sun®
microsystems

Setting Up Your UNIX Environment: Beginner's Guide

Part No: 800-1285-03
Revision A of 17 February 1986

Credits and Trademarks

Contents

Preface

This manual describes the setup files for the C-Shell command interpreter, and the interactive programs `vi` and `mail`. Each of these files is read in automatically by the appropriate program, and contains commands and instructions to set up (or disable) various features of that program.

In this manual, the term *environment* is loosely defined as the various options and features that affect how the system or interactive program responds to you.

There is a sample of each setup file and a line-by-line explanation of its contents. Culled from a variety of expert users, these files contain some very convenient combinations of features and commands. Most importantly, these samples provide a starting point from which begin tailoring the system to your specific needs and style.

Another aspect of your environment is the file system and the current working directory. Included is an appendix to help in finding your way within the file system and its directories.

Chapter 1 is an overview of various setup files and a description of how they are used by the interactive programs.

Chapter 2 describes the `.cshrc` file for the C-Shell.

Chapter 3 describes the `.login` file for the C-Shell.

Chapter 4 describes the `.logout` file for the C-Shell.

Chapter 5 describes the `.exrc` file for `vi` (and the line editor `ex`).

Chapter 6 describes the `.mailrc` file for `mail`.

Appendix A is an outline of the directory hierarchy on a typical UNIX† system.

Online copies of the sample files are located in:

```
.cshrc       /usr/lib/Cshrc
.login       /usr/lib/Login
.logout      /usr/lib/Logout
.mailrc      /usr/lib/Mailrc
.exrc        /usr/lib/Exrc
```

† UNIX is a trademark of AT&T Bell Laboratories.

Prerequisite Documents	*Getting Started With UNIX: Beginner's Guide*
Companion Documents	*Solving Problems: Self Help for Beginners*
	Doing More With UNIX: Beginner's Guide
	Games, Demos and Other Pursuits: Beginner's Guide
	Using the Network: Beginner's Guide
	Commands Reference Manual for the Sun Workstation

1

Overview

1

Overview

If you have been reading through *Getting Started With UNIX* and have been using the system for a while, you have probably discovered features that you like and features you *would* like. The interactive programs that you have used so far have many optional features that you may not know about. This manual describes a number of these features, and how set things up so that you get the features you want automatically.

1.1. The UNIX Environment

When working with the system, the interactive program that is currently running on your terminal provides a context in which you accomplish your work. When you first log in, you are said to be "in" the command interpreter or *shell*. When you change directories, you are said to be "in" a new one. When using the text editor, you are said to be "in" `vi`.

While in the shell, you typically run the commands described in the *Commands Reference Manual*. When in `vi`, you typically use text editing commands to read and modify files as described in *Editing Text Files*. While in `Mail`, you typically use commands to read and dispose of messages, or to compose and post messages.

Actually *you* are probably in your office while all this computing activity is going on, but the metaphor is helpful.

In keeping with this analogy you can think of the environment as characteristics of the system and the current interactive program that affect the way you work.

The technical meaning with respect to UNIX is more restricted: the *environment* is a body of information that is inherited from the "parent" of a process (program currently running). For example, the name of the current directory is passed along when you start *vi*, so it appears that you "stay in the same directory." See *Doing More With UNIX* for more information about processes.

When you change interactive programs (by entering `Mail` for instance), some characteristics, such as the commands that are accepted, also change. Others, such as the current working directory, may not. But most importantly, you usually wish to perform different sorts of tasks, so your expectations about what is a proper response from the system also change.

1.2. Interactive Programs and Setup Files

The interactive programs that you use most often, such as the C-Shell command interpreter, `mail`, and `vi`, each have a variety of optional settings that affect the way they respond to your commands. Unlike options that you type in on the command line (such as `ls -t`), you typically select interactive features by typing in commands while you are using that program.[1]

To save you time, most interactive programs allow you to put a list of commands (to select features that you normally want in effect) in a *setup* file in your home directory. Each program reads its setup file(s) automatically and performs the commands it contains.

Subsequent chapters present samples of the various setup files for these three interactive programs. Each file is described command by command.

1.3. Installing the Setup Files In Your Home Directory

The sample files described in this manual are located in the directory `/usr/lib`, and you can install copies in your home directory. But before you do, check to see if there are setup files already present:

```
mars% cd
mars% ls .cshrc .exrc .login .logout .mailrc
.cshrc not found
.exrc not found
.login not found
.logout not found
.mailrc not found
```

CAUTION **If one or more of these files *are* present in your home directory, check with your System Administrator before you install the samples.**

To install the setup files, type in these commands:

```
cd
cp /usr/lib/Cshrc  .cshrc
cp /usr/lib/Exrc   .exrc
cp /usr/lib/Login  .login
cp /usr/lib/Logout .logout
cp /usr/lib/Mailrc .mailrc
```

Once installed, you can modify them as you like using `vi`, or any other text editor.

These samples have been culled from the setup files of a variety of expert users. They contain many useful features and ideas. Even so, you will want to edit them to suit your own personal needs and tastes, and to remove references to features that you don't want. Generally speaking, the smaller the setup file is, the faster the program will start up.

1.4. Setting Up Your Terminal

Underlying all of these considerations is the terminal you are using, and *its* characteristics. As indicated in *Getting Started With UNIX* you can assign terminal functions such as `erase` and `kill` (erase the entire line typed in so far) to control keys such as [DEL] and [BACKSPACE].[2]

[1] Most interactive commands also have command-line options that you can specify. Refer to the entry for the command of interest in the *Commands Reference Manual* for information about its command-line options.

[2] If you are using `suntools` on the Sun Workstation, you can assign commands and functions to the special function keys on the Sun Workstation keyboard. See *Windows and Window-Based Tools* for details.

The command `stty` provides you with a means to set up these and other terminal characteristics. To find out what your current terminal characteristics are, type in the command:

```
stty everything
```

This gives you a list of all terminal characteristics currently in effect.

```
new tty, speed 9600 baud
even odd -raw -nl echo -lcase -tandem tabs -cbreak
crt: (crtbs crterase crtkill ctlecho) -tostop
-tilde -flusho -mdmbuf -litout -nohang
-pendin decctlq -noflsh
erase  kill   werase rprnt  flush  lnext  susp   intr   quit stop   eof
^?     ^U     ^W     ^R     ^O     ^V     ^Z/^Y  ^C     ^  ^S/^Q  ^D
```

The command

```
stty all
```

displays a shorter list. For now, you can ignore all but the the last two lines,[3] which describe terminal-control functions and the keys they are set to. These are described below:

```
mars% stty all
new tty, speed 9600 baud; -tabs
crt
decctlq
erase  kill   werase rprnt  flush  lnext  susp   intr   quit   stop   eof
^?     ^U     ^W     ^R     ^O     ^V     ^Z/^Y  ^C     ^      ^S/^Q  ^D
```

`erase`
: *Erase character*. Backspace and erase one character. This is the DEL key by default on some keyboards. On others, this function is assigned to the BACKSPACE key.

