be written and read consecutively. In general, this should be the optimal tuning, making the use of **tunefs -d** unnecessary.

Options

tunefs recognizes the following options and command-line arguments:

-a *maxcontig* Set the maximum number of contiguous blocks that will be laid out before forcing a rotational delay to *maxcontig* (see **-d** below). The default value is **1**, because most device drivers require one interrupt per disk transfer. For device drivers that can chain several buffers together in a single transfer, set *maxcontig* to the maximum chain length.

-d *rotdelay* *rotdelay* is the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to determine how much rotational spacing to place between successive blocks in a file.

-e *maxbpg* *maxbpg* specifies the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one fourth of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause large files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

-m *minfree* *minfree* specifies the percentage of space that is not available to normal users; i.e., the minimum free space threshold. The default value used is 10%. This value can be set to zero. If set to zero, throughput performance drops to as little as one-third of the efficiency expected when the threshold is set at 10%. Note that if *minfree* is raised above the current usage level, users cannot allocate files until enough files have been deleted to meet the new threshold requirement.

-v (visual) Display current values contained in the primary super-block to standard output.

-A (all) Modify redundant super-blocks as well as the primary super-block as stipulated by the configuration options and arguments.

special-device is the name of the file system to be tuned. It is either a block or character special file if the file system is not mounted, or a block special file if the file system is mounted.

WARNINGS

Root file system tuning is normally done during initial system software installation. Tuning the root file system after installation has little useful effect because so many files have already been written.

You can tune a file system, but you can't tune a fish.

AUTHOR

tunefs was developed by the University of California, Berkeley.

SEE ALSO

dumpfs(1M), mkfs(1M), newfs(1M), fs(4).

NAME

udpublickey - update the publickey database file and the NIS map

SYNOPSIS

udpublickey

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

udpublickey is executed from the *updaters*(1M) makefile when either **newkey** or **rpc.ypupdated** updates the **/etc/publickey** database file.

udpublickey receives the following information from **newkey** or **rpc.ypupdated:**

Requestor's name (a string)

Type of update

Number of bytes in key

Key

Number of bytes in data

Data

After receiving this information, **udpublickey** attempts to update the publickey database file, **/etc/publickey**.

If the update is successful, **udpublickey** makes the NIS map, **publickey.byname**.

If the update is completely successful, **udpublickey** exits with a zero (0) status; otherwise **udpublickey** exits with a valid NIS error.

This command should not be run interactively.

AUTHOR

udpublickey was developed by Sun Microsystems, Inc.

FILES

/etc/publickey

SEE ALSO

newkey(1M), rpc.ypupdated(1M), updaters(1M), publickey(4).

NAME

untic - terminfo de-compiler

SYNOPSIS

untic [*term*] [**-f** *file*]

DESCRIPTION

untic translates a terminfo file from the compiled format into the source format. If the environment variable **TERMINFO** is set to a path name, **untic** checks for a compiled terminfo description of the terminal under the path specified by **TERMINFO** before checking **/usr/share/lib/terminfo**. Otherwise, only **/usr/share/lib/terminfo** is checked.

Normally **untic** uses the terminal type obtained from the **TERM** environment variable. With the *term* (terminal type) argument, however, the user can specify the terminal type used.

With the *file* argument the user can specify the file used for translation. This option bypasses the use of the **TERM** and **TERMINFO** environment variables.

untic sends the de-compiled terminfo description result to standard output.

AUTHOR

untic was developed by HP.

FILES

/usr/share/lib/terminfo/?/*	compiled terminal capability data base

SEE ALSO

tic(1M), curses(3X), terminfo(4).

NAME

updaters - configuration file for NIS updating

SYNOPSIS

updaters

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

DESCRIPTION

updaters is a makefile used for updating the Network Information Service (NIS) databases. Databases can be updated only if the network is secure, that is, only if there is a NIS **publickey** database (**publickey.byname**). The default **updaters** script will update only the **publickey.byname** map.

An entry in the file is a make target for a particular NIS database. For example, if you wanted to add **passwd.byname** to this script, you would create a make target named passwd.byname with the command to update that database. See *udpublickey*(1M).

The information necessary to make the update is passed to the update command through standard input. The information passed is described below. All items are followed by a NEW LINE except for Actual bytes of key and Actual bytes of data.

Network name of client wishing to make the update (a string)

Kind of update (an integer)

Number of bytes in key (an integer)

Actual bytes of key

Number of bytes in data (an integer)

Actual bytes of data

After receiving this information through standard input, the command to update the particular database should decide whether the user is allowed to make the requested change.

If not, the command should exit with the status **YPERR_ACCESS**.

If the user is allowed to make the change, the command should make the change and exit with a status of zero.

If there are any errors that may prevent the updater from making the change, it should exit with the status that matches a valid NIS error code described in **<rpcsvc/ypclnt.h>**.

AUTHOR

updaters was developed by Sun Microsystems, Inc.

SEE ALSO

make(1), newkey(1M), rpc.ypupdated(1M), udpublickey(1M), publickey(4).

NAME

ups_mond - HP PowerTrust Uninterruptible Power System monitor daemon

SYNOPSIS

/usr/lbin/ups_mond [**-f** *configfile*] [**-s**]

DESCRIPTION

When it detects a loss of AC power for a period of time exceeding a configured limit, **ups_mond** ensures file system integrity by shutting down HP-UX. To do this, **ups_mond** uses the device special files specified in its configuration file (**/etc/ups_conf** by default) to monitor the state of each HP PowerTrust Uninterruptible Power System (UPS) attached to the system.

Use the **-f** option to specify a configuration file other than **/etc/ups_conf**. See *ups_conf*(4) for a description of the configuration file format.

By default, **ups_mond** is locked into memory (see *plock*(2)). That is, **ups_mond** is not swappable. Although extreme caution is required, you can make **ups_mond** swappable if all swap disks are powered by an uninterruptible power system (assured to have power when the primary power source fails). To make **ups_mond** swappable, use the **-s** option.

ups_mond is started by **init** (see *init*(1M)) by means of an entry in the file **/etc/inittab** (see *inittab*(4)). The **inittab** entry uses the **respawn** option to automatically restart **ups_mond** if **ups_mond** is terminated by the **kill** command (see *kill*(1)). This entry should follow the entry:

```
sqnc::wait:/sbin/rc </dev/console >/dev/console 2>&1 # system initialization
```

so that **ups_mond** is started after the system logging daemon (**syslogd**). It should also be run with real-time priority to assure its execution (see *rtprio*(1)).

ups_mond logs messages, and when appropriate invokes **/usr/sbin/shutdown** using the **-h** option, or **/usr/sbin/reboot**. For each configured UPS, **ups_mond** can be instructed (in **/etc/ups_conf**) to log messages only, without taking **shutdown** or **reboot** action. See MSG_ONLY in *ups_conf*(4). By default **ups_mond** performs the **shutdown** and **reboot** actions.

Note that when the shutdown is performed, UPSs that have lost AC line voltage will be turned off once the *shutdown_timeout_mins* time has expired (see *upd_conf*(4)).

ups_mond uses the **syslog** message logging facility to log these occurrences (see *syslog*(3C)). Messages are written to the console if **ups_mond** is unable to send them to **syslogd**. Critical messages (see DIAGNOSTICS section) are also sent to the console.

RETURN VALUE

ups_mond returns the following values:

zero (0)	Successful Completion
non-zero	Error encountered. See ERRORS below.

EXAMPLES

The entry in **/etc/inittab** should be similar to this:

```
ups::respawn:rtprio 0 /usr/lbin/ups_mond -f /etc/ups_conf
```

DIAGNOSTICS

Messages resulting from normal operation:

UPS Monitor daemon starting; using configuration file <*configfilename***>.**

UPS <*tty special file name***> OK: AC Power back on.**

AC Power to all recognized, system critical UPS's OK! System will not shutdown.

Messages resulting in exit of daemon:

usage: ups_mond [-f configfile].

cannot exec /usr/lbin/ups_mond -f <*configfilename***> -e ups_monchild due to <***error***>.**

permission denied; must be superuser.

exiting; unable to lock process in memory: <*errno***>.**

aborted, configfile <*configfilename***> open received error: <***errno***>.**

aborted, configfile <*configfilename***> fseek error: <***errno***>.**

aborted, malloc error: <*errno***>.**

terminated by signal <*decimal value of signal***>.**

Messages for which **shutdown** might be run (depends on UPS configuration):

UPS <*tty special file name***> AC POWER FAILURE - running on UPS battery.**

If power is not returned within previously configured time period, your system will automatically go to graceful shutdown.

Messages for which **reboot** might be run (depends on UPS configuration):

UPS <*tty special file name***> battery low.**

UPS <*tty special file name***> no output - either switch setting wrong on UPS or bad UPS.**

UPS <*tty special file name***> failed - requires repair.**

UPS <*tty special file name***> current overload; UPS turned itself off - either UPS bad or too many devices connected.**

UPS <*tty special file name***> ambient temperature too high; UPS turned itself off - reduce heat in area.**

UPS <*tty special file name***> output voltage too high; UPS turned itself off - requires repair.**

UPS <*tty special file name***> output voltage too low; UPS turned itself off - requires repair.**

cannot exec shutdown due to <*errno***>.**

The above messages are followed by the following message:

reboot -halt invoked due to UPS error cited in previous syslog message.

Messages that are only logged (no **shutdown**/**reboot** action is taken):

warning - no upstty: UPS's found in configfile <*configfilename***>; daemon running for no purpose.**

warning - shutdown delay or shutdown timeout parameter in configfile <*configfilename***> missing or not greater than zero; using default.**

UPS <*tty special file name***> in bypass-mode; no AC Power-loss protection.**

UPS <*tty special file name***> interrupted, but read of ups status failed - possible UPS hardware problem.**

upstty <*tty special file name***> failed open: <***errno***>; ignoring that tty and continuing.**

UPS <*tty special file name***> ioctl(TCGETA) failed: <***errno***>; ignoring that UPS.**

UPS <*tty special file name***> ioctl(TCSETAF) failed <***errno***>; ignoring that UPS.**

UPS <*tty special file name***> line too noisy; ignoring that UPS.**

UPS <*tty special file name***> could not enable; loss of power would not be detectable.**

UPS <*tty special file name***> read failed: <***errno***>; Uninterruptible Power Supply has not been connected correctly; loss of power would not be detectable.**

UPS <*tty special file name***> write failed: <***errno***>; ignoring that UPS.**

UPS <*tty special file name***> read of status received ILLEGAL CMD or NOISY LINE.**

UPS <*tty special file name***> read of status received <***number***> bytes of unexpected data (octal: <***octal returned***>): <***string returned***>.**

UPS <*tty special file name***> read of status failed: <***errno***>.**

UPS <*tty special file name***> write failed: <***errno***>.**

UPS <*tty special file name***> turned-off Failure Alarm.**

UPS <*tty special file name***> turned-off Inverter Failure Alarm.**

UPS <*tty special file name***> turned-off No Battery Alarm.**

UPS <*tty special file name***> turned-off Battery Charger Fault Alarm.**

UPS <*tty special file name***> turned-off Current Overload Alarm.**

UPS <*tty special file name***> turned-off High Ambient Temperature Alarm.**

UPS <*tty special file name***> turned-off Battery Failure Alarm.**

UPS <*tty special file name***> turned-off High Battery Voltage Alarm.**

UPS <*tty special file name***> turned-off Low Battery Voltage Alarm.**

UPS <*tty special file name***> turned-off High Output Voltage Alarm.**

UPS <*tty special file name***> turned-off Low Output Voltage Alarm.**

UPS <*tty special file name***> Inverter Failure requires repair.**

UPS <*tty special file name***> No Battery - ensure UPS battery installed.**

UPS <*tty special file name***> Battery Charger Fault- requires repair.**

UPS <*tty special file name***> Current Overload - either UPS bad or too many devices connected.**

UPS <*tty special file name***> High Ambient Temperature- reduce area temperature.**

UPS <*tty special file name***> Battery Failure- requires repair.**

UPS <*tty special file name***> High Battery Voltage - requires repair.**

UPS <*tty special file name***> Low Battery Voltage - requires repair.**

UPS <*tty special file name***> UNKNOWN status/alarm <***hex number***> - may require repair.**

write to UPS <*tty special file name***> of command <***cmd string***> Failed: <***errno***>.**

read from UPS <*tty special file name***> after sending command <cmd string> to it failed; <***errno***>.**

UPS <*tty special file name***> could not execute command <***cmd string***>; returned (octal: <***octal returned***>): <***string returned***> - possible bad signal cable.**

Messages relating to Timer Controlled Power On and Off:

Timer Controlled On/Off information invalid; ignored.

mknod error: <*errno***> for Timed On/Off fifo file /var/tmp/timed_off; continuing without.**

open error: <*errno***> for Timed On/Off fifo file /var/tmp/timed_off; continuing without.**

Timer Controlled On value exceeds UPS <*tty special file name***> maximum. The maximum value of <***maximum supported decimal value***> will be used for this UPS.**

ERRORS

ups_mond returns the following error values:

EINVAL	**ups_mond** encountered an incorrect parameter.
EPERM	Insufficient privileges. **ups_mond** must be started by a superuser.

EINTR	**ups_mond** was interrupted (terminated) by **signal**() or **kill**(). See *signal*(2) and *kill*(1).
one (1)	For all other error conditions.

FILES

/dev/tty*
/etc/ups_conf
/var/tmp/timed_off
/var/adm/syslog/syslog.log

SEE ALSO

kill(1) init(1M) plock(2) signal(2) syslog(3C) inittab(4) ups_conf(4).

NAME

useradd - add a new user login to the system

SYNOPSIS

useradd [**-u** *uid* [**-o**]] [**-g** *group*] [**-G** *group* [**,** *group* ...]] [**-d** *dir*] [**-s** *shell*] [**-c** *comment*] [**-m** [**-k** *skel_dir*]] [**-f** *inactive*] [**-e** *expire*] *login*

useradd -D [**-g** *group*] [**-b** *base_dir*] [**-f** *inactive*] [**-e** *expire*]

DESCRIPTION

The **useradd** command creates a user login on the system by adding the appropriate entry to the **/etc/passwd** file and any security files, modifying the **/etc/group** file as necessary, creating a home directory, and copying the appropriate default files into the home directory. The new login remains locked until the **passwd** (see *passwd*(1)) command is invoked. If the system is in trusted mode, **useradd** returns a random password to stdout: the new login is still locked until the **passwd** command is invoked.

New Behavior

login will not be added to the primary group entry in the **/etc/group** file, even if the primary group is specified in the command line. However, the *login* is added to the corresponding supplemental group in **/etc/group** file.

Options

The **useradd** command supports the following options:

-u *uid* Specifies the UID for the new user. *uid* must be a non-negative decimal integer less than **MAXUID** as it is defined in the <**param.h**> header file. *uid* defaults to the next available unique number above the maximum currently assigned number. UIDs from 0-99 are reserved.

-o Allows the UID to be non-unique (i.e., a duplicate).

-g *group* Specifies the integer group ID or character string name of an existing group. This defines the primary group membership of the new login. The default for this option can be reset by invoking **useradd -D -g** *group*.

-G *group* Specifies the integer group ID or character string name of an existing group. This defines the supplemental group memberships of the new login. Multiple groups may be specified as a comma separated list. Duplicates within *group* with the **-g** and **-G** options are ignored.

-d *dir* Specifies the home directory of the new login. It defaults to *base_dir/login*, where *login* is the new login and *base_dir* is the base directory for new login home directories.

-s *shell* Specifies the full pathname of the new login shell. The default is an empty field, which causes the system to use **/sbin/sh** as the login shell. The value of *shell* must be a valid executable file.

-c *comment* Specifies the comment field present in the **/etc/passwd** entry for this login. This can be any text string. A short description of the new login is suggested for this field.

-m Creates the home directory for the new login if it does not exist. If the home directory exists, the directory must have read and execute permission by *group*, where *group* is the primary group of the new login.

-k *skel_dir* Specifies the skeleton directory that contains information that can be copied to the new login's home directory. This directory must exist. The system provides a skeleton directory, **/etc/skel**, that can be used for this purpose.

-f *inactive* Specifies the maximum number of days of continuous inactivity of the login before the login is declared invalid. Normal values are positive integers, while a value of −1 defeats this status.

-e *expire* Specifies the date on which this login can no longer be used. After *expire*, no user will be able to access this login. This option is used to create temporary logins. *expire*, which is a date, may be typed in any format, except a Julian date. For example, a date may be entered in either of the following formats:

```
July 13, 1993
7/13/93
```

A value of **''''** defeats the expired date status.

-D Manages the defaults for various options. When **useradd** is invoked with this option only, the default values for *group*, *base_dir*, *skel_dir*, *shell*, *inactive*, and *expire* are displayed. Invoking **useradd** with this option and other allowed options sets the default values for those options.

-b *base_dir* Specifies the default base directory for the system. If **-d** *dir* is not specified, *base_dir* is concatenated with the new login name to define the path of the new home directory. *base_dir* must exist.

The **useradd** command may be used with the *login* argument, where *login* is the new login name, specified as a string of printable characters. It may not contain a colon (**:**) or a newline (**\n**).

Unless enhanced security is installed (see *pwconv*(1M)), the **-e** and **-f** options are not supported and will return an error.

Networking Features

NIS

This command is aware of NIS user and group entries. Only local users and groups may be modified with this command. Attempts to modify an NIS user or group will result in an error. NIS users and groups must be administered from the NIS server. NIS users are checked when verifying uniqueness of the new UID or new user name, which may result in the error

login *x* **not unique**

(return value 9), or the error

UID # is not unique (when -o is not used)

(return value 4) even though the user or UID is not present in the local **/etc/passwd** file. The error

Cannot modify /etc/group file, /etc/passwd was modified

(return value 10) is returned if a group specified with either the **-g** option or the **-G** option is an NIS group (see *group*(4)).

NFS

Errors may occur with the **-m** or **-k** options if the indicated directory is within an NFS mounted file system that does not allow root privileges across the NFS mount, and the directory or files within the directory do not have sufficient permissions.

RETURN VALUE

useradd exits with one of the following values:

0 Successful completion.

2 Invalid command syntax.

3 Invalid argument supplied to an option.

4 *uid* is not unique (when **-o** is not used).

6 The *group* specified with the **-g** option does not exist.

9 *login* is not unique.

10 Cannot modify the **/etc/group** file. The login was added to the **/etc/passwd** file, but not to the **/etc/group** file.

12 Unable to create the home directory (while using the **-m** option) or unable to complete the copy of *skel_dir* to the new home directory.

13 Unable to open **/etc/ptmp** file or **/etc/default** file, or **/etc/passwd** file is non-existent.

14 **/etc/passwd**, or **/etc/ptmp**, or **/etc/default** file busy. Another command may be modifying the **/etc/passwd** file.

16 Cannot add the entry into the **/etc/passwd** file.

EXAMPLES

Add the user **otto** to the system with all of the default attributes.

```
useradd otto
```

Add the user **otto** to the system with a UID of **222** and a primary group of **staff**.

```
useradd -u 222 -g staff otto
```

List the defaults for the primary group, base directory, inactivity timeout, and skeleton directory.

```
useradd -D
```

Change the default primary group to **staff**.

```
useradd -D -g staff
```

WARNINGS

A directory can be shared between the users belonging to the same group. If the home directory is in the unshared mode and a new user is allocated to that directory then it will be put into the shared mode by setting the permissions of that directory to **775** (i.e. includes the write permissions to the group as well). Also, the directory which will be shared should have **read** and **execute** permissions for the group. Otherwise, **useradd** will report an error.

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **useradd** terminates.

A group entry in the **/etc/group** file can have maximum of **LINE_MAX** bytes. If a user is added to a group that has reached **LINE_MAX** limit, another entry of the same group is created to which the new user is added. A warning message is also issued.

FILES

/etc/passwd
/etc/skel
/etc/group
/etc/ptmp

SEE ALSO

passwd(1), users(1), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), userdel(1M), usermod(1M), group(4).

STANDARDS COMPLIANCE

useradd: SVID3

NAME

userdel - delete a user login from the system

SYNOPSIS

userdel [**-r**] *login*

DESCRIPTION

The **userdel** command deletes a user login from the system by modifying the appropriate login related files.

The **userdel** command requires the *login* argument. *login* is the name to be deleted, specified as a string of printable characters. It may not contain a colon (:) or a newline (\n).

Option

The following option is available to this command:

-r The home directory of *login* is removed from the system. This directory must exist. Following the successful execution of this command, none of the files and directories under the home directory will be available.

If a user is deleted and the home directory is shared by others, then this directory is not deleted even with the **-r** option.

In the event where a directory is shared by users of the same group and the owner of that directory is deleted, then the ownership of that directory is propagated to the next user who is sharing that directory. The new owner is determined by looking at the order in which the users sharing this directory are added to the **/etc/passwd** file. If there is only one user remaining then the directory is brought back to unshared mode by resetting the permissions to **755** from **775**.

NETWORKING FEATURES

NIS

This command is aware of NIS user and group entries. Only local users and groups may be deleted or modified with this command. Attempts to delete or modify NIS users or groups will result in an error. NIS users and groups must be administered from the NIS server. The **userdel** command may fail with the error

login *x* **does not exist**

(return value 6) if the user specified is an NIS user (see *passwd*(4)). The error

Cannot modify /etc/group file, /etc/passwd was modified

(return value 10) is returned if a local user belongs to an NIS group (see *group*(4)).

NFS

Errors may occur with the **-r** option if the affected directory is within an NFS mounted file system that does not allow root privileges across the NFS mount, and the directory or files within the directory do not have sufficient permissions.

RETURN VALUES

u

userdel exits with one of the following values:

0 Successful completion.

2 Invalid command syntax.

3 Invalid argument supplied to an option.

6 The *login* to be removed does not exist.

8 The *login* to be removed is in use.

10 Cannot modify the **/etc/group** file, but the login was removed from the **/etc/passwd** file.

12 Unable to remove or modify the home directory.

13 Unable to open **/etc/ptmp** file or **/etc/passwd** file is non-existent.

14 **/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.