`kill`
: *Kill the whole line*. Erase the entire command line typed in so far.

`werase`
: *Delete word*. Erase the rightmost word typed in so far (back to a space or tab); usually assigned to CTRL-W.

`rprnt`
: *Reprint*. Reprint the line typed in so far. This is useful when you type ahead and the prompt gets displayed in the middle of your text.

`flush`
: *Wait for a keystroke*. Stops terminal output until you press a key.

[3] For more information about the remaining terminal characteristics, refer to `stty` in the *Commands Reference Manual*.

`lnext`
: *Literal next-character.* Interprets the next (control) character as literal text.

`susp`
: *Suspend the program.* Temporarily halts execution of the program currently running and puts it in the background. To resume execution of the command, type `%`.[4] When you type the suspend character, usually CTRL-Z, in the middle of a C-Shell command-line, the shell ignores that line and issues a new prompt.

`intr`
: *Interrupt.* Interrupt the program currently running.

`quit`
: Halt the current program and leave a binary image in a file called `core`.

`stop`
: *Stop the display.* To resume, press CTRL-Q.

`eof`
: *End-of-file.* Send the program an end-of-file character.

To assign any of these functions to another control key, supply the function, a space and the circumflex character (^), followed by the new key, as an arguments to `stty`.

```
mars% stty erase ^j
```

This avoids problems trying to type in a key that is already assigned.

To assign `erase` to the BACKSPACE key, use the command:

```
stty erase ^h
```

To assign `werase` to the DEL key, use:

```
stty werase ^?
```

To assign `kill` to the ESC key, use:

```
stty kill ^[
```

Chapter 3 has more information about setting up terminal characteristics. For a description of other terminal characteristics that you can set up, refer to `stty` in the *Commands Reference Manual*. To assign commands to the special function keys on the workstation keyboard, refer to *Windows and Window-Based Tools*.

[4] C-Shell only

1.5. The File System Hierarchy

As mentioned above, the file system's directory hierarchy is a part of the "landscape" that you will want to become familiar with. Appendix A outlines the organization of directories for a typical UNIX† system.

† UNIX is a trademark of AT&T Bell Laboratories.

2

The C-Shell and the `.cshrc` File

2

The C-Shell and the `.cshrc` File

The C-Shell is one of the two command interpreters available on the Sun Workstation, and the one that we recommend for interactive use. Whenever you start running the C-Shell, such as when you log in or open a terminal (`shelltool`) window, the C-Shell looks for the `.cshrc` file in your home directory for its initial instructions. You can include in this file any command that you might ordinarily type on the command line.

The name is derived from `csh`, which is the program that uses it. The `rc` suffix is derived from the term "run command." Setup files ending in this suffix are read at the time you *run* the *command.* The "dot" at the beginning of the filename indicates that this file is to remain hidden from view when you do an `ls`. Setup files are rarely of interest unless you are editing them specifically, and in that case you already know the filename and directory location. (To list hidden files, use the `-a` option of `ls`.)

2.1. Selecting C-Shell Features

While in the C-Shell, you can use the `set` command to select the options you would like. For instance, if you want the C-Shell to prevent you from accidentally logging out by typing a CTRL-D, you can set the `ignoreeof` option (or, technically speaking, *variable*):

```
mars% set ignoreeof
mars% ^D
Use "exit" to leave csh.
```

To turn off an option, use the `unset` command:

```
mars% unset ignoreeof
```

Some options allow you to supply a specific number or value. For instance, you can use the `history` variable to select the size of the history list; that is, the number of previous commands to remember:

```
mars% set history = 40
```

Or, you can alter the prompt that the shell displays:

```
mars% set prompt = "THIS IS A VERY LONG PROMPT: "
THIS IS A VERY LONG PROMPT: ■
```

To see what options are currently in effect, and their values (if any) use the `set` command with no arguments:

```
mars% set
argv    ()
cdpath  (. .. /usr/sam /usr/sam/bin /usr/sam/src)
cwd     /usr/sam/env
history 40
home    /usr/sam
ignoreeof
noclobber
notify
path    (. /usr/sam /usr/sam/bin /usr/local /usr/ucb /usr/bin /bin)
prompt  mars%
shell   /bin/csh
status  0
term    sun
user    sam
```

You can find descriptions of all C-Shell options (predefined variables)in Appendix E of *Doing More With UNIX*, or refer to `csh` in the *Commands Reference Manual*.

2.2. A Sample `.cshrc` File

The following pages contain an annotated listing of the sample `.cshrc` file located in `/usr/lib/Cshrc`. This is a very large sample file. Many of the commands and features it includes may not pertain to you, and we recommend that you delete those that you don't from your copy of the file.

A number of commands have been "commented out" by placing a pound-sign (#) to their left. The C-Shell will ignore these commands unless you remove the pound-sign character. Commands that are listed but commented out are felt to be interesting and educational, but not necessarily those that a beginner would use.

```
#################################################################
#
#          .cshrc file
#
#          initial setups for both interactive and noninteractive
#          C-Shells
#
#################################################################


#          set up search path

set lpath = ( )  #  add directories for local commands
set path = (. ~ ~/bin /usr/local /usr/ucb /usr/bin /bin)
set path = ($path[1-3] $lpath $path[4-])
#set path = ($path /etc /usr/etc)

#          cd path

set lcd = ( )  #  add parents of frequently used directories
cdpath = (.. ~ ~/bin ~/src $lcd)

#          set this for all shells

set noclobber

#          aliases for all shells

alias cd            'cd \!*;echo $cwd'
alias cp            'cp -i'
alias mv            'mv -i'
alias rm            'rm -i'
alias pwd           'echo $cwd'
#alias del          'rm -i'
#umask 002

#          skip remaining setup if not an interactive shell

if ($?USER == 0 || $?prompt == 0) exit

#           settings  for interactive shells

set history=40
set ignoreeof
#set notify
set savehist=40
#set prompt="% "
#set prompt="`hostname`{`whoami`}\!: "
#set time=100
```

```
#           commands for interactive shells

#date
#pwd

#          other aliases

#alias a               alias
#alias h               'history \!* | head -39 | more'
#alias u               unalias

alias ^L               clear
#alias list            cat
alias lock             lockscreen
alias m                more
alias mroe             more
#alias type            more

alias .                'echo $cwd'
alias ..               'set dot=$cwd;cd ..'
alias ,                'cd $dot '

#alias dir             ls
alias pdw              'echo $cwd'
#alias ff              'find . -name \!* -print'
alias la               'ls -a'
alias ll               'ls -la'
#alias ls              'ls -F'

#alias pd              dirs
#alias po              popd
#alias pp              pushd

alias open             'chmod go+r'
alias shut             'chmod go-r'
alias x                'chmod +x'

alias j                'jobs -l'
alias f                'fg %\!*'
alias lo               logout

alias bye              logout
alias ciao             logout
alias die              logout

#alias k               kill
alias psg              'ps -ax | grep \!* | grep -v grep'
alias punt             kill
```

```
#alias r              rlogin
#alias run            source
#alias slay 'set j=`ps -ax|grep \!*|head -1`; kill -9 `echo $j[1]`'

alias nms 'tbl \!* | nroff -ms | more'                  # nroff -ms
alias tms 'tbl \!* | troff -t -ms >! troff.output &'    # troff -ms
alias tpr 'tbl \!* | troff -t -ms | lpr -t &'           # troff & print
alias ppr 'lpr -t \!* &'                                # print troffed

alias lp1             'lpr -P1'
alias lp2             'lpr -P2'

alias lq1             'lpq -P1'
alias lq2             'lpq -P2'

alias lr1             'lprm -P1'
alias lr2             'lprm -P2'

#alias sd             'screendump | rastrepl | lpr -v &'

#alias c              'cc \!1.c \!:2* -o \!1 >>& c.errors'
#alias ccc            'cc \!*.c -o \!*'
#alias edit           textedit

alias fem             man
alias help            man
alias key             'man -k'

alias mkae            make
```