17 Cannot delete entry from **/etc/passwd** file.

EXAMPLES

Remove the user **otto** from the system:

```
userdel otto
```

Remove the user **bob** from the system and delete **bob**'s home directory from the system:

```
userdel -r bob
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **userdel** terminates.

FILES

/etc/passwd
/etc/group
/etc/ptmp

SEE ALSO

passwd(1), users(1), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), passwd(1M), useradd(1M), usermod(1M), group(4).

STANDARDS COMPLIANCE

userdel: SVID3

NAME

usermod - modify a user login on the system

SYNOPSIS

usermod [**-u** *uid* [**-o**]] [**-g** *group*] [**-G** *group* [**,** *group*...]] [**-d** *dir* [**-m**]] [**-s** *shell*]
[**-c** *comment*] [**-f** *inactive*] [**-l** *new_logname*] [**-e** *expire*] *login*

DESCRIPTION

The **usermod** command modifies a user login on the system by changing the appropriate login related files.

The **usermod** command requires the *login* argument. *login* is a new login name, specified as a string of printable characters. It may not contain a colon (**:**) or a newline (**\n**).

New Behavior

If the primary group of a user is modified, then the user name is not added to the primary group entry in **/etc/group** file. However, if **-G** option is specified the user is added to the corresponding supplemental group.

Options

The **usermod** command supports the following options:

-u *uid*	Specifies the UID for the new user. *uid* must be a non-negative decimal integer less than **MAXUID** as it is defined in the <**param.h**> header file.
-o	Allows the UID to be non-unique (i.e., a duplicate).
-g *group*	Specifies the integer group ID or character string name of an existing group. This redefines the primary group membership of the new login.
-G *group*	Specifies the integer group ID or character string name of an existing group. This redefines the supplemental group memberships of the new login. Duplicates within *group* with the **-g** and **-G** options are ignored.
-d *dir*	Specifies the new home directory of the login. It defaults to *base_dir/login*, where *login* is the new login and *base_dir* is the base directory for new login home directories.
-m	Move the user's home directory to the directory specified with the **-d** option. If the home directory exists, the directory must have read and execute permission by *group*, where *group* is the primary group of the login.
-s *shell*	Specifies the full pathname of the login shell. The value of *shell* must be a valid executable file.
-c *comment*	Specifies the comment field present in the **/etc/passwd** entry of this login. This can be any text string. A short description of the new login is suggested for this field.
-f *inactive*	Specifies the maximum number of days of continuous inactivity of the login before the login is declared invalid. Normal values are positive integers, while a value of –1 defeats this status.
-l *new_logname*	Specifies the new login name for the user. It consists of a string of printable characters that does not contain a colon (**:**) or a newline (**\n**).
-e *expire*	Specifies the date on which this login can no longer be used. After *expire*, no user will be able to access this login. This option is used to create temporary logins. *expire*, which is a date, may be typed in any desired format, except a Julian date. For example, a date may be entered as either of the following: **July 13, 1993** / **7/13/93** A value of **''''** defeats the expired date status.

Unless enhanced security is installed (see *pwconv*(1M)), the **-e** and **-f** options are not supported and will return an error.

u

A directory can be shared between the users belonging to the same group. If the home directory is in unshared mode and a new user is allocated to that directory, then it will be put into shared mode by setting the permissions of that directory to **775** (i.e., includes the write permissions to the group as well). Also, the directory which will be shared should have read and execute permissions for the group.

In the event where a directory is shared by users of the same group and the owner of that directory is modified, then the ownership of that directory is propagated to the next user who is sharing that directory. The new owner is determined by looking at the order in which the users sharing this directory are added to the **/etc/passwd** file. If there is only one user remaining then the directory is brought back to unshared mode by resetting the permissions to **755** from **775**.

If a directory is shared by users, then one cannot change the primary group of any of these users unless the home directory of that user is also changed.

Networking Features

NIS

The **usermod** command is aware of NIS user and group entries. Only local users and groups may be modified with this command. Attempts to modify an NIS user or group will result in an error. NIS users and groups must be administered from the NIS server. This command may fail with the error

login *x* **does not exist**

(return value 6) if the user specified is an NIS user (see *passwd*(4)). However, NIS users are checked when verifying uniqueness of the new UID or the new user name. Also, the error

Cannot modify /etc/group file, /etc/passwd was modified

(return value 10) may be returned if a group specified with either the **-g** option or the **-G** option is an NIS group (see *group*(4)).

NFS

Errors may occur with the **-m** option if either the source or the target directory is within an NFS mounted file system that does not allow root privileges across the NFS mount and the directory or files within the directory do not have sufficient permissions.

RETURN VALUE

usermod exits with one of the following values:

0	Successful completion.
2	Invalid command syntax.
3	Invalid argument supplied to an option.
4	*uid* is not unique (when **-o** is not used).
6	The *login* to be modified or the *group* specified with the **-g** option does not exist.
8	The *login* to be modified is in use.
9	*new_logname* is not unique.
10	Cannot modify the **/etc/group** file. The other parts of the update request will be performed.
11	There is insufficient space to move the home directory (with the **-m** option). The other parts of the update request will be performed.
12	Unable to complete the move of the home directory to the new home directory.
13	Unable to open **/etc/ptmp** file, or **/etc/passwd** file is non-existent.
14	**/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.
15	Cannot modify the entry in the **/etc/passwd** file.

EXAMPLES

Change **otto**'s primary group to **staff**.

```
usermod -g staff otto
```

Change **otto**'s user ID to **333** and change the login name to **bob**.

```
        usermod -u 333 -l bob otto
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was devised. If this locking fails after subsequent retrying, **usermod** terminates.

While modifying the user login, the username is not added to the primary group entry in the **/etc/group** file. If a supplemental group is specified, the user is added to the supplemental group. If the size of a group entry in **/etc/group** file exceeds **LINE_MAX** limit, a new entry of the same group is created and a warning message is issued.

FILES

/etc/passwd
/etc/group
/etc/ptmp

SEE ALSO

passwd(1), users(1), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), useradd(1M), userdel(1M), group(4).

STANDARDS COMPLIANCE

usermod: SVID3

NAME

uucheck - check the uucp directories and permissions file

SYNOPSIS

/usr/lbin/uucp/uucheck [**-v**] [**-x** *debug_level*]

DESCRIPTION

uucheck checks for the presence of the files and directories required by **uucp** (see *uucp*(1)). **uucheck** is executed from the UUCP makefile before the installation occurs. **uucheck** also checks for various obvious errors in the **/etc/uucp/Permissions** file.

Options

uucheck recognizes the following options and command-line arguments:

-v (verbose) Print a detailed explanation of how **uucp** programs will interpret the **Permissions** file.

-x *debug_level*
Debug. *debug_level* is a single digit; the higher the number, the more detail returned.

Note that **uucheck** can only be used by the super-user or **uucp**.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Maxuuxqts
/etc/uucp/Maxuuscheds
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1), uustat(1), uux(1), uucico(1M), uusched(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uucico - transfer files for the uucp system

SYNOPSIS

/usr/lbin/uucp/uucico -r1 -s *system* [**-x** *debug_level*] [**-d** *spool_directory*]

/usr/lbin/uucp/uucico [**-x** *debug_level*] [**-d** *spool_directory*]

DESCRIPTION

uucico scans the **/var/spool/uucp** directories for work files. If such files exist, a connection to a remote system is attempted using the line protocol for the remote system specified in file **/etc/uucp/Systems**. **uucico** then executes all requests for work and logs the results.

Options

uucico recognizes the following options:

-r1 Start **uucico** in the MASTER mode. The default is SLAVE mode.

-s *system* Do work only for the system specified by *system*. If there is no work for *system* on the local spool directory, initiate a connection to *system* to determine if *system* has work for the local system. This option must be used if **-r1** is specified.

-d *spool_directory*
Search directory *spool_directory* instead of the default spool directories (usually **/var/spool/uucp/***).

-x *debug_level* Use debugging option. *debug_level* is an integer in the range 1 through 9. More debugging information is given for larger values of *debug_level*.

uucico is usually started by a local program such as **cron**, **uucp**, or **uuxqt** (see *cron*(1M), *uucp*(1), and *uuxqt*(1M)). It when debugging should a user initiate **uucico** directly.

When started by a local program, **uucico** is considered the MASTER and attempts a connection to a remote system. If **uucico** is started by a remote system, it is considered to be in SLAVE mode.

For the **uucico** connection to a remote system to be successful, there must be an entry in the **/etc/passwd** file on the remote system of the form:

```
uucp::5:5::/var/spool/uucppublic:/usr/lbin/uucp/uucico
```

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Maxuuxqts
/etc/uucp/Maxuuscheds
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

u

SEE ALSO

uucp(1), uustat(1), uux(1), cron(1M), uusched(1M), uutry(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uuclean - uucp spool directory clean-up

SYNOPSIS

/usr/lbin/uucp/uuclean [*options*]

DESCRIPTION

uuclean scans the spool directories for files with the specified prefix and deletes all those that are older than the specified number of hours.

Options

uuclean recognizes the following options:

-d*directory*	Clean *directory* instead of the spool directory. If *directory* is not a valid spool directory, it cannot contain “work files”; i.e., files whose names start with **C.**. These files have special meaning to **uuclean** pertaining to **uucp** job statistics.
-p*pre*	Scan for files with *pre* as the file prefix. Up to 10 **-p** arguments can be specified. A **-p** without any *pre* following will cause all files older than the specified time to be deleted.
-n*time*	Files whose age is more than *time* hours are deleted if the prefix test is satisfied (default time is 72 hours).
-w*file*	The default action for **uuclean** is to remove files that are older than a specified time (see **-n** option). The **-w** option is used to find files older than *time* hours; however, the files are not deleted. If the argument *file* is present the warning is placed in *file*; otherwise, the warnings go to the standard output.
-s*sys*	Only files destined for system *sys* are examined. Up to 10 **-s** arguments can be specified.
-m*file*	The **-m** option sends mail to the owner of the file when it is deleted. If a *file* is specified, an entry is placed in *file*.

This program is typically started by **cron** (see *cron*(1M)).

FILES

/var/spool/uucp/* spool directory

SEE ALSO

uucp(1), uux(1), cron(1M), uucleanup(1M).

Tim O’Reilly and Grace Todino,
Managing UUCP and Usenet, O’Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O’Reilly & Associates, Inc. USA.

NAME

uucleanup - uucp spool directory clean-up

SYNOPSIS

/usr/lbin/uucp/uucleanup [**-C** *time*] [**-D** *time*] [**-W** *time*] [**-X** *time*] [**-m** *string*] [**-o** *time*] [**-s** *system*] [**-x** *debug_level*]

DESCRIPTION

uucleanup scans the spool directories for old files and takes appropriate action to remove them. Depending on the options selected, **uucleanup** performs the following:

- Informs the requestor of send and/or receive requests for systems that cannot be reached.
- Returns mail that cannot be delivered to the sender.
- Removes all other files.

In addition, **uucleanup** warns users of requestors who have been waiting for a given number of days (the default is 1 day). Note that unless *time* is specifically set, the default *time* values for the following options are used.

Options

uucleanup recognizes the following options:

-C*time* Any **C.** files greater or equal to *time* days old are removed with appropriate information to the requestor. The default *time* is 7 days.

-D*time* Any **D.** files greater or equal to *time* days old are removed. An attempt is made to deliver mail messages and execute news when appropriate. The default *time* is 7 days.

-W*time* Any **C.** files equal to *time* cause a message to be mailed to the requestor warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator can include a message line telling who to call to correct the problem (see the **-m** option). The default *time* is 1 day.

-X*time* Any **X.** files greater than or equal to *time* days old are removed. The **D.** files are probably not present (if they were, the **X.** could be executed). But, if **D.** files are present, they are taken care of by **D.** processing. The default *time* is 2 days.

-m*string* This string is included in the warning message generated by the **-W** option. The default string is **See your local administrator to locate the problem.**

-o*time* Other files whose age is more than *time* days are deleted. The default time is 2 days.

-s*system* Clean-up the spool directory for *system* only. The default is to clean-up all spool directories.

-x*debug_level* The debug level is a single digit between 0 and 9. The higher the numbers, the more detailed the debugging information returned.

This program is typically started by the script **uudemon.cleanu**, which should be started by **cron** (see *cron*(1M)).

FILES

/var/spool/uucp/* spool directory

SEE ALSO

cron(1M), uucp(1), uux(1), uuclean(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

/usr/sbin/uucpd - UUCP over TCP/IP server daemon

DESCRIPTION

uucpd is the server for supporting UUCP connections over TCP/IP networks.

uucpd is invoked by *inetd*(1M) when a UUCP connection is established (that is, a connection to the port indicated in the "uucp" service specification; see *services*(4)), and executes the following protocol:

1) The server prompts with "login:", the uucico process at the other end must supply a username.
2) Unless the username refers to an account without a password, the server then prompts with "Password:", the uucico process at the other end must supply the password for that account.

If the username is not valid or is valid but refers to an account that does not have **/usr/lbin/uucp/uucico** as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, *uucico*(1M) is run. Entries are made in **/var/adm/wtmp** traceable with *who*(1) and *last*(1M).

PROTOCOL RESTRICTION

Only 'g' protocol for uucico is supported.

DIAGNOSTICS

All diagnostic messages are returned on the connection, after which the connection is closed.

user read
An error occurred while reading the username.

passwd read
An error occurred while reading the password.

login incorrect
The username or the password is invalid or the user's login shell for this account is not /usr/lbin/uucp/uucico.

WARNINGS

On Trusted Systems uucpd prohibits uucico to start if any of the following are true :

- the login account is locked (several causes).
- current time doesn't match existing time-of-day restrictions for this account.

Under such conditions uucpd will return the message **login incorrect** to the connection. The connection is then dropped.

AUTHOR

uucpd was developed by the University of California, Berkeley and HP.

FILES

/etc/inetd.conf	configuration file for inetd
/var/adm/inetd.sec	optional security file for inetd
/etc/services	service name data base
/var/adm/wtmp	login data base

SEE ALSO

inetd(1M), services(4), uucico(1M).

NAME

uugetty - set terminal type, modes, speed and line discipline

SYNOPSIS

/usr/lbin/uucp/uugetty [**-h**] [**-t** *timeout*] [**-r**] *line* [*speed* [*type* [*linedisc*]]]

/usr/lbin/uucp/uugetty -c *file*

DESCRIPTION

uugetty sets terminal type, modes, speed and line discipline. It is similar to **getty**, except that **uugetty** supports using the line in both directions (see *getty*(1M)). This allows users to log in, but, if the line is free, **uucico**, **cu**, and **ct** can dial out (see *uucico*(1), *cu*(1), and *ct*(1)). When devices are used with **uucico**, **cu**, and **ct**, lock files are created. Therefore, when the call to **open()** returns (see *open*(2)) (or the first character is read when the **-r** option is used), the status of the lock files indicates whether the line is used by **uucico**, **cu**, **ct**, or someone trying to log in. See *getty*(1M) for more information.

Note that with the **-r** option, several carriage-return characters might be required before the login message is output. When **uucico** is trying to log in, it can be instructed to enter numerous carriage-return characters with the following login script:

```
\r\d\r\d\r\d\r in:-in: ...
```

where ... represents whatever would normally be used for the login sequence.

An entry for an intelligent modem or direct line that has a **uugetty** on each end must use the **-r** option (this causes *uugetty* to wait to read a character before it enters the login message, thus preventing two instances of **uugetty** from looping). If there is a **uugetty** on one end of a direct line, there must be a **uugetty** on the other end as well.

EXAMPLES

The following line is an **/etc/inittab** entry using **uugetty** on an intelligent modem or direct line:

```
30:2:respawn:/usr/lbin/uucp/uugetty -r -t 60 tty12 1200
```

WARNINGS

ct does not work when **uugetty** is used with an intelligent modem such as a Penril or a Ventel.

FILES

/etc/gettydefs
/etc/issue
/var/spool/locks/LCK*

SEE ALSO

ct(1), cu(1), login(1), uucico(1M), getty(1M), init(1M), ioctl(2), gettydefs(4), inittab(4), tty(7).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uuls - list spooled uucp transactions grouped by transaction

SYNOPSIS

uuls [**-m**] [*directories* ...]

uuls [**-s**] [**-m**] [*directories* ...]

uuls [**-k**] [**-m**] [*directories* ...]

DESCRIPTION

This command lists the contents of UUCP spool directories (default **/var/spool/uucp/***) with the files in each directory grouped into three categories:

- Transactions,
- Orphans, and
- Others.

Transactions

Each output line starts with a transaction control filename, and includes the name of each local (same-directory) subfile referenced by the control file (see below). Each is possibly followed by the total size in bytes (**-s** option) or Kbytes (**-k** option) in the transaction (see below). The **-m** (meanings) option replaces the subfile names with nodename, user, and *commandline* information (see below).

Orphans

All subfiles not referenced by any control file.

Others

All other files in the directory (all files not listed under one of the above categories).

Filenames are formatted into columns, so there can be more than one file per line. If a transaction has more subfiles than fit on one line, it is followed by continuation lines which are indented further.

The **-s** (size in bytes) and **-k** (Kbytes) options cause the command to follow each transaction in the **Transactions** section with a total size for all stat-able, sendable files in that transaction. This includes **D.*** files only, not **C.*** or **X.*** files. It does include stat-able files outside the spool directory that are indirectly referenced by **C.*** files. Sizes are either in bytes or rounded to the nearest Kbyte (1024 bytes), respectively. A totals line is also added at the end of the **Transactions** section.

The **-m** (meanings) option causes the command to follow **C.*** and **X.*** files with a *nodename***!***username commandline* line, instead of subfilenames. For **C** files, one line is printed per remote execution (**D***X**) subfile it references. *nodename* is truncated at seven characters, *username* at eight, and *commandline* at however much fits on one line.

If **-m** is given, for each **C** file with no remote execution files, the command instead shows the meaning of the **C** file itself on one or more lines. Each line consists of a username, then **R** (receive) or **S** (send), then the name of the file to be transferred. See below for details.

Filenames are listed in ascending collation order within each section (see Environment Variables below), except that the first section is only sorted by the control filename. Every file in the directory except **.** and **..** appears exactly once in the entire list, unless **-m** is used.

Details

Transaction files are those whose names start with **C.** or **X.**. Subfilenames, which usually start with **D.**, are gleaned from control file lines, at most one per line, from blank-separated fields, as follows:

```
C.*:  R <remotefrom> <localto> <user> -<options>
C.*:  S <localfrom> <remoteto> <user> -<options> <subfile> <mode>
X.*:  F <subfile>
```

Lines that do not begin with the appropriate character (**R**, **S**, or **F**) are ignored.

In the **R** (receive) case, <remotefrom> is used to print the **C**-file meaning, and its transaction size is taken as zero (unknown).

In the **S** (send) case, if <subfile> is **D.0**, <localfrom> is a file not in the spool directory, resulting from a typical **uucp** call without the **-C** (copy) option. In this case <localfrom> is used for the transaction size, if stat-able, and to print the **C**-file meaning.

uucp -C and **uux** both set <subfile> to a true (spooled) subfile name.

Orphan files are those whose names start with **D.** and which are not referenced by any control files.

This algorithm extracts from control files the names of all subfiles that should exist in the spool directory when the transaction is not being actively processed. It is not unusual to see "missing subfiles" and "orphans" if you **uuls** a spool directory while **uucico**, **uucp**, **uux**, or **uuxqt** is active.

Meanings information is obtained by reading each **D**∗**X**∗ subfile referenced by each **C.**∗ file, and by reading **X**∗**X**∗ files. *nodename***!***username* is taken from the last line in the file which is of the form:

U <username> <nodename>

Likewise, *commandline* is taken from the last line of the form:

C <commandline>

If a subfile name is referenced more than once, references after the first show the subfile as missing. If a subfile name appears in a (corrupt) directory more than once, the name is only found once, but then it is listed again under **Orphans**.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

If LC_COLLATE is not specified in the environment or is set to the empty string, the value of LANG is used as a default. If LANG is not specified or is set to the empty string, a default of “C” (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **uuls** behaves as if all internationalization variables are set to “C” (see *environ*(5)).

DIAGNOSTICS

The program writes an appropriate message to standard error if it has any problems dealing with a specified file (directory), including failure to get heap space. It always returns zero as its exit value.

If a control file is unopenable (wrong permissions or it disappeared while **uuls** was running), its name is preceded by a ∗ and the size of the transaction is zero. If a subfile is missing (filename not found in the directory being listed) or not stat-able (if required for **-s** or **-k**), its name is preceded by a ∗ and it contributes zero bytes to the size of the transaction.

If **-m** is specified and a **D**∗**X**∗ file is missing or unreadable, its name is given with a ∗ prefixed, as usual.

BUGS

This command uses *chdir*(2) to change to each directory in turn. If more than one is specified, the second through last directories must be absolute (not relative) pathnames, or the *chdir*() may fail.

AUTHOR

uuls was developed by HP.

SEE ALSO

mail(1), uucp(1), uuto(1), uux(1), uuxqt(1M), stat(2).

NAME

uusched - schedule uucp transport files

SYNOPSIS

/usr/lbin/uucp/uusched [**-u** *debug_level*] [**-x** *debug_level*]

DESCRIPTION

uusched is the UUCP file transport scheduler. It is usually started by the daemon **uudemon.hour**, which is started by **cron** (see *cron*(1M)) from the following entry in **/var/spool/cron**:

```
39 * * * * /usr/bin/su uucp -c */usr/lbin/uucp/uudemon.hour > /dev/null*
```

Options

uusched recognizes two options which are provided for debugging purposes only.

-x *debug_level*	Output debugging messages.
-u *debug_level*	Pass as **-x** to **uucico** (see *uucico*(1M)). The *debug_level* is a number between 0 and 9. The higher the number, the more detailed the information returned.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

cron(1M), uucico(1M), uusched(1M), uucp(1), uustat(1), uux(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uusnap - show snapshot of the UUCP system

SYNOPSIS

uusnap

DESCRIPTION

uusnap displays in tabular format a synopsis of the current UUCP situation. The format of each line is as follows:

```
site    N Cmds    N Data    N Xqts    Message
```

Where *site* is the name of the site with work, *N* is a count of each of the three possible types of work (command, data, or remote execute), and *Message* is the current status message for that site as found in the **STST** file.

Included in *Message* may be the time left before UUCP can re-try the call, and the count of the number of times that UUCP has tried to reach the site. The process id of **uucico** may also be shown if it is in a TALKING state.

AUTHOR

uusnap was developed by the University of California, Berkeley.

SEE ALSO

uucp(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uusnaps - sort and embellish uusnap output

SYNOPSIS

uusnaps

DESCRIPTION

uusnaps runs **uusnap** (see *uusnap*(1M)) and post-processes the output into a more useful form. It sorts output lines in "Pareto-style", showing first those remote systems with the greatest number of **Cmds** files, next **Data** files, and then **Xqts** files.

uusnaps inserts a ***** after the number of **Xqts** files on those lines where **Data** is not equal to (2 × **Cmds**) + **Xqts**. This may be a sign of missing or orphaned transaction parts. Use **uuls** to check (see *uuls*(1)).

uusnaps adds summary information after all **uusnap** output. The first line is a total of the numbers of **Cmds**, **Data**, and **Xqts** files. The second line contains a grand total number of transaction files, followed by the number of directory bytes this represents. This is an indication of the true size of the directory itself if all empty entries were squeezed out. Finally, if it appears that transaction files might be missing or orphaned, **uusnaps** returns the number of missing or excess files.

WARNINGS

uusnaps assumes that each directory entry takes 24 bytes.

SEE ALSO

uusnap(1M), uuls(1).