Explanation of Command Lines

Line 1:

```
set lpath = ( )   #  add directories for local commands
```

Creates a variable in which to add the pathanes of directories containing local commands. Add the pathnames for any such directories between the parentheses (ask your System Administrator for the appropriate names). These directories are incorporated into the `path` variable in line *3*.

Line 2:

```
set path = (. ~ ~/bin /usr/local /usr/ucb /usr/bin /bin)
```

Sets the `path` variable to include the standard directories containing UNIX commands.

Line 3:

```
set path = ($path[1-3] $lpath $path[4-])
```

Inserts the directories from `lpath` into the search path in their proper place (after `.`, `~` and `~/bin`, but before the remaining directories. When your system has locally defined versions of existing programs, you often want those versions to be selected ahead of the standard versions.

Line 4:

```
#set path = ($path /etc /usr/etc)
```

This line is "commented out." The # as the leftmost character instructs the C-Shell to ignore any remaining characters on the line (in this case the entire line). You can activate the line by deleting the #. When active, this line adds the directories `/etc` and `/usr/etc` to your search path. These directories contain system administration commands.

Line 5:

```
set lcd = ( )   #  add parents of frequently used directories
```

Creates a variable in which to add the pathnames of directories that are *parents* of those that you often `cd` to. For instance, if you often `cd` to `/usr/man/man1`, add put the pathname `/usr/man` between the parentheses. These directories are added to the `cdpath` variable in the next line.

Line 6:

```
cdpath = (.. ~ ~/bin ~/src $lcd)
```

Sets the `cdpath` variable. You need not specify pathnames when you `cd` to directories that are contained in any of those listed. With `..` set in your `cdpath` (as above), if you were in `/usr/man/man1` and you wanted to `cd` to `/usr/man/cat1`, you could use the command `cd cat1` to do so.

Line 7:

```
set noclobber
```

Prevents unintentional overwrites of files when you use the > symbol. See *Doing More With UNIX* for details.

Line 8:

```
alias cd        'cd \!*;echo $cwd'
```

Displays the new directory when you use `cd`.

Line 9:

```
alias cp        'cp -i'
```

Asks for confirmation before overwriting existing files with `cp`.

Line 10:

```
alias mv        'mv -i'
```

Asks for confirmation before overwriting existing files with `mv`.

Line 11:

```
alias rm        'rm -i'
```

Asks for confirmation before removing files.

Line 12:

```
alias pwd       'echo $cwd'
```

A faster way of seeing the current working directory.

Line 13:

```
#alias del       'rm -i'
```

A name for `rm` that is familiar to PC users. (Commented out.)

Line 14:

```
#umask 002
```

Sets the default permissions mask for new files to allow read and write access to the group as well as the owner. (Commented out.)

Line 15:

```
if ($?USER == 0 || $?prompt == 0) exit
```

Tests to see whether there is a variable called `USER`, or a variable called `prompt` currently set. If not, then the C-Shell stops processing commands from this file.

Line 16:

```
set history=40
```

The C-Shell records the last 40 commands typed in.

Line 17:

```
set ignoreeof
```

Prevents accidental logouts when you type [CTRL-D].

Line 18:

```
#set notify
```

Prevents waiting for display of C-Shell messages. Normally, the C-Shell waits until just before printing the prompt to print its messages. Commands, however, don't always wait to print their messages, so setting `notify` means that all messages will work the same way. (Commented out.)

Line 19:

```
set savehist=40
```

When you log out, the C-Shell saves the last 40 commands, and uses them as the starting history list for your next session.

Line 20:

```
#set prompt="% "
```

An alternate prompt favored by some UNIX wizards. (Commented out.)

Line 21:

```
#set prompt="`hostname`{`whoami`}\!: "
```

An alternate prompt favored by some network wizards. (Commented out.)

Line 22:

```
#set time=100
```

Display time statistics for commands that take longer than 100 CPU seconds. (Commented out.)

Line 23:

```
#date
```

Display the date and time when the C-Shell starts up. (Commented out.)

Line 24:

```
#pwd
```

Display the working directory when the C-Shell starts up. (Commented out.)

Line 25:

```
alias a         alias
```

Abbreviate the `alias` command.

Line 26:

```
#alias h        'history \!* | head -39 | more'
```

Abbreviate `history`, and delete the last line (containing `h`) from the display. (Commented out.)

Line 27:

```
#alias u        unalias
```

Abbreviate the `unalias` command. (Subsequent commented-out lines are not highlighted in the explanation, but can be recognized as starting with a pound-sign.)

Line 28:

```
alias ^L        clear
```

The CTRL-L character is character often used to begin a new page or clear the current one. This alias mimics that behavior.

Line 29:

```
#alias list      cat
```

A name for `cat` that is familiar to PC users.

Line 30:

```
alias lock      lockscreen
```

An abbreviation for `lockscreen`.

Line 31:

```
alias m         more
```

An abbreviation for `more`.

Line 32:

```
alias mroe      more
```

A remedy for "fat fingers."

Line 33:

```
#alias type      more
```

A name for `more` that is familiar to PC users.

Line 34:

```
alias .         'echo $cwd'
```

An abbreviation for `pwd`.

Line 35:

```
alias ..        'set dot=$cwd;cd ..'
```

A quick way to change from child directory to parent (and back again with alias on the next line).

Line 36:

```
alias ,         'cd $dot '
```

A quick way to change back after using the `..` alias above.

Line 37:

```
#alias dir       ls
```

A name for `ls` that is familiar to PC users.

Line 38:

```
alias pdw       'echo $cwd'
```

A remedy for fat fingers. Same as the alias for `pwd`.

Line 39:

```
#alias ff       'find . -name \!* -print'
```

Find a named file in any subdirectory of the current one.

Line 40:

```
alias la        'ls -a'
```

Abbreviation for command to list all filenames, including those that begin with a dot (`.`).

Line 41:

```
alias ll        'ls -la'
```

Abbreviation for a command to give a long listing of filenames, including those that begin with a dot.

Line 42:

```
#alias ls       'ls -F'
```

`ls` appends characters on the end of a filename to indicate that file's type.

Line 43:

```
#alias pd        dirs
```

Abbreviation to display the directory stack maintained by `pushd` and `popd`. See *Doing More With UNIX* for details.

Line 44:

```
#alias po        popd
```

Change directories to the one on the top of the stack, and remove its name from the stack.

Line 45:

```
#alias pp        pushd
```

Change directories, adding the current directory and the destination to the stack.

Line 46:

```
alias open      'chmod go+r'
```

Make a file readable to the group and public.

Line 47:

```
alias shut      'chmod go-r'
```

Make a file unreadable to all but you, the owner.

Line 48:

```
#alias x         'chmod +x'
```

Give a file execute permissions for all users.

Line 49:

```
alias j         'jobs -l'
```

Display the list of background jobs.

Line 50:

```
alias f         'fg %\!*'
```

Bring a job to the foreground.

Line 51:

```
alias lo        logout
```

Abbreviation for `logout`. Particularly nice when `ignoreeof` is set.

Line 52:

```
alias bye       logout
```

Another name for `logout`.

Line 53:

```
alias ciao      logout
```

Yet another name for `logout`.

Line 54:

```
alias die       logout
```

You guessed it.

Line 55:

```
#alias k        kill
```

Abbreviation for `kill`, the command to halt a process. See *Doing More With UNIX* for details.

Line 56:

```
alias psg       'ps -ax | grep \!* | grep -v grep'
```

Check on the status of a command by name. See *Processes and Other Users* in *Doing More With UNIX* for details.

Line 57:

```
alias punt      kill
```

Another name for `kill`.

Line 58:

```
#alias r        rlogin
```

Log in to another host machine on the net. See *Using the Network* for details.

Line 59:

```
#alias run       source
```

The `source` command instructs the C-Shell to take a file (such as the `.cshrc` file) as a list of commands to perform.

Line 60:

```
#alias slay 'set j=`ps -ax|grep \!*|head -1`; kill -9 `echo $j[1]`'
```

Kill a running command by name. Attempts to kill the first such command encountered.

Line 61:

```
alias nms 'tbl \!* | nroff -ms | more'
```

Format and display a document containing `tbl` instructions and `ms` macros on the terminal.

Line 62:

```
alias tms 'tbl \!* | troff -t -ms >! troff.output &'
```

Format a document using `tbl` and `ms` macros, and place the output in a file for later printing.

Line 63:

```
alias tpr 'tbl \!* | troff -t -ms | lpr -t &'
```

Format a document using `tbl` and `ms` macros and print it.

Line 64:

```
alias ppr 'lpr -t \!* &'
```

Print a preformatted troff-output file.

Line 65:

```
alias lp1       'lpr -P1'
```

Abbreviation to print on printer #1. See *Doing More With UNIX* for details.

Line 66:

```
alias lp2       'lpr -P2'
```

Abbreviation to print on printer #2.

Line 67:

```
alias lq1       'lpq -P1'
```

Abbreviation to check the queue for printer #1.

Line 68:

```
alias lq2       'lpq -P2'
```

Abbreviation to check the queue for printer #2.

Line 69:

```
alias lr1       'lprm -P1'
```

Abbreviation to remove a job or jobs from printer #1.

Line 70:

```
alias lr2       'lprm -P2'
```

Abbreviation to remove a job or jobs from printer #2.

Line 71:

```
#alias sd        'screendump | rastrepl | lpr -v &'
```

Abbreviation to print the contents of the Workstation screen.

Line 72:

```
#alias c        'cc \!1.c \!:2* -o \!1 >>& c.errors'
```

Abbreviation to run the C compiler and save compilation error messages.

Line 73:

```
#alias ccc      'cc \!*.c -o \!*'
```

Abbreviation to run the C compiler.

Line 74:

```
#alias edit     textedit
```

Abbreviation for the window-system text editor.

Line 75:

```
alias fem       man
```

Another name for the `man` command.

Line 76:

```
alias help      man
```

Yet another name for the `man` command.

Line 77:

```
alias key       'man -k'
```

Abbreviation for the `man -k` command (same as the `whatis` command described in *Doing More With UNIX*)

Line 78:

```
alias mkae      make
```

A remedy for fat fingers.

3

The C-Shell and the `.login` File

3

The C-Shell and the `.login` File

When you log in, after performing instructions in the `.cshrc` file, the C-Shell then performs instructions in the `.login` file. Subsequent C-Shells, such as those running within terminal (`shelltool`) windows, ignore the `.login` file.

Like the `.cshrc` file, you can include any command that you might type in on the command line. However, we reccommend that you use the `.login` file for initializing remote terminals (for when you log in by phone), starting your window system (when you first log in to the workstation), and setting up special variables called *environment* variables. Unlike shell variables, environment variables are passed along to subsequent commands and programs automatically. You need not set them up again every time you start a new C-Shell or run a new program such as `vi`.

Environment variables are useful for storing information that all programs need to know about. For instance, many commands and programs need to know what type of terminal you are using. This information is stored in the `TERM` environment variable. Commands that send output to the printer need to know which printer to send their output to. You can use the `PRINTER` environment variable, to store the name of a printer to use by default. If you want to specify an alternate font for window-system displays, you can set the `DEFAULT_FONT` environment variable to the name of a file containing that font.

To set an environment variable, use the `setenv` command. This command has two required arguments, the *name* of the variable, and its *value*.