NAME

uusub - monitor uucp network

SYNOPSIS

/usr/lbin/uucp/uusub [*options*]

DESCRIPTION

uusub defines a **uucp** subnetwork and monitors the connection and traffic among the members of the subnetwork.

Options

uusub recognizes the following options:

-a*sys*	Add *sys* to the subnetwork.
-d*sys*	Delete *sys* from the subnetwork.
-l	Report the statistics on connections.
-r	Report the statistics on traffic amount.
-f	Flush the connection statistics.
-u*hr*	Gather the traffic statistics over the past *hr* hours.
-c*sys*	Exercise the connection to the system *sys*. If *sys* is specified as **all**, exercise the connection to all the systems in the subnetwork.

The connections report is formatted as follows:

sys #call #ok time #dev #login #nack #other

Format interpretation:

sys	remote system name,
#call	number of times the local system tried to call *sys* since the last flush was done,
#ok	number of successful connections,
time	latest successful connect time,
#dev	number of unsuccessful connections because of no available device (e.g., ACU),
#login	number of unsuccessful connections because of login failure,
#nack	number of unsuccessful connections because of no response (e.g. line busy, system down),
#other	number of unsuccessful connections because of other reasons.

Traffic statistics are reported as follows:

sfile sbyte rfile rbyte

Format interpretation:

sfile	number of files sent,
sbyte	number of bytes sent over the period of time indicated in the latest *uusub* command with the **-u***hr* option,
rfile	number of files received,
rbyte	number of bytes received.

The command:

```
uusub -c all -u 24
```

is typically started by **cron** once a day.

FILES

/var/uucp/.Admin/L_sub	connection statistics
/var/uucp/.Admin/R_sub	traffic statistics

u

`/var/uucp/.Log/*`	system log file

SEE ALSO

uucp(1), uustat(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uuxqt - execute remote uucp or uux command requests

SYNOPSIS

/usr/lbin/uucp/uuxqt [**-s** *system*] [**-x** *debug_level*]

DESCRIPTION

uuxqt executes remote job requests generated by use of the **uux** command (see *uux*(1)). **uux** generates **X.** files and places them in the spool directory, where **uuxqt** searches for them. For each **X.** file, **uuxqt** determines whether the required data files are available and accessible, and if file commands are permitted for the requesting system. The **Permissions** file is used to validate file accessibility and command execute permission. Then **uuxqt** performs execution of the commands.

Two environment variables are set before the **uuxqt** command is executed: **UU_MACHINE** is the machine that sent the previous job and **UU_USER** is the user who sent the job. These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

uuxqt recognizes the following options:

-s *system*	Execute commands on the specified *system* .
-x *debug_level*	Produce debugging output on standard output. *debug_level* is a single digit between 0 and 9. The higher the number, the more detailed debugging information returned.

FILES

/etc/uucp/Permissions
/etc/uucp/Maxuuxqts
/var/spool/uucp/*
/var/spool/locks/LCK*

SEE ALSO

uucp(1), uustat(1), uux(1), uucico(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

vgcfgbackup - create or update LVM volume group configuration backup file

SYNOPSIS

/usr/sbin/vgcfgbackup [**-f** *vg_conf_path*] [**-u**] *vg_name*

DESCRIPTION

The **vgcfgbackup** command saves the LVM configuration for a volume group in a default or alternate configuration backup file (see the **-f** option).

By default, **vgcfgbackup** runs automatically each time an LVM command changes the LVM configuration. In this case, it always uses the default configuration backup file. An existing default configuration backup file is renamed with an extension of **.old**.

Options and Arguments

vgcfgbackup recognizes the following options and arguments:

vg_name The path name of a volume group.

-f *vg_conf_path* Save the configuration using an alternate file name specified by *vg_conf_path*.

If **-f** is omitted, the default file name is in the form:

/etc/lvmconf/ *base_vg_name* **.conf**

base_vg_name is the base name of *vg_name*. For example, if *vg_name* is specified as **/dev/vg00**, *base_vg_name* is **vg00**.

-u Update the configuration backup file with the latest LVM configuration. Only those physical volumes added since the configuration backup file was last modified need to be online.

If **-u** is omitted, all physical volumes for *vg_name* must be online.

RETURN VALUE

vgcfgbackup exits with one of the following values:

0 Successful completion.
>0 Failure. Errors occurred when information from the volume group was being accessed.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Back up LVM configuration information for volume group **/dev/vg00** in the default backup file **/etc/lvmconf/vg00.conf**:

```
vgcfgbackup /dev/vg00
```

Update LVM configuration information corresponding to volume group **/dev/vg00** in the default backup file **/etc/lvmconf/vg00.conf**:

```
vgcfgbackup -u /dev/vg00
```

Back up LVM configuration information for volume group **/dev/vg00** in the alternate configuration backup file **/tmp/vg00.backup**:

```
vgcfgbackup -f /tmp/vg00.backup vg00
```

WARNINGS

It is recommended that any alternate configuration backup file be created in the root file system (as is the case with the default path name). This facilitates easy volume group recovery during maintenance mode, such as after a system crash.

AUTHOR

vgcfgbackup was developed by HP.

SEE ALSO

vgcfgrestore(1M).

NAME

vgcfgrestore - display or restore LVM volume group configuration from backup file

SYNOPSIS

/usr/sbin/vgcfgrestore -n *vg_name* **-l**

/usr/sbin/vgcfgrestore [-R] -n *vg_name* [**-o** *old_pv_path*] *pv_path*

/usr/sbin/vgcfgrestore -f *vg_conf_path* **-l**

/usr/sbin/vgcfgrestore [-R] -f *vg_conf_path* [**-o** *old_pv_path*] *pv_path*

Remarks

vgcfgrestore cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **vgcfgrestore** command restores the LVM configuration data from a default (**-n** option) or alternate (**-f** option) configuration backup file to the physical volume named by *pv_path*. Or, it displays the configuration data on standard output (**-l** option).

The configuration stored for one physical volume, *old_pv_path*, can be copied to another physical volume *pv_path* (**-o** option).

Options and Arguments

vgcfgrestore recognizes the following options and arguments:

pv_path The raw (character) device path name of a physical volume that is currently online.

If the **-o** option is omitted, *pv_path* must specify a physical volume whose configuration is stored in the configuration backup file.

-f *vg_conf_path* Get configuration information from the alternate configuration backup file *vg_conf_path*.

-l List the configuration information saved in the specified configuration backup file.

-n *vg_name* Get configuration information from the default configuration backup file:

/etc/lvmconf/ *base_vg_name* **.conf**

vg_name is the path name of the volume group.

base_vg_name is the base name of *vg_name*. For example, if *vg_name* is specified as **/dev/vg00**, *base_vg_name* is **vg00**.

-o *old_pv_path* Restore the configuration information saved for physical volume *old_pv_path* to physical volume *pv_path*.

This option is useful when a physical volume's name has changed since the configuration backup file was created or updated.

old_pv_path must be the path name of a physical volume whose configuration is stored in the configuration backup file. It need not be currently online.

pv_path must be the path name of a physical volume that is currently online. Its configuration need not be stored in the configuration backup file.

-R This option will force restoring the LVM configuration data even if there is a physical volume mismatch between the kernel and the configuration backup file with the volume group still active. This option should not be used unless the configuration file is absolutely valid and up-to-date. Restoring invalid configuration data can result in data corruption later.

If there are alternate physical volume links configured in the system, the following message will appear when total number of physical volumes in the kernel does not match with the configuration backup file due to missing alternate physical volume links:

```
Mismatch between the backup file and the running kernel:
Kernel indicates X disks for /dev/vgname; /etc/lvmconf/vgname
```

V

```
indicates Y disks.  Cannot proceed with the restoration.
Deactivate the Volume Group and try again.
```

In this case, the user is advised to deactivate the volume group first, then use the **vgcfgrestore** command to restore configuration data when the volume group is unavailable. But if the volume group has to stay available and the user is absolutely sure the configuration file is correct, this option will restore data from the configuration file when the volume group stays available.

RETURN VALUE

vgcfgrestore exits with one of the following values:

0	Successful completion.
>0	Failure. Errors occurred during the restore operation.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Restore the LVM configuration information for the physical volume **/dev/rdsk/c0t7d0** that was saved in the default file **/etc/lvmconf/vg00.conf**:

```
vgcfgrestore -n /dev/vg00 /dev/rdsk/c0t7d0
```

Force to restore the LVM configuration data when volume group is still active

```
vgcfgrestore -R -n /dev/vg00 /dev/rdsk/c0t7d0
```

Restore the LVM configuration information to physical volume **/dev/rdsk/c0t4d0** using alternate configuration file **/tmp/vg00.backup**:

```
vgcfgrestore -f /tmp/vg00.backup /dev/rdsk/c0t4d0
```

List backup information saved in default configuration file **/etc/lvmconf/vg00.conf**:

```
vgcfgrestore -n /dev/vg00 -l
```

Above command might display the following:

```
Volume Group Configuration information in "/etc/lvmconf/vg00.conf"
VG Name /dev/vg00
 ---- Physical volumes : 2 ----
    /dev/rdsk/c0t6d0 (Bootable)
    /dev/rdsk/c0t5d0 (Non-bootable)
```

Restore LVM configuration information stored for **/dev/rdsk/c0t7d0** in default configuration file **/etc/lvmconf/vg01.conf** to physical volume **/dev/rdsk/c0t6d0**:

```
vgcfgrestore -n /dev/vg01 -o /dev/rdsk/c0t7d0 /dev/rdsk/c0t6d0
```

V

WARNINGS

Preferably, the volume group should be made unavailable before executing **vgcfgrestore** by executing the command

vgchange -a n *vg_name*

AUTHOR

vgcfgrestore was developed by HP.

SEE ALSO

vgcfgbackup(1M).

NAME

vgchange - set LVM volume group availability

SYNOPSIS

Activate volume group

/usr/sbin/vgchange -a *availability* [**-l**] [**-p**] [**-q** *quorum*] [**-s**] [**-P** *resync_daemon_count*] [*vg_name* ...]

Assign to high availability cluster and mark volume group sharable

/usr/sbin/vgchange -c *cluster* **-S** *sharable vg_name*

Remarks

MC/ServiceGuard cluster operations require the installation of the optional MC/ServiceGuard software, which is not included in the standard HP-UX operating system.

Lock Manager cluster operations require the installation MC/LockManager software which is not included with the standard HP-UX operating system.

Mirrored disk operations require the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

DESCRIPTION

The **vgchange** command with the **-a** option activates or deactivates one or more volume groups.

The **vgchange** command with the **-c** option controls the membership of one or more volume groups in a high availability cluster. The **vgchange** command with the **-c** and **-S** options control the membership of a volume group and mark it sharable.

The **vgchange** command without the **-P** *resync_daemon_count* option (default) will spawn one *nomwcsyncd* process for each **NOMWC/NONE** volume group being activated. This may create a lot of *nomwcsyncd* processes running concurrently when it activates a large number of **NOMWC/NONE** volume groups and overload.

The **-P** *resync_daemon_count* option provides a way to control the number of concurrent *nomwcsyncd* processes. The count is an advisory number and a different count might be chosen internally if load balance or other reason is needed. When specified, there are up to *resync_daemon_count* **+ 1** *nomwcsyncd* processes; one of them is the controlling processing to spawn others. **-P 0** will use the system default (currently defined to be 4).

vg_name must be defined as a volume group in the file **/etc/lvmtab**. If *vg_name* is omitted, all volume groups defined in **/etc/lvmtab** are affected.

High Availability Cluster Overview

Volume groups can be defined on disk volumes that are connected to two or more systems in a high availability cluster. This situation has a high potential for data corruption unless special software is used to coordinate shared access to the same volume group by all systems. This coordination is provided by MC/ServiceGuard or MC/LockManager.

A volume group can be marked as part of a MC/ServiceGuard cluster. When such a group is activated in exclusive mode, it can be accessed for exclusive read-write activity by only one of the systems at a time; the other systems can have read-only access to the data.

A volume group can be marked as a part of an MC/LockManager cluster. In this case, the volume group can be marked as sharable, and may be activated in shared mode for read-write access by all the nodes in the cluster. Shared read-write access by multiple cluster nodes is coordinated by MC/LockManager's distributed lock manager (DLM).

Options and Arguments

vgchange recognizes the following options and arguments:

vg_name	The path name of a volume group.
-a *availability*	Set volume group availability. *availability* can have one of the following values:

y	Activate each specified volume group and all associated physical and logical volumes for read-write access. If a volume group is marked as part of a high availability cluster, it is activated in exclusive read-write mode, as for the **-a e** option.

e Activate each specified volume group and all associated physical and logical volumes for exclusive read-write access. The volume group must be marked as part of a high availability cluster, and the availability software must be running on the system; otherwise, the volume group is not activated.

s Activate each specified volume group and all associated physical and logical volume for shared read-write access. The volume group must be marked as part of a high availability cluster and marked sharable; otherwise, the volume group is not activated.

If any of the logical volumes in the volume group are mirrored, and if there are more than two systems in the high availability cluster, this volume group will not be activated because HP MirrorDisk/UX software is only supported in a clustered environment with a maximum of two nodes configured.

If the **-a y** or **-a e** option is executed on a currently active volume group, **vgchange** attempts to include any physical volumes that were previously listed as missing. This is useful if a physical volume has come back online. However, no automatic synchronization of any mirrored logical volumes is done. If synchronization is required, execute the **vgsync** command (see *vgsync*(1M)).

r Activate each specified volume group and all associated physical and logical volumes for read-only access. This option is ignored for a volume group that is already activated.

If a volume group is marked as part of a high availability cluster, the high availability software must be running on the system; otherwise, the volume group is not activated.

n Deactivate each specified volume group and its associated logical volumes. You must close the logical volumes prior to executing this option. For example, if the logical volume contains a file system, the file system must be unmounted.

-c *cluster* Control the membership of volume groups in a high availability cluster. *cluster* can have one of the following values:

y Mark each specified volume group as a member of the high availability cluster. The high availability software must be running; otherwise, the volume group is not marked. Needs to be done on one node only.

n Remove each specified volume group from membership in the high availability cluster. The high availability software does not need to be running.

The volume group must be deactivated with the **-a n** option before a **-c y**|**n** option can be executed.

-S *sharable* Control the sharability of volume groups in a high availability cluster. *sharable* can have one of the following values:

y Mark each specified volume group as sharable. The high availability software must be running; otherwise, the volume group is not marked. Needs to be done on one node only.

n Remove the shared attribute from the volume group. The high availability software does not need to be running.

The volume group must be deactivated with the **-a n** option before a **-S y**|**n** option can be executed.

-l Disable the opening of logical volumes that belong to each specified volume group. If the **-l** option is set, later attempts to open the logical volumes will fail. To allow an opening of these logical volumes to succeed, execute **lvchange -a y**.

-p Activate each specified volume group only if all of the physical volumes that belong to it are available.

-q *quorum* Set the quorum enforcement for each specified volume group. *quorum* can have one of the following values:

y Enforce the quorum requirement. This is the default.

n Ignore the quorum requirement.

The **-q n** option can be used to activate the volume group when the disk quorum is not maintained because too many disks were lost. Since it ensures the integrity of the LVM configuration information, it is normally not advisable to override the quorum.

-s Disable the synchronization of stale physical extents within the volume group specified by *vg_name*. This option is only effective when used with the **-a y** or **-a e** option.

-P *resync_daemon_count*
gives the advisory count to control the number of *nomwcsyncd* processes on volume group activation.

Mirrored Disk Activation

When the optional HP MirrorDisk/UX software is running and a volume group is activated, LVM performs the necessary mirror consistency recovery for each logical volume in the volume group based on the state of Mirror Write Cache and Mirror Consistency Recovery (see the Consistency Recovery section of *lvdisplay*(1M)). In a non-shared environment, LVM supports **MWC**, **NOMWC** and the **NONE** recovery. But in shared environment, LVM only supports **NOMWC** and the **NONE** recovery.

MWC Recover mirror consistency by using the Mirror Write Cache and Mirror Consistency Record. This mode implies that the Mirror Write Cache is on.

NOMWC Recover mirror consistency by searching all logical extents and copying data from a non-stale copy to the other mirror copies. This mode implies that the Mirror Write Cache is off.

NONE Do not recover mirror consistency during volume group activation on this logical volume. This mode implies that the Mirror Write Cache is off.

Next, mirror synchronization refreshes stale mirror copies by copying data from a nonstale copy. If the **-s** option is specified on the command line, mirror synchronization does not occur. However, for those logical volumes that have Mirror Write Cache turned off, mirror synchronization is done independently of whether the **-s** option appears on the command line.

General Activation

If **vgchange** cannot access a physical volume, it lists the volume's status as missing. If too many physical volumes in the volume group are missing, **vgchange** reports that the group does not have a quorum and cannot be activated. The lack of a quorum can be overridden with the **-q n** option.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Activate volume group **/dev/vg03**:

```
vgchange -a y /dev/vg03
```

Deactivate volume group **/dev/vg03**:

```
vgchange -a n /dev/vg03
```

Activate volume group **/dev/vg03** without synchronizing extents that are not current on logical volumes that have Mirror Write Cache turned on:

```
vgchange -a y -s /dev/vg03
```

Exclusive Activation

Set up volume group **/dev/vg03** for use in a high availability cluster:

```
vgchange -a n /dev/vg03   # Deactivate volume group
vgchange -c y /dev/vg03   # Enable volume group for HA cluster
vgchange -c y -S y /dev/vg03 # Enable volume group for HA cluster
                               and mark as sharable
vgchange -a e /dev/vg03   # Activate volume group in exclusive mode
vgchange -a s /dev/vg03   # Activate volume group in shared mode
```

Activate all volume groups; activate those that are marked for membership in a high availability cluster in exclusive mode:

```
vgchange -a y
```

Activate all volumes that are marked for membership in a high availability cluster in exclusive mode:

```
vgchange -a e
```

WARNINGS

Ordinary Operation

In ordinary operation (i.e., without the optional high availability software), it is possible to activate a volume group for read-write access from more than one physically connected system, leading to a high potential for data corruption. Therefore, if access is desired from more than one system to a single volume group, it is important that only one system activate the volume group for read-write access; the other systems can use read-only access. There is no problem if all systems activate the volume group for read-only access.

Furthermore, volume group information is only read from the disks during volume group activation. Dynamic changes to the volume group such as the following are not propagated to other systems sharing the volume group:

Logical volume configuration changes.

Changes to the status of the mirrored extents.

Bad-block relocation that occurs during write operations.

Because of these limitations, when sharing volume groups between systems it is recommended that logical volumes be accessed only by one system at a time. If logical volumes need to be accessed simultaneously, the logical volumes should not be mirrored and should not have bad-block relocation turned on, or all systems should use read-only access to the logical volumes.

SEE ALSO

mount(1M), vgcreate(1M), vgextend(1M), vgreduce(1M), vgdisplay(1M).

If MC/ServiceGuard is installed: cmcheckconf(1M), cmquerycl(1M), and *Managing MC/ServiceGuard*.

NAME

vgcreate - create LVM volume group

SYNOPSIS

/usr/sbin/vgcreate [**-A** *autobackup*] [**-x** *extensibility*] [**-e** *max_pe*] [**-l** *max_lv*] [**-p** *max_pv*] [**-s** *pe_size*] [**-g** *pvg_name*] *vg_name pv_path* ...

DESCRIPTION

The **vgcreate** command creates a new volume group. *vg_name* is a symbolic name for the volume group and must be used in all references to it. *vg_name* is the path to a directory entry under **/dev** which must contain a character special file named **group**. Except for the **group** entry, the directory *vg_name* should be empty. The *vg_name* directory and the **group** file have to be created by the user (see *lvm*(7)).

vgcreate leaves the volume group in an active state.

Before assigning a physical volume to a volume group, the physical volume has to be created using the **pvcreate** command (see *pvcreate*(1M)).

If **vgcreate** fails to install the first specified physical volume into the volume group, the volume group is not created. If, for any reason, one of the remaining specified physical volumes cannot be installed into the volume group, an error message is printed, but the installation continues until the end of the list of physical volumes.

Options and Arguments

vgcreate recognizes the following options and arguments:

pv_path The block device path name of a physical volume that will be assigned to the new volume group. You can specify physical volume links *(pv-links)* for a physical volume providing different paths that reference the same physical volume in the *pv_path* list. The order in which the paths are listed is important. The first path becomes the **primary link** to the physical volume, the second becomes an **alternate link** to the physical volume. The **primary link** is the default path used to access the physical volume. If the **primary link** becomes unavailable, LVM automatically switches to the **alternate link** to access the physical volume.

vg_name The path name of a subdirectory of the **/dev** directory. *vg_name* must be empty except for a character special file named **group**. Typically, this directory name is in the form **/dev/vg***NN*, where *NN* numbers sequentially from **00**.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the volume group. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group.

- **n** Do not back up configuration changes this time.

-e *max_pe* Set the maximum number of physical extents that can be allocated from any of the physical volumes in the volume group. The default value for *max_pe* is **1016**. However, if the size of any physical volume exceeds **1016** times the *pe_size*, the default value for *max_pe* is adjusted to match the physical volume size. The maximum number of physical extents can be a value in the range 1 to 65535.

-g *pvg_name* Create a new physical volume group with the name *pvg_name*. All physical volumes specified in the *pv_path* parameter become a member of the newly created physical volume group.

The physical volume group information is stored in an ASCII file, **/etc/lvmpvg**. The file can be edited to create a physical volume group instead of using the **vgcreate** command. However, ensure that the physical volumes to be used have already been installed in the volume group prior to creating the physical volume group.

The physical volume group name must be unique within a volume group although identical physical volume group names can appear in different volume groups (see *lvmpvg*(4) for format details).

-l *max_lv* — Set the maximum number of logical volumes that the volume group is allowed to contain. The default value for *max_lv* is **255**. The maximum number of logical volumes can be a value in the range 1 to 255.

-p *max_pv* — Set the maximum number of physical volumes that the volume group is allowed to contain. The default value for *max_pv* is **16**. The maximum number of physical volumes can be a value in the range 1 to 255.

-s *pe_size* — Sets the number of megabytes in each physical extent, where *pe_size* is expressed in units of megabytes (MB) in the range 1 to 256. *pe_size* must be equal to a power of 2 (1, 2, 4, 8, etc.). The default value for *pe_size* is **4** (four megabytes).

-x *extensibility* — Set the allocation permission for adding physical extents on the physical volumes specified by the *pv_path* parameter. *extensibility* can have one of the following values:

- **y** Allow allocation of additional physical extents on the physical volume. This is the default.
- **n** Prohibit allocation of additional physical extents on the physical volume. Logical volumes residing on the physical volume can still be accessed after the volume group has been activated by the **vgchange -a y** command.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Create a volume group named **/dev/vg00** containing two physical volumes with extent size set to 2 MB, from scratch.