`setenv` *name value*

For example

```
mars% setenv PRINTER lw
```

Although not required, the convention is to use all capitals for names of environment variables (to distinguish them from ordinary shell variables). To see what environment variables are currently in effect, use the `printenv` command:

```
mars% printenv
HOME=/usr/sam
SHELL=/bin/csh
PATH=.:/usr/sam:/usr/sam/bin:/usr/local:/usr/ucb:/usr/bin:/bin
TERM=sun
USER=sam
DEFAULT_FONT=/usr/lib/fonts/fixedwidthfonts/serif.r.11
EDITOR=/usr/ucb/vi
PRINTER=lw
WINDOW_PARENT=/dev/win0
WMGR_ENV_PLACEHOLDER=/dev/win1
WINDOW_ME=/dev/win9
WINDOW_GFX=/dev/win9
```

To remove an environment variable, use the `unsetenv` command:

```
mars% unsetenv PRINTER
mars% echo $PRINTER
PRINTER: Undefined variable.
```

3.1. A Sample `.login` File

The following pages contain an annotated listing of the sample `.login` file located in `/usr/lib/Login`. If you do not plan to log in from a remote terminal over the phone, you can delete the lines that pertain to remote terminals. Again, some commands are commented out. And again, we recommend that you delete commands that do not pertain to you.

```
###################################################################
#
#          .login file
#
#          Read in after the .cshrc file when you log in.
#          Not read in for subsequent shells.  For setting up
#          terminal and global environment characteristics.
#
###################################################################


#          terminal characteristics for remote terminals:

#          Leave lines for all but your remote terminal commented
#          out (or add a new line if your terminal does not appear).

if ($TERM != "sun") then
set noglob
#eval `tset -sQ -m dialup:?925 -m switch:?925 -m dumb:?925 $TERM`
#eval `tset -sQ -m dialup:?h19 -m switch:?h19 -m dumb:?h19 $TERM`
#eval `tset -sQ -m dialup:?mac -m switch:?mac -m dumb:?mac $TERM`
#eval `tset -sQ -m dialup:?vt100 -m switch:?vt100 -m dumb:?vt100 $TERM`
#eval `tset -sQ -m dialup:?wyse-nk -m switch:?wyse-nk -m dumb:?wyse-nk $TERM`
#eval `tset -sQ -m dialup:?wyse-vp -m switch:?wyse-vp -m dumb:?wyse-vp $TERM`
unset noglob
endif

#          general terminal characteristics

#stty -crterase
#stty -tabs
#stty crt
#stty erase '^h'
#stty werase '^?'
#stty kill '^['
#stty new

#          environment variables

#setenv DEFAULT_FONT "/usr/lib/fonts/fixedwidthfonts/screen.r.11"
#setenv EXINIT 'set sh=/bin/csh sw=4 ai report=2'
setenv MORE '-c'
#setenv PRINTER lw

#          commands to perform at login

#echo "!=<"      # turn off key click

#w
```

```
if ("`tty`" != "/dev/console") exit
echo -n "Suntools? (^C to interrupt)
sleep 5
suntools
```

Explanation of Command Lines

Line 1:

```
if ($TERM != "sun") then
```

Perform the commands between this line and the `endif` line only when logging in on a terminal, than a Sun Workstation.

Line 2:

```
set noglob
```

Turn off filename substitution.

Line 3:

```
#eval `tset -sQ -m dialup:?925 -m switch:?925 -m dumb:?925 $TERM`
```

If logging in over a phone line, or some other remote means, set up terminal characteristics for a Televideo 925 terminal and place these characteristics in the environment for faster startup of interactive programs. Asks for confirmation before performing this set-up. If you respond with an `n`, terminal characteristics are set to those of the Workstation. Refer to `tset` in the *Commands Reference Manual* for more information.

All of the lines pertaining to specific terminal types are commented out. To activate the line that pertains to your terminal, remove the pound-sign. If your terminal does not appear, duplicate this line, change the `925` to the name of your terminal (see your System Administrator for this information) and remove the pound-sign.

Line 4:

```
#eval `tset -sQ -m dialup:?h19 -m switch:?h19 -m dumb:?h19 $TERM`
```

Set up terminal characteristics for a Heathkit H19 terminal.

Line 5:

```
#eval `tset -sQ -m dialup:?mac -m switch:?mac -m dumb:?mac $TERM`
```

Set up terminal characteristics for a Macintosh running Macterminal.

Line 6:

```
#eval `tset -sQ -m dialup:?vt100 -m switch:?vt100 -m dumb:?vt100 $TERM`
```

Set up terminal characteristics for a VT100 terminal.

Line 7:

```
#eval `tset -sQ -m dialup:?wyse-nk -m switch:?wyse-nk -m dumb:?wyse-nk $TERM`
```

Set up terminal characteristics for a Wyse 50 terminal.

Line 8:

```
#eval `tset -sQ -m dialup:?wyse-vp -m switch:?wyse-vp -m dumb:?wyse-vp $TERM`
```

Set up terminal characteristics for a Wyse 50 in ADDS Viewpoint mode with "enhance" turned on.

Line 9:

```
unset noglob
```

Restore filename substitution.

Line 10:

```
endif
```

Marks last line to be skipped when an `if ... then` statement is found to be false; in this case, when logging in to a Sun Workstation directly (or from another Sun on the network).

Line 11:

```
#stty -crterase
```

Set up the erase function to backspace without blotting out erased characters. Erased characters remain visible on the screen until you overwrite them with new ones, but are not transmitted to the C-Shell when you press [RETURN].

Line 12:

```
#stty -tabs
```

Convert tabs to spaces when displayed on the screen.

Line 13:

```
#stty crt
```

Set up standard CRT characteristics.

Line 14:

```
#stty erase '^h'
```

Set the erase character to [BACKSPACE]. Note that with `stty`, control characters are indicated by the two-character symbol *circumflex-character*: `^c`.

Line 15:

```
#stty werase '^?'
```

Set the erase-word character to [DEL].

Line 16:

```
#stty kill '^['
```

Set line kill character to [ESC].

Line 17:

```
#stty new
```

Use the new version of the terminal driver.

Line 18:

```
#setenv DEFAULT_FONT "/usr/lib/fonts/fixedwidthfonts/screen.r.11"
```

Set up a default font for window-system tools and displays.

Line 19:

```
#setenv EXINIT 'set sh=/bin/csh sw=4 ai report=2'
```

Another way to set up options for `vi`. If you use the `EXINIT` environment variable, `vi` ignores your `.exrc` file.

Line 20:

```
setenv MORE '-c'
```

Sets up `more` to overwrite the screen rather than scrolling. This makes reading `more` output much easier.

Line 21:

```
#setenv PRINTER lw
```

Indicate which printer is to receive jobs by default.

Line 22:

```
#echo "!=<"            # turn off key click
```

If your Workstation keyboard has keys that click, this command turns of the clicking.

Line 23:

```
#w                     # see who is logged in
```

See who is logged in on your system.

Line 24:

```
if ("`tty`" != "/dev/console") exit
```

If the terminal is not your Workstation console (the Workstation when not running the Window-system), then stop further processing of this file.

Line 25:

```
echo -n "Suntools? (^C to interrupt)
```

Warn you that `suntools` is about to start.

Line 26:

```
sleep 5
```

Wait 5 seconds before starting `suntools` to give you a chance to press CTRL-C

Line 27:

```
suntools
```

Start the window system.

4

The C-Shell and the `.logout` File

4

The C-Shell and the .logout File

When you log out completely (not just from a single window), the C-Shell performs instructions in the .logout file. This file is useful for running housekeeping type commands in the background while you are away.

Like .cshrc and .login you can include any command that you might type in on the C-Shell command line. We recommend that you use this file only for displaying information about the session just ending that you want to know about, and running background commands. You should *not* put commands that run interactively in this file, nor should you include commands that take any significant amount of time unless the command runs in the background. Otherwise someone may be able to interrupt the command and gain unauthorized access to your workstation or terminal.

4.1. A Sample .logout File

The following pages contain an annotated listing of the sample .logout file located in /usr/lib/Logout. Some commands are commented out, and we recommend that you delete commands that do not pertain to you.

```
############################################################
#
#          .logout file
#
#          Read in when you exit from the login shell.
#          For performing housekeeping while your are away.
#
############################################################


clear                                                              1
echo "`hostname`: `whoami` logged out at `date`                    2
#echo "Goodbye\!"                                                  3

if (-e /usr/games/fortune) /usr/games/fortune -a                   4
#if (-r /etc/motd) cat /etc/motd                                   5

#unalias rm                                                        6
#nice find ~ '(' -name core -o -name '*.BAK' -o -name '*.CKP' \    7
#          -o -name '#*' -o -name junk ')' \
#          -atime +3 -mtime +3 -user $USER -type f -exec \rm '{}' \; &
```

Explanation of Command Lines

Line 1:

```
clear
```

Clears the terminal screen.

Line 2:

```
echo "`hostname`: `whoami` logged out at `date`
```

Displays the name of the host machine, your user name, and the date and time you logged out.

Line 3:

```
#echo "Goodbye\!"
```

A more traditional parting wish.

Line 4:

```
if (-e /usr/games/fortune) /usr/games/fortune -a
```

If the `fortune` command is available, use it to display one of many humorous sayings.

Line 5:

```
#if (-r /etc/motd) cat /etc/motd
```

If the message of the day is readable, display it.

Line 6:

```
#nice find ~ '(' -name core -o -name '*.BAK' -o -name '*.CKP'
#       -o -name '#*' -o -name junk ')' \
#       -atime +3 -mtime +3 -user $USER -type f -exec \rm '{}
```

Run `find` at low priority in the background, starting with your home directory. Look for files named `core`, `*.BAK`, `*.CKP`, `'#*'` or `junk`. Of these, select only those that are at least 3 days old, haven't been modified for at least 3 days, belong to you, and are regular files (not directories). Remove each file selected, escaping any aliases that might be applied to `rm`.

To activate this command, you need to delete the first pound-sign in all three lines.

5

vi and the .exrc File

5

vi and the .exrc File

Whenever you run vi, the editor looks in the .exrc file for initial commands and option settings. The vi editor has a number of options that are described in detail in *Editing Text Files*. vi has a :set command with which you select the editing options that you want, but you cannot use it to create new variables, as you can with the C-Shell's set command.

5.1. Setting Options While in vi

To see the list of options that are currently in effect, type in :set with no arguments:

```
...
~
~
~
~
~
~
~
~
:set
autoindent beautify nomesg number redraw term=sun wrapmargin=8
```

To see the list of all possible settings, use the :set all command:

```
...
~
:set all
:set all
autoindent              open                        tabstop=8
autoprint               nooptimize                  taglength=0
noautowrite             paragraphs=IPLPPPQPP LIpplpipbp tags=tags /usr/lib/tags
beautify                prompt                      tagstack
directory=/tmp          noreadonly                  term=sun
noedcompatible          redraw                      noterse
noerrorbells            remap                       timeout
hardtabs=8              report=5                    ttytype=sun
noignorecase            scroll=16                   warn
nolisp                  sections=NHSHH HUnhsh       window=33
nolist                  shell=/bin/csh              wrapscan
magic                   shiftwidth=8                wrapmargin=8
nomesg                  noshowmatch                 nowriteany
nonumber                noslowopen
Hit return to continue
```

To select a specific option or options, include them as arguments to the `:set` command. Note that for options having values, there are *no* spaces between the name, the equal-sign, and the value for that option. When the value for an option includes spaces, such as that for `sections` above, the space is escaped with a backslash within the command:

```
:set sections=NHSHH\ HUnhsh
```

To turn off an option, add the prefix `no` to the name of that option as an argument to `:set`.

```
:set
autoindent beautify nomesg number redraw term=sun wrapmargin=8
```

```
:set noautoindent
:set
beautify nomesg number redraw term=sun wrapmargin=8
```

To change the value of a setting such as `wrapmargin`, use `set` to establish a new value:

```
:set wrapmargin=0
```

(This has the effect of eliminating automatic wrapping at the end of the line).

5.2. A Sample .exrc File

The following page contains an annotated listing of the sample .exrc file located in /usr/lib/Exrc. Since vi does not accept comments as with the C-Shell, there are no lines commented out. So, you may wish to delete all lines starting with :map, and add them (or others like them) when you have read through *Editing Text Files*.

```
set autoindent
set autoprint
set noignorecase
set nomesg
set noslowopen
set noterse
set nonumber

set report=2
set shell=/bin/csh
set tabstop=4
set wrapmargin=8

map ; :
map g :%
map v ~
map F !} fmt -c
map FF !G fmt -c
map S !} sort
map SS !G sort
map T :r!

map! ;b \fB
map! ;i \fI
map! ;p \fP
map! ;r \fR
map! ;- \-
map! ;u \s-2UNIX\s+2
```

Explanation of Command Lines

Line 1:

```
set autoindent
```

When adding new lines, maintain the same indention as the line above.

Line 2:

```
set autoprint
```

Automatically print each line altered within ex, the line editor.

Line 3:

```
set noignorecase
```

The case (upper or lower) of a character is significant in searches and substitutions. Use `set ignorecase` to make searches and substitutions case insensitive. But be careful if you do!

Line 4:

```
set nomesg
```

Messages to the terminal do not interfere with the vi display.

Line 5:

```
set noslowopen
```

Sets up vi for operation with a fast terminal or window. For terminals on slow dialup lines, use `set slowopen` to suspend updates of the screen during insertions for smoother operation.

Line 6:

```
set noterse
```

vi gives more complete error messages for beginning users. For shorter messages, use .setterse

Line 7:

```
set nonumber
```

Inhibits display of line numbers in both ex and vi. For a display of line numbers, use `set number`.

Line 8:

```
set report=2
```

Report on all substitutions or deletions that affect more than two lines.

Line 9:

```
set shell=/bin/csh
```

Set the shell to be a C-Shell for `!` escapes. Refer to *Editing Text Files* for more infomration.