First, create the directory **/dev/vg00** with the character special file called **group**.

```
mkdir /dev/vg00
mknod /dev/vg00/group c 64 0x030000
```

The minor number for the **group** file should be unique among all the volume groups on the system. It has the format **0x***NN***0000**, where *NN* runs from **00** to **09**. The maximum value of *NN* is controlled by the kernel tunable parameter **maxvgs**.

Initialize the disks using *pvcreate*(1M).

```
pvcreate /dev/rdsk/c1t0d0
pvcreate /dev/rdsk/c1t2d0
```

Create the volume group.

```
vgcreate -s 2 /dev/vg00 /dev/dsk/c1t0d0 /dev/dsk/c1t2d0
```

Create a volume group named **/dev/vg01** that can contain a maximum of three logical volumes, with extent size set to 8 MB:

```
vgcreate -l 3 -s 8 /dev/vg01 /dev/dsk/c3t4d0
```

Create a volume group named **/dev/vg00** and a physical volume group named **PVG0** with two physical volumes:

```
vgcreate -g PVG0 /dev/vg00 /dev/dsk/c1t0d0 /dev/dsk/c2t0d0
```

Using the **PV Links** feature to create a volume group named **/dev/vg00** with a physical volume which can be referenced by two different paths. **/dev/dsk/c3t0d0** and **/dev/dsk/c4t0d0** refer to the same physical volume, accessed via different controller hardware paths. In this example, **/dev/dsk/c3t0d0** becomes the **primary link** to the physical volume. **/dev/dsk/c4t0d0**

V

becomes an **alternate link** to the physical volume.

```
vgcreate /dev/vg00 /dev/dsk/c3t0d0 /dev/dsk/c4t0d0
```

WARNINGS

It is not possible to create a volume group that contains both HP-IB devices and devices using another type of interface.

SEE ALSO

pvcreate(1M), vgchange(1M), vgdisplay(1M), vgextend(1M), vgreduce(1M), lvm(7).

NAME

vgdisplay - display information about LVM volume groups

SYNOPSIS

/usr/sbin/vgdisplay [**-v**] [*vg_name* ...]

DESCRIPTION

The **vgdisplay** command displays information about volume groups. For each *vg_name* specified, **vgdisplay** displays information for that volume group only. If no *vg_name* is specified, **vgdisplay** displays names and corresponding information for all defined volume groups.

The volume group must be activated (see *vgchange*(1M)) before it can be displayed.

Options and Arguments

vgdisplay recognizes the following options and arguments:

vg_name
The path name of the volume group, for example, **/dev/vg00**.

-v For each volume group, display additional information about logical volumes, physical volumes, and physical volume groups.

Display Without –v Option

If you omit the **-v** option, only the following information is displayed:

--- Volume groups ---

VG Name	The path name of the volume group.
VG Write Access	Current access mode of the volume group. Possible values are **read/write** and **read-only**.
VG Status	State of the volume group: always **available**, as after a **vgchange -a y** command, since deactivated volume groups are not displayed.
Max LV	Maximum number of logical volumes allowed in the volume group.
Cur LV	Current number of logical volumes in the volume group.
Open LV	Number of logical volumes currently open in the volume group.
Max PV	Maximum number of physical volumes allowed in the volume group.
Cur PV	Current number of physical volumes in the volume group.
Act PV	Number of physical volumes that are currently active.
Max PE per PV	Maximum number (limit) of physical extents that can be allocated from any of the physical volumes in the volume group.
VGDA	Number of Volume Group Descriptor Areas within the volume group.
PE Size	Size of each physical extent.
Total PE	Total number of physical extents within the volume group: the sum of the number of physical extents belonging to each available physical volume in the volume group. (This does not include physical extents belonging to stand-by spare physical volumes; presence of these is only possible if you are using mirrored disks -- see below).
Alloc PE	Number of physical extents currently allocated to logical volumes.
Free PE	Number of physical extents not allocated (not including physical extents belonging to stand-by spares).
Total PVG	Total number of physical volume groups within the volume group.
Total Spare PVs	Total number of physical volumes that are designated as spares for this volume group. This will include both stand-by and active spares -- see below.

Total Spare PVs in use
Total number of spare physical volumes that are active in place of (containing all data from) a failed physical volume.

Display With –v Option

If you specify the **-v** option, **vgdisplay** lists the following additional information for each logical volume, for each physical volume, and for each physical volume group in the volume group:

--- Logical volumes ---

Information about logical volumes belonging to *vg_name*:

LV Name The block device path name of a logical volume in the volume group.

LV Status State of the logical volume:

- **available/stale** Logical volume available but contains physical extents that are not current.
- **available/syncd** Logical volume available and synchronized.
- **available** Logical volume available; **stale**/ **syncd** state cannot be confidently determined because logical volumes have both the Mirror Write Cache and Mirror Consistency Recovery turned off.
- **unavailable** Logical volume is not available for use.

LV Size (Mbytes)
Size of the logical volume.

Current LE Number of logical extents in the logical volume.

Allocated PE Number of physical extents used by the logical volume.

Used PV Number of physical volumes used by the logical volume.

--- Physical volumes ---

Information about physical volumes belonging to *vg_name*:

PV Name The block device path name of a physical volume in the group. When an alternate link to a physical volume has been added, **Alternate Link** is displayed next to the device path name. (See *vgextend*(1M) for definition.)

PV Status State of the physical volume: (*NOTE*: spare physical volumes are only relevant if you have installed HP MirrorDisk/UX software):

- **available** The physical volume is available and is not a spare physical volume.
- **available/data spared**
  The physical volume is available. However, it's data still resides on an active spare.
- **available/active spare**
  The physical volume is available and is an active spare physical volume. (An active spare is a spare that has taken over for a failed physical volume.)
- **available/standby spare**
  The physical volume is a spare "standing by" in case of a failure on any other physical volume in this volume group. It can only be used to capture data from a failed physical volume.
- **unavailable** The physical volume is unavailable and is not a spare physical volume.
- **unavailable/data spared**
  The physical volume is unavailable. However, it's data now resides on an active spare, and its data is available if the active spare is available.

unavailable/active spare
The physical volume is unavailable and it's an active spare. Thus, the data on this physical volume is unavailable.

unavailable/standby spare
The physical volume is a spare "standing by" that is not currently available to capture data from a failed physical volume.

Total PE Total number of physical extents on the physical volume.

Free PE Number of free physical extents on the physical volume.

Spared from PV
If the physical volume represents an active spare, this field will show the name of the failed physical volume whose data now resides on this spare. This information can be used to manually move the data back to the original physical volume once it has been repaired (see *pvmove*(1M)). If it cannot be determined which physical volume that the data came from, this field will instead display **Missing PV**. A missing PV would indicate that when the volume group was last activated or reactivated (see *vgchange*(1M)), the "failed" physical volume was not able to attach to the volume group.

Spared to PV If the physical volume represents a failed physical volume, this field will show the name of the active spare physical volume that now contains the data that originally residing on this volume. This information can be used to manually move the data back to the original physical volume (see *pvmove*(1M)) once it has been repaired.

--- Physical volume groups ---

Information about physical volume groups belonging to *vg_name*:

PVG Name Name of a physical volume group in the volume group.

PV Name The block device path name of a physical volume in the physical volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Display information about all the volume groups within the system:

```
vgdisplay
```

Display all of the information about one volume group, including the characteristics and status of both the logical and physical extents of the volume group:

```
vgdisplay -v /dev/vg02
```

SEE ALSO

lvdisplay(1M), pvdisplay(1M), vgchange(1M), vgcreate(1M).

NAME

vgexport - export an LVM volume group and its associated logical volumes

SYNOPSIS

/usr/sbin/vgexport [**-m** *mapfile*] [**-p**] [**-v**] *vg_name*

/usr/sbin/vgexport -m *mapfile* **-s -p -v** *vg_name*

DESCRIPTION

Using the format of the first command line of the *SYNOPSIS* above, the **vgexport** command can be used to remove a volume group from the system. The volume group will be removed without modifying the logical volume information found on the physical volumes.

The volume group identified by *vg_name* is removed from the **/etc/lvmtab** file, and the associated device files including the *vg_name* directory and **group** file are removed from the system.

The volume group information and data is untouched on the physical volume. These disks can be imported to other system with the **vgimport** command (see *vgimport*(1M)).

Sharable Option, Series 800 Only

Using the format of the second command line of the *SYNOPSIS* above, the **vgexport** command generates a *mapfile* that can be copied to other systems that are part of a high availability cluster and the **vgimport** command (see *vgimport*(1M)) can be used to recreate the volume group. See also *vgchange*(1M). The *mapfile* contains a description of the volume group and its associated logical volume(s) (if any). The logical volume information found on the physical volumes is not modified. Note that with this option, the volume group is not removed from the system. (See the second example below).

The volume group specified in the *mapfile* can be shared with the importing systems. The volume group is not removed from the exporting system.

Options and Arguments

vgexport recognizes the following options and arguments:

vg_name The path name of the volume group.

-m *mapfile* By default, a file named **mapfile** is created in the current directory. This file contains a description of the volume group and its associated logical volume(s) (if any). Use this option to specify a different name for the file, *mapfile*. This file can be used as input to **vgimport** (see *vgimport*(1M)). When used with the **-s** option, the volume group specified in the *mapfile* can be shared with other systems in the high availability cluster.

-p Preview the actions to be taken but do not update the **/etc/lvmtab** file or remove the devices file. This option is best used in conjunction with the **-v** option.

-v Print verbose messages including the names of the physical volumes associated with this volume group.

-s Sharable option, Series 800 only. When the **-s** option is specified, then the **-p**, **-v**, and **-m** options must also be specified. A *mapfile* is created that can be used to create volume group entries on other systems in the high availability cluster (with the **vgimport** command).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Export the volume group **/dev/vg01** into *mapfile* **vg01.mymap**. The volume group will be removed from the exporting system.

```
vgexport -m  vg01.mymap /dev/vg01
```

Create a *mapfile* to be copied to other systems in a high availability cluster to build the volume group information for the volume group, **/dev/vg02**. Note that the volume group is not removed from the exporting system. The importing systems will create the volume group with the **vgimport** command using the **-s** and **-m** options.

```
    vgexport -s -p -m -v  vg02.mymap /dev/vg02
```

SEE ALSO

vgimport(1M), vgscan(1M).

NAME

vgextend - extend an LVM volume group by adding physical volumes

SYNOPSIS

/usr/sbin/vgextend [**-A** *autobackup*] [**-g** *pvg_name*] [**-x** *extensibility*] [**-z** *sparepv*] *vg_name* *pv_path* ...

Remarks

vgextend cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **vgextend** command assigns additional physical volumes to volume group *vg_name*. The volume group must be active.

Volume groups are extended by adding one or more physical volumes specified by *pv_path* ...

After the physical volumes have been successfully added to the volume group, the disk space they contain can be allocated to logical volumes.

Before assigning an additional physical volume to a volume group, create the physical volume with the **pvcreate** command (see *pvcreate*(1M)). Then, create the volume group with the **vgcreate** command, assigning at least one physical volume (see *vgcreate*(1M)).

If, for any reason, a specified physical volume cannot be installed into the volume group, an error message is displayed. However, the installation continues to the end of the list of physical volumes.

When a *pv_path* refers to one of the physical volumes already in the volume group by a different *pv_path* name to indicate the use of a different controller, this new path becomes an **alternate link** to the physical volume. When two paths that reference the same disk are provided in the *pv_path* list, the order of the paths is important. The first path becomes the "primary link" to the physical volume, the second becomes an "alternate link" to the physical volume. The primary link is the path used to access the physical volume unless the primary link becomes unavailable in which case LVM automatically switches to the alternate link to access the physical volume.

Options and Arguments

vgextend recognizes the following options and arguments:

pv_path The block device path name of a physical volume.

vg_name The path name of the volume group.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the volume group. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group.

- **n** Do not back up configuration changes this time.

-g *pvg_name* Extend an existing physical volume group while the volume group is being extended by adding all the physical volumes in the *pv_path* parameter to the physical volume group specified by *pvg_name*.

If the specified physical volume group does not exist, it is created, thus providing a means for creating new physical volume groups after the volume group has been created. Another way to extend or add a physical volume group is to edit the **/etc/lvmpvg** file as described in *vgcreate*(1M). See *lvmpvg*(4) for format details.

-x *extensibility* Set allocation permission for additional physical extents on the physical volume specified by *pv_path*. *extensibility* can have one of the following values:

- **y** Allow allocation of additional physical extents on the physical volume.
- **n** Prohibit allocation of additional physical extents on the physical volume. Logical volumes residing on the physical volume can still be accessed.

-z *sparepv* This option requires the installation of the optional HP MirrorDisk/UX software. It allows you to mark the physical volume(s) specified by *pv_path* to be either a spare physical volume or a regular, non-spare physical volume. (A spare physical volume can be used to replace an existing physical volume within a volume group when mirroring is in effect, in the event the existing physical volume fails.) *sparepv* can have one of the following values:

- **y** The physical volume(s) will be used as spare(s). No physical extents from a spare physical volume will be available as part of the "free" pool of extents in the volume group. The spare physical volume(s) will only be used in the event of another physical volume within this volume group becomes unavailable (fails).
- **n** The physical volume(s) will be used as regular, non-spare members of the volume group. This is the default.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Add physical volumes **/dev/dsk/c0t1d0** and **/dev/dsk/c0t2d0** to volume group **/dev/vg03**:

```
vgextend /dev/vg03 /dev/dsk/c0t1d0 /dev/dsk/c0t2d0
```

Extend physical volume group **PVG0** while adding physical volumes **/dev/dsk/c0t3d0** and **/dev/dsk/c0t4d0** to volume group **/dev/vg03**:

```
vgextend -g PVG0 /dev/vg03 /dev/dsk/c0t3d0 /dev/dsk/c0t4d0
```

Add an alternate link to one of the physical volumes in the volume group where **/dev/dsk/c0t4d0** and **/dev/dsk/c1t4d0** refer to the same physical volume (referenced via different controllers), and the volume group already contains **/dev/dsk/c0t4d0**. **/dev/dsk/c0t4d0** remains the primary link (in use) and **/dev/dsk/c1t4d0** becomes the alternate link.

```
vgextend  /dev/vg03 /dev/dsk/c1t4d0
```

Add a spare physical volume to a volume group:

```
vgextend  -z y /dev/vg03 /dev/dsk/c2t4d0
```

WARNINGS

It is not possible to extend a volume group such that it contains both HP-IB devices and devices that use another type of interface.

The new physical volume which has been added to the volume group could potentially have a different block size compared to physical volumes already in the volume group.

If a logical volume is created on two or more physical volumes which have a different block size, it is not possible to use such logical volume for file system purposes. See *extendfs*(1M).

For example, when a logical volume contains physical volumes that all have 1k block size, and then it is extended to contain a physical volume with 2k block size, then the block size of the volume group is increased to 2k.

SEE ALSO

pvchange(1M), pvcreate(1M), vgchange(1M), vgcreate(1M), vgdisplay(1M).

NAME

vgimport - import an LVM volume group onto the system

SYNOPSIS

/usr/sbin/vgimport [**-m** *mapfile*] [**-p**] [**-v**] *vg_name pv_path* ...

/usr/sbin/vgimport -m *mapfile* **-s -v** *vg_name*

DESCRIPTION

The **vgimport** command adds the specified volume group to the system. The physical volumes, specified as *pv_path* ..., are scanned to obtain the volume group information and logical volume information. This command works much like **vgcreate** by requiring that the volume group device directory and the **group** special file be created before the command is executed (see *vgcreate*(1M)). The *vg_name* is added from the **/etc/lvmtab** file, and the associated logical volume device files are added to the system.

vgimport assumes that the volume group information has already been created on the physical volumes.

This command is useful in conjunction with the **vgexport** command (see *vgexport*(1M)), to move volume groups from one system to other systems within a high availability cluster.

vgimport creates logical volume devices files under the *vg_name* directory using the naming convention given in *mapfile* or using the default naming convention used by the **lvcreate** command (see *lvcreate*(1M)).

Sharable Option, Series 800 Only

In the second format of the command line shown in **SYNOPSIS**, the volume group specified in the *mapfile* is shared.

vgimport does not activate the imported volume group due to the many possible options at volume group activation time. To activate the volume group once it has been successfully imported, use the **vgchange** command (see *vgchange*(1M)).

Options and Arguments

vgimport recognizes the following options and arguments:

pv_path	The block device path names of a physical volume. This argument is not used with the **-s** and related options.
vg_name	The path name of the volume group.
-m *mapfile*	Specify the name of the file from which logical volume names and numbers are to be read. This option is optional when used as in the first command line format of the *SYNOPSIS*. If this option is not specified, logical volume names are created using the default naming convention **lvol***nn* where *nn* is the logical volume minor number. When used with the **-s** option, the volume group specified in the *mapfile* can be shared among the exporting system and the importing systems.
-s	Sharable option, Series 800 only. When the **-s** option is specified, then the **-p**, **-v**, and **-m** options must also be specified. The specified *mapfile* is the same *mapfile* specified by using the **vgexport** command also using the **-p**, **-m**, and **-s** options. The *mapfile* is used to create the volume groups on the importing systems.
-p	Preview the actions to be taken but do not update the **/etc/lvmtab** file or add the devices file. This option is best used in conjunction with the **-v** option.
-v	Print verbose messages including names of the logical volumes.

WARNINGS

The following warnings should only apply to the **-s** option when importing devices such as (NIKE) or disks with alternate path:

Since the **-s** option causes a search on the system for each disks with the same **vg_id.** When **vgimport** reconstruct the newly imported volume group entry in **/etc/lvmtab** file, the order of disks could be different than it was before. And the following will happen:

The designated primary and alternate link might not be the same as it was configured before.

Alternate links will be added to the importing volume group even if they might not be configured in the volume group initially.

If the original primary path of a disks become an alternate path after the newly imported volume group entry is created in **/etc/lvmtab,** the order can be easily reverted by using **vgreduce** to remove the primary path and then use **vgextend** to add the path back again.

If additional alternate paths were added to the newly imported volume group, use **vgreduce** to reduce any alternate paths that were added but they were not needed.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Import the volume group **/dev/vg01** that is located on physical disks **/dev/dsk/c0t1d0** and **/dev/dsk/c0t3d0**:

```
vgimport -v /dev/vg01 /dev/dsk/c0t1d0 /dev/dsk/c0t3d0
```

Activate the volume group following a successful import:

```
vgchange -a y vg01
```

Import the volume group **/dev/vg01** using the *mapfile*, **/tmp/vg01.mymap**. **mymap** was previously specified by the **vgexport** command on another system. The volume group, **/dev/vg01**, is specified in **mymap** and will be used by the importing system only:

```
vgimport -v -m /tmp/vg01.mapfile /dev/vg01 \
/dev/dsk/c0t5d0 /dev/dsk/c0t7d0
```

Import the volume group **/dev/vg02** using the *mapfile*, **/tmp/vg02.mymap**. **mymap** was previously specified by the **vgexport** command on another system. The volume group, **/dev/vg02**, is specified in **mymap** and will be shared among the exporting system, this system, and other systems importing the volume group as shared:

```
vgimport -v -s -m /tmp/vg02.mymap dev/vg02
```

SEE ALSO

vgexport(1M), vgscan(1M).

NAME

vgreduce - remove physical volumes from an LVM volume group

SYNOPSIS

/usr/sbin/vgreduce *vg_name pv_path* ...

/usr/sbin/vgreduce -f *vg_name*

Remarks

vgreduce cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **vgreduce** command removes each physical volume specified by a *pv_path* argument from volume group *vg_name*.

The **vgreduce** command with -f option removes all missing physical volume from the volume group.

All but one physical volume can be removed. The last physical volume must remain in the volume group so that the logical volume driver can continue to operate. The last physical volume in the volume group can be removed with the **vgremove** command (see *vgremove*(1M)).

Before executing **vgreduce**, remove all logical volumes residing on each physical volume represented by a *pv_path* by executing **lvremove** (see *lvremove*(1M)).

Any physical volume in the *pv_path* list that is also a member of a physical volume group (as defined in **/etc/lvmpvg**) is also removed from the physical volume group. If the physical volume happens to be the last one in the physical volume group, the physical volume group is also removed from the volume group.

When a physical volume in the *pv_path* list has multiple **PV-links**, the physical volume is *not* removed from the volume group, until all the links to the volume are removed. When a physical volume in the *pv_path* list is the **primary link** (in use) to a physical volume, removing the link forces LVM to switch to the **alternate link** to access the physical volume. When the *pv_path* removed is an **alternate link** to the device, only the link is removed; the volume group and physical volume are otherwise unchanged.

Options and Arguments

vgreduce recognizes the following option and arguments:

-f *vg_name*	force reduction of missing physical volume(s) in a given volume group. **vgreduce** obtains the name of each physical volume (PV) belonging to the volume group from the file **/etc/lvmtab**. It then reads the LVM structures from each PV and compares these with that held by the kernel to work out which PVs are missing. PVs which are missing will be candidates for removal. If all the physical extents on the missing PV are free then it will be removed from the volume group. Otherwise **vgreduce** will report the physical to logical extent mapping. For missing PVs which have extents in use you must free up all the extents by using *lvreduce*(1M) and re-run **vgreduce** with the **-f** option. This option is most commonly used when the *vgdisplay*(1M) command shows "Cur PV" higher than "Act PV" and all of the PVs belonging to the volume group are attached.
pv_path	The block device path name of a physical volume.
vg_name	The path name of the volume group.

V

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Remove physical volume **/dev/dsk/c0t1d0** from volume group **/dev/vg01**:

```
    vgreduce /dev/vg01 /dev/dsk/c0t1d0
```

Force reduction of missing PVs from volume group: **vg01**

```
vgreduce -f /dev/vg01
```

The following messages will appear after missing PVS has been removed successfully:

> PV with key 0 sucessfully deleted from vg **/dev/vg01**
>
> Repair done, please do the following steps.....:
>
> 1. Save **/etc/lvmtab** to another file.
> 2. Remove **/etc/lvmtab**.
> 3. Use **vgscan -v** to recreate **/etc/lvmtab**.
> 4. NOW use *vgcfgbackup*(1M) to save the LVM setup.

SEE ALSO

vgchange(1M), vgcreate(1M), vgdisplay(1M), vgextend(1M).

NAME

vgremove - remove LVM volume group definition from the system

SYNOPSIS

/usr/sbin/vgremove *vg_name* ...

DESCRIPTION

The **vgremove** command removes from the system the last physical volume of the volume group and the definition of the volume group or groups specified by *vg_name* .... Since all system knowledge of the volume group and its contents are removed, the volume group can no longer be accessed.

To move a volume group from one system to another, use the **vgexport** command instead (see *vgexport*(1M)).

Before executing **vgremove**, remove all logical volumes residing on the last physical volume by executing **lvremove** (see *lvremove*(1M)).

vgremove is equivalent to the inverse of executing **vgcreate** for one physical volume (see *vgcreate*(1M)).