Line 10:

```
set tabstop=4
```

Set tab stops every 4 characters.

Line 11:

```
set wrapmargin=8
```

When a space is typed within 8 characters of the right screen edge, insert a carriage-return at the end of the previous word, starting a new line automatically.

Line 12:

```
map ; :
```

While in `vi` command (*visual*) mode, interpret a semicolon as if you had typed a colon. This allows you to use either a semicolon or a colon as the first character in a substitution command.

Line 13:

```
map g :%
```

Wile in visual mode, interpret a `g` as if you typed the characters `:%`. This allows you to start commands to substitute throughout the file with either a `g` or a `:%`.

Line 14:

```
map v ~
```

While in visual mode, interpret a `v` as if you typed a `~`, the command to invert the case of a character.

Line 15:

```
map F !} fmt -c
```

While in visual mode, interpret an `F` as if you typed in the command

```
!} fmt -c
```

to adjust line-breaks for the lines between the cursor and the endo of the paragraph as close to column 80 as possible (without breaking across words). Refer to *Editing Text Files* and `fmt` in the *Commands Reference Manual* for more information.

Line 16:

```
map FF !G fmt -c
```

While in visual mode, interpret the characters FF as if you typed in

```
!G fmt -c
```

a command to right-adjust the contents of the file from the current line to the end.

Line 17:

```
map S !} sort
```

While in visual mode, interpret an S as if you typed in the command

```
!} sort
```

a command to sort the remaining lines in the paragraph.

Line 18:

```
map SS !G sort
```

While in visual mode, interpret the characters SS as if you typed in

```
!G sort
```

a command to sort the remaining lines in the file.

Line 19:

```
map T :r!
```

While in visual mode, interpret a T as if you typed in a :r!, which when followed by a shell command, inserts the output of that command into the file (after the current line). T is used because both r and R are already vi commands.

Line 20:

```
map! :b \fB
```

While in vi *append* mode (notice the exclamation point), interpret the character sequence ;b as if the string \fB were typed. When you press ; and then b in rapid succession, the editor appends the characters \fB (a troff command to change to **bold** font) in their place. This can make preparation of troff input files with complicated font changes much easier.

Line 21:

```
map! ;i \fI
```

While in vi append mode, interpret the sequence ;i as if \fI were typed (troff change to *italic* font).

Line 22:

```
map! ;p \fP
```

While in vi append mode, interpret the sequence ;p as if \fP were typed (troff change to previous font).

Line 23:

```
map! ;r \fR
```

While in `vi` append mode, interpret the sequence `;r` as if `\fR` were typed (`troff` change to roman font).

Line 24:

```
map! ;- \-
```

While in `vi` append mode, interpret the sequence `;-` as if `\-` were typed (`troff` minus-sign).

Line 25:

```
map! ;u \s-2UNIX\s+2
```

While in `vi` append mode, interpret the sequence `;u` as if `\s-2UNIX\s+2` were typed. This slightly reduces the point size of the word UNIX on the page, and makes for a better-looking line when formatted through `troff`.

6

Mail and the .mailrc File

6

Mail and the .mailrc File

When you run Mail, the program looks for the .mailrc file for initial option settings. Mail has a number of options that are described in detail in or*Mail and Messages*, Mail in the *Commands Reference Manual*.

NOTE *If you are using the* suntools *window system, then we recommend that you use* mailtool *instead of* Mail. *Refer to Windows and Window-Based Tools for information on how to set up and use* mailtool.

Like the C-Shell, Mail has a set command with which you select options or create new variables.

6.1. Setting Options While in mail

To see the list of options that are currently in effect, type in the set command with no arguments:

```
& set
DEAD="~/dead.letter"
EDITOR="/usr/ucb/ex"
MAILRC="/usr/titan/rdh/.mailrc"
MBOX="/usr/titan/rdh/mbox"
PAGER="cat -s | more -22"
SHELL="/bin/csh"
VISUAL="/usr/ucb/vi"
alwaysignore
append
askcc
asksub
autoprint
cmd="lpr -p &"
crt="15"
dot
header
hold
keep
keepsave
metoo
prompt="&"
record="~/mbox"
save
&
```

To select a specific option or options, include them as arguments to the `set` command. Note that for options having values, there are *no* spaces between the name, the equal-sign, and the value for that option. As shown above, when a value contains spaces, you can surround it with quotes.

To turn off a `mail` option, use the `unset` command, followed by the name of the option (variable).

6.2. A Sample `.mailrc` File

The following page contains an annotated listing of the sample `.mailrc` file located in `/usr/lib/Mailrc`. As with `vi`, `mail` does not accept comments. So you will need to delete from your copy any lines that you don't want.

```
set alwaysignore                                                   1
set askcc                                                          2
set asksub                                                         3
set autoprint                                                      4
set cmd="lpr -p &"                                                 5
set crt=15                                                         6
set DEAD=~/dead.letter                                             7
set EDITOR=/usr/ucb/ex                                             8
set hold                                                           9
set keepsave                                                       10
set metoo                                                          11
set PAGER="cat -c -s | more -22"                                   12
set prompt="{Mail}& "                                              13
set record=~/.record                                               14
set SHELL=/bin/csh                                                 15
set VISUAL=/usr/ucb/vi                                             16

ignore apparently-to date errors-to from id in-reply-to \          17
        message-id precedence received references remailed-date \
        remailed-from return-path sent-by status via
```

Explanation of Command Lines

Line 1:

```
set alwaysignore
```

Omit display and printing of the message-routing fields indicated by the `ignore` command (below).

Line 2:

```
set askcc
```

Ask for a list of users to send copies of the message being composed.

Line 3:

```
set asksub
```

Ask for a subject.

Line 4:

```
set autoprint
```

Print the next message automatically after a `d` (delete) or `u` (undelete) command.

Line 5:

```
set cmd="lpr -p &"
```

Set the | (pipe) command to send output to the line printer, unless you indicate some other command to pipe output through.

Line 6:

```
set crt=15
```

Set the length of a message that can be printed without paging to be 15 lines.

Line 7:

```
set DEAD=~/dead.letter
```

Indicate the name of your dead-letter file.

Line 8:

```
set EDITOR=/usr/ucb/ex
```

Use `ex` to edit the message being composed when you type the `~e` on a line by itself, followed immediately by a RETURN.

Line 9:

```
set hold
```

Retain current messages in the system mailbox until each is disposed of.

Line 10:

```
set keepsave
```

Keep copies of saved messages in the system mailbox until explicitly deleted.

Line 11:

```
set metoo
```

When sending to a mailing list, if your username appears in the list, send yourself a copy.

Line 12:

```
set PAGER="cat -c -s | more -22"
```

Use the command

```
cat -c -s | more -22
```

to break long messages into pages.