Before removing a volume group, two steps are necessary:

1. Remove all the logical volumes belonging to the group by using the **lvremove** command (see *lvremove*(1M)).
2. Remove all but one physical volume belonging to the volume group by using the **vgreduce** command (see *vgreduce*(1M)).

If there is any physical volume group created under *vg_name* ..., the physical volume group information is also removed from file **/etc/lvmpvg**.

Arguments

vgremove recognizes the following argument:

vg_name	The path name of a volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Remove volume group **/dev/vg02** from the system:

```
vgremove /dev/vg02
```

SEE ALSO

lvremove(1M), vgchange(1M), vgreduce(1M).

V

NAME

vgscan - scan physical volumes for LVM volume groups

SYNOPSIS

/usr/sbin/vgscan [**-a**] [**-p**] [**-v**]

DESCRIPTION

The **vgscan** command allows the re-creation of the **/etc/lvmtab** file and possibly the associated volume group device files. This command should be run only in the event of a catastrophic error such as the deletion of the **/etc/lvmtab** file or the mismatch of names of the physical volumes in the **/etc/lvmtab** file to the actual physical volume path configuration. If the **/etc/lvmtab** file exists, the information contained in the file is used to assist in rebuilding the file, but the existing file is updated with the new corrected configuration.

vgscan searches each physical volume connected to the system, looking for logical volumes. If there are dual controller devices, only the primary controller device path is scanned, unless you specify the **-a** option to allow access to all paths. It groups these physical volumes into volume groups by matching the volume group information found on the physical volumes. Then it searches the **/dev** directory for all **group** device files with the LVM major number, and tries to match device files with the logical volumes' information found on the physical volumes.

If matches occur, it determines the volume group name from the device file path, and updates the **/etc/lvmtab** file with the volume group name and the list of physical volumes paths contained in that volume group. For volume groups where the device files cannot be matched, it prints the list of physical volumes for each volume group.

After **vgscan** completes successfully, run the **vgimport** command on each set of unmatched physical volumes (see *vgimport*(1M)).

Options

vgscan recognizes the following options:

-a Scan all controller device paths for all disks.

-p Preview the actions that would be taken but do not update file **/etc/lvmtab**. This option is best used in conjunction with the **-v** option.

-v Print verbose messages.

WARNINGS

The following warning only applies to dual controller devices (NIKE), or disks with alternate path:

Since **vgscan** search each disks on the system in the order of where they have configured. When **vgscan** reconstruct **/etc/lvmtab** file, the order of disks in the file could be different than it was before. The following will happen:

The designated primary and alternate link might not be the same as it was configured before.

Alternate links will be added to the **/etc/lvmtab** file even if they might not be configured in the volume group initially.

The boot information might be incorrect due to different order of disks in the new **/etc/lvmtab** file.

In order to correct the above problems, do the following:

Use **vgchange** with **-a** option to activate all volume groups.

Use **lvlnboot** with -R option to correct boot information on disk.

Use **vgreduce** to reduce any alternate links that were added to the **/etc/lvmtab** file by **vgscan,** but they were not needed.

If the original primary path of a disks become an alternate path after **/etc/lvmtab** file is reconstructed, the order can be easily reverted by using **vgreduce** to remove the primary path and use **vgextend** to add the path back again.

If **/etc/lvmtab** is destroyed, do not use **vgscan** to re-construct **/etc/lvmtab** if the system is heavily loaded by an application. Otherwise, **vgscan** will create an incomplete **/etc/lvmtab** due to a known NIKE/LVM limitation issue. It's important to quiesce the logical volume's I/O before re-constructing the **/etc/lvmtab**.

If for some reason, there is a need to re-construct **/etc/lvmtab** when the system is running production application, **vgscan** will create a partial **/etc/lvmtab**. In this case, most of the primary paths should be included in the **/etc/lvmtab**. Use **vgextend** to include any missing alternate paths in the VG.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Scan the primary controller device paths for all the physical volumes on the system, but do not update the **/etc/lvmtab** file:

```
vgscan -p -v
```

Scan all controller device paths to all physical volumes on the system:

```
vgscan -a -v
```

Scan the primary controller device paths for all the physical volumes on the system and re-create **/etc/lvmtab** file:

```
vgscan -v
```

The following messages will appear after **/etc/lvmtab** file is re-created.

```
*** LVMTAB has been created successfully.
*** If PV links are configured in the system
*** Do the following to resync information on disk.
*** #1.  vgchange -a y
*** #2.  lvlnboot -R
```

SEE ALSO

vgexport(1M), vgimport(1M).

(Requires Optional HP MirrorDisk/UX Software)

NAME

vgsync - synchronize stale logical volume mirrors in LVM volume groups

SYNOPSIS

/usr/sbin/vgsync *vg_name* ...

Remarks

This command requires the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

DESCRIPTION

The **vgsync** command synchronizes the physical extents of each mirrored logical volume in the volume group specified by *vg_name* .... Synchronization occurs only on the physical extents that are stale mirrors of the original logical extent.

The synchronization process can be time consuming, depending on the hardware characteristics and the amount of data. Unless disabled, the mirrors within a volume group are synchronized automatically when the volume group is activated by the **vgchange -a y** command.

Arguments

vgsync recognizes the following argument:

vg_name	The path name of a volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Synchronize the mirrors on volume group **/dev/vg04**:

```
vgsync /dev/vg04
```

WARNINGS

It is not advisable to interrupt a **vgsync** process.

SEE ALSO

lvsync(1M), vgchange(1M), vgdisplay(1M).

NAME

vhe_altlog - login when Virtual Home Environment (VHE) home machine is not available

SYNOPSIS

/usr/sbin/vhe/vhe_altlog

DESCRIPTION

vhe_altlog is a shell script that permits a user to log in when the home machine is not accessible through Virtual Home Environment (VHE). This script is executed when a login using the user name of *altlogin* is completed. After the user logs in using the user login name **altlogin**, **vhe_altlog** asks for a user name and password. If these are valid, the user is logged in on the machine and the home directory is a temporary directory such as **/tmp**. This provides user access to other machines in the group of VHE nodes even though a home machine is not available.

A user entry for **altlogin** must be present in **/etc/passwd** for it to be a valid login name. A typical entry resembles the following:

```
altlogin::6:1::/tmp:/usr/sbin/vhe/vhe_altlog
```

DIAGNOSTICS

If an invalid user name or password is supplied, the attempted login is rejected.

AUTHOR

vhe_altlog was developed by HP.

FILES

/etc/passwd

SEE ALSO

vhe_mounter(1M), vhe_u_mnt(1M), vhe_list(4).

NAME

vhe_mounter - start the Virtual Home Environment (VHE)

SYNOPSIS

/usr/sbin/vhe/vhe_mounter

DESCRIPTION

vhe_mounter is a shell script that configures a machine to operate with the Virtual Home Environment (VHE). VHE enables users to have the same view of their execution environments when logging in on machines interconnected with VHE. Machines connected with VHE must also be running the Network File System (NFS).

Information needed by **vhe_mounter** is provided by file **/etc/vhe_list** which contains a list of host names included in the group of VHE machines.

DIAGNOSTICS

vhe_mounter always returns exit code **0**.

AUTHOR

vhe_mounter was developed by HP.

FILES

/etc/vhe_list

SEE ALSO

vhe_altlog(1M), vhe_u_mnt(1M), vhe_list(4).

NAME

vhe_u_mnt - perform Network File System (NFS) mount to remote file system

SYNOPSIS

/usr/sbin/vhe/vhe_u_mnt

DESCRIPTION

vhe_u_mnt enables a user to perform a Network File System (NFS) mount to a remote file system. **vhe_u_mnt** is executed upon completion of a login using user name **mounter**. After logging in as user **mounter**, **vhe_u_mnt** asks for the name of the machine to which an NFS mount is to be done. If that machine name is listed in file **/etc/vhe_list**, mounts that are valid for that machine are made. This prevents the command from giving a user the ability to do NFS mounts to arbitrary machines. File **/etc/vhe_list** contains a list of hostnames that are part of the VHE group.

User name **mounter** must be present in file **/etc/passwd** file for it to be a valid login name. A typical entry resembles:

```
mounter::6:1::/:/usr/sbin/vhe/vhe_u_mnt
```

DIAGNOSTICS

If a machine name is supplied that is not contained in **/etc/vhe_list**, an error message is produced indicating that the machine is not on the list of machines available for mounting.

AUTHOR

vhe_u_mnt was developed by HP.

FILES

/etc/passwd
/etc/vhe_list

SEE ALSO

vhe_altlog(1M), vhe_mounter(1M), vhe_list(4).

NAME

vipw - edit the password file

SYNOPSIS

vipw

DESCRIPTION

vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, you will be told to try again later. The **vi** editor is used unless the environment variable **EDITOR** indicates an alternate editor. **vipw** performs a number of consistency checks on the password entry for **root**, and does not allow a password file with an incorrectly formatted root entry to be installed.

WARNINGS

An **/etc/passwd.tmp** file not removed when a system crashes prevents further editing of the **/etc/passwd** file using **vipw** after the system is rebooted.

AUTHOR

vipw was developed by the University of California, Berkeley.

FILES

/etc/passwd.tmp

SEE ALSO

passwd(1), passwd(4).

NAME

volcopy, labelit (generic) - copy a file system with label checking

SYNOPSIS

/usr/sbin/volcopy [*options*] *fsname special1 volname1 special2 volname2*

/usr/sbin/labelit [*options*] *special* [*fsname volume* [**-n**]]

DESCRIPTION

The **volcopy** command makes a literal copy of the file system using a block size matched to the device.

Options

volcopy recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-a Invoke a verification sequence requiring a positive operator response instead of the standard delay before the copy is made.

-o *specific_options*
Specify options specific to the file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for an *FStype*-specific module of the command. See the file system specific manual entries for a description of the *specific_options* that are supported, if any.

-s (default) Invoke the DEL-if-wrong verification sequence.

-y Assume a **yes** response to all questions

-V Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and arguments with other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Other options are used with 9-track magnetic tapes:

-bpi *density*	Bits per inch.
-feet *size*	Size of reel in feet.
-reel *num*	Beginning reel number for a restarted copy.
-buf	Use double buffered I/O.

The **volcopy** command requests length and density information if they are not given on the command line and they are not recorded on an input tape label. If the file system is too large to fit on one reel, the **volcopy** command prompts for additional reels. Labels of all reels are checked. Tapes can be mounted alternately on two or more drives. If the **volcopy** command is interrupted, it asks if the user wants to quit or wants to escape to the command interpreter. In the latter case, other operations (such as executing the **labelit** command) can be performed before returning to the **volcopy** command by exiting the command interpreter.

fsname	The file system name on the device (e.g., **root**) being copied.
special	The physical disk section or tape (e.g., **/dev/rdsk/1s3** or **/dev/rmt/c0t0d0BEST**).
volname	The physical volume name; it should match the external sticker. Such label names are limited to six or fewer characters. The argument *volname* can be **-** to use the existing volume name.
special1	The device from which the copy of the file system is being extracted.
volname1	The volume from which the copy of the file system is being extracted.
special2	The target device.
volname2	The target volume.

The **labelit** command can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, the **labelit** command prints current label values. The **-n** option provides for initial labeling of new tapes only (this destroys previous contents). The **-F**, **-V**, and **-o** options can be specified for the **labelit** command. The behavior of the **-F**, **-V**, and **-o** options is similar to their behavior in the **volcopy** command.

FILES

/etc/default/fs	File that specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

volcopy_*FStype*(1M), fs_wrapper(5).

NAME

volcopy, labelit (hfs) - copy an HFS file system with label checking

SYNOPSIS

/usr/sbin/volcopy [*options*] *fsname special1 volname1 special2 volname2*

/usr/sbin/labelit [*options*] *special* [*fsname volume* [**-n**]]

DESCRIPTION

The **volcopy** command makes a literal copy of an HFS file system using a block size matched to the device.

Options

volcopy recognizes the following options:

-F hfs	Specifies the HFS file system type.
-a	Invoke a verification sequence requiring a positive operator response instead of the standard delay before the copy is made.
-s	(default) Invoke the DEL-if-wrong verification sequence.
-y	Assume a **yes** response to all questions
-V	Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and arguments with other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Other options are used with 9-track magnetic tapes:

-bpi *density*	Bits per inch.
-feet *size*	Size of reel in feet.
-reel *num*	Beginning reel number for a restarted copy.
-buf	Use double buffered I/O.

The **volcopy** command requests length and density information if they are not given on the command line and they are not recorded on an input tape label. If the file system is too large to fit on one reel, the **volcopy** command prompts for additional reels. Labels of all reels are checked. Tapes can be mounted alternately on two or more drives. If the **volcopy** command is interrupted, it asks if the user wants to quit or wants to escape to the command interpreter. In the latter case, other operations (such as executing the **labelit** command) can be performed before returning to the **volcopy** command by exiting the command interpreter.

fsname	The file system name on the device (e.g., **root**) being copied.
special	The physical disk section or tape (e.g., **/dev/rdsk/1s3** or **/dev/rmt/c0t0d0BEST**).
volname	The physical volume name; it should match the external sticker. Such label names are limited to six or fewer characters. The argument *volname* can be **-** to use the existing volume name.
special1	The device from which the copy of the file system is being extracted.
volname1	The volume from which the copy of the file system is being extracted.
special2	The target device.
volname2	The target volume.

The **labelit** command can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, the **labelit** command prints current label values. The **-n** option provides for initial labeling of new tapes only (this destroys previous contents). The **-F** and **-V** options can be specified for the **labelit** command. The behavior of the **-F** and **-V** options is similar to their behavior in the **volcopy** command.

SEE ALSO

fstyp(1M), volcopy(1M), fs_wrapper(5).

NAME

volcopy, labelit (vxfs) - copy a VxFS file system with label checking

SYNOPSIS

/usr/sbin/volcopy [**-F vxfs**] [**-V**] [**-a**] [**-s**] [**-y**]
fsname special1 volname1 special2 volname2

/usr/sbin/labelit [**-F vxfs**] [**-V**] [**-n**] *special* [*fsname volume*]

DESCRIPTION

The **volcopy** command makes a literal copy of a VxFS file system using a block size matched to the device.

Options

volcopy recognizes the following options and command-line arguments:

-F vxfs	Specify the file-system type (**vxfs**).
-V	Validate the command line options, however, the command is not executed. If the options specified are valid, the complete command line is echoed. If the options specified are not valid, an error message is printed.
-a	Normally, before copying the file system, **volcopy** delays so that the user can interrupt the copying if needed. This option invokes a verification sequence requiring a positive operator response instead of the standard delay.
-s	(default) Before copying the file system, prints a message and allows the user to interrupt the copy if needed.
-y	Assume a **yes** response to all questions

Other options are used with 9-track magnetic tapes:

-bpi *density*	Bits per inch.
-feet *size*	Size of reel in feet.
-reel *num*	Beginning reel number for a restarted copy.
-buf	Use double buffered I/O.

The **volcopy** command requests length and density information it is not given on the command line or if it is not recorded on an input tape label. If the file system is too large to fit on one reel, **volcopy** prompts for additional reels. Labels of all reels are checked. Tapes can be mounted alternately on two or more drives. If **volcopy** is interrupted, it asks if the user wants to quit or to escape to the command interpreter. In the later case, other operations (such as **labelit**) can be performed before returning to **volcopy** by exiting the command interpreter.

The *fsname* argument represents the file system name on the device (e.g., **root**) being copied.

special should be the physical disk section or tape (e.g., **/dev/rdsk/c0t4d0s0** or **/dev/rmt/0mb**).

volname is the physical volume name; it should match the external sticker. Such label names are limited to six or fewer characters. The argument *volname* can be **-** to use the existing volume name.

The arguments *special1* and *volname1* are the device and volume, respectively, from which the copy of the file system is being extracted. The arguments *special2* and *volname2* are the target device and volume, respectively.

The **labelit** command can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, **labelit** prints current label values. **-F** and **-V** options can be specified for **labelit**. The behavior of **-F** and **-V** options are similar to the behavior of **volcopy**. The option **-n** of **labelit** provides for initial labeling of new tapes (this destroys previous contents).

SEE ALSO

volcopy(1M), labelit(1M).

NAME

vtdaemon - respond to vt requests

SYNOPSIS

vtdaemon [**-g**[*ngateway*]] [**-n**] *lan_device lan_device* ...

DESCRIPTION

vtdaemon responds to requests from other systems (via local area network) made by **vt** (see *vt*(1)). **vtdaemon** spawns a server to respond to each request that it receives.

Options

vtdaemon recognizes the following command-line options and arguments:

-g[*ngateway*]
Causes **vtdaemon** to rebroadcast all requests received on one lan device to all other lan devices specified on the command line. The optional parameter *ngateway* specifies the maximum number of *vtgateway* servers that can be in operation concurrently. If *ngateway* is not specified, there is no limit on the number of vtgateway servers that can be in operation concurrently.

-n Causes *vtdaemon* to ignore all requests that have come through a gateway.

The remaining arguments are the full path names of lan devices that *vtdaemon* looks for requests on. If no lan devices are specified, the default lan device is used.

The major number for this device must correspond to a IEEE 802.3 local area network device.

Another function of **vtdaemon** is to create *portals* and service portal requests. A *portal* is a callout device that can be used by **uucico** to communicate to another machine via local area network (see *uucico*(1M)). Portals are created by **vtdaemon** according to the configuration information found in the file **/etc/uucp/L-vtdevices**. Each line in **L-vtdevices** has the format:

<calldev>[,<lan device>] <nodename>

For each line, **vtdaemon** creates a portal named *calldev* in **/dev**. Whenever this device is opened, **vtdaemon** spawns a server that creates a connection to the system specified by *nodename* via the lan device specified. If no lan device is specified, the first one specified on the command line when **vtdaemon** was started is used (or the default lan device is used if no lan devices were specified on the command line).

vtdaemon should be terminated by sending signal **SIGTERM** to it. When **vtdaemon** receives this signal it removes all of the portals it created in **/dev** before exiting.

DIAGNOSTICS

Diagnostics messages produced by **vtdaemon** are written to **/var/adm/vtdaemonlog**.

WARNINGS

vtdaemon uses the Hewlett-Packard **LLA** (Link Level Access) direct interface to the HP network drivers. **vtdaemon** uses the multicast address **0x01AABBCCBBAA**. It should not be used or deleted by other applications accessing the network. **vtdaemon** uses the following IEEE 802.3 *sap* (service access point) values: **0x90**, **0x94**, **0x98**, **0x9C**, **0xA0**, **0xA4**, **0xA8**, **0xAC**, **0xB0**, **0xB4**, **0xB8**, **0xBC**, **0xC0**, **0xC4**, **0xC8**, **0xCC**, **0xD0**, and **0xD4**. They should not be used by other applications accessing the network.

Desktop HP-UX

If your system has been installed with the Desktop HP-UX product, then both **ptydaemon** and **vtdaemon** will not be started by default. In order to start these daemons, change *PTYDAEMON_START* and *VTDAEMON_START* from a "0" to a "1" in the **/etc/rc.config.d/ptydaemon** and **/etc/rc.config.d/vt** files, respectively. The system must either be rebooted for these changes to take effect, or you can manually start both daemons by typing :

```
/usr/sbin/ptydaemon
/usr/sbin/vtdaemon /dev/lan0
```

where */dev/lan0* is the character special device file corresponding to the IEEE802.3 local area network device.

FILES

`/var/adm/vtdaemonlog`	logfile used by *vtdaemon*.
`/dev/lan0`	default lan device name.

SEE ALSO

vt(1), uucico(1M).

NAME

vxdiskusg (vxfs) - generate disk accounting data of VxFS file systems by user ID

SYNOPSIS

/usr/sbin/acct/vxdiskusg [*options*] [*file*...]

DESCRIPTION

vxdiskusg generates intermediate disk accounting information from data in *file*, or the standard input if the **-s** flag is specified and *file* is omitted. **vxdiskusg** outputs lines on the standard output, one per user, in the following format:

uid login #blocks

where:

uid	User's numerical user ID,
login	User's login name, and
#blocks	Total number of disk blocks allocated to this user.

Without the **-s** option, *file* ... is the special filename of device(s) containing file systems. **vxdiskusg** reads only the inodes of file systems for disk accounting.

Options

vxdiskusg recognizes the following options:

-s	Input data is already in **vxdiskusg** output format. **vxdiskusg** combines all lines for a single user into a single line.
-v	verbose. Print a list on standard error of all files that are charged to no one.
-i *ignlist*	Ignore the data on those file systems whose file system name is in *ignlist*. *ignlist* is a list of file system names, separated by commas or separated by spaces and enclosed within quotes. **vxdiskusg** compares each name in this list with the file system name stored in the *volume name* (see *volcopy_vxfs*(1M)), if it exists.
-p *file*	Use *file* as the name of the password file to generate login names. **/etc/passwd** is used by default.
-u *file*	Write records to *file* of files that are charged to no one. Records consist of the special file name, the inode number, and the user ID.

The output of **vxdiskusg** is normally the input to **acctdisk** (see *acct*(1M)) which generates total accounting records that can be merged with other accounting records. **vxdiskusg** is normally run in **dodisk** (see *acctsh*(1M)).

EXAMPLES

The following generates daily disk accounting information for the file systems on these disks:

```
for i in /dev/vg00/lvol1 /dev/vg00/lvol6 /dev/vg00/lvol7; do
    vxdiskusg $i > dtmp.‘basename $i‘ &
done
wait
vxdiskusg -s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

V

FILES

/etc/passwd	used for user-ID-to-login-name conversions

SEE ALSO

acct(1M), acctsh(1M), volcopy(1M), volcopy_vxfs(1M), acct(4), diskusg(1M).

STANDARDS CONFORMANCE

vxdiskusg: SVID2, SVID3

NAME

vxdump, rvxdump (vxfs) - incremental file system dump, local or across network

SYNOPSIS

/usr/sbin/vxdump [**-nuwW**] [**-0123456789**] [**-f** *file_name*]
[**-d** *density*] [**-s** *size*] [**-T** *time*] [**-b** *block_size*]
[**-B** *records*] *filesystem*

/usr/sbin/rvxdump [**-nuwW**] [**-0123456789**] [**-f** *file_name*]
[**-d** *density*] [**-s** *size*] [**-T** *time*] [**-b** *block_size*]
[**-B** *records*] *filesystem*

/usr/sbin/vxdump [*option* [*argument* ...] *filesystem*]

/usr/sbin/rvxdump [*option* [*argument* ...] *filesystem*]

DESCRIPTION

vxdump and **rvxdump** copy to magnetic tape all files in the **vxfs** *filesystem* that have been changed after a certain date. This information is derived from the files **/var/adm/dumpdates** and **/etc/fstab**.

vxdump and **rvxdump** support both *getopt*(3C) and traditional **dump** command line invocations as shown above. The original **dump** command line style is supported for compatibility with previous versions of **vxdump** and for synonymy with the existing **dump** program used for hfs file systems. For the traditional command line style, *option* consists of characters from the set **0123456789bBdfnsTuWw** without any intervening white space.

On most devices **vxdump** can detect end-of-media and prompt for the media to be changed, so it is not necessary to specify the size of the device. However, if the dump will require multiple tapes and the tapes are to be read using an older version of **vxrestore**, or if the tape device handles end-of-media in a way that **vxdump** doesn't understand, then the size of the device must be specified using either the **-B** option or a combination of the **-d** and **-s** options.

Options

-*number* Where *number* is in the range [0-9]. This number is the dump level. All files modified since the last date stored in the file **/var/adm/dumpdates** for the same file system at a lesser dump level will be dumped. Thus, the option **-0** causes the entire file system to be dumped. If no date is determined by the level, the beginning of time is assumed.

-B *records*
The number of logical records per volume. The **vxdump** logical record size is 1024 bytes. *records* can also be specified with a suffix to indicate a unit of measure other than 1024 bytes. A **k**, **m**, or **g** can be appended to the number to indicate that the value is in kilobytes, megabytes, or gigabytes, respectively. This option overrides the calculation of tape size based on length and density.

-b *block_size*
The blocking factor is taken from the *block_size* option argument. (default is 63 if **-b** is not specified). Block size is defined as the logical record size times the blocking factor. **vxdump** writes logical records of 1024 bytes. Older versions of **vxdump** used a blocking factor of 10 for tapes with densities less than 6250 BPI, and 32 for tapes with densities of 6250 BPI or greater. **vxrestore** will dynamically determine the blocking factor.

-d *density*
The density of the tape (expressed in BPI). This is used in calculating the amount of tape used per tape reel. If the **-s** option is specified, a default density value of 1600 is assumed a for a reel tape.

-f *file_name*
Place the dump on the file *file_name* instead of the tape. If the name of the file is **-**, **vxdump** writes to the standard output. This option can be of the form *machine***:***device* to specify a tape device on a remote machine.

-n Whenever **vxdump** requires operator attention, notify all users in group **operator** by means similar to that described by *wall*(1M).

-s *size* *size* is the size of the dump tape, specified in feet. When the specified size is reached, **vxdump** waits for reels to be changed. If the **-d** option is specified, a default size value of 2300 is assumed a for a reel tape.

-u If the dump completes successfully, write on file **/var/adm/dumpdates** the date when the dump started. This file records a separate date for each file system and each dump level. The format of **/var/adm/dumpdates** is user-readable and consists of one free-format record per line: file system name, increment level and dump date in *ctime*(3C) format. The file **/var/adm/dumpdates** can be edited to change any of the fields if necessary.

-T *date* Use the specified date as the starting time for the dump instead of the time determined from looking in **/var/adm/dumpdates**. The format of *date* is the same as that of *ctime*(3C) This option is useful for automated dump scripts that wish to dump over a specific period of time. The **-T** option is mutually exclusive with the **-u** option.

-W For each file system in **/var/adm/dumpdates**, print the most recent dump date and level, indicating which file systems should be dumped. If the **-W** option is set, all other options are ignored and **vxdump** exits immediately.

-w Operates like **W**, but prints only file systems that need to be dumped.

If no arguments are given, the options are assumed to be **-9u** and a default file system is dumped to the default tape.

Operator Interaction

vxdump requires operator intervention for any of the following conditions:

- end of tape,
- end of dump,
- tape-write error,
- tape-open error, or
- disk-read error (if errors exceed threshold of 32).

In addition to alerting all operators implied by the **-n** option, **vxdump** interacts with the control terminal operator by posing questions requiring **yes** or **no** answers when it can no longer proceed or if something is grossly wrong.

Since making a full dump involves considerable time and effort, **vxdump** establishes a checkpoint at the start of each tape volume. If, for any reason, writing that volume fails, **vxdump** will, with operator permission, restart from the checkpoint after the old tape has been rewound and removed and a new tape has been mounted.

vxdump periodically reports information to the operator, including estimates (typically low) of the number of blocks to write, the number of tapes it will require, time needed for completion, and the time remaining until tape change. The output is verbose to inform other users that the terminal controlling **vxdump** is busy and will be for some time.

Compatibility

The dump tape format is independent of the VxFS disk layout. A dump of a file system with the Version 3 disk layout can be restored on a file system using the Version 2 disk layout or even a file system of another file system type, with the following exceptions:

- Files larger than 2 Gbyte cannot be restored by earlier versions of **vxrestore**. If a file larger than 2 Gbyte is encountered, **vxrestore** will skip the file and produce the diagnostic:

  Resync restore, skipped *num* **blocks**

- Files larger than 2 Gbyte cannot be restored on a file system that does not support large files (see *mount_vxfs*(1M)).

- A file with a large *uid* (user ID of the file owner) or large *gid* (group ID of the file owner) cannot be restored correctly on a file system that does not support large IDs. Instead, the owner and/or group of the file will be that of the user invoking **vxrestore**. (A large ID is a value grater than 65535. The VxFS Version 2 disk layout does not support lage IDs).

- Files with VxFS extent attributes (see *setext*(1M)) cannot be restored on a file system of a type that does not support extent attributes.

If you use **vxdump** to produce a dump intended for an earlier version of **vxrestore,** and if the dump requires multiple tapes, you should use the **-s**, **-d**, or **-B** option.

Dumps produced by older versions of **vxdump** can be read by the current version of **vxrestore**.

NOTES

Dumps should be performed with the file system unmounted or the system in single-user environment (see *init*(1M)) to insure a consistent dump. If the VxFS Advanced package is installed, the dump can be performed in the multi-user environment using a snapshot file system with the online backup facility (see the **snapof=file** option of *mount_vxfs*(1M)).

Up to 32 read errors on the file system are ignored.

Each reel requires a new process; thus parent processes for reels already written remain until the entire tape is written.

vxdump creates a server, **/usr/sbin/rmt**, on the remote machine to access the tape device.

EXAMPLES

In the following example, assume that the file system **/mnt** is normally attached to the file tree at the root directory, (**/**).

This example causes the entire file system (**/mnt**) to be dumped on **/dev/rmt/0m** and specifies that the the size of the tape is 2 gigabytes.

```
vxdump -0 -B 2g -f /dev/rmt/0m /mnt
```

Or, using the traditional command line syntax and specifying the tape size in logical records:

```
vxdump 0Bf 2097152 /dev/rmt/0m /mnt
```

where the option argument “2097152” goes with the option letter **B** as it is the first option letter that requires an option argument, and where the option argument “/dev/rmt/0m” goes with the option letter **f** as it is the second option letter that requires an option argument.

AUTHOR

vxdump and **rvxdump** are based on the **dump** and **rdump** programs from the 4.4 Berkeley Software Distribution, developed by the the University of California, Berkeley, and its contributors.

FILES

/dev/rdsk/c0t0d0	Default file system to dump from.
/dev/rmt/0m	Default tape unit to dump to.
/var/adm/dumpdates	New format-dump-date record.
/etc/fstab	Dump table: file systems and frequency.
/etc/mnttab	Mounted file system table.
/etc/group	Used to find group **operator**.

SEE ALSO

rmt(1M), vxrestore(1M), fstab(4).

NAME

vxrestore, rvxrestore (vxfs) - restore file system incrementally, local or across network

SYNOPSIS

/usr/sbin/vxrestore [**-rRtxihmvy**] [**-s** *number*]
[**-b** *block_size*] [**-e** *opt*] [**-f** *file*] [*file_name* ...]

/usr/sbin/rvxrestore [**-rRtxihmvy**] [**-s** *number*]
[**-b** *block_size*] [**-e** *opt*] [**-f** *file*] [*file_name* ...]

/usr/sbin/vxrestore *key* [*file_name* ...]

/usr/sbin/rvxrestore *key* [*file_name* ...]

DESCRIPTION

rvxrestore is another name for **vxrestore**. **vxrestore** reads tapes previously dumped by the **vxdump** or **rvxdump** command (see *vxdump*(1M)).

vxrestore and **rvxrestore** support both *getopt*(3C) and traditional **restore** command line invocations as shown above. The original **restore** command line style is supported for compatibility with previous versions of **vxrestore** and for synonymy with the existing **restore** program used for hfs file systems.

For the original **restore** command line style, actions taken are controlled by the *key* argument where *key* is a string of characters containing exactly one function letter from the group **rRxtsi**, and zero or more function modifiers from the group **befhmvy**. One or more *file_name* arguments, if present, are file or directory names specifying the files that are to be restored. Unless the **h** modifier is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

Options

-r Read the tape and load into the current directory. **-r** should be used only after careful consideration, and only to restore a complete dump tape onto a clear file system or to restore an incremental dump tape after a full-level zero restore. Thus,

```
/usr/sbin/newfs -F vxfs /dev/rdsk/c0t0d0
/usr/sbin/mount -F vxfs /dev/dsk/c0t0d0 /mnt
cd /mnt
vxrestore -r
```

is a typical sequence to restore a complete dump. Another **vxrestore** can then be performed to restore an incremental dump on top of this. Note that **vxrestore** leaves a file **restoresymtab** in the root directory of the file system to pass information between incremental **vxrestore** passes. This file should be removed when the last incremental tape has been restored.

-R Resume a full restore. **vxrestore** restarts from a checkpoint it created during a full restore (see **-r** above). It requests a particular tape of a multi-volume set on which to restart a full restore. This provides a means for interrupting and restarting a multi-volume **vxrestore**.

-x Extract named files from the tape. If the named file matches a directory whose contents had been written onto the tape, and the **-h** option is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no *file_name* argument is given, the root directory is extracted, which results in the entire contents of the tape being extracted, unless **-h** has been specified.

-t Names of *file_names*, as specified on the command line, are listed if they occur on the tape. If no *file_name* is given, the root directory is listed, which results in the entire content of the tape being listed, unless **-h** has been specified.

-s*number*
number is used as the dump file number to recover. This is useful if there is more than one dump file on a tape.

-i This option allows interactive restoration of files from a dump tape. After reading in the directory information from the tape, **vxrestore** provides a shell-like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

add [*arg*]	The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, it and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). File names on the extraction list are displayed with a leading * when listed by **ls**.
cd [*arg*]	Change the current working directory to the specified argument.
delete [*arg*]	The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, it and all its descendents are deleted from the extraction list (unless **h** is specified on the command line). The most expedient way to extract most files from a directory is to add the directory to the extraction list, then delete unnecessary files.
extract	All files named on the extraction list are extracted from the dump tape. **vxrestore** asks which volume the user wants to mount. The fastest way to extract a few files is to start with the last volume, then work toward the first volume.
help	List a summary of the available commands.
ls [*arg*]	List the current or specified directory. Entries that are directories are displayed with a trailing **/**. Entries marked for extraction are displayed with a leading *. If the verbose key is set, the inode number of each entry is also listed.
pwd	Print the full pathname of the current working directory.
quit	**vxrestore** immediately exits, even if the extraction list is not empty.
set-modes	Set the owner, modes, and times of all directories that are added to the extraction list. Nothing is extracted from the tape. This setting is useful for cleaning up after a restore aborts prematurely.
verbose	The sense of the **v** modifier is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes **vxrestore** to print out information about each file as it is extracted.

The following options can be used in addition to the letter that selects the primary function desired:

-b *block_size*
Specify the block size of the tape in Kbytes. If the **-b** option is not specified, **vxrestore** will determine the tape block size dynamically. [This option is exists to preserve backwards compatibility with previous versions of **vxrestore**.]

-e *opt*
Specify how to handle a *vxfs* file that has extent attribute information. Extent attributes include reserved space, a fixed extent size, and extent alignment. It may not be possible to preserve the information if the destination file system does not support extent attributes, has a different block size than the source file system, or lack free extents appropriate to satisfy the extent attribute requirements. Valid values for *opt* are:

warn	Issue a warning message if extent attribute information cannot be kept (the default).
force	Fail to restore the file if extent attribute information cannot be kept.
ignore	Ignore extent attribute information entirely.

-f *file*
Specify the name of the archive instead of **/dev/rmt/0m**. If the name of the file is **-**, **vxrestore** reads from standard input. Thus, **vxdump** and **vxrestore** can be used in a pipeline to vxdump and vxrestore a file system with the command

```
vxdump 0f - /usr | (cd /mnt; vxrestore xf -)
```

An archive name of the form *machine***:***device* can be used to specify a tape device on a remote machine.

-h Extract the actual directory, rather than the files to which it refers. This prevents hierarchical restoration of complete subtrees.

-m Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted and one wants to avoid regenerating the complete pathname to the file.

-v Type the name of each file restored, preceded by its file type. Normally **vxrestore** does its work silently; the **-v** option specifies verbose output.

-y Do not ask whether to abort the operation if **vxrestore** encounters a tape error. Normally **vxrestore** asks whether to continue after encountering a read error. With this option, **vxrestore** continues without asking, attempting to skip over the bad tape block(s) and continue as best it can.

vxrestore creates a server, **/usr/sbin/rmt**, on the remote machine to access the tape device.

DIAGNOSTICS

vxrestore complains if a read error is encountered. If the **-y** option has been specified, or the user responds **y**, **vxrestore** attempts to continue the restore.

If the dump extends over more than one tape, **vxrestore** asks the user to change tapes. If the **-x** or **-i** option has been specified, **vxrestore** also asks which volume the user wants to mount. The fastest way to extract a few files is to start with the last volume and work towards the first volume.

There are numerous consistency checks that can be listed by **vxrestore**. Most checks are self-explanatory or can "never happen". Here are some common errors:

filename: **not found on tape**
The specified file name was listed in the tape directory but not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file *inumber*, **got** *inumber*
A file not listed in the directory showed up. This can occur when using a dump tape created on an active file system. Dumps should be performed with the file system unmounted or the system in single-user environment (see *init*(1M)) to insure a consistent dump. If the VxFS Advanced package is installed, the dump can be performed in the multi-user environment using a snapshot file system with the online backup facility (see the **snapof=file** option of *mount_vxfs*(1M)).

Incremental tape too low
When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high
When doing an incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level has been loaded.

Tape read error while restoring *filename*
Tape read error while skipping over inode *inumber*
Tape read error while trying to resynchronize
A tape-read error has occurred. If a file name is specified, the contents of the restored files are probably partially wrong. If **vxrestore** is skipping an inode or is trying to resynchronize the tape, no extracted files are corrupted, although files may not be found on the tape.

Resync restore, skipped *num* **blocks**
After a tape-read error, **vxrestore** may have to resynchronize itself. This message indicates the number of blocks skipped over. This message will also be generated by older versions of **vxrestore** while skipping over files larger than 2 Gbyte dumped by a more recent version of **vxdump**.

NOTES

If the dump tape contains files larger than 2 Gbyte, and if the file system being restored to does not support files larger than 2 Gbyte, the file will not be restored correctly. Instead it will be truncated to 2 Gbyte.

A file with a large *uid* (user ID of the file owner) or large *gid* (group ID of the file owner) cannot be restored correctly on a file system that does not support large IDs. Instead, the owner and/or group of the file will be that of the user invoking **vxrestore**. (A large ID is a value grater than 65535. The VxFS Version 2 disk layout does not support lage IDs).

Dumps produced by older versions of **vxdump** can be read by the current version of **vxrestore**.

vxrestore can restore files to a file system of a type other than VxFS. If the file system type does not support extent attributes, than the extent attributes will not be restored (see the **-e** option).

WARNINGS

vxrestore can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level-zero dump (see the *vxdump*(1M) manual page) must be done after a full restore. Since **vxrestore** runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files are unchanged.

AUTHOR

vxrestore and **rvxrestore** are based on the **restore** program distributed in the 4.4 Berkeley Software Distribution, developed by the the University of California, Berkeley, and its contributors.

FILES

/dev/rmt/0m	default tape drive
/tmp/rstdr*	file containing directories on the tape
/tmp/rstmd*	owner, mode, and time stamps for directories
./restoresymtab	information passed between incremental restores

SEE ALSO

vxdump(1M), extendfs_vxfs(1M), fsadm_vxfs(1M), mkfs(1M), mount(1M), newfs(1M), rmt(1M).

NAME

vxupgrade - upgrade the disk layout of a VxFS file system

SYNOPSIS

/usr/sbin/vxupgrade [**-n** *new_version*] [**-r** *rawdev*] *mount_point*

DESCRIPTION

vxupgrade prints the current disk layout version number for a VxFS file system or upgrades the file system to a new disk layout. **vxupgrade** operates on file systems mounted for read/write access: *mount_point* must be a mounted VxFS file system. Only a privileged user can query or upgrade a VxFS file system.

When invoked with the **-n** option, **vxupgrade** upgrades the disk layout to the specified version. When invoked without the the **-n** option, vxfs prints the disk layout version number of the file system.

Options

-n *new_version*	Disk layout version number to upgrade to. *new_version* is 3.
-r *rawdev*	Pathname of raw device to use. This option can be used when **vxupgrade** cannot determine what the raw device corresponding to the mount point is (when **/etc/mnttab** is corrupted, for instance).

To perform an upgrade, **vxupgrade** freezes the file system, allocates and initializes the new structures, frees the space used by the old structures, and then thaws the file system. This process should not keep the file system frozen for more than a few seconds.

vxupgrade makes use of a lock file (**lost+found/.fsadm**) on the file system to ensure that only one instance of **vxupgrade** is running at any time. **vxupgrade** and **fsadm** cannot be run simultaneously, so the lock file also ensures that **vxupgrade** is not run while a file system reorganization is in progress. When **vxupgrade** is invoked for an upgrade, it opens the lock file in the root of the file system specified by *mount_point*. If the file doesn't exist, it is created. The *fcntl*(2) system call is used to obtain a write lock on the file. If the write lock fails, **vxupgrade** will assume that another **vxupgrade** or an **fsadm** is running and will fail.

NOTES

Once a file system has been upgraded to Version 3, it is no longer mountable with releases of VxFS prior to VxFS 3.0.

File systems cannot be downgraded.

Free Space Requirement

vxupgrade requires some free space on the file system in order to perform the upgrade, and the upgrade may fail if not enough free space is available. It is difficult to determine the exact amount of space required to upgrade a VxFS file system; however, one can estimate the maximum space required.

To upgrade a Version 2 file system with $n * 1024$ inodes (allocated only) and $m * 32768$ blocks to Version 3, the worst-case minimum value is at least $n * 2432$ bytes + $m * 8220$ bytes + 115 Kbytes, in extents of 8 Kbytes or larger. Free extents of larger than 8 Kbytes may be required, so this is only a lower bound on the worst-case minimum required. Since this is the worst-case minimum, it may be possible to upgrade with less free space available. After the upgrade to Version 3 is completed, all of this free space, plus some additional free space, will be reclaimed.

DIAGNOSTICS

An exit value of 0 is returned if the upgrade was successful, 1 if the upgrade failed due to a lack of free space, and 2 if the upgrade failed for some other reason.

V

FILES

mount_point/**lost+found/.fsadm**	lock file

SEE ALSO

fsadm_vxfs(1M), mkfs_vxfs(1M), fs_vxfs(4), and vxfsio(7).

NAME

wall - write message to all users

SYNOPSIS

/usr/sbin/wall [**-g***groupname*] [*file*]

/usr/sbin/cwall [**-g***groupname*] [*file*]

DESCRIPTION

Without arguments, the **wall** command reads a message from standard input until end-of-file. Then it sends this message to all currently logged-in users preceded by:

> **Broadcast Message from** ...

If the **-g***groupname* option is specified, **wall** sends the message to all currently logged-in *groupname* members (as specified in **/etc/group**) preceded by:

> **Broadcast Message from** ... **to group** *groupname*

If *file* is specified, **wall** reads *file* instead of standard input.

wall is typically used to warn all users prior to shutting down the system.

The sender must have appropriate privileges to override any protections the users may have invoked (see *mesg*(1)).

wall has timing delays, and takes at least 30 seconds to complete.

You must have appropriate privileges to override any protections users may have invoked (see *mesg*(1)).

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Cannot send to ...

The open on a user's tty file failed.

WARNINGS

The **wall** command will be WITHDRAWN from X/Open standard and may not be portable to other vendor's platforms.

AUTHOR

wall was developed by AT&T and HP.

FILES

/dev/tty*

SEE ALSO

mesg(1), write(1).

STANDARDS CONFORMANCE

wall: SVID2, SVID3, XPG2, XPG3

NAME

whodo - which users are doing what

SYNOPSIS

/usr/sbin/whodo [**-h**] [**-l**] [**user**]

DESCRIPTION

The **whodo** command produces merged, reformatted, and dated output from the **who**, **ps** and **acctcom** commands (see *who*(1) , *ps*(1) and *acctcom(1M)).*

If user is specified, output is restricted to all sessions pertaining to that user.

The following options are available:

-h	Suppress the heading.
-l	Produce a long form of output. The fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in hours:minutes), the idle time - that is, the time since the user last typed anything (in hours:minutes), the CPU time used by all processes and their children on that terminal (in minutes:seconds), the CPU time used by the currently active processes (in minutes:seconds), and the name and arguments of the current process.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **whodo** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

FILES

/etc/passwd

/var/adm/pacct

SEE ALSO

ps(1), who(1), acctcom(1M).

STANDARDS CONFORMANCE

whodo: SVID2, SVID3

NAME

xntpd - Network Time Protocol daemon

SYNOPSIS

xntpd [**-ab**] [**-c** *conffile*] [**-e** *authdelay*] [**-f** *driftfile*] [**-k** *keyfile*] [**-l** *loopfile*]
[**-p** *pidfile*] [**-r** *broaddelay*] [**-s** *statsdir*] [**-t** *trustedkey*]

DESCRIPTION

xntpd is a daemon which maintains a UNIX system's time-of-day in agreement with Internet standard time servers. **xntpd** is a complete implementation of the Network Time Protocol (NTP) version 3 standard as defined by RFC 1305 and also retains compatibility with version 2 servers as defined by RFC 1119 and version 1 servers as defined by RFC 1059. **xntpd** does all computations in fixed point arithmetic and is entirely free of floating point code. The computations done in the protocol and clock adjustment code are carried out with high precision to try to maintain an accuracy suitable for synchronizing with even the most precise external time source.

xntpd exits if it detects that the system clock is off by 1000 seconds or more. Ordinarily, **xntpd** reads its configuration from a file at startup time. The default configuration file is **/etc/ntp.conf**, though this may be overridden from the command line. It is also possible to specify a working, though limited, **xntpd** configuration entirely on the command line, obviating the need for a configuration file. This may be particularly appropriate when **xntpd** is to be configured as a broadcast client, with all peers being determined by listening to broadcasts at run time.

The following command line arguments are understood by **xntpd** (see the configuration file description for a more complete functional description):

-a	**xntpd** will run in authenticate mode.
-b	**xntpd** will listen for broadcast NTP and sync to this if available.
-c	It specifies an alternate configuration file *filename.*
-e	It specifies the time (in seconds) **xntpd** takes to compute the NTP encryption field on this computer.
-f	It specifies the location of the drift file.
-k	It specifies the location of the file which contains the NTP authentication keys.
-l	It specifies the name of the file to record loop filter statistics.
-p	It specifies the name of the file to record the daemon's process id.
-r	It specifies the default round trip delay (in seconds) to be used when synchronizing to broadcasts
-s	It specifies a directory to be used for creating statistics files.
-t	It adds a key number to the trusted key list.

Configuration File Options

xntpd's configuration file is relatively free format. Comments, which may be freely inserted, begin with a **#** character and extend to the end of the line. Blank lines are ignored. Configuration statements include an initial keyword followed by white space separated arguments, some of which may be optional. Configuration statements may not be continued over multiple lines. Arguments may be network numbers (which must be written in numeric, dotted-quad form), integers, floating point numbers (when specifying times in seconds) and text strings. Optional arguments are delimited by **[]** in the following descriptions, while alternatives are separated by **|** .

peer *host_address* [**key** *#*] [**version** *#*] [**minpoll** *interval*] [**prefer**]
server *host_address* [**key** *#*] [**version** *#*] [**minpoll** *interval*] [**prefer**]
broadcast *host_address* [**key** *#*] [**version** *#*] [**minpoll** *interval*]

These three statements specify various time servers to be used and/or time services to be provided. The **peer** statement specifies that the given host is to be polled in **symmetric active** mode, i.e. that the host is requested to provide time to which you might synchronize and, in addition, indicates that you are willing to have the remote host synchronize to your time. The **server** statement specifies that the given host is to be polled in **client** mode, i.e. that the host is requested to provide time to which you might synchronize but that you are unwilling to have the remote host synchronize to your own time. The **broadcast** statement requests your local daemon to transmit broadcast NTP to the specified address. The latter is usually the broadcast address on [one of] your local network[s].

X

key This option, when included, indicates that all packets sent to the address are to include authentication fields encrypted using the specified key number (the range of which is that of an unsigned 32 bit integer). The default is to not include an encryption field.

version
This option allows one to specify the version number to be used for outgoing NTP packets. Versions 1, 2, and 3 are the choices, version 3 is the default.

minpoll
It specifies the polling interval. The valid value for *interval* should be between 6-10 inclusive, which specifies that the minimum polling interval is 2** *interval* seconds minimum even when the local daemon isn't using the remote server's data for synchronization. The default minpoll interval value is 6 (64 seconds).

prefer This option marks the host as a preferred host. Preferred hosts determined the validity of the PPS signal and are the primary selection for synchronization when found in the set of suitable synchronization sources.

precision *#*

Indicates the precision of local timekeeping. The value is an integer which is approximately the base 2 logarithm of the local timekeeping precision in seconds. By default this value is set to -6.

The precision declared by an implementation can affect several aspects of server operation, and can be used as a tuning parameter for your synchronization subnet. It should probably not be changed from the default value, however, unless there is a good reason to do so.

driftfile *filename*

Specifies the name of the file used to record the "drift" (or frequency error) value **xntpd** has computed. If the file exists on startup, it is read and the value used to initialize **xntpd**'s internal value of the frequency error. The file is then updated once every hour by replacing the old file with a new one containing the current value of the frequency error. Note that the file is updated by first writing the current drift value into a temporary file and then using *rename*(2) to replace the old version. This implies that **xntpd** must have write permission for the directory the drift file is located in, and that file system links, symbolic or otherwise, should probably be avoided.

broadcastclient yes|no

This indicates whether the local server should listen for, and attempt to synchronize to, broadcast NTP. The default is **no**.

broadcastdelay *seconds*

Specifies the default round trip delay to the host whose broadcasts are being synchronized to. The value is specified in seconds and is typically (for ethernet) a number between 0.007 and 0.015 seconds. This initial estimate may be improved by polling each server to determine a more accurate value. Defaults to 0.008 seconds.

authenticate yes|no

Indicates whether the local server should operate in authenticate mode or not. If **yes**, only peers which include an authentication field encrypted with one of our trusted keys (see below) will be considered as candidates for synchronizing to. The default is **no**.

authdelay *seconds*

Indicates the amount of time it takes to encrypt an NTP authentication field on the local computer. This value is used to correct transmit timestamps when the authentication is used on outgoing packets. The value usually lies somewhere in the range 0.0001 seconds to 0.003 seconds, though it is very dependent on the CPU speed of the host computer.

keys *filename*

Specifies the name of a file which contains the encryption keys which are to be used by **xntpd**. The format of this file is described below.

trustedkey *#* [*#* ...]

Allows the specification of the encryption key numbers which are trusted for the purposes of determining peers suitable for time synchronization, when authentication is enabled. Only peers using one of these keys for encryption of the authentication field, and whose authenticity can be verified by successful decryption, will be considered as synchronization candidates. The arguments are 32 bit unsigned integers. Note, however, that NTP key 0 is fixed and globally known. If meaningful authentication is to be performed the 0 key should not be trusted.

controlkey *#*

Certain changes can be made to the **xntpd** server via mode 6 control messages, in particular the setting of leap second indications in a server with a radio clock. The **controlkey** statement specifies an encryption key number to be used for authenticating such messages. Omitting this statement will cause control messages which would change the state of the server to be ignored.

restrict *address* [**mask** *numeric_mask*] [*flag*] [...]

xntpd implements a general purpose address-and-mask based restriction list. The list is sorted by address and by mask, and the list is searched in this order for matches, with the last match found defining the restriction flags associated with the incoming packets. The source address of incoming packets is used for the match, with the 32 bit address being and'ed with the mask associated with the restriction entry and then compared with the entry's address (which has also been and'ed with the mask) to look for a match. The **mask** argument defaults to 255.255.255.255, meaning that the *address* is treated as the address of an individual host. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and, given the sort algorithm, is always the first entry in the list. Note that, while *address* is normally given as a dotted-quad address, the text string *default*, with no mask option, may be used to indicate the default entry.

In the current implementation, flags always restrict access; i.e. an entry with no flags indicates that free access to the server is to be given. The flags are not orthogonal, in that more restrictive flags will often make less restrictive ones redundant. The flags can generally be classed into two categories, those which restrict time service and those which restrict informational queries and attempts to do run time reconfiguration of the server. One or more of the following flags may be specified:

ignore	Ignore all packets from hosts which match this entry. If this flag is specified neither queries nor time server polls will be responded to.
noquery	Ignore all NTP mode 6 and 7 packets (i.e. information queries and configuration requests) from the source. Time service is not affected.
nomodify	Ignore all NTP mode 6 and 7 packets which attempt to modify the state of the server (i.e. run time reconfiguration). Queries which return information are permitted.
noserve	Ignore NTP packets whose mode is other than 6 or 7. In effect, time service is denied, though queries may still be permitted.
nopeer	Provide stateless time service to polling hosts, but do not allocate peer memory resources to these hosts even if they otherwise might be considered useful as future synchronization partners.
notrust	Treat these hosts normally in other respects, but never use them as synchronization sources.
ntpport	This is actually a match algorithm modifier, rather than a restriction flag. Its presence causes the restriction entry to be matched only if the source port in the packet is the standard NTP UDP port (123).

Default restriction list entries, with the flags *ignore, ntpport,* for each of the local host's interface addresses are inserted into the table at startup to prevent the server from attempting to synchronize to its own time. A default entry is also always present, though if it is otherwise unconfigured no flags are associated with the default entry (i.e. everything besides your own NTP server is unrestricted).

The restriction facility was added to allow the current access policies of the time servers running on the NSFnet backbone to be implemented with **xntpd** as well. While this facility may be otherwise useful for keeping unwanted or broken remote time servers from affecting your own, it should not be considered an alternative to the standard NTP authentication facility. Source address based restrictions are easily circumvented by a determined cracker.

statsdir */directory/[prefix]*

Indicates the full path of a directory where statistics files should be created (see below). This optional *prefix* allows the modification of the filename prefix of the file generation sets used for handling statistics logs (see **filegen** and **statistics** statement below).

statistics *names*

Enables writing of statistics records. *names* can be one or more statistics names separated by space. Currently two kinds of statistics are supported:

loopstats enables recording of loop filter statistics information. Each computation of the local clock parameters outputs a line of the following form to the file generation set named “loopstats”:

```
48773 10847.650 0.0001307 17.3478 2
```

The first two fields show the date (modified Julian) and time (seconds and fraction past UTC midnight). The next three fields show last offset, current drift compensation value and time constant of the loop filter.

peerstats enables recording of peer statistics information. This includes statistics records of all peers of a NTP server and of the PPS filter, if PPS signal handling is supported by the server. Each valid update appends a line of the following form to the current element of a file generation set named “peerstats”:

```
48773 10847.650 127.127.4.1 9714 -0.001605 0.00000 0.00142
```

The first two fields show the date (modified Julian) and time (seconds and fraction past UTC midnight), while the next two fields are the peer address and status, respectively. The final three fields show offset, delay and dispersion.

Statistic files are managed using file generation sets (see **filegen** below). The information obtained by enabling statistics recording allows analysis of temporal properties of a **xntpd** server. It is usually only useful to primary servers or maybe main campus servers.

filegen *name* [**file** *filename*] [**type** *typename*] [**link** | **nolink**] [**enable** | **disable**]

Configures setting of generation file set *name.* Generation file sets provides a mean for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files. Generation file sets provide access to a set of files used to store the actual data. At any time at most one element of the set is being written to. The **type** given specifies when and how data will be directed to a new element of the set. This way, information stored in elements of a file set that are currently unused are available for administrational operations without the risk of disturbing the operation of **xntpd**. (Most important: they can be removed to free space for new data produced.) Filenames of set members are built from three elements. *name* is name of the statistic to be collected. Currently only two kinds of statistics are supported: *loopstats* and *peerstats*.

file Defines a *filename* string directly concatenated to the *prefix* mentioned above (no intervening / (slash)) if *prefix* is defined in the **statsdir** statement.

type This part reflects individual elements of a file set. It is generated according to the *type* of a file set as explained below. A file generation set is characterized by its type. The following *typenames* are supported:

none The file set is actually a single plain file.

pid One element of file set is used per incarnation of a **xntpd** server. This type does not perform any changes to file set members during runtime, however it provides an easy way of separating files belonging to different **xntpd** server incarnations. The set member filename is built by appending a dot **.** to the concatenated *prefix* and *filename* strings, and appending the decimal representation of the process id of the **xntpd** server process. (e.g *<prefix><filename>.<pid>*)

day One file generation set element is created per day. The term *day* is based on *UTC*. A day is defined as the period between 00:00 and 24:00 UTC. The file set member suffix consists of a dot and a day specification in the form *<YYYYMMDD>*. *YYYY* is a 4 digit year number (e.g. 1992). *MM* is a two digit month number. *DD* is a two digit day number. Thus, all information written at December 10th, 1992 would end up in a file named *<prefix><filename>.19921210*

week Any file set member contains data related to a certain week of a year. The term *week* is defined by computing day of year modulo 7. Elements of such a file generation set are distinguished by appending the following suffix to the file set filename base: A dot, a four digit year number, the letter **W** and a two digit week number. For example, information from January, 10th 1992 would end up in a file with suffix *.1992W1.*

month One generation file set element is generated per month. The file name suffix consists of a dot, a four digit year number, and a two digit month.

year One generation file elment is generated per year. The filename suffix consists of a dot and a 4 digit year number.

age This type of file generation sets changes to a new element of the file set every 24 hours of server operation. The filename suffix consists of a dot, the letter **a,** and an eight digit number. This number is taken to be the number of seconds the server is running at the start of the corresponding 24 hour period.

enabled/disabled
Information is only written to a file generation set when this set is **enabled**. Output is prevented by specifying **disabled**. The default is **enabled**.

link/nolink
It is convenient to be able to access the *current* element of a file generation set by a fixed name. This feature is enabled by specifying **link** and disabled using **nolink.** The default is **link**. If **link** is specified, a hard link from the current file set element to a file without suffix is created. When there is already a file with this name and the number of links of this file is one, it is renamed appending a dot, the letter "C", and the pid of the **xntpd** server process. When the number of links is greater than one, the file is unlinked. This allows the current file to be accessed by a constant name.

Authentication Key File Format

The NTP standard specifies an extension allowing verification of the authenticity of received NTP packets, and to provide an indication of authenticity in outgoing packets. This is implemented in **xntpd** using the DES encryption algorithm. The specification allows any one of a possible 4 billion keys, numbered with 32 bit unsigned integers, to be used to authenticate an association. The servers involved in an association must agree on the value of the key used to authenticate their data, though they must each learn the key independently. The keys are standard 56 bit DES keys.

Additionally, another authentication algorithm is available which uses an MD5 message digest to compute an authenticator. The length of the key or password is limited to 8 characters. **xntpd** reads its keys from a file specified using the **-k** command line option or the **keys** statement in the configuration file. While key number 0 is fixed by the NTP standard (as 56 zero bits) and may not be changed, one or more of the keys numbered 1 through 15 may be arbitrarily set in the keys file.

The key file uses the same comment conventions as the configuration file. Key entries use a fixed format of the form

keyno type key

where *keyno* is a positive integer, *type* is a single character which defines the format the key is given in, and *key* is the key itself.

The key may be given in one of four different formats, controlled by the "type" character. The four key types, and corresponding formats, are listed following.

S The "key" is a 64 bit hexadecimal number in the format specified in the DES document, that is the high order 7 bits of each octet are used to form the 56 bit key while the low order bit of each octet is given a value such that odd parity is maintained for the octet. Leading zeroes must be specified (i.e. the key must be exactly 16 hex digits long) and odd parity must be maintained. Hence a zero key, in standard format, would be given as *0101010101010101* .

N The "key" is a 64 bit hexadecimal number in the format specified in the NTP standard. This is the same as the DES format except the bits in each octet have been rotated one bit right so that the parity bit is now the high order bit of the octet. Leading zeroes must be specified and odd parity must be maintained. A zero key in NTP format would be specified as *8080808080808080*

A The "key" is a 1-to-8 character ASCII string. A key is formed from this by using the lower order 7 bits of the ASCII representation of each character in the string, with zeroes being added on the right when necessary to form a full width 56 bit key, in the same way that encryption keys are formed from Unix passwords.

M The "key" is a 1-to-32 character ASCII string, using the MD5 authentication scheme. Note that both the keys and the authentication schemes (DES or MD5) must be identical between a set of peers sharing the same key number.

Primary Clock Support

xntpd can be optionally compiled to include support for a number of types of reference clocks. A reference clock will generally (though not always) be a radio timecode receiver which is synchronized to a source of standard time such as the services offered by the NRC in Canada and NIST in the U.S. The interface between the computer and the timecode receiver is device dependent and will vary, but is often a serial port.

For the purposes of configuration, **xntpd** treats reference clocks in a manner analogous to normal NTP peers as much as possible. Reference clocks are referred to by address, much as a normal peer is, though an invalid IP address is used to distinguish them from normal peers. Reference clock addresses are of the form *127.127.t.u* where *t* is an integer denoting the clock type and *u* indicates the type-specific unit number. Reference clocks are normally enabled by configuring the clock as a server using a **server** statement in the configuration file which references the clock's address. Clock addresses may generally be used anywhere else in the configuration file a normal IP address can be used, for example in **restrict** statements.

There is one additional configuration statement which becomes valid when reference clock support is used. The format is:

fudge *127.127.t.u* [**time1** *secs*] [**time2** *secs*] [**value1** *int*] [**value2** *int*] [**flag1** *0*|*1*] [**flag2** *0*|*1*]

There are two times (whose values are specified in fixed point seconds), two integral values and two binary flags available for customizing the operation of a clock. The configuration and interpretation of these values, and whether they are used at all, is a function of the needs of the particular clock driver.

xntpd on HP-UX currently supports the Spectracom's Netclock/2 plus a special pseudo-clock which synchronizes to the local system clock and can be used for backup or when no other clock source is available. The clock drivers, and the addresses used to configure them, are described as the followings:

127.127.1.u - Local synchronization clock driver

This driver doesn't support an actual clock, but rather allows the server to synchronize to its own clock, in essence to free run without its stratum increasing to infinity. This can be used to run an isolated NTP synchronization network where no standard time source is available, by allowing a free running clock to appear as if it has external synchronization to other servers. By running the local clock at an elevated stratum it can also be used to prevent a server's stratum from rising above a fixed value, this allowing a synchronization subnet to synchronize to a single local server for periods when connectivity to the primary servers is lost.

The unit number of the clock (the least significant octet in the address) must lie in the range 0 through 15 inclusive and is used as the stratum the local clock will run at. Note that the server, when synchronized to the local clock, will advertise a stratum one greater than the clock peer's stratum. More than one local clock may be configured (indeed all 16 units may be active at once), though this hardly seems useful.

The local clock driver uses only the **time1** parameter of the **fudge** statement. This parameter actually provides read and write access to the local clock drift compensation register. This value, which actually provides a fine resolution speed adjustment for the local clock, is settable but will remain unchanged from any set value when the clock is free running without external synchronization. The **fudge time1** parameter thus provides a way to manually adjust the speed of the clock to maintain reasonable synchronization with, say, a voice time announcement.

127.127.4.u

This driver provides an interface to the Spectracom Netclock/2 WWVB Synchronized Clocks in either Format-0 or Format-2 mode of operation at 9600 bps. When the clock is set to Format-0, the **time zone** switch on the clock should be set to 0 (**UTC** time). When the clock is set to Format-2, the **time zone** switch can be set to the zone which reflects the local time.

A device file */dev/wwvb%d* needs to be created, where *%d* is the unit number *u*. The valid unit number lies in the range 0 through 4 inclusive. The driver opens the RS232 output on the */dev/wwvb%d* device file. This driver does not require a 1-pulse-per-second (pps) signal and automatically compensates for the baud rate on the serial port. It does not require the clock discipline or STREAMs modules.

Fudge statement is used as a general calibration factor for Netclock2. A positive **time1** value will advance the time and a negative **time1** value will retard the time. The parameter defaults to zero, which should be appropriate if the clock's propagation delay switches have been set appropriately. The **value1** parameter can be used to set the stratum at which the peer operates. The default is 0, which is correct if you want the clock to be considered for synchronization whenever it is operating.

AUTHOR

xntpd was developed by Dennis Ferguson at the University of Toronto.

Text amended by David Mills at the University of Delaware.

FILES

/etc/ntp.conf	The default name of the configuration file.
/etc/ntp.drift	The conventional name of the drift file.
/etc/ntp.keys	The conventional name of the key file.

SEE ALSO

ntpq (1M), ntpdate (1M).

NAME

ypinit - build and install Network Information Service databases

SYNOPSIS

/usr/sbin/ypinit -m [**DOM=***NIS_domain*]

/usr/sbin/ypinit -s *NIS_server_name* [**DOM=***NIS_domain*]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypinit is a shell script that creates Network Information Service (NIS) databases on either a master or slave NIS server. **ypinit** asks a few self-explanatory questions, and reports success or failure to the terminal. For an overview of Network Information Service, see *ypfiles*(4) and *ypserv*(1M).

Options

ypinit recognizes the following options and command-line arguments:

-m	Create the local host as the master server to all maps (databases) provided in the NIS domain (see *domainname*(1)). All maps are built from scratch, either from information provided to **ypinit** at run-time, or from ASCII files in **/etc**. All such files should be complete and unabbreviated, unlike how they may exist on a NIS client machine (see *passwd*(4) for examples of abbreviated files). See *ypmake*(1M) for more information on how NIS databases are built on the master server. Note that **ypinit** uses the **NOPUSH=1** option when invoking **make**, so newly formed maps are not immediately copied to slave servers (see *ypmake*(1M)).
-s	Create NIS databases on a slave server by copying the databases from an existing NIS server that serves the NIS domain. The *NIS_server_name* argument should be the host name of either the master server for all the maps or a server on which the maps are current and stable.
DOM=*NIS_domain*	Causes **ypinit** to construct maps for the specified *NISdomain*. **DOM** defaults to the NIS domain shown by the **domainname** command (see *domainname*(1).

DIAGNOSTICS

ypinit returns exit code 0 if no errors occur; otherwise, it returns exit code 1.

AUTHOR

ypinit was developed by Sun Microsystems, Inc.

FILES

/etc/group
/etc/hosts
/etc/netgroup
/etc/networks
/etc/passwd
/etc/protocols
/etc/publickey
/etc/rpc
/etc/services
/etc/vhe_list
/etc/auto_master
/etc/mail/aliases

y

SEE ALSO

domainname(1), makedbm(1M), vhe_altlog(1M), vhe_mounter(1M), vhe_u_mnt(1M), ypmake(1M), yppush(1M), ypserv(1M), ypxfr(1M), ypxfrd(1M), group(4), hosts(4), netgroup(4), networks(4), passwd(4), protocols(4), publickey(4), rpc(4), services(4), vhe_list(4), ypfiles(4).

NAME

ypmake - create or rebuild Network Information Service databases

SYNOPSIS

/var/yp/ypmake [**DIR=***source_directory*] **[**DOM=*NIS_domain*] \
[**NOPUSH=1**] [**PWFILE=***passwd_file*] [*map* [*map* ...]]

cd /var/yp; make [**DIR=***source_directory*] **[**DOM=*NIS_domain*] \
[**NOPUSH=1**] [**PWFILE=***passwd_file*] [*map* ...]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypmake is a shell script that builds one or more Network Information Service (NIS) maps (databases) on a master NIS server. If no arguments are specified, **ypmake** either creates maps if they do not already exist or rebuilds maps that are not current. These maps are constructed from ASCII files. **ypmake** then executes **yppush** to notify slave NIS servers of the change and make the slave servers copy the updated maps to their machines (see *yppush*(1M)).

If any *maps* are supplied on the command line, **ypmake** creates or updates those *maps* only. Permissible names for *maps* are the filenames in **/etc** listed under FILES below. In addition, specific *maps* can be named, such as **netgroup.byuser** or **rpc.bynumber**.

The **make** command can be used instead of **ypmake** (see *make*(1)). The **Makefile** no longer calls the **ypmake** script but now actually constructs the maps. All NIS commands have been modified to use the **Makefile** instead of **ypmake.** The **Makefile** and **ypmake** can co-exist, but it is recommended that you consider using the **Makefile** which is the standard mechanism for building maps on other vendor's machines.

Both the **Makefile** and **ypmake** script use four variables:

DIR=*source_directory*	The directory containing the ASCII source files from which maps are constructed. **DIR** defaults to **/etc**.
DOM=*NIS_domain*	Causes **ypmake** to construct maps for the specified *NIS_domain*. **DOM** defaults to the NIS domain shown by **domainname** (see *domainname*(1)).
NOPUSH=1	When non-null (null by default), **NOPUSH** inhibits copying the new or updated databases to the slave NIS servers. Only slave NIS servers in the specified *domain* receive **yppush** notification when **NOPUSH** is null.
PWFILE=*passwd_file*	Specifies the full pathname of the ASCII file that **ypmake** should use when building the NIS passwd maps. **PWFILE** defaults to **$DIR/passwd**.

The order of arguments passed to **ypmake** is unimportant, but the maps are built or updated in the left-to-right order provided.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of Network Information Service.

DIAGNOSTICS

ypmake returns one of the following exit codes upon completion:

0	Normal termination; no problems.
1	One or more unrecognized arguments were passed.
2	The NIS domain name is not set.
3	The subdirectory used to contain maps for a specific NIS domain, **/var/yp/domain_name**, does not exist or is not writable.
4	An error was encountered when building at least one of the maps.
5	One or more maps' ASCII files do not exist or are unreadable.

EXAMPLES

Create or rebuild the password databases (both the **passwd.byname** and **passwd.byuid** maps) from **/etc/passwd** and use **yppush** to copy the databases to any slave NIS servers in the default NIS *domain*:

```
ypmake passwd.byname
```

Create or rebuild the hosts databases from **/etc/hosts** but do not copy the databases to any slave NIS servers:

```
ypmake hosts NOPUSH=1
```

Create or rebuild the network maps from **/nis/sourcefiles/networks** and copy the maps to any slave NIS servers in NIS domain **DAE_NIS**:

```
ypmake DOM=DAE_NIS networks DIR=/nis/sourcefiles
```

AUTHOR

ypmake was developed by Sun Microsystems, Inc.

FILES

/etc/group
/etc/hosts
/etc/netgroup
/etc/networks
/etc/passwd
/etc/protocols
/etc/publickey
/etc/rpc
/etc/services
/etc/vhe_list

SEE ALSO

domainname(1), make(1), makedbm(1M), vhe_altlog(1M), vhe_mounter(1M), vhe_u_mnt(1M), ypinit(1M), yppush(1M), ypserv(1M), group(4), hosts(4), netgroup(4), networks(4), passwd(4), protocols(4), publickey(4), rpc(4), services(4), vhe_list(4), ypfiles(4).

NAME

yppasswdd - daemon for modifying Network Information Service passwd database

SYNOPSIS

/usr/lib/netsvc/yp/rpc.yppasswdd *passwd_file* [**-l** *log_file*] [**-m** [*arg1 arg2 ...*]]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

The **yppasswdd** daemon handles password change requests from **yppasswd** (see *yppasswd*(1)). It changes a password entry in *passwd_file*, which must be in the format defined by *passwd*(4). The change is made only if the old password provided by **yppasswd** matches the encrypted password of that entry.

yppasswdd should be executed only on the master Network Information Service (NIS) server for the passwd database (map). The **yppasswdd** daemon is not executed by default, nor can it be started by **inetd** (see *inetd*(1M)). To enable automatic startup of **yppasswdd** at boot time, the **NIS_MASTER_SERVER** variable should be set to 1 in file **/etc/rc.config.d/namesvrs** on the master NIS server.

Options

yppasswdd recognizes the following options and command-line arguments:

-l *log_file*	Log diagnostic and error messages to *log_file*. These messages are not available if **yppasswdd** is started without the **-l** option. Information logged to the file includes date and time of the message, the host name, process ID and name of the function generating the message, and the message itself. Note that different services can share a single log file because enough information is included to uniquely identify each message.
-m [*arg1 arg2 ...*]	After *passwd_file* is modified, and if using the **-m** option, **yppasswdd** executes **make** to update the NIS passwd database (see *ypmake*(1M) Any arguments following the **-m** flag are passed to **make**. To ensure that the passwd map is rebuilt to contain the new password and all slave NIS servers have their passwd maps properly updated to include the change, always use the **-m** option to **yppasswdd**, but do not use the **NOPUSH=1** argument to **make**.

EXAMPLES

Assume the **yppasswdd** daemon is started on the master NIS server as follows:

```
/usr/lib/netsvc/yp/rpc.yppasswdd /var/yp/src/passwd \
                                -l /var/adm/yppasswdd.log \
                                -m passwd DIR=/var/yp/src
```

This indicates that the ASCII file from which the NIS passwd database is built is **/var/yp/src/passwd**. When this file is updated by a request from **yppasswd**, the NIS passwd database is rebuilt and copied to all slave NIS servers in the master's NIS domain (see *domainname*(1)).

Log messages are written to the file **/var/adm/yppasswdd.log**.

WARNINGS

yppasswdd uses lock file **/etc/ptmp** to get exclusive access to *passwd_file* when updating it. The file **/etc/ptmp** may persist if *passwd_file* is being updated and

- The system crashes or
- **yppasswdd** is killed using **SIGKILL** (see *kill*(1) and *signal*(2)).

File **/etc/ptmp** must be removed before **yppasswdd** can function properly again.

vipw also uses **/etc/ptmp** when updating **/etc/passwd** (see *vipw*(1M)). As a result, **yppasswdd** competes with **vipw** when it updates *passwd_file* if *passwd_file* is **/etc/passwd**.

y

AUTHOR

yppasswdd was developed by Sun Microsystems, Inc.

FILES

/etc/ptmp lock file used when updating *passwd_file*

SEE ALSO

domainname(1), yppasswd(1), vipw(1M), ypmake(1M), yppasswd(3N), passwd(4), publickey(4), ypfiles(4).

NAME

yppoll - query NIS server for information about NIS map

SYNOPSIS

/usr/sbin/yppoll [**-h** *host*] [**-d** *domain*] *mapname*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

yppoll asks a Network Information Service (NIS) server process (see *ypserv*(1M)) to return the order number (the **time** in seconds when the map was built – *time*(2)) and master NIS server's host name for a NIS database named *mapname*. **yppoll** then writes them to standard output. If the server uses Version 1 NIS protocol, **yppoll** uses this older protocol to communicate with it. **yppoll** also prints the old style diagnostic messages in case of failure.

See *ypfiles*(4) and *ypserv*(1M) for an overview of Network Information Service.

Options

-h *host*	Ask the **ypserv** process on *host* to return the map information (see *ypserv*(1M)). If **-h** *host* is not specified, the host returned by **ypwhich** is used (see *ypwhich*(1)).
-d *domain*	Use *domain* instead of the domain returned by **domainname** (see *domainname*(1)).

AUTHOR

yppoll was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypwhich(1), ypserv(1M), ypfiles(4).

NAME

yppush - force propagation of Network Information Service database

SYNOPSIS

/usr/sbin/yppush [**-d** *domain*] [**-m** *maxm*] [**-t** *mint*] [**-v**] *mapname*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

yppush copies a Network Information Service (NIS) map (database), *mapname*, from the map's master NIS server to each slave NIS server. It is usually executed only on the master NIS server by shell script **ypmake** which is run either after changes are made to one or more of the master's NIS databases or when the NIS databases are first created. See *ypmake*(1M) and *ypinit*(1M) for more information on these processes.

yppush constructs a list of NIS server host names by reading the NIS map **ypservers** within the *domain*. Keys within the **ypservers** map are the host names of the machines on which the NIS servers run. **yppush** then sends a "transfer map" request to the NIS server at each host, along with the information needed by the transfer agent (the program that actually moves the map) to call back **yppush**.

When the transfer attempt is complete, whether successful or not, and the transfer agent sends **yppush** a status message, the results can be printed to standard output. Messages are printed when a transfer is not possible, such as when the request message is undeliverable or when the timeout period on responses expires.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of Network Information Service.

Options

yppush recognizes the following options:

-d *domain* Copy *mapname* to the NIS servers in *domain* rather than to the *domain* returned by **domainname** (see *domainname*(1)).

-m *maxm* Attempt to run *maxm* transfers in parallel to as many servers simultaneously. Without the **-m** option, **yppush** attempts to transfer a map to each server, one at a time. When a network has many servers, such serial transfers can result in long delays to complete all transfers. A *maxm* value greater than 1 reduces total transfer time through better utilization of CPU time at the master. *maxm* can be any value from 1 through the number of NIS servers in the domain.

-t *mint* Set the minimum timeout value to *mint* seconds. When transferring to one slave at a time, **yppush** waits up to 80 seconds for the transfer to complete, after which it begins transferring to the next slave. When multiple parallel transfers are attempted by use of the **-m** option, it may be necessary to set the transfer timeout limit to a value larger than the default 80 seconds to prevent timeouts caused by network delays related to parallel transfers.

-v Verbose mode: messages are printed when each server is called and when each response is received. If this option is omitted, only error messages are printed.

WARNINGS

In the current implementation (Version 2 NIS protocol), the transfer agent is *ypxfr*(1M) which is started by the *ypserv*(1M) program at *yppush*'s request (see *ypxfr*(1M) and *ypserv*(1M)). If *yppush* detects it is interacting with a Version 1 NIS protocol server, it uses the older protocol to send a Version 1 **YPPROC_GET** request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for Version 1 servers. **yppush** prints a comment saying that a Version 1 message was sent. The system administrator should then verify by other means that the transfer actually occurred.

AUTHOR

yppush was developed by Sun Microsystems, Inc.

FILES

/usr/sbin/*domain***/ypservers.{dir, pag}**

/usr/sbin/*domain***/***mapname***.{dir, pag}**

SEE ALSO

domainname(1), ypserv(1M), ypxfr(1M), ypfiles(4).

NAME

ypserv, ypbind, ypxfrd - Network Information Service (NIS) server, binder, and transfer processes

SYNOPSIS

/usr/lib/netsvc/yp/ypserv [**-l** *log_file*]

/usr/lib/netsvc/yp/ypbind [**-l** *log_file*] [**-s**] [**-ypset**|**-ypsetme**]

/usr/sbin/ypxfrd

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (YP). The functionality remains the same; only the name has changed.

DESCRIPTION

The Network Information Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are files in a directory tree rooted at **/var/yp** (see *ypfiles*(4)). The processes are **/usr/lib/netsvc/yp/ypserv**, the NIS database lookup server, and **/usr/lib/netsvc/yp/ypbind**, the NIS binder. Both **ypserv** and **ypbind** are daemon processes activated at system startup time when the **NIS_MASTER_SERVER** or **NIS_SLAVE_SERVER** variable is set to 1, for **ypserv,** and the **NIS_CLIENT** variable is set to 1, for **ypbind,** in the **/etc/rc.config.d/namesvrs** file.

The NIS programmatic interface is described in *ypclnt*(3C). Administrative tools are described in *ypwhich*(1), *yppoll*(1M), *yppush*(1M), *ypset*(1M) and *ypxfr*(1M). Tools to see the contents of NIS maps (databases) are described in *ypcat*(1) and *ypmatch*(1). Database generation and maintenance tools are described in *makedbm*(1M), *ypinit*(1M), and *ypmake*(1M). The command to set or show the default NIS domain is *domainname*(1).

ypxfrd transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers will be faster, depending on the map. **ypxfrd** should be run on a server running HP-UX release 10.0. **ypxfr** (see *ypxfr*(1M)) will attempt to use **ypxfrd** first. If that fails, it will use the older transfer method. The **ypxfrd** daemon is activated at system startup time when the **NIS_MASTER_SERVER** or **NIS_SLAVE_SERVER** variable is set to 1 in the **/etc/rc.config.d/namesvrs** file.

The **ypserv** daemon's primary function is to look up information in its local collection of NIS maps. It runs only on NIS server machines providing data from NIS databases. Communication to and from **ypserv** is by means of RPC. Lookup functions are described in *ypclnt*(3C).

Four lookup functions perform on a specific map within a NIS domain: **Match**, **Get_first**, **Get_next**, and **Get_all**. The **Match** operation matches a key to a record in the database and returns its associated value. The **Get_first** operation returns the first key-value pair (record) from the map, and **Get_next** enumerates (sequentially retrieves) the remainder of the records. **Get_all** returns all records in the map to the requester as the response to a single RPC request.

Two other functions supply information about the map other than normal map entries: **Get_order_number** and **Get_master_name**. The order number is the time of last modification of a map. The master name is the host name of the machine on which the master map is stored. Both order number and master name exist in the map as special key-value pairs, but the server does not return these through the normal lookup functions. (If you examine the map with **makedbm** or **yppoll** (see *makedbm*(1M) or *yppoll*(1M)), they will be visible.) Other functions are used within the NIS system and are not of general interest to NIS clients. They include:

Do_you_serve_this_domain?
Transfer_map
Reinitialize_internal_state

The **ypbind** daemon remembers information that lets client processes on its machine communicate with a **ypserv** process. The **ypbind** daemon must run on every machine using NIS services, both NIS servers and clients. The **ypserv** daemon may or may not be running on a NIS client machine, but it must be running somewhere on the network or be available through a gateway.

The information that **ypbind** remembers is called a **binding**: the association of a NIS domain name with the Internet address of the NIS server and the port on that host at which the **ypserv** process is listening for service requests. This information is cached in the directory **/var/yp/binding** using a filename in the form *domainname***.***version*.

Client requests drive the binding process. As a request for an unbound domain comes in, the **ypbind** process broadcasts on the network trying to find a **ypserv** process serving maps within that NIS domain. Since the binding is established by broadcasting, at least one **ypserv** process must exist on every network. Once a binding is established for a client, it is given to subsequent client requests. Execute **ypwhich** to query the **ypbind** process (local and remote) for its current binding (see *ypwhich*(1)).

Bindings are verified before they are given to a client process. If **ypbind** is unable to transact with the **ypserv** process it is bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind again. Requests received for an unbound domain fail immediately. Generally, a bound domain is marked as unbound when the node running **ypserv** crashes or is overloaded. In such a case, **ypbind** binds to any NIS server (typically one that is less heavily loaded) that is available on the network.

The **ypbind** daemon also accepts requests to set its binding for a particular domain. **ypset** accesses the **Set_domain** facility; it is for unsnarling messes and is not for casual use.

Options

ypserv recognizes the following options:

-l *log_file* Log diagnostic and error messages to the file, *log_file*.

If **ypserv** is started without the **-l** option, **ypserv** writes its messages to **/var/yp/ypserv.log** if that file exists.

If **ypbind** is started without the **-l** option, **ypbind** writes its messages directly to the system console, **/dev/console**.

Information logged to the file includes the date and time of the message, the host name, the process id and name of the function generating the message, and the message itself. Note that different services can share a single log file since enough information is included to uniquely identify each message.

ypbind recognizes the following options:

-l *log_file* Log diagnostic and error messages to the file, *log_file*. See the description above.

-s Secure. When specified, only NIS servers bound to a reserved port are used. This allows for a slight increase in security in completely controlled environments, where there are no computers operated by untrusted individuals. It offers no real increase in security.

-ypset Allow **ypset** to be used to change the binding (see *ypset*(1M)). For maximum security, this option should be used only when debugging the network from a remote machine.

-ypsetme Allow **ypset** to be issued from this machine (see *ypset*(1M)). Security is based on IP address checking, which can be defeated on networks where untrusted individuals may inject packets. This option is not recommended.

AUTHOR

ypserv, **ypbind**, and **ypxfrd** were developed by Sun Microsystems, Inc.

FILES

/var/yp/binding/*domainname***.***version*

These files cache the last successful binding created for the given domain, in order to to speed up the binding process. When a binding is requested, these files are checked for validity and then used.

/var/yp/securenets This file is read by **ypxfrd** and **ypserv**. It contains a list of IP addresses that these servers will allow a binding to.

/var/yp/secureservers This file is read by ypbind. It contains a list of IP addresses that ypbind will receive a binding from.

SEE ALSO

domainname(1), ypcat(1), ypmatch(1), yppasswd(1), ypwhich(1), makedbm(1M), rpcinfo(1M), ypinit(1M), ypmake(1M), yppasswdd(1M), yppoll(1M), yppush(1M), ypset(1M), ypxfr(1M), ypclnt(3C), yppasswd(3N), ypfiles(4).

NAME

ypset - bind to particular Network Information Service server

SYNOPSIS

/usr/sbin/ypset [**-V1**] [**-h** *host*] [**-d** *domain*] *server*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypset tells **ypbind** to get Network Information Service (NIS) services for the specified *domain* from the **ypserv** process running on *server* (see *ypserv*(1M) and *ypbind*(1M)). *server* is the NIS server that the NIS client binds to, and is specified as either a host name or an IP address. If *server* is down or is not running **ypserv**, this is not discovered until a local NIS client process tries to obtain a binding for the domain. The **ypbind** daemon then tests the binding set by **ypset**. If the binding cannot be made to the requested server, **ypbind** attempts to rebind to another server in the same domain.

The **ypset** command is useful for binding a client node that is not on a broadcast network, since broadcasting is the method by which **ypbind** locates a NIS server. If a client node exists on a broadcast network which has no NIS server running, and if there is a network with one running that is available via a gateway, **ypset** can establish a binding through that gateway. It is also useful for debugging NIS client applications such as when a NIS map exists only at a single NIS server.

In cases where several hosts on the local net are supplying NIS services, it is possible for **ypbind** to rebind to another host, even while you attempt to find out if the **ypset** operation succeeded. For example, typing **ypset host1** followed by **ypwhich** and receiving the reply **host2** may be confusing. It could occur when *host1* does not respond to **ypbind** because its **ypserv** process is not running or is overloaded, and *host2*, running **ypserv**, gets the binding.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of the Network Information Service.

Options

ypset recognizes the following options and command-line arguments:

-V1	Bind *server* for the (old) Version 1 NIS protocol.
-h *host*	Set the binding on *host* instead of locally. *host* can be specified as a host name or an IP address.
-d *domain*	Use *domain* instead of the default domain returned by **domainname** (see *domainname*(1)).

DIAGNOTICS

Sorry, ypbind on host 'name' has rejected your request.

The user is not root, or ypbind was run without the **-ypset** flags. See *ypserv*(1M) for explanations of the **-ypset** flags.

Sorry, I couldn't send my rpc message to ypbind on host 'name'.

The user is not root, or ypbind was run without one of the **-ypset** flags. See *ypserv*(1M) for explanations of the **-ypset** flags.

WARNINGS

The *server* is the NIS server to bind to, specified as either a host name or an IP address. If *server* is a host name, **ypset** uses the NIS services' **hosts** database (built from **/etc/hosts** on the master server) to resolve the name to an IP address. This process works only if the node currently has a valid binding for the domain in question. In most cases, *server* should be specified as an IP address.

AUTHOR

ypset was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypwhich(1), ypserv(1M), ypfiles(4).

NAME

ypupdated, rpc.ypupdated - server for changing NIS information

SYNOPSIS

`/usr/lib/netsvc/yp/rpc.ypupdated [-is]`

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

`ypupdated` is a daemon that updates information in the **`Network Information Service (NIS)`** databases. It is activated at system startup when the **`NIS_MASTER_SERVER`** variable is set to 1 in **`/etc/rc.config.d/namesvrs`** file on the NIS master server. **`ypupdated`** consults the file **`updaters`** in the directory **`/var/yp`** to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (without augmented security).

Options

`ypupdated` supports the following options:

- **`-i`** Accept RPC calls with the insecure AUTH_UNIX credentials. This allows programmatic updating of the NIS maps in all networks.
- **`-s`** Accept only calls authenticated using the secure RPC mechanism (AUTH_DES authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

AUTHOR

`ypupdated` was developed by Sun Microsystems, Inc.

FILES

`/var/yp/updaters`

SEE ALSO

keyserv(1M), updaters(1M).

NAME

ypxfr, ypxfr_1perday, ypxfr_1perhour, ypxfr_2perday - transfer NIS database from server to local node

SYNOPSIS

/usr/sbin/ypxfr [**b**] [**-c**] [**-d** *domain*] [**-f**] [**-h** *host*] [**-s** *domain*] [**-C** *tid prog ipaddr port*] *mapname*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypxfr copies a Network Information Service (NIS) map (database) to the local host from a NIS server by using the NIS services. A map can be copied regardless of its age, or it can be copied depending on whether its modification time (order number) is more recent than that of the local map.

The **ypxfr** command creates a temporary map in directory **/var/yp/domain** where *domain* is the NIS *domain*. The **ypxfr** command fills the map with *mapname* entries, obtains the map parameters (master and order number), and loads them. It then clears the old version of *mapname* and moves the temporary map to the existing *mapname*.

If **ypxfr** is run interactively, it writes messages to standard output. If **ypxfr** is invoked without a controlling terminal and if the log file **/var/yp/ypxfr.log** exists, **ypxfr** appends all its messages to that file. Since **ypxfr** is usually run from root's **crontab** file (see *crontab*(1)) or by **yppush** (see *yppush*(1M)), the log file can retain a record of what **ypxfr** attempted and what the results were.

To maintain consistency between NIS servers, **ypxfr** should be executed periodically for every map in the NIS. Different maps change at different rates. For example, the **services.byname** map may not change for months at a time, and might therefore be checked for changes only once a day, such as in the early morning hours. However, **passwd.byname** may change several times per day, so hourly checks for updates might be more appropriate.

A **crontab** file can perform these periodic checks and transfers automatically. Rather than having a separate **crontab** file for each map, **ypxfr** requests can be grouped in a shell script to update several maps at once. Example scripts (mnemonically named) are in **/var/yp**: **ypxfr_1perday**, **ypxfr_2perday**, and **ypxfr_1perhour**. They serve as reasonable rough drafts that can be changed as appropriate.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of the Network Information Service.

Options

ypxfr recognizes the following options and command-line arguments:

-b Preserve the resolver flag in the map during transfer.

-c Do not send a "clear current map" request to the local **ypserv** process. Use this flag if **ypserv** is not running locally when you are running **ypxfr**. Otherwise, **ypxfr** complains that it cannot talk to the local **ypserv**, and the transfer fails. If **ypserv** is running locally, do not use this flag.

-d *domain* Copy the map from a NIS server in *domain* rather than the *domain* returned by **domainname** (see *domainname*(1)).

-f Force the map to be copied, even if its order number at the remote NIS server is not more recent than the order number of the local map.

-h *host* Obtain the map from *host*, regardless of its master server. If this option is not used, **ypxfr** asks the NIS service for the master's host name and tries to obtain its map. The *host* can be a name or an IP address of the form *a.b.c.d*.

-s *domain* Specify a source domain from which to transfer a map that should be the same across domains (such as the **services.byname** map.

-C *tid prog ipaddr port*
This option is used only by **ypserv**. When **ypserv** invokes **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process (that initiated the transfer) at the host with IP address *ipaddr*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.

AUTHOR

ypxfr was developed by Sun Microsystems, Inc.

FILES

/usr/sbin/ypxfr.log	log file

The following scripts are suggested for use with **cron**.

/usr/sbin/ypxfr_1perday	run one transfer per day
/usr/sbin/ypxfr_2perday	run two transfers per day
/usr/sbin/ypxfr_1perhour	hourly transfers of "volatile" maps

SEE ALSO

crontab(1), domainname(1), cron(1M), ypinit(1M), yppush(1M), ypserv(1M), ypfiles(4).