Line 13:

```
set prompt="Mail& "
```

Use the string `Mail&` as your `Mail` prompt.

Line 14:

```
set record=~/mbox
```

Keep a record of outgoing mail in the named file (`~/.record`) in this case. Note that this file may contain copies of confidential mail, and so should be protected. If you use a record file, its name should begin with a dot (`.`), and you should type in the commmand:

`chmod 600` *filename*

to make getting access to it more difficult.

Line 15:

```
set SHELL=/bin/csh
```

Start a C-Shell with the `!` shell-escape command.

Line 16:

```
set VISUAL=/usr/ucb/vi
```

Use `vi` to edit the message being composed when you type the `~v` on a line by itself, followed immediately by a RETURN.

Line 17:

```
ignore apparently-to date errors-to from id in-reply-to \
        message-id precedence received references remailed-date \
        remailed-from return-path sent-by status via
```

Do not display any of the routing-information fields listed above.

A

The File System Hierarchy

A

The File System Hierarchy

The chart on the pages that follow outlines the file system hierarchy on a typical Sun Workstation. To clarify the relationships of the various directories, each filename is shown as a complete pathname, nested underneath its parent. Files that contain interesting information are noted.

Filename		Description
		root directory
/bin		*utilitiy programs*
	/bin/ar	
	/bin/as	
	/bin/awk	
	/bin/cat	
	/bin/cc	
	. . .	
	/bin/who	
/dev		*devices and special files*
	/dev/console	*console terminal*
	/dev/drum	*memory paging device*
	. . .	
	/dev/*mem	*memory special files*
	/dev/null	*system wastebasket*
	. . .	
	/dev/pty[p-z]*	*pseudo-terminal driver(s)*
	. . .	
	/dev/tty*	*terminals*
	/dev/tty[p-z]*	*pseudo-terminals*
	/dev/vme*	*VME bus special files*
	. . .	
	/dev/win*	*window system special files*
/etc		*system administration files & programs*
	. . .	
	/etc/cron	
	/etc/fastboot	
	/etc/fasthalt	
	. . .	
	/etc/fsck	
	/etc/fstab	*table of mountable filesystems*
	/etc/group	*system group membership table*
	. . .	
	/etc/hosts	*list of systems on the network*
	/etc/hosts.equiv	*list of trusted systems*
	. . .	
	/etc/motd	*message-of-the-day file*
	/etc/mount	
	/etc/mtab	*table of mounted filesystems*
	. . .	
	/etc/passwd	*password file*
	/etc/printcap	*table of printers and capabilities*
	. . .	

	/etc/termcap	*table of terminal devices and capabilities*
	. . .	
	/etc/ttys	*terminal initialization info*
	/etc/ttytype	*table of connected terminals*
	. . .	
	/etc/utmp	*table of users logged in*
	/etc/yp	*system yellow-pages directory*
	. . .	
	/lost+found	*detached filesystems for* `fsck`
	/private	*client workstation files*
	/private/usr2	*directory for guest accounts*
	/stand	*standalone programs (not run under UNIX)*
	/usr	*general-purpose directory*
Your files are here. ⇒	**/usr/***name*	*home directory for* name
	/usr/*name***/.cshrc**	
	/usr/*name***/.login**	
	/usr/*name***/.logout**	
	/usr/*name***/.exrc**	
	/usr/*name***/.mailrc**	
	. . .	
	/usr/*name/filename*	
	. . .	
	/usr/*name/directory*	
	/usr/*name/directory/filename*	
	. . .	
	. . .	
	/usr/*host*	`/usr` *directory mounted from another* host
	/usr/adm	*system administration files*
	. . .	
	/usr/adm/lastlog	*table of most recent logins*
	. . .	
	/usr/bin	*more utility programs*
	/usr/bin/addbib	
	/usr/bin/adjacentscreens	
	/usr/bin/align_equals	
	/usr/bin/at	
	/usr/bin/basename	
	/usr/bin/bc	

/usr/bin/cal	
. . .	
/usr/bin/ypwhich	
/usr/crash	*system crash files & programs*
/usr/dict	*dictionary files*
. . .	
/usr/dict/words	*dictionary wordlist*
/usr/etc	*more system administration files and programs*
/usr/etc/ac	
. . .	
/usr/etc/catman	
. . .	
/usr/etc/yp	*yellow pages directory*
/usr/games	*games and demos*
/usr/include	*standard C* `#include` *files*
. . .	
/usr/include/f77	*Fortran include files*
. . .	
/usr/include/images	*icon images*
. . .	
/usr/include/nfs	*NFS include files*
. . .	
/usr/include/pascal	*Pascal include files*
/usr/include/pixrect	`pixrect` *include files*
. . .	
/usr/include/sys	*system internals include files*
. . .	
/usr/lib	*library routines and other useful stuff*
/usr/lib/.rootmenu	*sample setup files for* `suntools`
/usr/lib/.suntools	
. . .	
/usr/lib/.textswrc	
/usr/lib/Cshrc	*sample setup files for for* `csh`, `mail` *and* `vi`
/usr/lib/Exrc	
/usr/lib/Login	
/usr/lib/Logout	
/usr/lib/Mailrc	
. . .	
/usr/lib/atrun	
/usr/lib/calendar	
. . .	
/usr/lib/crontab	

. . .	
/usr/lib/defaults	*directory for window-system defaults*
. . .	
/usr/lib/font	*troff fonts*
. . .	
/usr/lib/tmac	*troff macro-package files*
. . .	
/usr/local	*local utility programs*
/usr/local/lib	*local libraries*
/usr/man	*Manual Page Sources*
/usr/man/cat[1-8]	*formatted pages*
/usr/man/man[1-8]	*source files*
/usr/preserve	*preserves editor files from crashes*
/usr/sccs	*sccs programs*
/usr/spool	*delayed execution files*
/usr/spool/mail	*system mailboxes*
/usr/spool/lpd	*printer queue(s)*
/usr/tmp	*temporary files*
/usr/ucb	*programs developed at U.C. Berkeley*
/usr/ucb/Mail	
/usr/ucb/biff	
/usr/ucb/ccat	
/usr/ucb/checknr	
/usr/ucb/chsh	
. . .	

Index

Revision History

Version	Date	Comments
A	24 December 85	First edition of this manual.

Notes

Notes

Notes

Notes

Notes

Corporate Headquarters
Sun Microsystems, Inc.
2250 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 287815

For U.S. Sales Office locations, call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Sun House
31-41 Pembroke Broadway
Camberley
Surrey GU15 3XD
England
0276 62111
TLX 859017

Australia: 61-2-436-4699
Canada: 416 477-6745
France: (1) 46 30 23 24
Germany: (089) 95094-0
Japan: (03) 221-7021
The Netherlands: 02155 24888
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276 62111

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales