Ø Site Operations

symbolics

Suggested Reading Paths for Different Users

Book	Information	Read This	LISP Programmer	System Maintainer	System User	Application User
Ø	about site operations and maintenance.	before or during site installation.		Ø		
1	needed by first-time Symbolics computer users. ▯	for an overview of the Symbolics computer and its functions.			1	1
2	**a.** about Symbolics Common Lisp concepts.	for an overview and description of Symbolics Common Lisp features.	2			
	b. about Symbolics Common Lisp Objects (dictionary).	for reference while developing Symbolics Common Lisp programs.				
3	about the Genera editing facilities. ▯	for detailed information about text and font editing.			3	
4	about Symbolics Common Lisp programming style ▯ and tools.	to become familiar with the Symbolics Common Lisp programming and debugging environment.	4			
5	about device-independent input and output.	for reference while writing I/O code.	5			
6	about the electronic communications facilities. ▯	to set up, manage, and use your electronic mail system.			6	6
7	**a.** about dynamic window facilities for programming the user interface.	to set up your program's window interface and command loop; to control user input and program output.	7			
	b. about additional facilities for programming the user interface.	to learn about the input editor, command processor, window system and other facilities.				
8	about the inner workings of the software environment.	if you want your programs to take advantage of the low-level system facilities.		8		
9	about the network system, namespace system, and protocols.	to understand Symbolics networks and the namespace system, and to write new network applications.		9		
10	System Index	to locate topics anywhere in the Documentation Set.	10	10	10	10

See also the binder (*Installation, Notes, and Bulletins*) for software installation procedures, highlights of new features in Genera 7.0, and HOSS bulletins.

▯ A reference card contains summary information from this book.

Ø Site Operations

symbolics™

Cambridge, Massachusetts

Site Operations

# 999007

June 1986

This document corresponds to Genera 7.0 and later releases.

Symbolics, Inc.
4 New England Tech Center
555 Virginia Road
Concord, MA 01742

Text written and produced on Symbolics 3600-family computers by the Documentation Group of Symbolics, Inc.
Text masters produced on Symbolics 3600-family computers and printed on Symbolics LGP2 Laser Graphics Printers.
Cover design: Schafer/LaCasse
Printer: CSA Press
Printed in the United States of America.

Printing year and number: 88 87 86 9 8 7 6 5 4 3 2 1

Table of Contents

List of Figures

1. Introduction to Site Operations

This document provides information for various operations that are necessary to maintain Symbolics machines at your site. These include the following:

- performing backups and dumps
- using the distribution subsystem
- using the carry tape system
- using the FEP-Tape system
- adding LMFS partitions
- maintaining the file system
- saving new versions of world loads
- installing new software

2. File System Editing Operations Program

The File System Editing Operations Program includes the File System Editor as well tools for doing other local file system maintenance operations, such as manipulating backup tapes, and running the salvager. See the section "File System Editor" in *Reference Guide to Streams, Files, and I/O*. See the section "Lisp Machine File System" in *Reference Guide to Streams, Files, and I/O*. Although it includes commands for manipulating remote file systems, including the interface to the File System Editor, the File System Editing Operations Program is intended for manipulating the local file system.

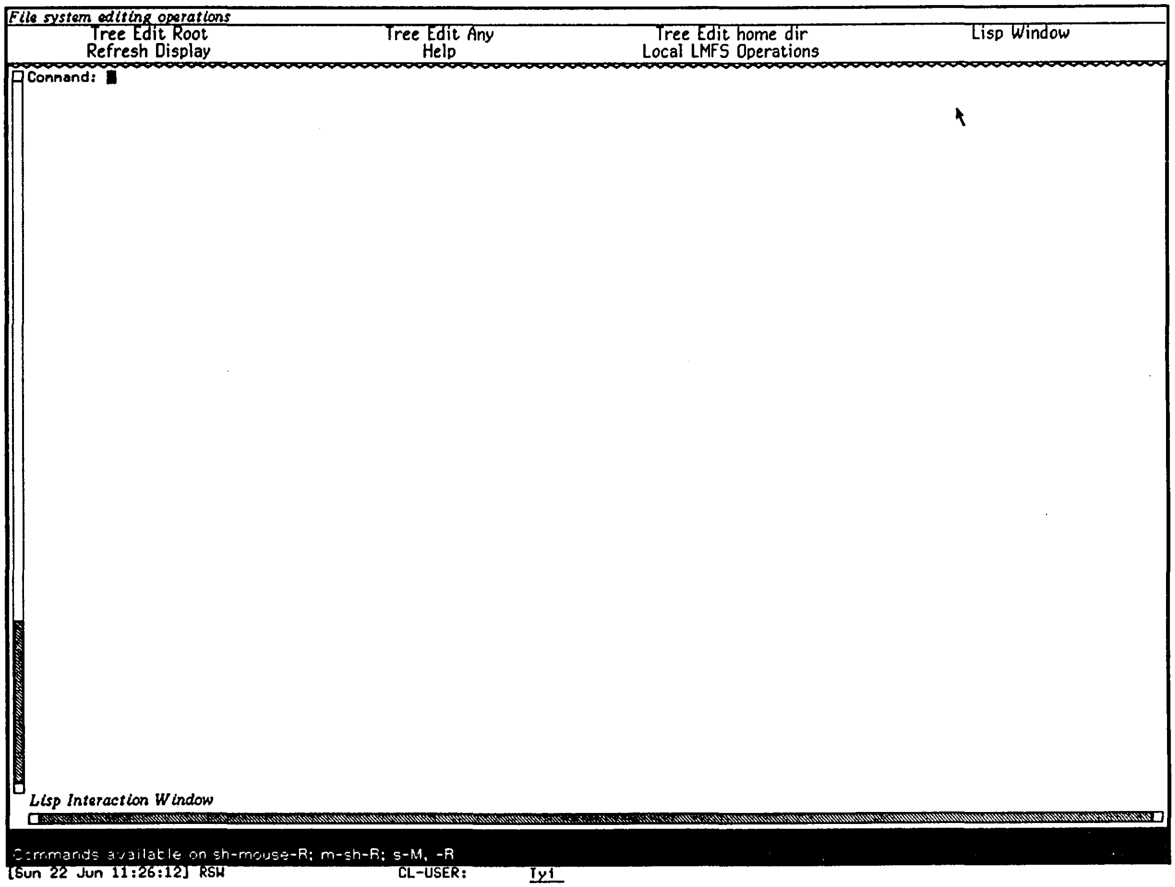
You enter the File System Editing Operations program by doing one of the following:

- pressing SELECT F
- giving the command Select Activity File System Operations to the Command Processor prompt.
- clicking on File System in the system menu

Figure 1 shows the initial window of the File System Editing Operations.

The File System Editing Operations program uses a frame with a command menu pane at the top and a large pane beneath it. You give commands by clicking on the command menu pane. The large pane is initially a *Lisp Interaction Window*. The menu has four levels of operations, which unfold as you click on certain operations. For example, clicking right on [Local LMFS Operations] in the first level of the command menu invokes the second level. Clicking right on [LMFS Maintenance Operations] invokes the third level. Then, clicking right on [Local LMFS Tools] invokes the fourth level of operations. Figure 2 shows the four menu levels.

In addition, if are in the first level of the menu, which is initially a *Lisp Interaction Window*, and click on a File System Editor command in the command menu, for example, [Tree Edit Root], the large window changes to a *File System Editor* window. If you click on [Lisp Window], this command changes the large window back to a Lisp Interaction Window. When it is a Lisp Listener, you can type Lisp forms and have them evaluated and printed. When it is a File System Editor window, you are in the File System Editor and can click on the items in the menu to activate File System Editor commands.



```
Connand: ▮
```

```
Commands available on sh-mouse-R; m-sh-R; s-M, -R
[Sun 22 Jun 11:26:12] RSW          CL-USER:     Tyi
```

Figure 1. Initial File System Editing Operations Menu

2.1 File System Editing Operations Commands

The following table describes the menu structure of the File System Editing Operations Program. The commands are organized into four levels, in increasing order of potential for causing problems if you don't use them correctly. The table is followed by explanations of the commands at each level. Discussions of how these commands are used in various file maintenance procedures make up the next few chapters of this book.

Level 1	General file system editing operations.
Level 2	Local file system control operations.

These are to be used by the person at your site responsible for maintaining the file system and doing backup dumps. This menu is summoned by clicking on [Local LMFS Operations] in the Level 1 menu.

Level 3 Server and maintenance operations.

These should be used only by someone who is very knowledgeable about the file system. This menu is summoned by clicking on [LMFS Maintenance Operations] in the Level 2 menu.

Level 4 File system internal data structure editing operations.

You should not attempt to use these commands unless you are an expert in Lisp Machine File System internal data structures. This menu is summoned by clicking on [LMFS Internal Tools] in the Level 3 menu.

Note: *The commands in menus three and four can damage your file system if used incorrectly. If you have any questions about these operations, please call Symbolics Software Support.*

Some commands are described as "typing out" certain information. If the large pane is a *Lisp Interaction Window,* such information is simply displayed on that window; if it is a *File System Editor,* then the information is displayed in a typeout window over the file system editor information. You can flush the typeout window and restore the display of the File System Editor by pressing any character or by clicking on [Refresh Display].

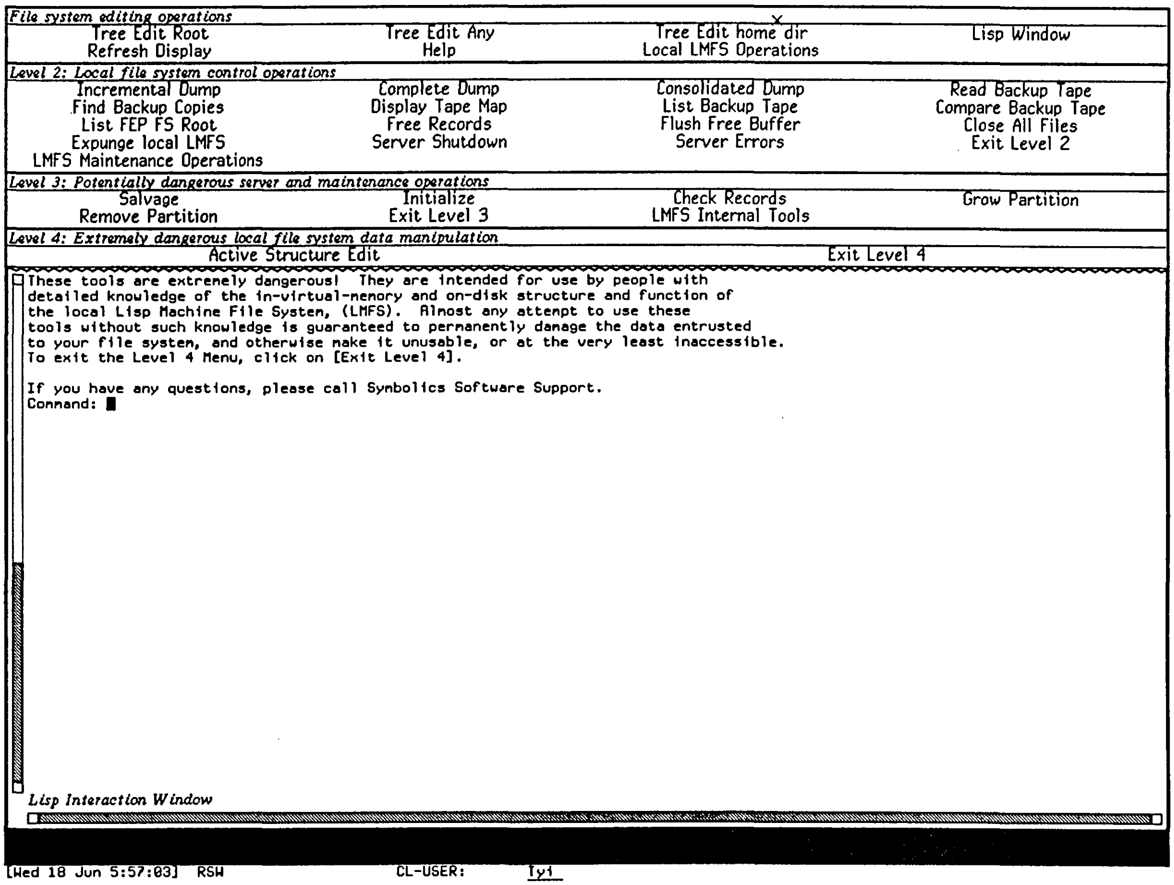
2.1.1 Level 1 Menu

File System Editing Operations:

[Tree Edit Root] Enter the File System Editor, using the root directory of the local file system as the base directory. See the section "File System Editor" in *Reference Guide to Streams, Files, and I/O.* This puts the large pane into the *File System Editor* state.

[Tree Edit Any] Enter the File System Editor; it prompts you for the name of the base directory. See the section "File System Editor" in *Reference Guide to Streams, Files, and I/O.* This puts the large pane into the *File System Editor* state.

[Tree Edit home dir] Enter the File System Editor, using your home directory as the base directory. See the section "File System Editor" in *Reference Guide to Streams, Files, and I/O.* This puts the large pane into the *File System Editor* state.



```
These tools are extremely dangerous!  They are intended for use by people with
detailed knowledge of the in-virtual-memory and on-disk structure and function of
the local Lisp Machine File System, (LMFS).  Almost any attempt to use these
tools without such knowledge is guaranteed to permanently damage the data entrusted
to your file system, and otherwise make it unusable, or at the very least inaccessible.
To exit the Level 4 Menu, click on [Exit Level 4].

If you have any questions, please call Symbolics Software Support.
Command: █
```

```
[Wed 18 Jun 5:57:03]  RSW          CL-USER:       Tyi
```

Figure 2. The Four Levels of the File System Editing Operations Menu

[Lisp Window] Put the large pane back into the *Lisp Interaction Pane* state. This is useful for getting out of the File System Editor.

[Refresh Display] When the large pane is in the *File System Editor* state, and you use one of the commands that "types out" information, the information appears on top of the File System Editor window, and you are told "`Type any char to flush:`". You can use this command to clear the screen and redisplay the File System Editor window without removing your hand from the mouse. You can also use this command to proceed from **MORE** pauses.

[Help] Type out general information about the File System Maintenance program and File System Editor.

[Local LMFS Operations]
Bring up the Level 2 menu Local File System Control Operations.

2.1.2 Level 2 Menu

The commands in this menu are intended to be used by the person at your site who is responsible for maintaining the file system and doing backup dumps.

Local File System Control Operations:

[Incremental Dump] Do an incremental dump of the local root directory. Offers an Accept-Values menu to adjust all parameters. See the section "Dumping, Reloading, and Retrieving", page 13.

[Complete Dump] Do a complete dump of the local root directory. Offers you an Accept-Values menu to adjust all parameters. See the section "Dumping, Reloading, and Retrieving", page 13.

[Consolidated Dump] Do a consolidated dump of the local root directory. Offers you and Accept-Values menu to adjust all parameters. See the section "Dumping, Reloading, and Retrieving", page 13.

[Read Backup Tape] Retrieve single files or reload full tapes. You select the activity you want from a pop-up menu. You can also use [Read Backup Tape] to list or compare tapes, if you change the default operation in the menu by clicking on Compare in the menu.

[Find Backup Copies] Locate files on backup tape. It prompts you for a file specification to locate.

[Display Tape Map] Display a directory listing of what should be on a backup tape. It prompts you for the tape number. You can display a listing of any backup tape directory on any machine connected to your network by giving the machine name and the pathname of the tape directory. Standard pathname defaulting and merging work.

[List Backup Tape] List the contents of a backup tape that you have mounted on a tape drive. It prompts you for the tape specification.

[Compare Backup Tape]
Compare the contents of a backup tape that you have mounted to the contents of the local file system. It prompts

you for the tape specification. To compare the tape to another (remote) file system, use [Read Backup Tape].

[List FEP FS Root] List the FEP file system's root directory from the default disk unit. See the section "Show FEP Directory Command", page 67. See the function **zl:print-disk-label**.

[Free Records] Type out information about the number of free records in the local file system. The last line tells you how many records are marked as free, how many are marked as used, and the sum of these numbers, which is the total number of records in the file system.

Clicking middle on [Free Records] prepares a directory-by-directory usage report of record use, indicating how many records are in use by files in each directory. It prompts you for the name of a file in which to place the report.

Clicking right on [Free Records] displays how many records are in use in each partition. This information is necessary for the commands that allow you to change the size of, add, or remove partitions.

See the section "Free Records", page 42.

[Flush Free Buffer] Write the internal pool of free disk records back to the disk. This happens automatically when you log out. After doing this, you can cold boot without losing records. See the section "Free Records", page 42.

[Close All Files] Call **fs:close-all-files**. This has nothing to do with the Lisp Machine File System as such; it closes any open files in use by your machine, whether local or remote. This is occasionally useful for cleaning up after problems occur, but be aware that by using [Close All Files], you can cause new problems for any programs in the machine that are validly using files at the time.

[Expunge Local LMFS]

Expunge all directories on the local file system. It tells you how much space was recovered.

[Server Shutdown] Shut down file servers at a future time, or reschedule or cancel a shutdown.

This command lets you shut down a file server cleanly. You run this command only on a 3600-family computer that is acting as a file server for other users. Clicking left on [Server Shutdown] means that you plan to shut down the file system soon. It asks you for a short message to be sent to people using the file server, which you can use to explain why it is being shut down and when it will return. It also asks you when you want the shutdown to take place; the default is five minutes. All users of the file system are sent periodic messages warning them that the server is going to be shut down. Finally, when the time comes, it closes all Chaosnet servers on the machine, and disables creation of new servers. When servers are shut down, you can cold boot the machine or whatever else you want to do. While the shutdown is "in progress" (the messages are being sent), you can cancel it by clicking middle on [Server Shutdown] or reschedule it by clicking right on [Server Shutdown].

[Server Shutdown] only shuts down the network server; it does not affect the local operation of the file system itself. It shuts down all servers, not just file servers, since anything that requires the file servers to be shut down also requires that all servers be shut down.

[Server Errors] Display all the error messages associated with errors encountered by the file server. When such errors occur, you get a message that begins as follows:

```
[File Server got an error: ...]
```

The message contains descriptive information about the error.

[Exit Level 2] Return from this menu to the top-level menu of General File System Editing Operations.

[LMFS Maintenance Operations]
Bring up the Level 3 menu of Server and Maintenance Operations. **Note:**The operations on the Level 3 menu can damage your file system if misused. They should only be performed by someone who is very knowledgeable about the file system.

2.1.3 Level 3 Menu

The commands in this menu are intended only for users who are thoroughly familiar with the operations of the file system; misuse of these operations can destroy or damage the file system.

Server and Maintenance Operations (Potentially Dangerous):

[Salvage] Run the salvager. See the section "Salvager", page 43.

[Initialize] Create a new file system. Use this tool to add file storage space to your local Lisp Machine File System. This operation asks you several questions and prints out information to make sure you really want to initialize the FEP file that would contain the file system. These verifications are to ensure that you do not accidentally destroy any previous file system residing there. [Initialize] takes about a minute for each four thousand records (a record is four 256-word disk blocks). It] queries you about the FEP file size be$^{\text{f}}$ore it initializes the new file system.

Clicking right on [Initialize] presents a menu of initialization. This is how you can add new FEP files to the running file system. For a description of this operation: See the section "Adding a Partition to LMFS", page 47.

Note: Added partitions should never be initialized. The partitions are automatically initialized by the system when they are created.

This tool is intended only for manipulating the local Lisp Machine File System, specifically, adding partitions to it for the purpose of storing files in them. If you want to create a FEP file for some other purpose, or if you want to create an additional paging file, you must do so from Lisp, using the Create FEP File command. See the section "Create FEP File Command" in *User's Guide to Symbolics Computers.* See the section "Allocating Extra Paging Space", page 50.

Warning: *Attempting to create an additional paging file using [Initialize] will destroy the file system on your machine, permanently and irretrievably.*

[Check Records] Check each record in the local file system for consistency, and notify you about any problems. (This is also available from the salvager). This option scans the hierarchy, going through the directories, making sure that each directory entry really describes a file that agrees with it, and that each record of each file is validly identified as a part of that file.

[Grow Partition] Increases the number of blocks available in a partition. It offers you a menu to select the partition to grow and prompts for the number of blocks you want to increase it by.

[Remove Partition] Remove active partitions from the file system and delete them. It does this by walking over the local file system, evacuating files and directories from the partitions to be removed, to other partitions. For medium and large file systems, this operation takes a long time. In order for it to succeed, there must be enough room in other partitions to contain the evacuated files and directories. This tool determines whether or not sufficient room exists for the operation to complete successfully, and queries if it suspects that sufficient room is not available. You can click right on [Free Records] to get a partition-by-partition report. Salvaging might be necessary to properly identify all free records. If you need to do this: See the section "Salvager", page 43.

[Remove Partition] provides the option of deleting the FEP file when all LMFS files have been removed from it. If it detects that a partition is not completely empty, it reports this and allows you to abort the process.

Do not attempt to use this tool to manipulate FEP files for any other purpose, or to manipulate FEP files in use by LMFS in any other way, for example, lmfs.file, lmfs1.file, or fspt.fspt. Misuse of this tool causes irretrievable destruction of data in your file system.

[Exit Level 3] Return from this menu to the Level 2 menu of Local File System Control Operations.

[LMFS Internal Tools]

Bring up the Level 4 menu of File System Data Manipulation Operations.

Warning:*Editing the internal structures of your file system can result in data being irretrievably lost. You should not use these tools unless you are sure you know what you are doing.*

2.1.4 Level 4 Menu

The commands in this menu are intended only for users who are thoroughly familiar with the internal organization and implementation of the file system; misuse of these operations can destroy or damage the file system.

Local File system data manipulation (Extremely Dangerous):

[Active Structure Edit]
Edit the active file system data structure. This displays "active" internal data structure as a scroll window, and is intended to be used by those debugging local file system problems.

[Exit Level 4]
Return from this menu to the Level 3 menu of Server and Maintenance Operations.

3. Dumping, Reloading, and Retrieving

A file system can be damaged or destroyed in any number of ways. Users can delete files by accident. To guard against such a disaster, it is wise to *dump* the file system periodically, that is, write out the contents of the files, their properties, and the directory information onto magnetic tapes. If the file system is destroyed, it can then be *reloaded* from the tapes. Individual files can also be *retrieved* from tapes, in case a single file is destroyed, or just accidentally deleted (and expunged). Dump tapes can also be used to save a copy of all the files on a system for archival storage.

In a *complete dump*, all of the files, directories, and links in the file system are written out to tapes. This, obviously, saves all the information needed to reload the file system. However, a complete dump can take a long time and use a lot of tape, especially if the file system is large. In order to make it practical and convenient to dump the file system at short intervals, a second kind of dump can be done, called an *incremental dump.*

In an incremental dump, only those files and links that have been created or modified since the last dump (of either kind) are dumped; things that have stayed the same are not dumped. (All directories are always dumped in an incremental dump.) Now, if the file system is destroyed, you reload it by first reloading from the most recent complete dump and then reloading each of the incremental dump tapes made since that complete dump, in the same order in which they were created. Therefore, you do not need to retain incremental dump tapes that were made *before* the most recent complete dump was done; you can reuse those tapes for future dumps.

Since all tapes containing incremental dumps done since the last complete complete dump must be reloaded in order to restore the file system, doing a complete dump regularly makes recovery time faster. Doing complete dumps also lets you reuse incremental dump tapes, as described above. The more incremental dump tapes you must load at recovery time, the longer it takes to recover, and thus the more chance there is that something will go wrong. Thus, it is advantageous to perform complete dumps periodically.

A *consolidated dump* is like an incremental dump, in that it only dumps files that have been created or changed recently. However, a consolidated dump backs up only those files that have been created or changed since a specified *consolidation date.* A consolidated dump is the appropriate kind to take if some event destroys recent incremental dump tapes, or they are found to be unreadable. If a complete dump extends through several days, it is wise to take an incremental dump between-tape stopping points as appropriate.

3.1 Performing Dumps

To perform a dump, follow these steps:

1. Mount a magnetic tape on an available and usable tape drive which need not be connected to a machine on which you are performing the dump.

2. Press `SELECT F` to select the File System Editing Operations Program and click on [Local LMFS Operations].

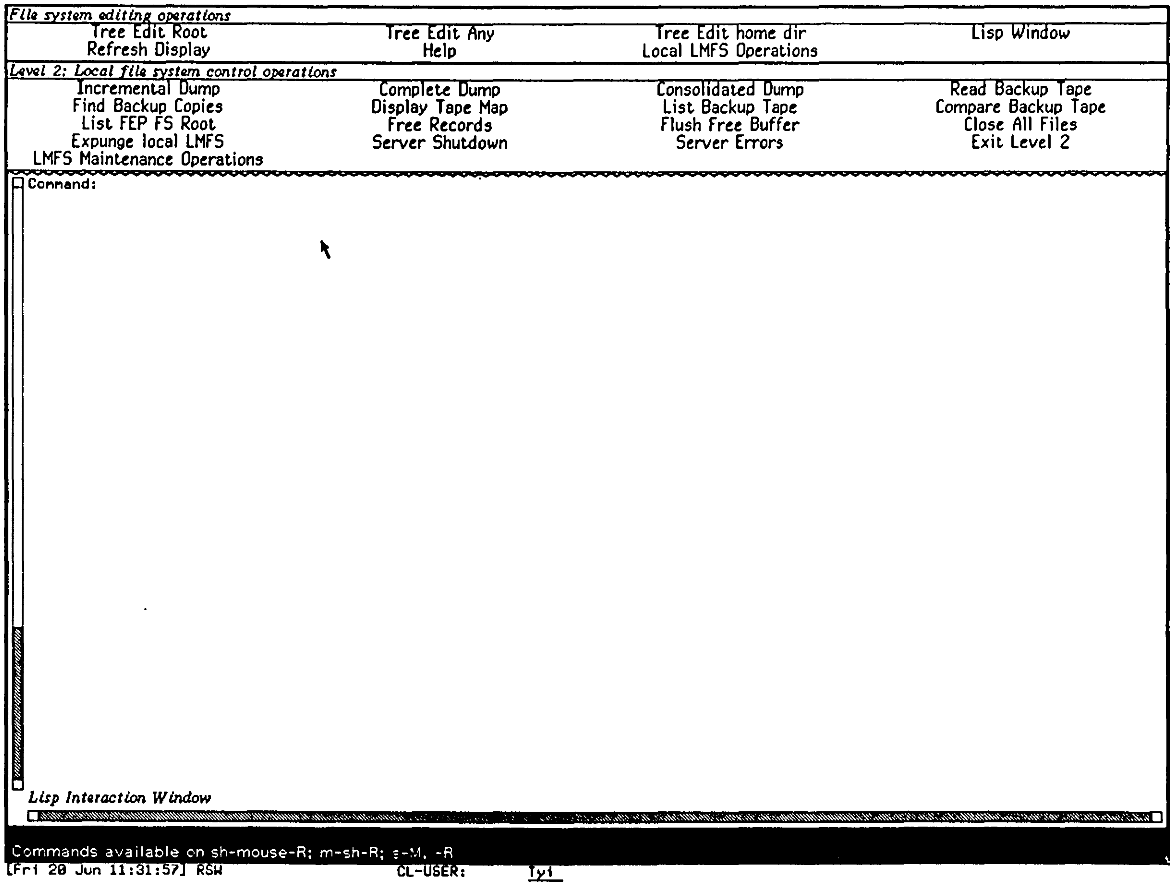
   This invokes the second level of the File System Editing Operations Program, which is called *Local File System Control Operations.*

3. Choose either [Incremental Dump], [Complete Dump] or [Consolidated Dump] by clicking on the menu.

   These commands respond with an Accept Values menu that lets you set the parameters of the dump; the only difference among the commands is the initial value of some of the parameters in this window.

4. Change the values in this window as needed.

Figure 3 shows the second level of the File System Editing Operations Program.



```
Command:
```

```
Commands available on sh-mouse-R; m-sh-R; s-M, -R
[Fri 20 Jun 11:31:57] RSW          CL-USER:      Tyi
```

Figure 3. Performing a Dump

Here is an explanation of the parameters offered for modification in this window:

Dump Type

There are three possible types of dump: *incremental*, *consolidated*, and *complete*. The three File System Maintenance commands initialize this field according to their own requirements; the [Dump] command initializes it to *complete*.

Pathnames

The pathname, or pathnames, specifying what is to be dumped. If there is more than one pathname, they are separated by commas. This value controls what files and directories are inspected for dumping. The type of the dump (complete, incremental, or consolidated) and the status of the individual files controls what subset of these files are actually dumped. For information about the different types of dumps: See the section "Dumping, Reloading, and Retrieving", page 13. Names of single files or links can be used to dump single files or links.

Wildcard specifications can also be used: this is the normal way to dump many files from one directory, or from a subtree. Subtrees are dumped via recursive (**:wild-inferiors**, "**") directory wildcards. The pathname you type is merged with "local:>**>*.*.*".

To dump the whole file system, which is the normal default, the appropriate pathname is:

```
>**>*.*.*
```

To dump all the files in directory >foo>bar, and all of its inferiors, the appropriate pathname is:

```
>foo>bar>**>*.*.*
```

To dump all the latest Lisp files in directory >abel>baker, but not any of its inferiors, the appropriate pathname is:

```
>abel>baker>*.lisp.newest
```

See the section "Naming of Files" in *Reference Guide to Streams, Files, and I/O*. See the section "LMFS Pathnames" in *Reference Guide to Streams, Files, and I/O*.

Tape Reel ID

Every reel of tape produced by the dumper must have a Tape Reel ID, which is a string of up to eight characters. You must explicitly supply a value for this option. The reel ID is used to identify this reel of tape to the backup system; it appears in the dump maps and in any messages about the tape. The **:complete-dump-tape** or **:incremental-dump-tape** property of any file dumped is set to this value, as well. The Tape Reel ID should be written with a pen onto the label of the tape so that the tape can be identified by sight. *Note that you must supply a Tape Reel ID.*

Tape Drive Spec

The host name of the machine on which the tape drive to be used appears. This is initialized to a reasonable default, which is usually Local: if there is a tape drive present on the local host.

Dump deleted files

This can be either *Yes* or *No*, and says whether files marked as deleted but not yet expunged should be dumped on the backup tape. The default value is *No*; deleted files normally are not dumped.

Tape when done

This controls what the dumper does with the tape when it has finished passing over all the specified pathnames. These are the available options:

Offline

Rewinds the tape and puts it offline. It declares the dump finished. This is the default.

Rewind

Rewinds the tape without putting it offline. It declares the dump finished. It facilitates listing or verifying the tape contents.

Leave

Leaves the tape positioned at the end without rewinding it or putting it offline. It declares the dump finished. It facilitates more dumping later, by leaving the tape in the correct position for using "Append to tape" in a later dump invocation.

Query

At the end of the dump, all of these options are presented, and you can choose whether to rewind and set the tape offline, rewind it, leave it at end, or dump some more files. If you choose to dump more files, the dumper menu is offered again, and the new files are appended to this tape. The dump is not declared finished until you click on "Abort".

Person operating

The identification of the person doing the dump. This is entered into the backup map, and sets this person as the file author of that map. Normally, this is the same as the login ID of the user performing the dump, and that is its default value. However, if the user who is performing the dump is not logged in to the machine from which the dump is invoked,this field should be filled with that user's name. It is important for documentation purposes and site record-keeping.

Consolidate from

This field is only used during a consolidated dump. It is a date and time in the past, entered in any acceptable Lisp Machine format. The consolidated dump dumps all, and only, files that have been created or modified since this date.

Set date dumped

When the dumper finishes writing a tape, it marks all the files it has dumped as having been dumped on that tape at this time, and creates a *tape directory*, as described below. These measures allow the file to be retrieved later, and indicate that the file no longer needs to be dumped in incremental dumps. This is the default action, which corresponds to a value of *Yes*. **Note:** The LMFS dumper should not be used to move software between sites as it is far too general, and system-independent; use the carry system and the distribution tape system instead. However, if you do use the dumper to make tapes that are not part of your site's backup, such as for moving software between machines, you do *not* want to indicate that the files were dumped, or to make a tape directory. Select *No* in this case.

Restart pathname

The purpose of this feature is to allow restarting of complete dumps that are interrupted by any sort of failure. When the dumper finishes a tape, it prints out the pathname of the last file dumped. Although this is recorded in the dump map, *the pathname and the name of the tape should be recorded on paper by the person doing the dump*, especially if a complete dump is being done.

To restart a dump, fill out the menu as usual, but type in the pathname of the last file known to have been dumped as the value of [Restart Pathname]. The dumper scans the sub-hierarchy indicated, but does not dump files already dumped, or even progress down, seeking files to be dumped, into directories that the restart pathname indicates have already been processed. The skipping of files and directories already dumped is based on sorting order, not whether the file has actually been dumped. Thus, if files A, C, E, G, and I exist, and files A through E get dumped one day, and the dump is interrupted and restarted from E the next day, a D created in the interim is not dumped.

Comment A string, of arbitrary contents, written on each reel of the dump and in the dump map. This might say why the dump was performed, or any other special information about this dump.

When you are done filling in values, press `END`; if you decide not to do a dump after all, press `ABORT`. If there is something wrong about the set of parameters you have specified, the program displays a message and presents you with the Accept Values window again. Otherwise, it displays a message saying that the dump has started successfully, and proceeds. While the dump is in progress, the name of the file that is being dumped is shown in the far right-hand field of the status line (this is the field that normally shows you the names of files that are being read or written).

The dumper creates a file called the *dump map*. The dump map is a character file, giving a complete description of what has been dumped, directory by directory and file by file, including the time of dumping, the tape on which the file was dumped, the tape reel ID of the previous tape on which the file appears (if any), and so on. The dump map is created in the >dump-maps directory. Its name is constructed from the type of dump and the date and time at which it was started; the file type is **map** and the version is **1**. A typical dump map might have the pathname:

```
>dump-maps>complete-3/15/86-9:02.map.1
```

The dumper puts all information about the dump, the operator, the time of day, the options, and so forth, in the map. It also puts error recovery information there, and descriptions of tape-changings, as well as the number of files dumped on each tape. The dumper performs a **:finish** operation on the map file at the end of each tape, so that if the system crashes during a multi-tape dump, information about previous tapes is guaranteed to be intact and accessible.

The dumper also creates a file called the *tape directory* for each separate reel of each dump. This is a binary file saying what is on the tape, with more or less the same information as the dump map. You use this file when you try to locate dumped copies of a file. See the section "Finding Backup Copies cf Files", page 25. The tape directory is also created in the >dump-maps directory. Its name is the tape reel ID of the tape, its file type is **directory**, and its version is **1**. A typical directory map might have the pathname:

```
>dump-maps>INC00001.directory.1
```

The dumper dumps files successively to tape, and at the end of each tape, rewinds and unloads the tape, asking for a new tape if there are more files to be dumped. It is only after it has done this that it sets backup dates for the files and makes the dump directory.

If the dumper gets an irrecoverable error while writing a tape, it attempts to write end-of-file marks on the tape and inform you of what has happened. It gives you the option of either considering the files on that tape to have been validly dumped, in which case the dump continues on the new tape, or discarding the tape, in which case it redumps all the files that it had dumped on the bad tape onto the new tapes. The problem and its chosen recovery are described in the dump map.

By default, the dumper tries to read each tape before writing on it. This is to avoid accidentally overwriting valuable tapes. For tapes to be appended to, this is necessary. For other tapes, it is desirable. It often takes a long time to attempt to read blank tape, to prove that a new tape is really new. The dumper explains and queries if it is not confident that the tape being written on is the right one.

Some sites may want to waive this checking. This is necessary when tape hardware is in use that cannot time out while reading blank tape, and therefore reads the whole tape when a new tape is checked, with no way to stop it. The site option **validate-lmfs-dump-tapes**, an attribute of the Site object for a site, which is normally elected, enables the suppression of this checking.

3.2 Reloading and Retrieving

Reloading is the process of moving all the files on a backup tape into a local file system. *Retrieving* is the process of moving selected files.

Reloading and retrieving can load files onto any LMFS.

Two other functions are related to reloading and retrieving: listing the contents of backup tapes with the List tape option, and verifying the contents of the dump with the Compare option. The *reloader* program implements all four of these functions.

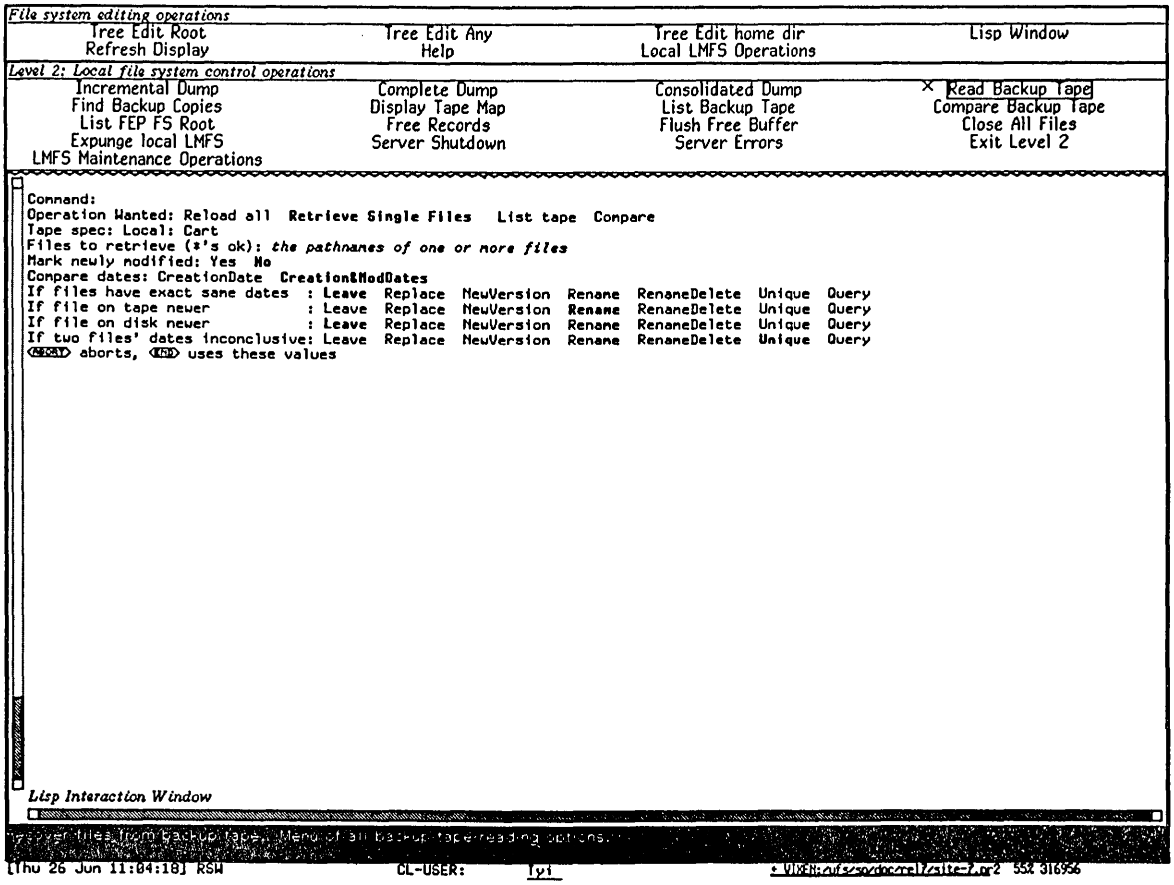
To invoke the reloader:

1. Press SELECT F to get to Level 1 of the File System Editing Operations program.

2. Click on [Local LMFS Operations] to invoke Level 2.

3. Click on [Read Backup Tape].

Figure 4 shows the Accept-Values menu that appears when you click on [Read Backup Tape].



```
Connand:
Operation Wanted: Reload all  Retrieve Single Files   List tape  Conpare
Tape spec: Local: Cart
Files to retrieve (*'s ok): the pathnanes of one or nore files
Mark newly nodified: Yes  No
Conpare dates: CreationDate  Creation&ModDates
If files have exact sane dates  : Leave  Replace  NewVersion  Renane  RenaneDelete  Unique  Query
If file on tape newer           : Leave  Replace  NewVersion  Renane  RenaneDelete  Unique  Query
If file on disk newer           : Leave  Replace  NewVersion  Renane  RenaneDelete  Unique  Query
If two files' dates inconclusive: Leave  Replace  NewVersion  Renane  RenaneDelete  Unique  Query
<ABORT> aborts, <END> uses these values
```

```
Recover files from backup tape.  Menu of all backup tape-reading options.
[Thu 26 Jun 11:04:18] RSW            CL-USER:      Tyi            * VIXEN:/ufs/so/doc/rel7/site-7.or2  55% 316956
```

Figure 4. The Read Backup Tape Menu

The reloader Accept-Values menu contains the following items:

Operation Wanted This selects which function the reloader is to perform.

These are your operation choices:

Reload all The reloader is to read the tape, and reload all files on it. It will not reload files that are already present.

Retrieve Single Files
The reloader will expect a list of wildcard pathnames, to tell it which files to reload. It will not reload files that are already present. You specify these pathnames by clicking on the menu item [Files to retrieve]. For example, a wildcarded pathname for a single file looks like this:

```
E:>trees>.*
```

List tape The reloader will read the tape and display a description of its contents on the screen. It lists the dumps appearing on the tape, and which files are on the tape. This does *not* verify that the files were dumped correctly; use "Compare" for that.

Compare The reloader will read each file on the tape, look for the same file in the file system, and compare the two, bit for bit, reporting any discrepancies. Use this option to verify that a dump tape just created contains good data.

After selecting the operation to be performed, you supply the following information:

Tape Spec: The tape spec. Normally, for a 3600 with an attached cartridge tape the default is Local:, which is an acceptable expression of the local machine. Otherwise, a reasonable default will be chosen, based on available tape servers at your site. The tape drive is a character string identifying the tape drive to be used on the tape host. You only need to supply this value if the machine selected has more than one tape drive, and needs this information to select one.

Files to retrieve (*'s ok):
One or more pathnames, which are usually wildcard pathnames. This is used when you click on [Retrieve single files]. Any file matching one of these wildcard pathnames will be reloaded, unless it is already there. For more information, see the following sections:

See the section "Naming of Files" in *Reference Guide to Streams, Files, and I/O*.
See the section "Wildcard Pathname Mapping" in *Reference Guide to Streams, Files, and I/O*.
See the section "LMFS Pathnames" in *Reference Guide to Streams, Files, and I/O*.

Mark newly modified: Yes No

Specify yes to mark the newly reloaded files as *not yet dumped* so that they will get dumped by the next incremental or consolidated dump. Choose no if you don't want the files to be dumped in the next dump. The default is No.

The reloader will move all the files from the backup tape into the local file system unless files of the same name, type, and version exist. If this is the case, the reloader compares the creation and or creation and modification dates of these files according to guidelines you specify here. These are the guidelines you can choose:

Compare Dates: CreationDate Creation&ModDates

Compares the creation dates or creation dates and modification dates of the files you are trying to reload, depending on what you choose.

If files have exact same dates:

Your choices are to: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Leave. The meaning of these choices is explained below.

If file on tape newer:

Your choices are to: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Rename. The meaning of these choices is explained below.

If file on disk newer:

Your choices are to: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Leave. The meaning of these choices is explained below.

If you choose the creation and modification dates for the Compare Dates option in the first menu item, then this option is presented:

If two files' dates inconclusive:

Your choices are to: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Unique. The meaning of these choices is explained below.

Here is the meaning of each choice for the questions above:

Leave	Leaves the file in the file system and does not reload it.
Replace	Loads the file from tape, completely replacing the file in the file system.
NewVersion	Loads the file from tape as the newest version.
Rename	Renames the file uniquely, and loads the file from tape.
RenameDelete	Renames the file uniquely, deletes it, and loads the file from tape.
Unique	Loads the file from tape under a unique name in the proper directory.
Query	Stops the reload and asks you what to do when the tape is ready.

When you have made your choices from the reloader Accept-Values window, press END to begin the reloader. If you do not wish to proceed with the reload, tape list, or compare, press ABORT.

The reloader prints information about each file it reloads or about every file on the tape, if you have selected [List tape]. When it gets to the end of the tape, it stops. If you want to continue reloading, you must mount another tape and restart the reloader. Tapes can be reloaded in any order, but choose your reload options carefully. Generally you should load tapes in the opposite order of creation, and not replace any newer files. This prevents writing over files, such as mailboxes.

The reloader does not delete or expunge files. If you are reconstructing a file system from backup tapes that include many incremental backups, you must occasionally intervene to delete and expunge unwanted old files that are reloaded, in order to ensure adequate file space.

3.2.1 Comparing Backup Tapes

A few notes about the comparer: Occasionally, hardware problems can cause bad backup tapes to be written, without any error having been detected by the dumper. You should always verify a backup by using the "Compare" option of the reloader to *verify* the backup tape. The comparer verifies each bit of every file on the tape, and, for files that still exist, reports any discrepancy.

If you find that an incremental backup tape is deficient, and you decide that the files must be redumped, you must perform a consolidated dump, with a consolidation date equal to the time the bad incremental dump started. You must delete the backup dump tape directory ("*tape-name*.directory" in >dump-maps) by hand, if the the backup-copy finding mechanism is to be aware that the tape has been abandoned.

The reloader produces an error log file in the local directory >reloader-logs.

The dumper assumes that no undetected problems occurred. Thus, it does not run the comparer automatically. The dumper sets backup times when it finishes each tape.

3.3 Finding Backup Copies of Files

In order to retrieve individual files, or groups of files, rather than reloading an entire tape, you must know on what tape the files were dumped. The LMFS backup mechanism provides a way to search the binary tape directories, which are produced by the dumper, to find backup copies of files. This is the Lisp function **lmfs:find-backup-copies**, which you can invoke by clicking on [Local LMFS Operations] in Level 1 of the Files System Editing Operations program. Then click on [Find Backup Copies]. This prompts you for files pathnames.

lmfs:find-backup-copies searches all the binary tape maps at the site for the tape locations of all backup copies of a file. It prompts for names, which it parses with respect to the local host unless you name an explicit host in the pathname. All of the files requested must be from the same host. This applies only to Lisp Machine hosts. It looks at the binary tape maps on the specified host in the directory >dump-maps.

Normally, you specify wildcard pathnames for this function to match. A non-wild pathname is considered to be a valid degenerate case of a wildcard pathname. The default with which all of the names are merged is "local:>**>*.*.*". This function uses the same pathname interpretation conventions as the reloader.

Here is a sample interaction with **lmfs:find-backup-copies**:

```
Enter file pathnames for which to search, separated by commas.
Wildcard names are allowed.  [default LOCAL-LISPM:>**>*.*.*:]
Paths: f:>sys>io>*.lisp, f:>bsg>*.init, f:>lisp>*.q*
Searching for:
  F:>sys>io>*.lisp.*
  F:>bsg>*.init.*
  F:>lisp*>**>*.q*.*
-----
F:>LISPM>COLDLD.QBIN.39, created 2/17/86 00:26:58, on tape fscns001
  (Backup dump of 2/24/86 13:05:44)
F:>LISPM>COLDUT.QFASL.57, created 2/18/86 19:06:36, on tape fscns001
  (Backup dump of 2/24/86 13:05:44)
F:>LISPM>UTILS>NEW>EVAL.QBIN.13, created 2/19/86 04:45:17, on tape fscns001
  (Backup dump of 2/24/86 13:05:44)
....and so on
----- Scan complete.
```

4. Using Tape Facilities on Symbolics Computers

Symbolics supports different tape formats for different purposes. Each format is specific to one Lisp Machine tool. Many people wonder which tool is appropriate for which application.

The *LMFS dumper and reloader* are used for backing up files from a local Lisp Machine file system and reloading those files on the same local Lisp Machine file system, in the same place, at a later date. The intended use of LMFS backup is to reload files onto the same machine from which they were dumped.

The *Distribution dumper and loader* are intended to distribute transportable systems and libraries, defined by system declarations on logical hosts, from one site to another. This tool is best used when transporting many files within a system, rather than transporting just a few unrelated files. The distribution system specializes in finding appropriate source and object versions of systems, appropriate patch files, and so forth. Its use of logical pathnames allows it to create a tape which can be easily loaded into the filesystem of a foreign host.

TAPEX is a format for transferring character (source) files between hosts. It is the only one of these formats which can be read or written by other than a Lisp Machine. TAPEX programs exist for TOPS-20 and Multics, as well as for the Lisp Machine (**tape:tapex**). It cannot deal with any type of file except character files, which are written in standard ASCII. Each individual dump or load requires an interaction.

Carry tape is the most general tool, and is used to dump individual files and sets of files (specified by wildcard filespec) and load them at any site. Recent improvements to the wildcard facility make this a very powerful and easy-to-use tool.

FEP Tapes:

- *IFS tape* is the Initial File System tape. A different tape is shipped with each disk, and is used in case of dire emergency to restore the structure of the FEP file system. This tape should not be used without consulting with Software Support. The FEP reads the IFS tape, and discards all of the data on the disk. The FEP then creates a new root directory, and **empty** files to hold world loads and microcode files. After you initialize the disk with the IFS tape, you read a FEP tape containing the world loads and microcode files.

- *FEP-tape* is a format which allows the user to read and write world loads and microcode files both from and to cartridge tape. Tapes written with this program can be read with the FEP commands Load Microcode CART: and Disk Restore.

4.1 Tape Facilities and Their Uses

Here are the possible tape facilities you can use with a Symbolics computer. Below are the purposes, restrictions, and capabilities of each tape.

Note: The 3600 uses a Cypher tape drive, which is capable of writing tapes of only 20 megabytes. The other machine models use an Archive tape drive, which is capable of writing tapes of more than 20 megabytes. If you write a tape containing more than 20 megabytes (on any machine model excluding the 3600) you will not be able to read it on a 3600. Since only one tape utility (see chart below) limits the data written to tape to below 20 megabytes, you should be aware of this potential incompatibility.

If you use a machine which does not have a local tape drive, make sure that the machine with the tape drive has the **rtape** option in its namespace object. For further information: "Registering a Tape Drive in the Namespace", page 82.

LMFS

Purpose: Backing-up the file system on Symbolics computers.
Restrictions: Use only for LMFS files; may only restore to Symbolics computers.
Tape drive must be local? No
Can span multiple tapes? No
Is tape verifiable? Yes
Can write more than 20 megabytes per tape? Yes
Tapes readable by Lisp or the FEP? Lisp

Distribution

Purpose: Distributing software which is layered (not world loads).
Restrictions: Requires definition of logical pathnames.
Tape drive must be local? No
Can span multiple tapes? No
Is tape verifiable? No
Can write more than 20 megabytes per tape? Yes
Tapes readable by Lisp or the FEP? Lisp

Carry

Purpose: Moving files between hosts.
Restrictions: More difficult to use when many pathnames are needed.
Tape drive must be local? No
Can span multiple tapes? No
Is tape verifiable? Yes
Can write more than 20 megabytes per tape? Yes
Tapes readable by Lisp or the FEP? Lisp

FEP-Tape
Purpose: Writing and reading world load and microcode files.
Restrictions: World load or microcode files only.
Tape drive must be local? No
Can span multiple tapes? Yes
Is tape verifiable? Yes
Can write more than 20 megabytes per tape? Optionally
Tapes readable by Lisp or the FEP? FEP

TAPEX
Purpose: Making tapes in format for compatibility with TOPS-20 and Multix systems.
Restrictions: Use only for Source files.
Tape drive must be local? No
Can span multiple tapes? No
Is tape verifiable? Yes
Can write more than 20 megabytes per tape? Yes

4.2 Distribution Subsystem

The distribution subsystem is used at Symbolics to write and load distribution tapes. Customers use the subsystem mostly for loading the distribution tapes they receive from Symbolics; that is, they restore the files on these tapes to their own file systems. However, users can also use this facility subsystem to write their own tapes.

The distribution tape subsystem runs on a Lisp Machine and requires a tape drive. The tape drive can exist on that machine, or it can be part of another machine (including a non-Lisp Machine) that is accessible through the network and has a remote tape server program.

To restore data from tape to disk, use the Restore Distribution command. To write systems to tape for distribution, use the Distribute Systems command.

Restore Distribution Command

Restore Distribution *keywords*

Restores data from tape to disk. This command reads the directory listing of the files on tape, and then restores the files according to the pathname and property information on the tape. For each file it restores, Restore Distribution checks to see if the file already exists in the file system. If it does not exist, it restores the file. If the file does exist, however, Restore Distribution states that the file that it was supposed to restore already exists in your file system, and asks where to restore it instead. You supply a pathname. *Nowhere* is the Default, meaning *skip this file.*

keywords :Use Disk

:Use Disk {Yes, No} Reads the input from disk (test mode), rather than from tape. Test mode writes a special file which is an image of what would be written from tape. Use this when you are preparing a distribution and want to see what files would be written from tape. The default is No.

Distribute Systems Command

Distribute Systems *Systems keywords*

Writes systems to tape for distribution. Expects the input to be one or more systems. The default is the first system loaded into the current world. After you confirm the command, the Distribute Systems command lists the systems to write to tape, and asks if you want to perform the operation. Your choices are Y, N, Q, or S. Type Y for Yes, N for No, Q for Quit, or S for Selective. If you choose Selective, each file is listed, and you are asked if you want to load that particular file. You can select as many or as few files as you want. After you enter this information, you are prompted for the name of a tape spec.

keywords :File Types :Include Component Systems :Output Destination :Source Category, :Use Disk, :Version

:File Types {Sources, Binaries, Both, Patches-Only} What file types to distribute. The default is sources.

:Include Component Systems
Whether to include the component systems of the systems to be distributed. The default is Yes.

:Output Destination
{Buffer, File, Printer, Stream} Redirects the output of this command to specified streams.

:Source Category
{Basic, Optional, Restricted} Indicates which source category or categories to write to tape for distribution. Basic is the default.

:Use Disk {Yes, No} Reads the input from disk (test mode), rather than from tape. Test mode writes a special file which is an image of what would be written to tape. Use this when you are preparing a distribution and want to see what files would be written to tape. The default is No.

:Version {Latest, Newest, Released} Indicates which version of the system to write to tape for distribution. The type of input expected is: Latest, Newest, or Released. Released is the default.

4.3 Carry-Tape System

The carry-tape system provides a means of dumping selected files or sets of files to magnetic tape (cartridge or industry-compatible) and loading them at a later time, possibly at a different site. Using the Carry-Tape System, you can dump files from any host or set of hosts, and reload them to any place on any host.

The carry-tape system provides a standard, system-independent interchange medium for exchanging single programs and files between sites. It is meant to fill in a gap between the LMFS backup dumpers and the distribution tape system. It does not require you to prepare files or declarative forms in advance.

The carry-tape system has three components:

- The carry-tape dumper
- The carry-tape loader
- The carry-tape lister

4.3.1 The Carry-Tape Dumper

tape:carry-dump *file-or-files* &key *tape-host density reel* (*report* **t**) *Function*

Dumps a file or set of files to a carry-tape. You can dump any type of file. Character files are dumped and reloaded using the Genera character set as an interchange medium. Binary files are dumped and reloaded with the proper byte size as long as either of the following is true:

- The file is of one of the system's known canonical types.
- The operating system on which the file resides knows and can supply the byte size.

file-or-files a pathname, filespec, or list of pathnames and/or filespecs. Wildcard pathnames or filespecs may be used. Recursive ("accordion") wildcards may be used to dump subtrees on those hosts that support them. An example of a pathname which has recursive wildcards is:

```
E:>trees>**>*.*.*
```

tape-host a host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If you do not specify a host, the dumper uses the standard tape host prompt and defaulting mechanism.

density a fixnum, specifying tape density, which may be used when the applicable default is not appropriate.

reel can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

report tells the carry-tape dumper to report its progress as it dumps files. A value of **nil** tells it not to. A value of **t** tells it to report. The default is to report to **zl-user:*standard-output***. Any value besides **nil** or **t** is expected to be a stream to which the reports will be written.

Currently, carry dumps must fit on one tape.

The carry-tape dumper starts by finding out all available information about the files to be dumped, verifying their existence. It then asks for confirmation, and proceeds to dump all the files specified, without intervention.

Here is an example of using the Carry-Tape Dumper:

```
(tape:carry-dump "swanee:>minerals>*.d*")
To be dumped:
swanee:>minerals>*.d*: 7 files
Is this right? (Y or N) Yes.
Type name of tape host (default (CR) = POINTER): scrc
Tape mounted on drive mta0:.
Dumping swanee:>minerals>abel.data.3 (5-bit bytes)
Dumping swanee:>minerals>abel.patch-directory.7
.....
End of dump.
```

4.3.2 The Carry-Tape Loader

The carry-tape loader loads files from a carry-tape. The loader makes no attempt to copy any file properties, including author and creation date. It copies only file contents, and provides reasonable defaults for the target file name.

tape:carry-load &key *host density reel* (*report* **t**) *Function*

host a host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If you do not specify a host, the loader uses the standard tape host prompt and defaulting mechanism.

density a fixnum, specifying tape density, which can be used when the applicable default is not appropriate.

reel can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

report tells the carry-tape loader to report its progress as it loads files. A value of **nil** tells it not to. A value of **t** tells it to report. The default is to report to **zl-user:*standard-output***. Any value besides **nil** or **t** is expected to be a stream, to which the reports will be written.

These arguments are rarely needed.

The carry-tape loader begins its operation by reporting the pathnames given to the dumper, and asks if you wish to load all of the files dumped. If only one filespec or pathname was given, it is assumed that you want to load it all, and no question is asked:

```
(tape:carry-load)
Type name of tape host (default (CR) = APSO): beta
Tape mounted on drive mta0:.
Carry dump made by DCF.
Dump taken at 6/13/86 09:05:22.
Dumped on machine EAGLE.
Dumped: e:>trees>apple.orchard
```

The set of files dumped as a result of each pathname given to the dumper is called a *group*. If many groups were dumped, the loader lists the pathname of each group at the start of its operation, and asks for instructions about which groups are to be loaded (selectively) and which groups are to be skipped:

```
The following groups of files were dumped:
e:>trees>apple.orchard
e:>animals>whales>tails.tales
e:>baseball>runs>foul.*
-----
Load all these files? (ABORT to get out) (Y, Q, or M)
```

The possible responses are:

Y Yes Ignore distinctions of group, and proceed as described below.

Q Query — Query about each group, and proceed as below for those groups that are selected for loading.

M Menu — Same as Q, but present a multiple-choice menu instead of querying for each group.

If you do not want to load anything, you can press ABORT at any time to stop the loader.

The carry-tape loader can either query for the target location of each file to be loaded, or proceed in semi-automatic mode, in which the host and directory from which each file was dumped are used as a key to target loading of subsequent files from that host and directory. The name, type, and version of each file to be loaded are developed automatically from the name, type, and version of the file that was dumped, by means of the same mechanism used by ordinary file copying.

The normal action of the carry-tape loader is to query for each file, with a query of the following form:

```
Load SWANEE:>minerals>rock5.data.6 into BULLWINKLE:/usr2/jones/rock5.data?
          (Y, N, O, or A)?
```

The following responses apply:

Y, SPACE — Load the file into the place specified. The host and directory shown remain the default target directory for all files from this host and directory at the site writing the tape.

N — Do not load this file at all. The host and directory shown remain the default target directory for all files from this host and directory at the site writing the tape, in spite of this.

O — Prompt for another place in which to put this file. The host and directory into which this file is then loaded become the default for all subsequent files from the same host and directory at the site writing the tape. You are queried again at the time such files are encountered.

A — Load the file into the place specified. All further files from the same host and directory at the site writing the tape are then automatically loaded into the same host and directory as this file without querying you.

4.3.3 The Carry-Tape Lister

The carry-tape lister describes what is on a carry-tape. Once started, it does not interact in any way.

tape:carry-list &key *host density reel* (*report* **t**) *Function*

host a host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If you do not specify a host, the lister uses the standard tape host prompt and defaulting mechanism.

density a fixnum, specifying tape density, which may be used when the applicable default is not appropriate.

reel can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

report tells the carry-tape lister to report its progress as it lists files. A value of **nil** tells it not to. The default is to report to **zl-user:*standard-output***. Any value besides **nil** or **t** is expected to be a stream, to which the reports will be written.

These arguments are rarely needed.

4.4 FEP-Tape System

The FEP-Tape system provides a means of writing and reading the cartridge tapes used to distribute world loads and microcode files. World loads and microcode files are distributed using the FEP-tape format because the FEP commands "Load Microcode CART:" and "Disk Restore" can only read this format. Consequently, tapes written with the FEP-tape system can be read by a machine that does not have a working lisp world, although ordinarily they are read by Lisp. The only requirement is that the machine have an initialized FEP file system and a working set of FEP overlay files. For more information about the FEP system and overlay files: See the section "Introduction to the FEP", page 123.

4.4.1 The Contents of a Tape Made with the FEP-Tape System

Tapes made with the FEP-Tape system are created in *sets*. A FEP-Tape tape set is a series of one or more cartridge tapes. Ordinarily, a FEP-Tape set contains a set of microcode files and a world load file. There can be additional world-load files or other files on a tape set. However, there is no particular advantage in writing unrelated world load files onto the same tape.

The first tape in the set may begin with one or more *initial microcode files*. These are files written in cartridge tape stream mode which can be read by the FEP command Load Microcode CART:.

You only have to put initial microcode files on a tape if it has to be readable by a machine that does not have any microcode loaded. As of Genera 7.0, there are no known reasons to write initial microcode files. At some time in the future they might be necessary.

The initial microcode files are usually followed by the microcode and world load files and can continue onto additional tapes. These files are read by the FEP command Disk Restore or by the FEP-Tape application itself. When the FEP reads these files, it restores each one into a previously existing FEP file. **The FEP cannot create new files**. Thus, in order to read a FEP-Tape from the FEP, you must either pre-create a suitable set of files while running Lisp, or write over some other existing files.

4.4.2 How Much Will Fit on a FEP-Tape Tape?

There are two kinds of cartridge tape drives on Symbolics computers, Cypher (4-track) and Archive (9-track). Some 3600s (only the 3600 model, not the series of 3600 machines) have Cypher four-track drives.. These drives can read or write the first 20MBytes of a cartridge tape. By default, the FEP-Tape application writes tapes of no more than 19MBytes. This allows the tapes to be read by Cypher tape drives. The FEP-Tape application assumes that the tapes it reads were written for this type of drive.

Other types of Symbolics computers of the 3600 series, such as the 3640 and the 3670, have Archive nine-track cartridge tape drives. Currently, the FEP-Tape application writes or reads 39MBytes using an Archive tape drive, if, and only if, you give the optional argument to the Read Tape and Write Tape commands. The optional argument is *:Full Length Tapes Yes*. This 39MBytes is less than the full capacity of the cartridge tape.

You should only write 39MByte tapes when you are absolutely sure all of the machines that will read the tape have 9-track cartridge tape drives.

4.4.3 Invoking the FEP-Tape System

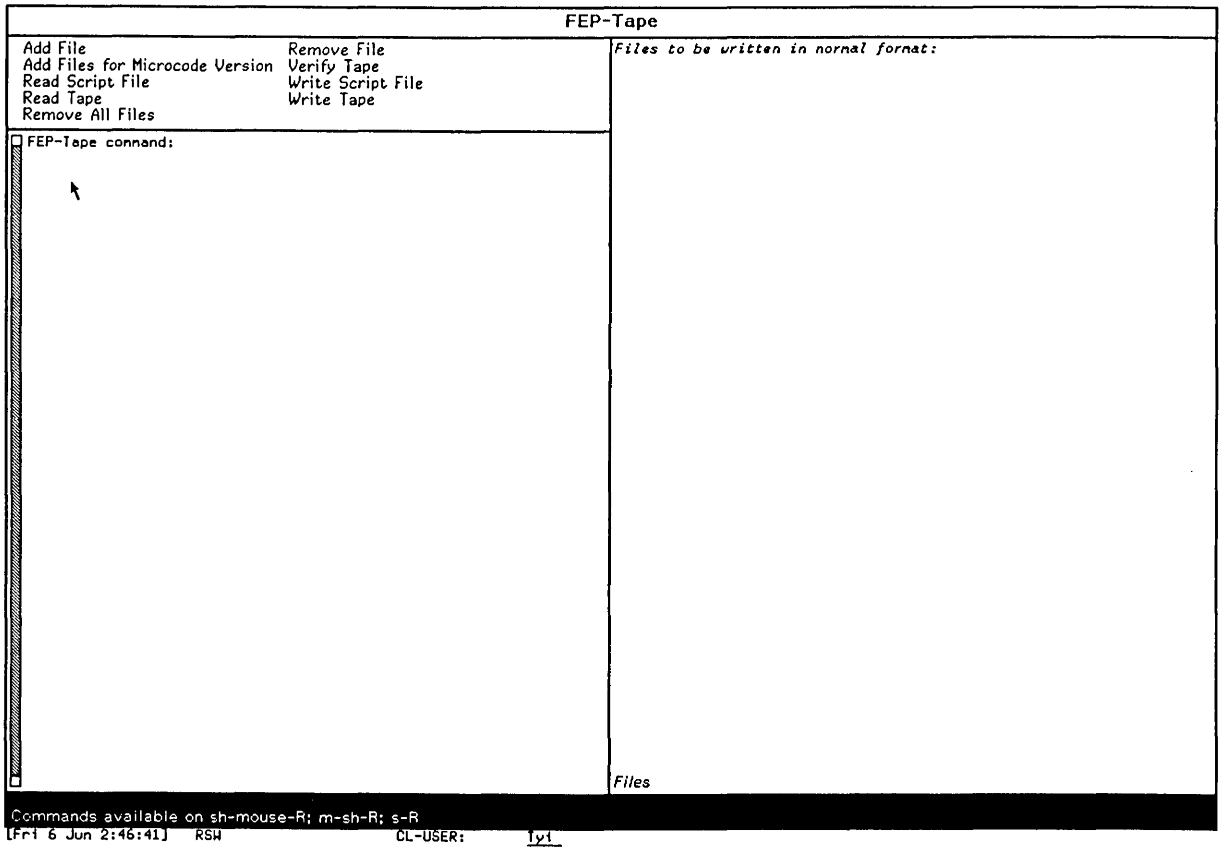
To invoke the FEP-Tape system, type this command to the Command Processor prompt:

```
Select Activity  FEP-Tape
```

This invokes a frame that consists of a command menu, a file display pane, and a listener pane with the prompt:

```
FEP-Tape Command:
```

The command menu frame lists all possible command options. You type commands in the listener pane at the FEP-Tape Command: prompt. When preparing to write a tape, the FEP-Tape application lists files to be written to the tape in the file display pane, as shown in Figure 5.



```
FEP-Tape command:
```

```
Files to be written in normal format:
```

```
Commands available on sh-mouse-R; m-sh-R; s-R
[Fri 6 Jun 2:46:41]   RSW                CL-USER:      Tyi
```

Figure 5. FEP-Tape Menu

The following command menu options are available:

Command *Definition*

Add File Adds a single file to the list of files to be written to tape. The command takes arguments that specify the pathname, whether the file should be written as an initial microcode file, an other file, or both, and a comment to be associated with the file. Only files "other" than the initial microcode files can have comments.

Add Files for Microcode Version
Adds all the microcode files for a particular version of standard microcode to the tape. The command takes arguments to specify the microcode version, whether to write row-major (Genera 7) and column-major (Release 6) array microcode, and whether to set up the initial microcode files.

Read Script File Reads a script file containing a file set that you have prepared in advance. Use this command to read in a list of files that you have selected to write to tape.

Read Tape Reads a FEP-Tape tape. This command prompts for the name of each file, and you specify where to put the file or if you want to skip reading it all together.

Remove All Files Removes all of the files listed for writing to tape.

Remove File Removes one file from the set of files listed for writing to tape.

Verify Tape Reads a newly-written tape set and compares the content of files on disk to the files on the tape.

Write Script File After specifying your tape-set, use this command to save the list of files in a script.

Write Tape Writes the files to tape. This command takes arguments to specify the host with the tape drive to use, (default is local) and whether to write 4-track or 9-track tapes.

4.4.4 Writing a FEP-Tape Tape Set

1. **Select the set of files to be written on the tape.**

   There are two ways to do this. One way is to read in a script file containing a file set that you have prepared beforehand. You can use the command Read Script File to read the list of files.

Another way to select the files to be written to tape is to use one or more FEP-Tape commands to specify the files to be written on the tape. You should use the Add File command or the Add Files for Microcode Version command.

The Add File command adds a single file to the list of files to be written to tape. The command takes arguments that specify the pathname, whether the file should be written as an initial microcode file, an other file, or both, and a comment to be associated with the file. Only files other than initial microcode files can have comments.

The Add Files for Microcode Version command adds all the microcode files for a particular version of the standard microcode (non-Prolog) to the tape. The command takes arguments to specify the microcode version, whether to write row-major (Genera 7.0) or column-major (Release 6.0) array microcode, and whether to set up the initial microcode files.

If the script file containing a file set has some files you do not want on your FEP-Tape tape, use the Remove File or the Remove All Files command to remove some files. The Remove All Files command removes every file listed for writing to tape, and the Remove File command removes just one file form the set of files listed for writing to tape.

2. **Save the list of files in a script file.**

   You can do this with the Write Script File command.

3. **Write the files to tape.**

   Use the Write Tape command to do this. This command takes arguments to specify the host with the tape drive to use , (default is local) and whether to write 4-track or 9-track tapes. The argument to specify the host is *:host,* and the argument to specify the length is *:Full Length Tapes.*

4. **After writing the files to tape you can verify with the Verify Tape command.**

   This command reads a newly written tape set and compares the content of files on disk to the files on the tape.

4.4.5 Reading a FEP-Tape Tape

To read a FEP-Tape tape, use the Read Tape command. This scans the tape. For each file, it prompts with the name of the file, and you specify either where to put the file in the file system or that you want to skip reading it all together. This command takes the arguments *:Full Length Tapes* and *:host.*

The FEP command Disk Restore also reads FEP-Tapes.

5. Multiple Partitions

The Lisp Machine File System (LMFS) allows the use of multiple partitions residing on one or more disk drives. It utilizes one or more files of the FEP file system as the vessels in which it stores its files and directories. These FEP files are called *partitions*. Normally, there is one large partition, usually called LMFS.FILE.1. All the files created by LMFS actually reside inside this FEP file, but the existence of these files is known only to LMFS, whose purpose it is to manage them; they are not known to the FEP file system. Since FEP files are limited to one particular disk drive, if a LMFS file is to utilize the space available on multiple drives, partitions must be created on each drive on which it is desired that LMFS store files. Then, LMFS must be instructed to use these partitions.

The selection of partitions to be used by LMFS is determined by a database called the *file system partition table* (FSPT). It is contained in a FEP file named >fspt.fspt on a boot drive. The FSPT is optional. If it is not present, LMFS uses lmfs.file on the FEP boot drive. The FSPT is a simple character database containing the actual pathnames (in the FEP file system) of the partitions to be used for file system access.

If any machine at your site has more than one disk, it may be difficult to find the disk location of the FSPT. In order to make finding the location of a FSPT easy, insert the Set LMFS FSPT Unit command in your Hello.boot file. This command causes LMFS to look for the file named >fspt.fspt on the unit (disk unit) specified. For example, if you put your FSPT on disk unit 2, put the following in your Hello.boot file:

```
Set LMFS FSPT Unit 2
```

Each partition in the file system knows how many partitions make up the file system. Only the FSPT, which is used only at LMFS startup time, indicates the locations of these partitions. That is, the file system databases in the actual partitions do not contain drive and partition numbers or FEP pathnames. Thus, when LMFS is down, partitions can be moved around using Copy File (m-X); as long as the FSPT is edited to indicate their new locations, LMFS comes up (when required) using the moved partitions. **Note:** Since the Copy File (m-X) command copies files according to byte size, you may need to edit the byte count of the partition for the copy file command to work. To do this, multiply the number of blocks by 1152, since partitions were previously created with a byte size of 0. For example:

```
1152 * number of blocks
```

The FSPT is edited only to move partitions around or to add a partition. When you add partitions to the file system, the file system automatically rewrites the FSPT database to include the locations of new partitions.

5.1 Free Records

The basic unit of allocation in the Lisp Machine File System is the *record*. A record is 1152 32-bit words, or four disk blocks. Each file system object is made from an integral number of records. At any time, each record is *in use* (representing an existing file system object) or *free* (not representing anything and free to be used in new objects). When the file system needs to find a new free block to create or grow an object, it does not search through the records looking for a free one, because that would require many disk operations and be very slow. Instead, the file system uses a redundant data structure called the *free record map*, kept in several blocks in a known location in the file system partition. The map has one bit for each record in the file system; this bit marks whether the record is free or in use. The file system can find a free record quickly by examining this map.

If the file system crashes, or something else goes wrong, the contents of the free record map can become inconsistent with the contents of the file system itself. For each record, two different errors are conceivable.

- The record might actually be in use, representing part of an object, but marked as "free" in the map. The system is designed so that this cannot happen, but hardware problems might cause it to happen anyway.

- The record might actually be free, but marked as "in use" in the map.

The first error is much worse than the second; the file system might use the record for a new object even though it is currently representing some existing object, which could destroy the existing object. If the second error occurs, the record simply is not allocated even though it could be. Such a record is said to be *lost.*

The file system is written so that a crash can only cause the second kind of error. While the file system is operating, it maintains a *free buffer* in its data structures in virtual memory. The free buffer is a pool of records that are not actually in use, but are marked as being in use in the free record map on the disk. When it needs to allocate a record, it draws on one of these; when it frees up a record, it adds the record to this buffer. When the buffer gets too big, some records are removed from the buffer and marked as "free" in the map on the disk; when the buffer runs low, more records are marked as "in use" in the map on the disk, and are added to the buffer. So, if the machine is cold booted, or the file system crashes, the records that are in the buffer are lost, but no errors of the first kind are caused. The size of the buffer is maintained at about 30 records, so each crash loses 30 records. To recover, log out of the machine or use the [Flush Free Buffer] command to flush the entire free buffer and mark the records as "free" in the map on the disk. To use the [Flush Free Buffer] command, press SELECT F to enter the File System Editing Program. Click right on [Local LMFS Operations] to invoke the second level of the progam, where you can click on [Flush Free Buffer]. After the buffer has been flushed, you can cold boot the machine without losing any blocks.

Lost records can be found again by the salvager. See the section "Salvager", page 43.

You can check the number of free records in the file system by using the File System Editing Operations program. First, press SELECT F to select the program. Then, click right on [Local LMFS Operations], to invoke the second level of operations. In the second level, if you click left on [Free Records], the program displays a line for each block of the file map, telling you which records are covered by that block, the number of such records, and how many are marked as free. It also tells you how many free records (marked as "in use" in the map) are in the free buffer, and finally displays a grand total of the number of free, used, and total records in the file system.

To find out how many records are actually in use, click middle on [Free Records] to prepare a printable report of record use throughout the file system. This has to pass over every object in the file system, and so it takes some time, especially on large file systems. The discrepancy between the answer of this function and the answer you get when you click left on [Free Records], tells you how many lost records there are; if there are a lot, you might want to run the salvager.

Clicking right on [Free Records] displays how many records are in use in each partition. This information is necessary for commands such as [Grow Partition] that allow you to change the size of partitions, add partitions, or remove partitions.

5.2 Salvager

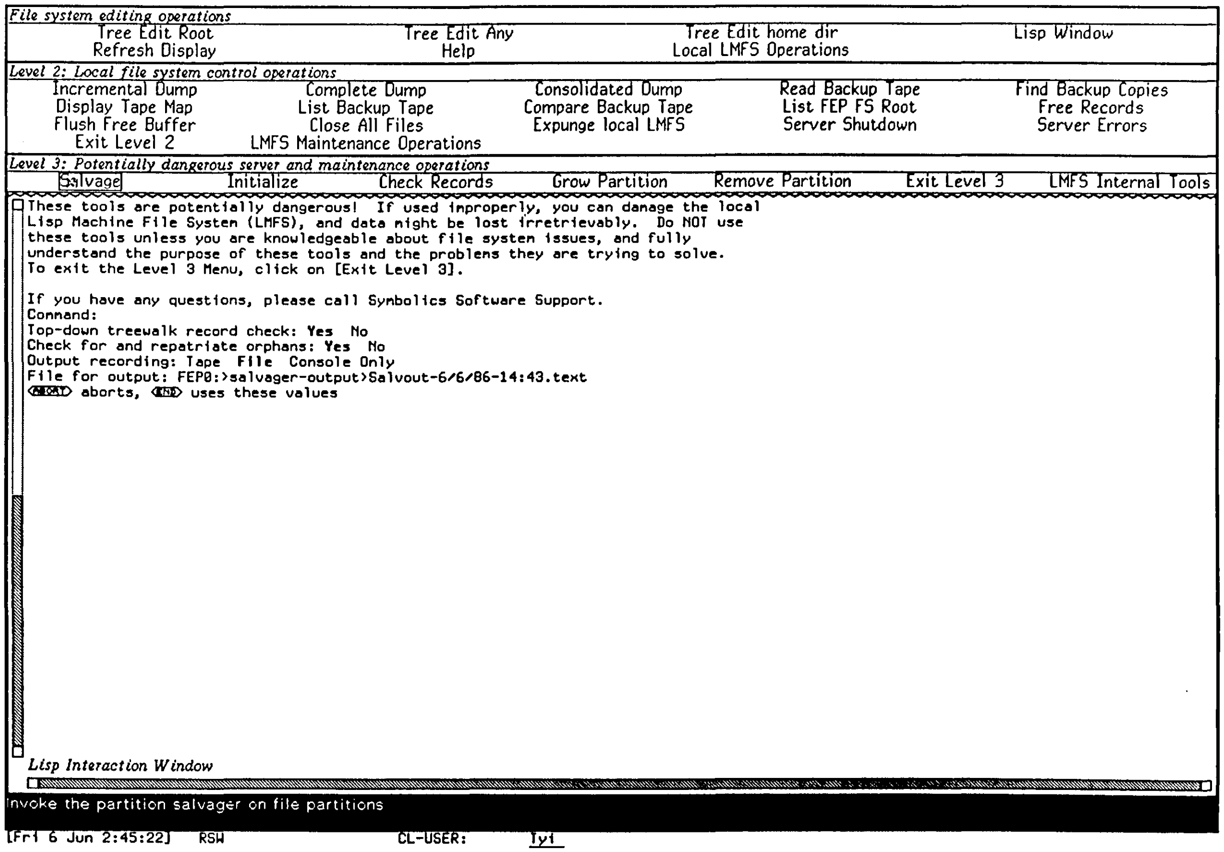
The *salvager* is a program that reads every LMFS record of the file system and finds and fixes certain inconsistencies and errors. It can fix two classes of problems.

- It can see which records are in use and which are free, and update the free record map to reflect the current state of the file system. This is how you recover lost records.

- It can find objects that are stored in a file system partition but are not referenced by any directory. Such objects are called *orphans*; they exist only if some problem has occurred, such as a file system crash during the creation of a file, or an unanticipated failure of some sort. The salvager finds such objects and puts them back into the directory hierarchy (*repatriates* them).

5.2.1 Using the Salvager

To run the salvager, Press SELECT F to select the File System Editing Operations program. Click on [Local LMFS Operations] to invoke the second level of the program. Next, click on [LMFS Maintenance Operations] to invoke the third level of the program. Now click on [Salvage] to obtain a menu of options. If you have a local file system of multiple partitions (occupying multiple FEP files), you are presented with a menu of partitions to process. This menu, which is an Accept-Values menu, also includes questions about salvager operations. In addition to listing the partitions to be salvaged, the menu offers you the options as shown in figure 6.



```
These tools are potentially dangerous!  If used improperly, you can damage the local
Lisp Machine File System (LMFS), and data might be lost irretrievably.  Do NOT use
these tools unless you are knowledgeable about file system issues, and fully
understand the purpose of these tools and the problems they are trying to solve.
To exit the Level 3 Menu, click on [Exit Level 3].

If you have any questions, please call Symbolics Software Support.
Command:
Top-down treewalk record check: Yes  No
Check for and repatriate orphans: Yes  No
Output recording: Tape  File  Console Only
File for output: FEP0:>salvager-output>Salvout-6/6/86-14:43.text
<ABORT> aborts, <END> uses these values
```

```
Invoke the partition salvager on file partitions
[Fri 6 Jun 2:45:22]   RSW          CL-USER:    Tyi
```

Figure 6. Salvager Options

Here are the options:

```
Top-down treewalk record check:  yes no
```

```
Check for and repatriate orphans:  yes no

Output recording:  Tape File Console only

File for output:
```

The first items on the menu constitute a list of partitions you can select for processing by the salvager. You can choose some or all of the partitions for processing.

The second menu option, `Top-down treewalk record check`, offers to scan all of the directories and files in the local LMFS and report any damaged records (hardware or software), disappeared files, or any other problems. This search starts at the root and goes through all of the file system, directory by directory, and is performed after all other salvaging activity.

Note: If you deselect any partition for repatriation, then the next menu item, which offers to check for and repatriate orphans, disappears. This happens because it is impossible to construct an accurate model of the hierarchy if each partition is not scanned.

The third menu option, `Check for and repatriate orphans`, offers to find orphaned objects and put them back into the directory hierarchy. During this scan, the salvager also replaces bad directory records with good ones.

The fourth menu option, `Output recordings`, offers to log the salvager output either to tape, in a file, or only to the console.

- If you choose the `Tape` option for output recording, every message goes onto the tape as soon as it is produced because of a special format that is used. Using an industry-compatible tape ensures that all messages appear on tape. If you use a cartridge tape, this is not fully guaranteed. The following forms may be used to view the tape produced in this way:

  lmfs:print-salvager-output-tape &optional *tape-spec* (*stream* **zl:standard-output**) *Function*

  Prints the contents of the tape created by the salvager. If you do not supply any arguments you are prompted for a tape spec, and output prints to the console.

  lmfs:copy-salvager-output-tape-to-file &optional *tape-spec* *pathname* *Function*

  Prints the contents of the tape created by the salvager to a file. You are prompted for any arguments not given.

- If you choose the `File` option for output recording, you must supply a file name. The default file is a FEP file, on boot unit 0. Every time a message is written to file a **:finish** is done to the file, so that even if the system crashes, the file is intact with all the messages up to the point of failure.

If you decide to put the output recording in a FEP file, make sure there is enough room, probably about 100 blocks. If you have your output recording sent to another host, choose a host that you are sure will stay on the network during the logging process.

Warning: If you only have one Symbolics computer or one file server, you can't use the `File` option because you may not put the output recording in a local LMFS file.

There are currently no tools for automatically processing a file containing a log of salvager output.

- If you choose the `Console only` option for output recording, note that this is not usually the device of choice. You should choose this option when there is no other means of logging available.
- If any problems occur while the log is running, such as a file closing or a disrupted network connection, a menu appears. This menu asks what to do about continuing the salvager's log. If you enter the debugger while the log is being recorded you are offered restart options for discontinuing or reselecting log options.

5.2.2 What the Salvager Does

The salvager always reconstructs the free record maps. Running the salvager takes about two minutes per thousand records of file partition.

When the salvager is repatriating an orphan and it cannot find the directory in which the orphan is supposed to reside, it creates a new directory as an inferior of the directory >repatriations, with a name like lost-1 or lost-2. After a repatriating salvager run, you should examine these directories. When the salvager repatriates an object, it types out a message saying that it did so. One of these messages might cause a **MORE** pause. If you plan to leave your console unattended while the salvager is working, you might want to disable **MORE** pauses before you start it.

Note: The salvager always considers storage occupied by orphans to be "in use" for purposes of the free record map, even if it is not repatriating the orphans. Thus, if many orphans existed, they could use up a great deal of disk space. But normally, orphans do not occur at all. When the salvager repatriates it also "fixes" disk errors and misplaced records or directories by replacing them with fresh, empty ones. By nature of the repatriation process, no files are lost in this way.

5.3 Adding a Partition to LMFS

You can add partitions to LMFS by using the File System Editing Operations program. First, press SELECT F to select the menu for that program. Click on [Local LMFS Operations] to invoke the second level of the menu. Then, click on [LMFS Maintenance Operations] to invoke the third level of the menu. In the third level of the menu, clicking right on [Initialize] yields a menu of initialization options, which offers [New File System] and [Auxiliary Partition] as choices. Choosing [New File System] is similar to clicking *left* on [Initialize]; it initializes a partition to be the basis of a file system. Clicking left on [Initialize] prompts for an initial LMFS partition location, offering FEP0:>lmfs.file as a default location.

When you add a new partition or a partition on another disk, the disk should be free of errors and properly initialized and formatted, and the partition should exist.

To add another partition, choose [Auxiliary Partition]. Enter the pathname of the FEP file to be used as the new partition. (The default pathname presented, which is correct for [New File System], is never correct for adding [Auxiliary Partition].) Then choose [Do It]. The system then performs much verification and error checking, roughly as much as when initializing a new partition. It must not be interrupted while performing these actions. When finished, it adds the partition and edits the FSPT automatically.

6. Creating More Room on the Local Disk

There are two file systems available on the Symbolics computer: the Lisp Machine File System (LMFS) and the FEP File System (FEP FS). LMFS is a general-purpose, highly flexible file system, suitable for everyday use. Currently, only the Symbolics processor understands how to operate on LMFS files. The FEP FS is a simple, basic file system that both the Symbolics computer and front-end processors understand how to access.

The FEP FS contains two kinds of files. The first kind, called a *FEP file* is used to store information the FEP uses to do things like boot Lisp and manage virtual memory; this includes world load files, microcode load files, paging files and boot files. The second kind of file is also a FEP file, but it is a very large file and it is called a *file system partition.* One or more partitions are what LMFS uses to store its structure and data. User files are stored by LMFS in partitions.

Sometimes the Save World or Copy World commands might inform you that you have run out of FEP file system space, and offers you the option of editing your FEP directory. For systems with 167-Mbyte or more of storage, you should delete and expunge old, unneeded world loads, and then resume from the Save World "out of room" error or retry the Copy World operation. You should not delete any world loads from a 140-Mbyte system. See the section "Instructions for Managing Disk Space on the 3640 with a 140 Megabyte Disk", page 71.

It is wise to keep a large (about 40K), noncritical world load or extra paging file on the Symbolics computer's disk, so it will be available for the FEP Disk Restore command to use in case all world loads become nonfunctional.

Sometimes, writing a file out to a LMFS produces an "out of room" error. This means that the present allocation of that particular LMFS is not large enough to accommodate your request for space. It might help to expunge directories with deleted files in them to remove unneeded versions of files, using the Zmacs command Dired (m-X).

If you still do not have enough space after you have deleted and expunged unnecessary files, consider creating an auxiliary file partition. You should only do so, however, on systems that have at least 280 Mbytes of storage. This is because 140-Mbyte systems have no room at all for an auxiliary file partition, and allocating an auxiliary file partition on a 167-Mbyte system can limit your space for large world loads. Even for 280-Mbyte systems, you are trading off world load space for file space when you create auxiliary partitions.

Be sure to reserve enough FEP file system space for a two world loads (the amount of FEP file space required for this depends on the size of the released worlds): a disk copy of your current world and a spare world load for the FEP Disk Restore command to use.

For details on how to create auxiliary file partitions: See the section "LMFS Multiple Partitions", page 41.

Warning: Once you have created an auxiliary file partition, you should never delete it, because you would lose all the data contained in that partition and make the entire Lisp Machine File System unusable.

If you run out of room while writing a LMFS file and then create a new partition to increase the LMFS space, you cannot resume the file operation that failed. Instead, you must abort that operation by pressing c-ABORT, and then retry the operation.

6.1 Allocating Extra Paging Space

Programs that use large amounts of virtual memory might require you to allocate additional paging space, to perform better or to perform at all. Only systems with at least 280 Mbytes of disk storage have enough room to permit additional paging files without adversely affecting the maintenance of worlds on the machine. In order to add an extra paging file to your virtual memory set, you must first create a FEP file using the Create FEP File command. Then, you can activate the paging file from Lisp by using the Add Paging File command. To create a 20-K block paging file on unit 0 type:

```
Create FEP File fep0:>page1.page 20000
```

After creating the extra paging file, any boot files should be modified to use this new paging file. Use the Declare Paging-files command in the boot file to load any paging files you want to use. A typical boot file before inserting the command to load the paging file might look something like this:

```
Clear Machine
Load Microcode >3600-mic.mic.389
Load World >Dist-7-0.load
Set Chaos-Address 401
Start
```

After creating the new paging file, edit your boot file to include the Declare Paging-files command. The new Boot.boot file might look something like this:

```
Clear Machine
Load Microcode >3600-mic.mic.389
Declare Paging-files fep0:page1
Load World >Dist-7-0.load
Clear Paging
Set Chaos-Address 401
start
```

For information about the Declare Paging-files command: See the section "FEP System Commands: General Usage", page 133.

It is safe to delete extra paging files, but only if they are not in active use. You cannot change a paging file that is being used, without booting. To change the paging area you have set up, first boot without adding the paging file to be deleted. Be sure to cold boot by hand, and when you type the Declare Paging-Files command, *do not* specify the extra paging file that you intend to delete. Once you have booted, you can delete the unwanted paging file by editing the FEP directory. Be sure to remove any references to the file from your boot file as well.

6.2 Adding a Paging File From Lisp

If you want to add a paging file from Lisp, use the new command:

```
Command:  Add Paging File
```

Prior to adding the paging file you may have to create the FEP file by using the command:

```
Create FEP File
```

to create the paging file.

Add Paging File Command

Add Paging File *pathname :prepend*

Adds a pathname as a paging file.

pathname — The pathname of the new paging file. The default pathname is the disk unit from which you most recently booted. For example, if you most recently booted from FEP1:>, the default paging file might look like:

```
FEP1:>.page
```

keywords — :prepend

:prepend — {*yes no*} Yes means to put the paging file at the beginning of the list of swap space to use when new space is needed. This makes the new paging file used almost immediately. No, which is the default, puts the paging file at the end of the list of paging files. Consequently, this new paging file will not be used until the previous swap space is completely used.

7. Manipulating World Loads

Once you receive a distribution world from Symbolics you have many choices about modifying and customizing that world to suit your needs. Here is a typical sequence of events:

1. You boot the original distribution world on one machine and make changes to the world to customize it for your site.

2. You save the site-customized world to use over again.

3. You distribute the site-customized world to other machines at the site.

4. Individuals may further modify the distributed site-customized world by loading special programs or applications into it.

5. If you have loaded additional systems into your world, you may want to optimize your world load to enhance paging performance.

6. Finally, any individually-customized world should be saved, as well as the site-customized world.

The commands you use to accomplish these tasks: booting worlds, customizing and optimizing worlds, saving modified worlds (complete worlds and Incremental Disk Saved worlds), and distributing world loads at your site, are described in this chapter.

7.1 Booting a World

Booting a world involves a series of commands given to the FEP. Putting these commands in a file called Boot.boot is an easy way to avoid typing these commands manually every time you wish to boot your machine.

Here is the sequence of commands found in a typical boot file:

```
Clear Machine
Load Microcode microcode-file-name
Declare Paging-files files-names
Load World distribution-world-file-name
Set Chaos-Address this-machine's-chaos-address
Start
```

If you encounter any errors with a particular command, try that command again.

microcode-file-name refers to the microcode needed for a specific system and hardware configuration. For example, if you are installing Genera 7.0 on a 3640, then the appropriate microcode file name is 3640-MIC.MIC.*version-number*. *Version-number* is the released microcode version, which changes from release to release. Note that if you have an IFU, you need a different microcode. For a list of Genera 7.0 microcode types: See: "Genera 7.0 Microcode Types" in the *Software Installation Guide.*

Declare Paging-files declares *file-names* to be the paging files for all subsequent Load World commands until a new Declare Paging-files command overrides it. This command replaces the Add Paging-files command in boot files. For more information about the Declare Paging-files command: See the section "FEP System Commands: General Usage", page 133.

Distribution-world-file-name refers to the name of the world distributed on the disk (for example, >release-7-0-dist.load).

This-machine's-chaos-address refers to the chaosnet address of the Symbolics computer. You must select a chaos address for each machine at your site. For more information about chaosnet addresses: See the section "Choosing Chaosnet Addresses", page 94.

To avoid typing all those commands every time you boot a world, use a file named >Boot.boot. This file contains text similar to the above example. You can edit >Boot.boot at any time, to reflect your current world load and microcode file names. The Boot command activates the file; for example, the command `Boot >boot.boot` tells the FEP to execute the commands in the file >Boot.boot.

You can have more than one Boot.boot file. This organization is useful if your site maintains various world loads, some of which may have special programs loaded into their environment. If you want to switch back and forth between world loads, having multiple boot files makes it easy to load these worlds. For example, you might want to have one boot file named Boot.boot, which has the commands for loading one world load, and another boot file, boot2.boot, which loads contains commands to load a different world load.

7.2 Customizing and Saving Worlds

Symbolics distributes new software on distribution tapes in the form of world loads. These distribution worlds, once they are installed on your machines, may be altered in many ways. The sequence of steps you can take with the initial distribution world load to achieve a desired outcome is varied. This section outlines the procedures for accomplishing certain customizations for your site. All of the commands mentioned in the procedures are documented in this section.

If you want to simply load a new world and save an incremental version of it, here is a basic sequence of steps you might take:

- Boot the new world. For information on booting: See the section "Booting a World", page 53.

- Use the Set Site command to make the world site-specific

- Use the Optimize World command to optimize the world

- Use the Save World Incremental command to save an incremental world on your machine

If you want to add some special programs, or systems, to the initial distribution world load, and then create an incremental world, this is the sequence of steps you might take:

- Boot the new world. For information on booting: See the section "Booting a World", page 53.

- Use the Set Site command to make the world site-specific

- Use the Load System command to load special software into your world

- Use the Optimize World command to optimize the world

- Use the Save World Incremental command to save an incremental world on your machine

If you want to add patches (updates to the software distributed by Symbolics) to the initial distribution world load, and then create an incremental world, this is the sequence of steps you might take:

- Boot the new world. For information on booting: See the section "Booting a World", page 53.

- Use the Set Site command to make the world site-specific

- Use the Load Patches command to add updated software to your world

- Use the Optimize World command to optimize the world

- Use the Save World Incremental command to save an incremental world on your machine

For information about Incremental Disk Save (IDS): See the section "Incremental Disk Save (IDS)", page 62.

For information about the Optimize World command: See the section "Optimizing a World", page 59.

If your site needs to build a distribution world, follow these steps:

- Boot the distribution world. For information on booting: See the section "Booting a World", page 53.
- Use the Set Site command to make the world site-specific
- Use the Load System command to load any additional software
- Use the Set Site command with the site name as Distribution
- Use the function **(si:full-gc)** to garbage-collect the world
- Use the function **(si:reorder-memory)** to optimizes paging performance
- Use the Save World Complete command to save the world

7.2.1 Commands Used to Customize and Save Worlds

7.2.1.1 The Load World Command

Load World *file-name*
Restores enough of the saved world in the computer so that you can start up the machine. It prints both the desired microcode version for this world and the currently loaded microcode version. The default value of *file-name* is the last file name given to the Load World command. Its initial default is >Genera-World.load.

```
    FEP Command:  Load World FEP1:>Genera-7-0.load
```

7.2.1.2 Set Site Command

Set Site *site name*

Starts a dialogue to set the current site to be *site name.* This command is used to configure the software and identify your machine before you use a new world load. It should be the first thing you type to your machine after booting the new software.

When a new world is booted for the first time, the herald gives the machine name as *DIS-LOCAL-HOST*. You are prompted in the course of the Set Site dialogue for a name for the machine.

You need the following information to use the Set Site command:

Site name What you call the location of your machines. This might be the name of your company, or, if you are more whimsical-minded, it might be related to the machine names you have chosen. In the sample dialogues, we have chosen the site name Downunder.

Name of the local host

The name of the Symbolics computer you are configuring. See the section "Why Do You Name Machines and Printers?" in *User's Guide to Symbolics Computers.* In the sample dialogues, we have chosen machine names Koala and Kangaroo.

Name of the namespace server

The name of the machine where the namespace database is stored.

Chaosnet address of the namespace server

The octal number that identifies the location of the namespace server on the network. You can use Show Host *machine name* or (**zl:hostat** "*machine name*") to find this number.

If you are configuring a new site, you also need:

SYS host

The machine where the sources are to be stored.

Host for bug reports

The machine to which bug reports are to be sent.

SYS:SITE; directory translation

The physical pathname that sys:site; translates to on the sys host. See the section "What is a Logical Pathname?" in *User's Guide to Symbolics Computers.* In the sample dialogues, this is

```
koala:>sys>site>
```

System account

The user-id that the system uses when a server logs into a machine. In the sample dialogues, we have chosen Wombat.

Load System Command

Load System *system keywords*

Loads a system into the current world.

system

Name of the system to load. The default is the last system loaded.

keywords

:Condition :Load Patches :Query :Redefinitions Ok :Silent :Simulate :Version

:Condition

{always, never, newly-compiled} Under what conditions to load each file in the system. Always means load each file. Newly-compiled means load a file only if it has been compiled since the last load. The default is always.

:Load Patches {yes, no} Whether to load patches after loading the system. The default is yes.

:Query {Everything, Confirm-only, No} Whether to query before loading. Everything means query before loading each file. Confirm-only means create a list of all the files to be loaded and then ask for confirmation before proceeding. No means just go ahead and load the system without asking any questions. The default is Confirm-only. The mentioned default is Everything.

:Redefinitions Ok
{yes, no} Controls what happens if the system asks for confirmation of any redefinition warnings during the loading process. Yes means assume that all requests for confirmation are answered yes and proceed. No means pause at each redefinition and await confirmation. The default is No. The mentioned default is Yes. This allows you to start loading a system that you know will take a long time to load and leave it to finish by itself without interruption for questions such as "Warning: *function-name* being redefined, ok? (Y or N)".

:Silent {yes, no} Whether to turn off output to the console while the load is going on. The default is no. Adding this keyword to your Load System command string is the same as :silent yes.

:Simulate {yes, no} Print a simulation of what compiling and loading would do. The default is no. Adding this keyword to your Load System command string is the same as :simulate yes.

:Version {released, latest, newest, use-default, *version-number*, *version-name*} Which version number to load. The default is use-default, that is latest.

Note: This command only loads a system. If you want to compile and load a system: See the section "Compile System Command" in *User's Guide to Symbolics Computers*.

Load Patches Command

Load Patches *system keywords*

Loads patches into the current world for the indicated systems or for all systems. See the function **load-patches** in *Program Development Utilities*.

system {All *system-name1, system-name2* ... } The default is All.

keywords :Query, :Save, :Show

:Query {yes no ask} Whether to ask for confirmation before loading each patch. The default is no.

:Save {*file-spec*, Prompt, No-Save} The file in which to save the world with all patches loaded. Omitting this keyword means do not save the world. The default when this keyword is added to your command is Prompt which means save the world and then prompt for a pathname.

:Show {yes no ask} Whether to print the patch comments as each patch is loaded. The default is yes.

7.2.1.3 Optimizing a World

If you load special software or programs into your world you may want to use the Optimize World command to improve paging performance.

The Optimize World command prepares a world to be saved after you have loaded your own special programs or layered software into the original distribution world. Optimize World improves paging performance by reorganizing compiled functions and certain data in virtual memory, so related things are on the same page. Paging performance is the speed at which virtual memory is swapped in order to access data. Once you have used Optimize World, your world load will page less than if it had been saved without running this program beforehand. Optimize World does not move objects that were already optimized in the distribution world load; it only moves things that you have loaded into that world.

Why to use the Optimize World Command

You should use Optimize World if you load your own programs or layered software into the distribution world and want to save this world for later use. Once you have processed a world with the Optimize World command you can boot and reuse this world over-and-over again, with the continued benefit of improved paging performance. The Optimize World command is one of the SmartStore storage management facilities.

When to use the Optimize World Command

It is recommended that you use the Optimize World command shortly before you plan to disk-save a world load.

When it is unnecessary to use the Optimize World Command

It is not necessary to use Optimize World if you have simply loaded the distribution world and customized it for your site without loading any additional programs.

What happens when the command is used

After you enter the Optimize World command it asks you if you are ready to begin an optimization. If you respond yes, the command begins optimization. When the program finishes, another message appears informing you that the process is complete. However, if you would like to see status reports of the reordering process, you can specify the keyword **:show** when you enter the command.

Optimize World typically takes about one-half hour to execute, but this time period varies according to the size of the world load and the total amount of main memory. **NOTE: During the time that Optimize World is running your machine does not respond to the network nor to the terminal. You are not able to use your machine.**

Optimize World Command

Optimize World *keywords*

Optimizes the world that is currently loaded into your environment. Use this command if you load special programs or systems in addition to the distribution by reorganizing the world to improve paging performance.

keywords
:show

:show Displays the progress of the optimization process on the screen.

7.2.1.4 Using the Optimize World Command

Here is the recommended procedure for making a customized world with your own programs loaded into the world:

1. Boot a suitable base world, such as the site-customized version of the distribution world load.

2. Then type the following sequence of commands at the Command Processor prompt. (Note that the default Command Processor prompt in an uncustomized system is `Command:`. The prompt for your site may be different, however, due to customization.)

   ← geschieht in Rel-7 angeblich, automatisch

   ```
   Command:  Start GC :Ephemeral
   Command:  Disable Services
   Command:  Login a suitable user :Init File none
   Command:  Load System name of system
   Command:  Optimize World
   ```

   To see the progress of the optimization display on the screen type:

   ```
   Command:  Optimize World :show
   ```

   And then use the Save World command:

   ```
   Command:  Save World Incremental filename
   ```

- For most applications it is better not to do **(si:full-gc)**. This is because using Incremental Disk Save after **(si:full-gc)** renders your world the same size as the world from which you started. **(si:full-gc)** generally does not gain anything if done in a freshly booted world that has only had a program loaded with the EGC turned on, assuming that the freshly booted world is a distribution world (which is a garbage-free world). If you decide to use **(si:full-gc)**, do it immediately before Optimize World, and use Complete rather than Incremental in the Save World.

- At the point in the sequence above when you use the Load System command, you can type Load System and/or Load File as many times as you need to in order to load all of your special systems and additional programs.

- If you choose to save your optimized world with the Save World Incremental command, then you must keep the parent world from which you made the optimized world. The parent world must be available to the machine in order for the incremental world to boot. For more information about incremental worlds, see the section *Incremental Disk Save* in *Installation and Site Operations.*

- Although the Optimize World command reorganizes the pages in the world, it has little effect on the actual size of the world.

7.2.1.5 Saving a World

To save a world, use the Save World command.

Save World Command

Save World (Complete or Incremental) *file-spec*

Saves the current world. The system prompts (Complete or Incremental) if Incremental Disk Save is enabled. You specify Complete to do a save of the entire world, or Incremental to do an Incremental Disk Save. If Incremental Disk Save is not enabled, the prompt is (Complete). You enter Complete by pressing `SPACE` or by typing `complete`.

file-spec The pathname to use for the saved world. The default is the FEP file specification for the local machine, based on the version number of the current system and on whether the save is to be complete or incremental.

7.2.1.6 Incremental Disk Save (IDS)

Incremental Disk Save (IDS) is a facility that allows you to save modified worlds using much less disk space than if you saved the entire world. IDS saves a world by saving only the pages that have been changed; it does not save a copy of the entire world. This copy is called the *incremental world* and contains only the pages that have been modified. The original world, before it is modified, is the *ancestor* of the modified world and is referred to as the *parent world.*

IDS has two major advantages over a full world save. One is that you can save an incremental world using much less disk space than if you saved the entire world. The second is that you can easily keep multiple incremental versions of a world on the same disk because the incremental disk saves use so much less disk space. In other words, you can take a parent world, make multiple different incremental worlds, and save all the incremental worlds using a minimum of disk space.

Performance is another consideration when using IDS. IDS uses extra wired memory which slightly impairs page faulting. Thus, an IDS-world will be slower than a non-IDS world.

The following diagram shows how you can have multiple versions of an incremental world. You might start with a distribution world (world 1) and modify it to reflect your site-specific information. You would then have one parent world and one incremental world (incremental world 2). You could then load another layered product, such as Pascal, into the incremental world. After using IDS to save the incremental changes, you would have one parent world and two incremental worlds (incremental worlds 2 and 3). You could continue creating more incremental worlds from the site-specific incremental world. For example, you might boot your machine with the site-specific world and then load your own application into it. After incrementally disk-saving this, you would now have one parent world and three incremental worlds (incremental worlds 2, 3, and 4). You might continue to modify one of these incremental worlds, such as by loading patches into the incremental world with Pascal. This would give you incremental world 5 and so on.

Here is another example that shows how disk space is conserved. Assume you have a distribution world that is 28,000 blocks in size; this is the parent world. You might want to set the site and load site modifications and save the resulting world. Doing a nonincremental disk save results in a world containing the parent and the modifications, which might be 30,000 or more blocks. Doing an incremental disk save would result in a world of perhaps 3,000 to 5,000 blocks that consists of only the new blocks and modified blocks of the parent. You might then boot this incremental world, load a system into it, and incrementally disk save the result, which might again be 3,000 to 5,000 blocks. You now have ready access to three worlds (a distribution world, a world customized for your site, and a world with an additional system) that total perhaps 36,000 blocks instead of

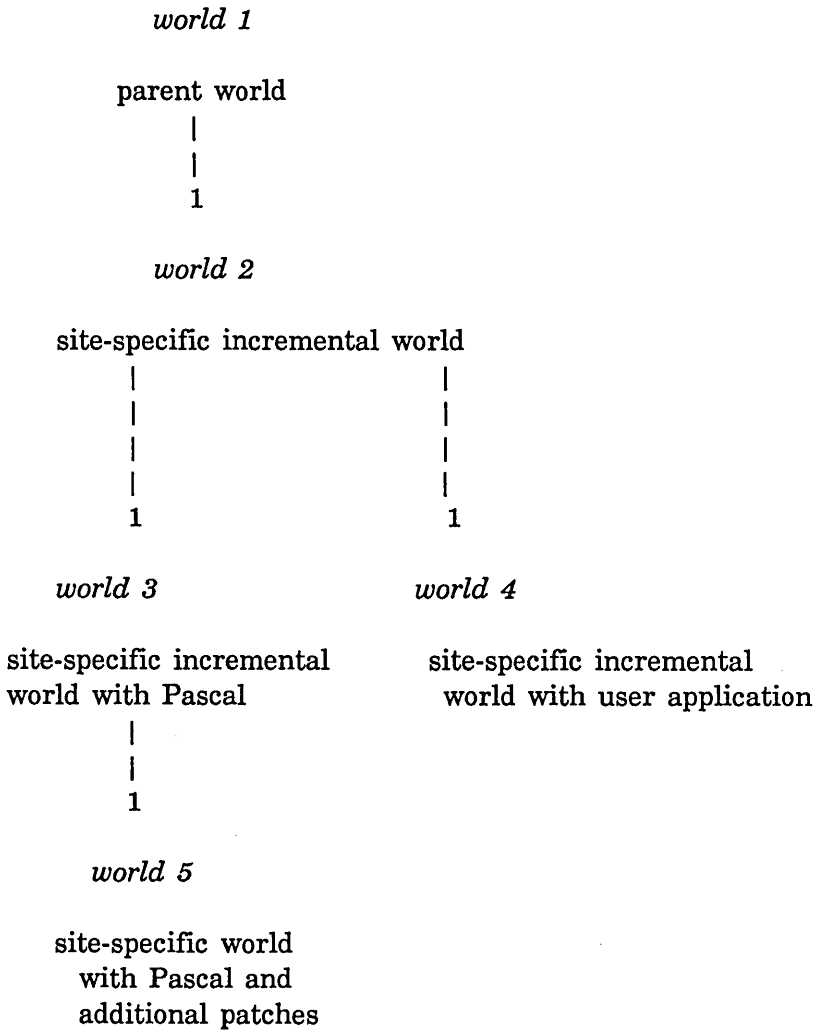

world 1
parent world
1
world 2
site-specific incremental world
1
1
world 3
site-specific incremental world with Pascal
world 4
site-specific incremental world with user application
1
world 5
site-specific world with Pascal and additional patches

Figure 7. Incremental Worlds

90,000 blocks. You always have the option to do a full disk save on a multigeneration incremental world; for example, you could do a full disk save on the incremental world with the system loaded into it.

Note: It is important to keep all ancestors for all incremental worlds you intend to use. If you delete some ancestor, a descendent will not be usable because it requires blocks that are in the ancestor.

The format of world files (full and incremental) is such that they can freely be moved around within a site and (less practically) between sites. Thus, a site can place the distribution world supplied by Symbolics on all the machines at the site. You can boot this distribution world on one machine, modify it, and incrementally disk-save it. You can then distribute the resulting incremental world to all the other machines. The parent world, as well as any other intermediate incremental worlds, must be available on that machine for this incremental world to boot.

7.2.1.7 Using Incremental Disk Save (IDS)

You can enable IDS with the Enable IDS command in your boot file.

The following steps describe the procedure for enabling IDS:

1. Edit the boot file to include the Enable IDS command, as shown in the following example:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Load World world-file-name
   Set Chaos-Address this-machine's-chaos-address
   Enable IDS
   Start
   ```

   The Enable IDS command must precede the Start command.

2. Boot the machine using the recently modified boot file.

3. Save the world using either the Save World command or **zl:disk-save**. IDS is enabled by default. To disable IDS, replace the Enable IDS command with the Disable IDS command in your boot file.

   This IDS-enabled world can be copied to other Symbolics computers at your site.

A world that is saved (either completely or incrementally) with IDS enabled retains this characteristic; IDS is enabled in any of the subsequently saved worlds. Therefore, you do not need to use the Enable IDS command again. However, doing so does not harm anything and ensures that IDS is enabled. (Resetting the FEP does disable IDS.)

To do an incremental disk save, proceed normally to modify your world; for example, doing site modifications, loading local site patches, loading any systems that will be used, and so on.

After making the modifications, save the world by using either the Save World command.

Garbage
Collection
lohnt
nicht

Note: Using **gc-immediately** or **si:full-gc** on a world prior to using IDS negates the benefits of using the command. This is because IDS saves only those pages of the world that have been changed; both the Start GC command and **si:full-gc** change the organization of the world. Thus, the size of a world which is garbage-collected and then saved with IDS is not substantially smaller. However, it is recommended that you use the ephemeral-object garbage collector (EGC) on your world. The EGC is enabled by default in Genera 7.0.

If IDS is enabled, you are asked:

```
(Complete or Incremental)
```

Answer Incremental if you want to save an incremental world; answer Complete if you want to save a full image, for example, if you want to condense collected patches. You are then prompted for the pathname for the saved file. A full save gives the same default as before, namely Genera-*major-minor* or System-*nnn-mmm*. An incremental save changes this default in the following ways:

- "Genera-" is prefaced with "Inc-" or "System-" is replaced with "Inc-".

- "-from-" is appended.

- A trimmed version of the loaded world name (the pathname that appears in the first line of the herald) is appended.

- If the result is longer than 32 characters (the limit of filenames in the FEP file system), the filename is truncated and the last 4 characters within the limit are replaced with -etc.

Examples: the pathname for "loaded world" is a possible modification of the *previous* default):

Loaded world	*Action*	*Default*
`Genera-7-0`	`Set Site`	`Inc-Genera-7-0-from-Genera-7-0`
`Inc-SITE-7-0-from-Rel-7.-0`	`Load Patches`	`Inc-271-99-from-IncSITE-7-0--etc`
`Inc-SITE2-from-Inc-SITE-7-0-etc`	`Load Fortran`	`Inc-271-99-from-IncSITE2-fro-etc`
`Inc-FORTRAN-from-Inc-SITE2-etc`		

7.2.1.8 Suggestions for Using IDS

For maximum flexibility, you should try using many levels of Incremental Disk Save. For example:

Steps		*World*
0		Distribution World
1	Boot the distribution world Use Set Site Incrementally save	world-1
2	Reboot with world-1 Load site patches Incrementally save	world-2
3	Reboot with world-2 Load a system Incrementally save	world-3

This encompasses a total of four worlds (a distribution world and three incrementally saved worlds) instead of two worlds (a distribution world and one collective save). The most stable modifications should be made first. That way, if the system that was loaded in the third step changes, it is not necessary to redo the Set Site or load the site patches; it is only necessary to reboot the world with site patches (world-2), reload the system, and incrementally save. If the site maintainer wishes to load new local site patches, the second and third steps need to be repeated, but not the first. The users of the result of the third save have the option to use the old site patches (and not get the new site patches) or rebuild their system with the new site patches.

Note that enabling IDS is pervasive across disk-saves. That is, if you turn it on before saving a world load, the resulting world has the facility enabled in it when that world is again booted.

7.3 Transferring Worlds to Other Machines

Once you have loaded a distribution world on one machine, you may want to transfer it to another machine. You use the Copy World command to do this. When a machine receives a new world, it must also receive new microcode. To receive new microcode, you must use the Copy Microcode command.

Before transferring a world to another machine, it is necessary to look at the contents of FEP directory on that machine, to see what world loads and microcode files are currently on the machine. To look at the FEP directory, use the Show FEP Directory command.

Show FEP Directory Command

Show FEP Directory *host unit*

Displays a description of the FEP files on *unit.*

host A host on the network. The default is local.

unit Disk structure. The default is FEP0:.
unit can be one of the following:

- An integer smaller than 20., interpreted as a disk unit number on the local host.
- An integer larger than 19., interpreted as the Chaosnet address of a remote host. Displays the contents of unit 0 on that host.
- A symbol, interpreted as the name of a remote host. Displays the contents of unit 0 on that host.
- A string of the form "*host|unit*", where *host* is the name or Chaosnet address of a remote host and *unit* is an integer representing a disk unit number on that host.

Show FEP Directory first displays an estimate of the number of free blocks and the proportion of blocks used on *unit.* It then displays a summary of the files on *unit*, one line per file. For each file, it displays the file name, the length in blocks and in bytes, the byte size, the creation date, the comment, and the author.

The following function copies a world from another machine:

Copy World Command

Copy World *file destination keywords*

Makes a copy of a world load.

file FEP file spec. *file* is required (no default).

destination FEP file spec. Required when copying a world from the local host to another host. When copying a world to the local host the default is same as the source file specification.

keywords :Block Count, :Start Block, :Update Boot File

:Block Count Number of blocks to copy. The default is the length of the band, meaning copy the entire band.

:Start Block Number of the block to start with. The default is 0, meaning begin at the beginning.

:Update Boot file
{FEP-file-spec none}. The default is the current default boot file name if *destination* is the local host.

You install microcode on a specific machine using the Copy Microcode command. You can do this after installing the new microcode at the site, with the distribution loader.

Copy Microcode Command

Copy Microcode {*version or pathname*} *destination keywords*

Installs a version of microcode.

version or pathname
Microcode version number or pathname to copy. *version* is a microcode version number (in decimal). *pathname* rarely needs to be supplied. It defaults to a file on FEP*n*:> (where *n* is unit number of the boot disk) whose name is based on the microcode name and version. (The file resides in the logical directory sys:l-ucode;.) The *version* actually stands for the file *appropriate-hardware*-MIC.MIC.*version* on FEP*n*:>. (See the Section "Genera 7.0 Microcode Types" in the Software Installation Guide).

destination FEP file specification. The pathname on your FEP*n*:> directory. The default is created from the microcode version.

keywords :update boot file

:update boot file
{FEP-file-spec none}. The default is the current default boot file name.

The logical directory *sys:l-ucode;* includes multiple types of microcode for each version number. The correct microcode to install depends upon the particular hardware configuration of your machine. When your machine is shipped, the default microcode filename is correct, but if your machine is upgraded (for example, an FPA board is installed) you might need to override the default used by the Copy Microcode command to get the correct microcode type for your configuration. Below is an example of how you would get the microcode type for your configuration.

```
Copy Microcode 3600-fpa-mic.mic.389
```

If you use the wrong microcode for your configuration, your machine will not boot, except in the case where your system has an FPA and you use a non-FPA microcode. In this case, the machine functions normally, but does not make use of the FPA at all.

If you are copying copying microcode from a machine running Genera 7.0 to a machine running Release 6.1, **you must specify the microcode file name**, rather than just specifying the microcode version. The name of the microcode files changed from Release 6.1 to Genera 7.0, and the Copy Microcode command does not understand the name change. It tries to find a microcode version with the same name components as the old microcode.

For example, to distribute a new microcode using the Copy Microcode command on the machine that will receive the new microcode, type:

```
    Copy Microcode  SYS:L-UCODE; 3600-mic.mic.389
```

For a list of microcode types for Genera 7.0: See the section "Genera 7.0 Microcode Types" in the *Software Installation Guide.*

When the Copy Microcode command asks if you want to update your boot file, answer Yes. The file is now updated.

8. Instructions for Managing Disk Space on the 3640 with a 140 Megabyte Disk

Since the 140 megabyte disk drive of the 3640 contains a smaller paging file than the 3600 or 3670, you must manage your 3640 FEP file system differently. For a complete description of paging files: See the section "FEP File Types", page 150.

This section describes the different procedures that you follow to manipulate paging space when:

- Loading the world.
- Customizing and saving that world load.
- Saving future world loads.

The disk of your 3640 contains a world load file, a large paging file (called Page.page), and an auxiliary file (called Aux.page) that is the same size as the world load file. You use the auxiliary file in one way for normal operation and in another way when putting a new world on the disk.

WARNING: If your system does not contain an auxiliary file (use the Show FEP Directory command to look for the file named *Aux.page*), call your field representative.

In normal operation, you boot a world load file and use both Page.page and the auxiliary file for paging. In this case, you call the auxiliary file Aux.page.

When you want to create a new world or transfer a new world to the disk, you boot your world load file and use only Page.page for paging. Instead of using the auxiliary file for paging, you rename it and use it to receive a new world you are creating. Once you have successfully created the new world, you rename the old world load file to Aux.page and use it as your auxiliary paging file. For specific instructions for this procedure, see "Installing Genera 7.0 on a 3640 with One 140-Mbyte Disk" in the *Software Installation Guide.*

The auxiliary file is always actively in use, either as:

- A paging file (in normal operation)
- The target file for new world load

8.1 Customizing and Saving the World on a 3640

The shipped configuration assumes the auxiliary file (Aux.page) is to receive your site's customized world load and so contains just one actual paging file (Page.page). **Note:** The size of the distribution world, the Page.page file, and Aux.page file in this example are only examples, since the sizes of these files vary from release to release.

distribution-world.load.1	30,000
Aux.page.1	30,000
Page.page.1	45,000

A customized, normal configuration uses the auxiliary file as a paging file and so contains two paging files:

New-world.load.1	30,000
Aux.page.1	30,000
Page.page.1	45,000

To create your customized world, follow these instructions:

1. Boot the distribution world using the correct microcode. Use only Page.page.1 for paging and reserve the auxiliary file. **Do not boot with the Aux.page file.** You should initially boot by hand rather than use the boot file so that you can set your chaos address and give the correct Add Paging instruction:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Declare Paging-Files  fep:page
   Load World world-load-file-name
   Set Chaos-Address this-machine's-chaos-address
   Start
   ```

2. Log in by using **si:login-to-sys-host**, for example:

   ```
   (si:login-to-sys-host)
   ```

3. Rename the auxiliary file to whatever name you wish, for example:

   ```
   Rename File FEP:>Aux.page.1 FEP:>New-world.load.1
   ```

4. Customize the booted world and then save it into your new world load file:

```
Save World FEP:>New-world.load.1
```

Since you are asking to save the world into an existing file, you are prompted for an action with which to proceed. The correct answer is Overwrite. Then you are asked if you want to update the boot file. Answer yes. The Set Chaos line that you manually typed is added to the boot file at this time.

5. Rename the distribution world to be the auxiliary file:

```
Rename File FEP:>distribution-world.LOAD.1 FEP:>Aux.page.1
```

6. At this point, you should edit the boot file, FEP:>Boot.boot, to add the auxiliary file as an additional paging file. Insert this line:

```
Declare Paging-Files  fep0:page aux
```

after the Load Microcode command. For an explanation of the Declare Paging- Files FEP command: See the section "FEP System Commands: General Usage", page 133.

Your edited boot file should look like this:

```
Clear Machine
Load Microcode microcode-file-name
Declare Paging-Files  fep0:page aux
Load World world-load-file-name
Set Chaos-Address this-machine's-chaos-address
Start
```

7. Save the edited version.

8. Log out and halt the machine.

9. Boot the new world using the boot file.

8.2 Saving Subsequent Worlds on a 3640

Whenever you wish to create a new world on your 3640 disk, you must follow a procedure similar to that shown above.

1. Boot manually, and do *not* type the Declare Paging-Files fep0:aux command, since you will be saving the latest world into it:

```
Clear Machine
Load Microcode microcode-file-name
Declare Paging-Files  fep0:page
Load World world-load-file-name
Set Chaos-Address this-machine's-chaos-address
Start
```

2. Login by using **si:login-to-sys-host**, for example:

   ```
   (si:login-to-sys-host)
   ```

3. Rename the auxiliary file to whatever name you wish, for example:

   ```
   Rename File FEP:>Aux.page.1 FEP:>Newer-world.load.1
   ```

4. Either customize the booted world and save it into your new world load file, or transfer the world from some other machine:

   ```
   Save World FEP:>Newer-world.load.1
   ```

   or:

   ```
   Copy World "other-machine-name"|FEP0:>remote-world-name.load
   FEP0:>newer-world.load
   ```

   Since you are asking to save the world into an existing file, you are prompted for an action with which to proceed. The correct answer is Overwrite. Then you are asked if you want to update the boot file. Answer yes.

5. Rename the old world to the auxiliary file to be used for paging:

   ```
   Rename File FEP:>Old-world.LOAD.1 FEP:>Aux.page.1
   ```

6. Log out and halt the machine.

7. Boot the machine using the boot file.

9. Enabling the Who-Calls Database At Your Site

The **who-calls** database is a cache that maps *names*, which are symbols, to code and variables that use those symbols in some way. A name can be used as a constant, a variable, a function, a macro, an instance variable, a condition, and a few others.

The **who-calls** database is activated when you use the Set Site command during site customization if the database has not already been enabled or disabled. By default, the Set Site command calls the function **(si:enable-who-calls :new)**, which records only the callers in any layered products, special software, or programs loaded into the world. The database is turned on in this way to make it easy to include new software or layered products in the database. **Note: (si:enable-who-calls :new)** does not cause the callers in code in the Distribution world to be recorded.

If you prefer to have all of the Symbolics-supplied software in the database, you can use the function **si:enable-who-calls** with the argument **:all.** Using the argument **:all** has the additional advantage that you can create the database once and then save the database when you save the world. However, creating a full database takes a long time and about 2000 pages of storage.

If you want only explicitly-named files to be in the database, use the function **si:enable-who-calls** with the argument **:explicit**.

If you use the function **si:enable-who-calls** without any arguments, it prompts you for an argument and offers help.

si:enable-who-calls *mode* *Function*

mode can be one of the following:

:all Creates a full callers database. This takes many minutes and about 2000 pages of storage. **:all** also queries about the old state.

:all-remake Creates a full callers database but does not query about the old state. This takes many minutes and about 2000 pages of storage.

:new Creates a callers database that includes only new functions.

:all-no-make Creates a callers database that includes only new functions. When you follow it with a **si:full-gc** the entire database is created. This takes many minutes and about 2000 pages of storage.

:explicit Enables items to be added to the callers database explicitly by using **si:add-files-to-who-calls-database** or **si:add-system-to-who-calls-database.**

After you use the function **si:enable-who-calls** with the argument best suited for the type of database you want to create, you can compress the database by using either **(si:compress-who-calls-database)** or **(si:full-gc).** **(si:compress-who-calls-database)** makes the database smaller, thus making the world load smaller if you save the world after using the function. If the database is large, this function may take many minutes to compress the database. Here are examples of the ways in which you can couple these functions:

- If you use the function **(si:enable-who-calls :new)**, you can load any special files (these may be layered products, private systems, the local site system, etc.) and then you use either either **(si:compress-who-calls-database)** or **(si:full-gc)** to compress the database.

- If you want to have the entire body of Symbolics-supplied software in your **who-calls** database, there are two options you can take during the customization of the distribution world. For the first option use this form:

  (si:enable-who-calls ':all-no-make)

  Followed by this form:

  (si:full-gc)

  (si:enable-who-calls ':all-no-make) creates a callers database that includes only new functions. When you do a **si:full-gc** the entire database is created. This takes a long time and about 2000 pages of storage. The world load file, when you save it, will be about 2000 pages bigger than it would be if you used the **:new** mode.

  The second option is to use this form:

  (si:enable-who-calls ':all)

  Followed by this form:

  (si:compress-who-calls-database)

  (si:enable-who-calls ':all) creates a full callers database. This also takes a long time and about 2000 pages of storage. **:all** also queries about the old state. **:all-remake** suppresses the query. **si:compress-who-calls-database** compresses the who-calls database by garbage-collecting the database.

- If you load your own programs into the database and wish to include any specific items in the database, you can use the function:

  (si:enable-who-calls ':explicit)

  followed by either **(si:compress-who-calls-database)** or **(si:full-gc)**.

  (si:enable-who-calls :explicit) enables you to add items to the callers database explicitly, by using **(si:add-files-to-who-calls-database)** or **(si:add-system-to-who-calls-database)**.

Note: If you use **(si:enable-who-calls :explicit)** or **(si:enable-who-calls :new)**, load a small amount of software into the world, and then save the world, there is no advantage to compressing or doing a full garbage collection.

It is better to use **(si:compress-who-calls-database)** rather than **(si:full-gc)**, since it is faster and does not prohibit the possibility of saving the results using Incremental Disk Save (IDS). (Using Incremental Disk Save (IDS) after **(si:full-gc)** renders your world the same size as the world from which you started.)

To disable the recording of callers completely, call the function **si:disable-who-calls**.

10. Namespace Attributes for a New LISPM Host

To create the new host object in the site namespace, type:

```
Edit Namespace Object Any RETURN
```

Click on [Create] and then specify the name of the new host.

Now add the attributes as shown in the sample template for your host type:

System Type*: LISPM
Service: *Set:* CHAOS-STATUS CHAOS-SIMPLE CHAOS-STATUS *Global-name*
Service: *Set:* SHOW-USERS CHAOS NAME *Global-name*
Service: *Set:* TIME CHAOS-SIMPLE TIME-SIMPLE *Global-name*
Service: *Set:* UPTIME CHAOS-SIMPLE UPTIME-SIMPLE *Global-name*
Service: *Set:* LOGIN CHAOS TELNET *Global-name*
Service: *Set:* LOGIN CHAOS SUPDUP *Global-name*
Service: *Set:* LOGIN CHAOS 3600-LOGIN *Global-name*
Service: *Set:* SEND CHAOS CONVERSE *Global-name*
Service: *Set:* SEND CHAOS SEND *Global-name*
Service: *Set:* NAMESPACE CHAOS NAMESPACE *Global-name*
Service: *Set:* NAMESPACE-TIMESTAMP CHAOS-SIMPLE NAMESPACE-TIMESTAMP *Global-name*
Service: *Set:* LISPM-FINGER CHAOS-SIMPLE LISPM-FINGER *Global-name*
Service: *Set:* FILE CHAOS NFILE *Global-name*
Service: *Set:* *Global-name*
Address: *Pair:* CHAOS *nnnnn Global-name*

In the Address attribute line, *nnnnn* represents a valid 5-digit octal Chaos address.

11. Maintaining the Namespace Database

After you bring up the new release, you can perform the following steps as part of your site administration activities:

- Register users, hosts, printers, and networks in the site's namespace
- Move the new release to other Lisp Machines at the site
- Install new releases distributed in patch tape format
- Install new releases distributed in world load format
- Install world loads from other sites

You should reflect any changes, such as new users or changes in the site's hardware, in the namespace database. Register new hosts and printers in the namespace database before connecting them to the network or supporting host. Register new users in the namespace database either before they use the new release or as part of the process when they log in for the first time. Whether you are registering new users or new hardware, the process is most easily done by copying and modifying an existing object of the same type using the namespace editor (invoked by the Edit Namespace Object command or the **tv:edit-namespace-object** function), and then saving the new object. Once it has been saved, it is part of your site's configuration and all machines running the new release know about the new object the next time they boot, or sooner in some cases. (See the section "Namespace System" in *Networks.*)

To use the namespace editor to create and update objects, click on [Namespace] on the System menu or select a Lisp Listener and type the Edit Namespace Object command or the **tv:edit-namespace-object** function.

11.1 Registering Users

The easiest registration strategy is to create your entry by copying someone else's entry. To copy another entry, use the Edit Namespace Object command and then click on the namespace editor's [Copy] command. If you are the first person at your site to register, copy the user object for user LISP-MACHINE.

Individual users can run the Edit Namespace Object command for themselves the first time they use a new release. If they have not created an appropriate user object, then logging into the new release fails because the system does not find the user object in the namespace database. Should this be the case, the system offers to create the user object with the Edit Namespace Object command. Click on the namespace editor's [Copy] command and copy the user object for user LISP-MACHINE.

For an overview of changing objects in the namespace database: See the section "Maintaining the Namespace Database", page 81.

11.2 Registering Hosts

To create the new host object in the site namespace, type:

```
Edit Namespace Object Any RETURN
```

Click on [Create] and then specify the name of the new host. Or, use the form:

```
(tv:edit-namespace-object :host New-Host :create t)
```

To determine the service attributes required for your LISPM: See the section "Namespace Attributes for a New LISPM Host", page 79.

11.3 Registering a Tape Drive in the Namespace

To register a tape drive for a Symbolics machine, use the `Edit Namespace Object` command to add the tape drive to the namespace database. Specify `tape chaos rtape` as the last `Service: Set:` entry in the host object. See the section "**service:** Host Object Attribute" in *Networks*. For example:

```
Edit Namespace Object :host Janis RETURN
```

pops up namespace menu that displays all the attributes of the host Janis. Add the tape service by clicking on `Set:` of the last `Service: Set:` entry, then type:

```
tape chaos rtape RETURN
```

12. Herald Functions and Variables

The herald is the multiline greeting message that is displayed when you cold boot or warm boot, or after you call the Show Herald command, the Save World command or the FEP Disk Restore command. When you cold or warm boot, your machine displays a full-screen herald, which gives some basic information about your machine and how to operate it. When you display the herald with the Show Herald command, you see an abbreviated herald.

Show Herald Command

Show Herald *keywords*

Displays the herald message. The herald is part of the screen display on a cold booted machine. It shows you the name of the FEP file or partition for the current world load, any comment added to the herald, a measure of the physical memory and swapping space available, the versions of the systems that are running, the site name, and the machine's own host name.

keywords :Detailed,:Output Destination

:Detailed {yes no} Whether or not to print the version information in full detail. The default is no.

:Output Destination
{*buffer file printer stream window*} Where to display the information. The default is the current window.

si:system-additional-info *Variable*
If you provide an additional comment to the herald using the Save World command or the **zl:disk-save** function, **zl:disk-save** sets this variable in the saved environment. The value of this variable should always be a string.

The following functions and variables are not actually used in printing the herald, but provide the same kind of information as does the Show Herald command.

si:describe-system-versions &optional *(stream* **zl:standard-output***)* *Function*
Prints the major and minor version numbers of the release and of any systems whose version number is not exactly the released version number. The output is sent to *stream,* which defaults to **standard-output**.

si:get-system-version &optional (*system* **"zl-user:system"**) *Function*
Returns three values. The first two are the major and minor version numbers of the version of *system* currently loaded into the machine. The

third is the status of the system, as a keyword symbol: **:experimental**, **:released**, **:obsolete**, or **:broken**. *system* defaults to **zl-user:system**. This returns **nil** if that system is not present at all.

si:print-system-status-warning &optional (*system "System"*) *Function*

If *system*'s status is **:experimental**, prints out a warning reminding the user to load patches. If *system*'s status is **:broken**, prints out a warning cautioning the user that the system may not work. Otherwise, it does nothing.

13. Installation Procedures

13.1 Adding a New Machine to an Existing Site

New Symbolics computers are shipped from the factory with Genera 7.0 preinstalled on the disk. Before using the machine, you must perform a configuration procedure, as described in this section.

Before performing the configuration procedure, register the machine as a new host in the site's namespace. For information about how to register users and hardware: See the section "Maintaining the Namespace Database", page 81.

The configuration procedure for adding a new machine to an existing site adds site-specific information and customizations to the preinstalled software. The site-specific information tells the machine its site's name and the location of the site's namespace server. The customizations include, for example, the installation of optional software products.

The procedure can be done either by copying an already configured world from another machine at the site or by booting the preinstalled world, adding the site-specific information, and saving that world again. If all your site's machines should have identical characteristics (for instance, the same optional software installed), it is easiest to copy a world from another machine.

The procedure differs for machines that have less than 167 Mbytes of total disk capacity (for example, model 3640 computers with single 140-Mbyte disks).

You need to perform only one of the procedures described here, depending on your circumstances. When copying a world to another machine, if your machine has a 470, 300, or 167 megabyte disk: See the section "Copying Worlds to Machines with Large Disks", page 85. If your machine has a 167 or 140 megabyte disk: See the section "Copying Worlds to Machines with Small Disks", page 87. When configuring worlds, if your machine has a 470, 300, or 167 megabyte disk: See the section "Configuring Preinstalled Worlds on Machines with Large Disks", page 91. If your machine has a 167 or 140 megabyte disk: See the section "Configuring Preinstalled Worlds on Machines with Small Disks", page 89.

13.1.1 Copying Worlds to Machines with Large Disks

Your machine's disk should have enough unused space for the copied world (about 40,000 blocks). Copy this world from another machine at your site. Perform the procedure as follows if you want to copy a world from another machine at the site and the new machine's disk capacity is at least 167 Mbytes:

1. Boot the distribution world on the new machine manually by typing the following command sequence:

```
Clear Machine
Load Microcode microcode-file-name
Load World distribution-world-file-name
Set Chaos-Address this-machine's-chaos-address
Start
```

This is an example of the commands contained in a boot file. Using this sequence of commands adds page.page as paging space and leaves aux.page unused.

microcode-file-name refers to the microcode needed for a specific system and hardware configuration. For example, if you are installing Genera 7.0 on a 3640, then the appropriate microcode file name is 3640-MIC.MIC.version-number. *Version-number* is the microcode version number, which changes from release to release. Note that if you have an IFU, you need a different microcode. For a list of Genera 7.0 microcode types see "Genera 7.0 Microcode Types" in the *Software Installation Guide.*

Distribution-world-file-name refers to the name of the world distributed on the disk (for example, >genera-7-0-dist.load).

This-machines's-chaos-address refers to the chaosnet address of the Symbolics computer. You must select a chaos address for each machine at your site. For more information about chaosnet address: See the section "Choosing Chaosnet Addresses", page 94.

2. After the machine boots, Enter the Set Site command and follow the instructions for configuring a non-namespace host at an established site. Use this dialogue if you are installing new software on a machine at your site that is not the namespace server. The machine on which you are installing new software is named Kangaroo.

   What you type is underlined in this example:

   ```
   Command: Set Site (site name) downunder
   Define a new site named DOWNUNDER (as opposed to looking for an existing
   definition of DOWNUNDER on disk)? (Y or N) N
   ```

   You answer no because your site is already defined.

   ```
   What host is the namespace server for DOWNUNDER (default: local): Koala
   Chaosnet address for KOALA: 401
   Host responds as KOALA, ok? (Y or N) Y
   The local host is now KANGAROO
   Command:
   ```

3. Save the world using the Save World command.

   ```
   Save World Complete  FEP0:>configured-genera-world.load
   ```

   To see the an example of using the Save World command: See the section "Using the Save World Command".

4. Use the Copy World command on the new machine to transfer a configured world from another machine at the site. For example:

   ```
   Copy World Kangaroo|FEP0:>configured-genera-world.load
   ```

   Here, the already configured world on host Kangaroo is copied to the machine on which this command is entered. Copy World asks if you want to update your boot file to load the configured world. Answer Yes. The boot file is automatically updated.

5. Log out, halt the machine using the Halt Machine command, and boot the new world. If you have backed up the world, you can then safely delete and expunge the preinstalled world to free up space on your disk. For information about backing up your world: See the section "Performing Dumps", page 14.

13.1.2 Copying Worlds to Machines with Small Disks

Perform the procedure as follows if you want to copy a world from another machine at the site and the new machine's disk capacity is less than 167 Mbytes:

1. Boot the distribution world on the new machine. To do this, find the boot file that is on the machine. The pathname is: FEP0:>Boot.boot. Edit the file to include the correct Chaos-Address for the machine you are going to boot. Save the file, halt the machine using the Halt Machine command, and boot again. Here is an example of the commands contained in a boot file.

   Clear Machine
   Load Microcode *microcode-file-name*
   Load World *distribution-world-file-name*
   Set Chaos-Address *this-machine's-chaos-address*
   Start

   Using this sequence of commands will add page.page as paging space and leave aux.page unused.

   microcode-file-name refers to the microcode needed for a specific system and hardware configuration. For example, if you are installing Genera 7.0 on a 3640, then the appropriate microcode file name is 3640-MIC.MIC.version-number. *Version-number* is the microcode version number, which changes which changes from release to release. Note that if you have an IFU, you need a different microcode. For a list of Genera 7.0 microcode types See: "Genera 7.0 Microcode Types" in the Software Installation Guide.

Distribution-world-file-name refers to the name of the world distributed on the disk (for example, >genera-7-0-dist.load).

This-machines's-chaos-address refers to the chaosnet address of the Symbolics computer. You must select a chaos address for each machine at your site. For more information about chaosnet address: See the section "Choosing Chaosnet Addresses", page 94.

If the boot file on disk contains the line `Add Paging >aux.page.1`, remove it from the command sequence. This is the file you use to receive the copied world. The file which contains extra paging space may also have a different name from aux.page, probably a name which denotes that it is additional paging space.

For information about assigning your machine a name and a chaosnet address, see *Choosing Machine Names and Chaosnet Addresses* in the *Software Installation Concepts Primer.*

2. Enter the Set Site command and follow the instructions for configuring a non-namespace host at an established site. Use this dialogue if you are installing new software on a machine at your site that is not the namespace server.

   What you type is underlined in this example:

   ```
   Command:  Set Site  (site name)  downunder
   Define a new site named DOWNUNDER (as opposed to looking for an existing
   definition of DOWNUNDER on disk)?  (Y or N) N
   ```

   You answer no because your site is aleady defined.

   ```
   What host is the namespace server for DOWNUNDER (default: local): Koala
   Chaosnet address for KOALA: 401
   Host responds as KOALA, ok? (Y or N) Y
   The local host is now KANGAROO
   Command:
   ```

3. Rename the auxiliary file to whatever name you want for the copied world. For example:

   ```
   Rename File FEP0:>aux.page.1  FEP0:>configured-genera-7-0-.load.1
   ```

   The file which contains extra paging space may also have a different name from aux.page, probably a name which denotes that it is additional paging space.

4. Use the Copy World command on the new machine to transfer a configured world from another machine at the site. For example:

   ```
   Copy World Kangaroo|FEP0:>configured-world.load
   ```

   Here, the already configured world on host Kangaroo is copied to the renamed auxiliary file. Since you are copying the world to an existing file, you are asked how you want to proceed; the correct answer is Overwrite. Copy World asks if you want to update your boot file to load the configured world. Answer Yes. The boot file is automatically updated.

5. The machine's disk now has the configured world; rename the old distribution world load file to aux.page:

   ```
   Rename File  FEP0:>distribution-world-file-name FEP0:>aux.page
   ```

6. Halt the machine and reboot it using the new world. The boot file uses the copied, configured world and uses both the standard file page.page and the renamed world load (aux.page) as paging space.

13.1.3 Configuring Preinstalled Worlds on Machines with Small Disks

You can configure a new machine with the distributed world load and the Set Site command. Proceed as follows if the new machine has less than 167 Mbytes of disk space:

1. Boot the distribution world on the new machine manually, by typing the following sequence:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Load World distribution-world-file-name
   Set Chaos-Address this-machine's-chaos-address
   Start
   ```

   This is an example of the commands contained in a boot file. Using this sequence of commands will add page.page as paging space and leave aux.page unused.

   microcode-file-name refers to the microcode needed for a specific system and hardware configuration. For example, if you are installing Genera 7.0 on a 3640, then the appropriate microcode file name is 3640-MIC.MIC.version-number. *Version-number* is the microcode version number, which changes from release to release. Note that if you have an IFU, you need a different if you have an IFU, you need a different microcode. For a list of Genera 7.0 microcode types See: "Genera 7.0 Microcode Types" in the Software Installation Guide.

Distribution-world-file-name refers to the name of the world distributed on the disk (for example, >release-7-0-dist.load).

This-machines's-chaos-address refers to the chaosnet address of the Symbolics computer. You must select a chaos address for each machine at your site. For more information about chaosnet address: See the section "Choosing Chaosnet Addresses", page 94.

If the boot file on disk contains the line Add Paging >aux.page.1, remove it from the command sequence. This is the file you use to receive the copied world. The file which contains extra paging space may also have a different name from aux.page, probably a name which denotes that it is additional paging space.

For information about assigning your machine a name and a chaosnet address, see *Choosing Machine Names and Chaosnet Addresses* in the *Software Installation Concepts Primer.*

2. Rename the auxiliary file to whatever name you want for the customized world. For example:

   ```
   Rename File FEP0:>aux.page.1 FEP0:>configured-genera-7-0.load.1
   ```

   The file which contains extra paging space may also have a different name from aux.page, probably a name which denotes that it is additional paging space.

3. Enter the Set Site command and follow the instructions for configuring a non-namespace host at an established site. Use this dialogue if you are installing new software on a machine at your site that is not the namespace server.

   What you type is underlined in this example.

   ```
   Command: Set Site (site name) downunder
   Define a new site named DOWNUNDER (as opposed to looking for an existing
   definition of DOWNUNDER on disk)? (Y or N) N
   ```

   You answer no because your site is already defined.

```
What host is the namespace server for DOWNUNDER (default: local): Koala
Chaosnet address for KOALA: 401
Host responds as KOALA, ok? (Y or N) Y
The local host is now KANGAROO
Command:
```

4. Use the Save World command to save the configured world. For example:

   ```
   Save World FEP0:>configured-genera-7-0-.load
   ```

   Save World asks if you want to update your boot file to load the configured world. Answer yes. The boot file is automatically updated.

5. The machine's disk now has the configured world; rename the distribution world load file to aux.page:

   ```
   Rename File FEP0:>distribution-world-file-name FEP0:>aux.page
   ```

6. Halt and reboot the machine, using the new boot file. The boot file uses the copied, configured world and uses both the standard file (page.page) and the renamed world (aux.page) as paging space.

13.1.4 Configuring Preinstalled Worlds on Machines with Large Disks

You can configure a new machine with the distributed world load and the Set Site command. Proceed as follows if the new machine has at least 167 Mbytes of disk space:

1. Boot the distribution world on the new machine manually, by typing the following sequence:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Load World distribution-world-file-name
   Set Chaos-Address this-machine's-chaos-address
   Start
   ```

   This is an example of the commands contained in a boot file. Using this sequence of commands adds page.page as paging space and leaves aux.page unused.

   microcode-file-name refers to the microcode needed for a specific system and hardware configuration. For example, if you are installing Genera 7.0 on a 3640, then the appropriate microcode file name is 3640-MIC.MIC.version-number. *Version-number* is the released microcode version, which changes from release to release. Note that if you have an IFU, you need a different microcode. For a list of Genera 7.0 microcode types See: "Genera 7.0 Microcode Types" in the Software Installation Guide.

Distribution-world-file-name refers to the name of the world distributed on the disk (for example, >release-7-0-dist.load).

This-machines's-chaos-address refers to the chaosnet address of the Symbolics computer. You must select a chaos address for each machine at your site. For more information about chaosnet address: See the section "Choosing Chaosnet Addresses", page 94.

For information about assigning your machine a name and a chaosnet address, see *Choosing Machine Names and Chaosnet Addresses* in the *Software Installation Concepts Primer.*

2. Enter the Set Site command and follow the instructions for configuring a non-namespace host at an established site. Use this dialogue if you are installing new software on a machine at your site that is not the namespace server.

   What you type is underlined in this example.

   ```
   Command: Set Site (site name) downunder
   Define a new site named DOWNUNDER (as opposed to looking for an existing
   definition of DOWNUNDER on disk)? (Y or N) N
   ```

   You answer no because your site is already defined.

   ```
   What host is the namespace server for DOWNUNDER (default: local): Koala
   Chaosnet address for KOALA: 401
   Host responds as KOALA, ok? (Y or N) Y
   The local host is now KANGAROO
   Command:
   ```

3. Make any other customizations you want for this world and then use the Save World command to save the configured world. For example:

   ```
   Save World FEP0:>configured-genera-7-0.load
   ```

   Save World asks if you want to update your boot file to load the configured world. Answer yes. The boot file is automatically updated. Back up the world if you have not already done so. For information about backing up your world: See the section "Performing Dumps", page 14.

4. Use the Copy World command on the new machine to transfer a configured world from another machine at the site. For example:

   ```
   Copy World Kangaroo|FEP0:>configured-genera-7-0.load
   ```

   Here, the already configured world on host Kangaroo is copied to the machine on which this command is entered. Copy World asks if you want to update your boot file to load the configured world. Answer Yes. The boot file is automatically updated.

5. Halt and reboot the machine, using the new boot file. The boot file uses the copied, configured world and uses both the standard file (page.page) and the renamed world (aux.page) as paging space.

13.2 Specifying a Time Zone for Your Site

All sites are in some time zone. The time zone translates user-specified time into universal time. Users specify site's time zone by modifying the time zone attribute of their site object by answering a question about it during the Set Site dialogue and/or by modifying the site's namespace object.

When you boot a distribution world, if the machine is unable to get the time from the network and does not trust its calendar clock (typically because the FEP has been reset, or the board has been replaced, or the machine is new) you are first prompted for a local time zone and then for the local time.

It is possible to set your time zone to be anything valid by specifying time zone mnemonic symbolics. For example, instead of limiting choices to Eastern Standard Time (EST), Central Standard Time (CST), and Pacific Standard Time (PST), you can use other commonly accepted mnemonic symbols for zones for all over the world. You can also specify a time zone as an offset from Greenwich Mean Time as well as by mnemonic name. You specify time zones west of GMT by their mnemonic if one is defined, or by a four digit number preceded by a - for west of GMT and a + for east of GMT. The numbers must be in the range -1200 to 1200 and can end either in 00 or 30. For example, -0500 means five hours west of Greenwich and is equivalent to EST. It is also possible to allow time zones on the half hour, like South Australian Standard time, which is 9.5 hours east of GMT. You specify this by either SAST or +0930.

If you want to use a function that returns a string requesting your time zone: See the function **time:timezone-string** in *Programming the User Interface, Volume B.*

13.2.1 Daylight Savings Time

Every area of the world has its own rules which control when daylight savings is in effect. For this reason, the system can only automatically switch to daylight savings time for time zones in the United States. For European countries or any other time zones, the system is unable to make this switch automatically. In this case, when a new time is in effect, you must must change the site's time zone by using the namespace editor to edit your site object.

To edit your site object, use the Edit Namespace Object command, with a namespace class of **site**. This invokes the namespace editor. The attribute **timezone*** is the first item in the menu. Clicking right on this attribute allows you to edit the time zone, since it repeats the name of the time zone in the operation section of the menu for editing. Clicking left on **timezone*** allows you to replace the time zone with a new one of your choice.

Note: When you initially set up your site you are queried for a time zone during the Set Site dialogue. If your site is in the United States, use the standard version of your local time zone, regardless of what time of year. For example, if your site is in the Eastern Standard Time zone, specify EST as your time zone regardless of whether or not daylight savings is in effect. If you enter your time zone in the namespace object, enter the standard time zone as well, regardless of the time of year.

Here is a complete list of the time zones that adjust automatically during daylight savings:

Time Zone	*Abbreviation*
Atlantic Standard Time	AST
Central Standard Time	CST
Eastern Standard Time	EST
Mountain Standard Time	MST
Pacific Standard Time	PST
Yukon Standard Time	YST
Alaska-Hawaii Standard Time	AHST

Time Zone	*Abbreviation for Daylight Savings*
Atlantic Daylight Time	ADT
Eastern Daylight Time	EDT
Central Daylight Time	CDT
Mountain Daylight Time	MDT
Pacific Daylight Time	PDT
Yukon Daylight Time	YDT
Alaska-Hawaii Daylight Time	AHDT

13.3 Choosing Chaosnet Addresses

A Chaos address is a 16-bit quantity in which the high-order 8 bits represent the subnet number, and the low-order 8 bits represent the host number on that subnet. Chaos addresses are expressed in octal.

```
  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

|<-------Subnet number--------->|<---------Host number--------->|
```

The subnet number is 1.
The host number is 1.
The Chaos address is 401 octal.

All machines at a site should have Chaosnet addresses with the same subnet number.

13.4 Boot File Contents

Here is a typical boot file sequence:

```
FEP Command:  Clear Machine
FEP Command:  Load Microcode microcode-file-name
FEP Command:  Load World distribution-world-file-name
FEP Command:  Set Chaos-Address this-machine's-chaos-address
FEP Command:  Start
```

Using this sequence of commands adds page.page as paging space and leaves aux.page unused.

microcode-file-name refers to the microcode needed for a specific system and hardware configuration. For example, if you are installing Genera 7.0 on a 3640, then the appropriate microcode file name is 3640-MIC.MIC.*version-number*. *Version-number* is the microcode version number, which changes from release to release. Note that if you have an IFU, you need a different microcode. For a list of Genera 7.0 microcode types: See: "Genera 7.0 Microcode Types" in the *Software Installation Guide.*

Distribution-world-file-name refers to the name of the world distributed on the disk (for example, >release-7-0-dist.load).

This-machines's-chaos-address refers to the chaosnet address of the Symbolics computer. You must select a chaos address for each machine at your site. For more information about chaosnet address: See the section "Choosing Chaosnet Addresses", page 94.

For information about assigning your machine a name and a chaosnet address, see

"Choosing Machine Names and Chaosnet Addresses" in the *Software Installation Concepts Primer.*

13.5 Editing Your Namespace Object After Installing a New Release

After you bring up a new release, you can perform the following steps as part of your site administration activities:

- Register users, hosts, printers, and networks in the site's namespace.
- Install printers.
- Move the new release to other Symbolics Lisp Machines at the site.
- Install new releases distributed in world load format.

You should reflect any changes, such as new users or changes in the site's hardware, in the namespace database. Register new hosts and printers in the namespace database before connecting them to the network or supporting host. Register new users in the namespace database either before they use the new release or as part of the process when they log in for the first time. You can register new users or new hardware most easily by copying and modifying an existing object of the same type, using the namespace editor (invoked by the Edit Namespace Object command) and then saving the new object. Once it has been saved, it is part of your site's configuration, and all machines running the new release know about the new object the next time they boot, or sooner in some cases. (See the section "Namespace System" in *Networks.*)

To use the namespace editor to create and update objects, click on [Namespace] on the System menu or select a Lisp Listener and type the Edit Namespace Object command.

For information about how to register users and hardware see the section "Maintaining the Namespace Database".

14. Installing Symbolics Dialnet

14.1 Introduction to Dialnet

Symbolics Dialnet is the component of the generic network system that supports the international dial network. The function of Dialnet is to provide a reliable transport medium over possibly unreliable common carrier facilities. The primary uses of this transport medium are mail transfer and remote login. Mail transfer is handled by the Symbolics mail reading and sending program (Zmail) and by the Symbolics store-and-forward mailer. Remote login is handled by the Terminal program.

14.2 Dialnet and Internet Domain Names

Internet Domain Names are part of a tree-structured naming scheme used by the Internet community for distributed administration of a very large namespace. Dialnet also uses the Internet Domain Names scheme of naming and addressing.

This section discusses how Dialnet uses the Internet Domain Names capability. For a complete discussion of Internet Domain Names and installation instructions: see the section "Internet Domain Names Capability" in *Genera 7.0 Release Notes.*

Symbolics networks are represented under the second-level domain name Symbolics.COM, and the Symbolics dial namespace is a third-level domain named DialNet.Symbolics.COM. The namespaces of individual Symbolics customer sites are usually represented as fourth-level domains, subdomains of the Symbolics Dial Network. For example, the dial namespace host named CSNY-Young would be in the CSNY.DialNet.Symbolics.COM domain.

The dialnet registries have a mechanism for associating Internet Domain Names with individual hosts, using the **:domain** keyword. This specifies that a given host serves as a mail gateway for a given Internet Domain. See the section "Contents of a Dialnet Registry", page 105.

14.2.1 Internet Domain Names Namespace Attribute

During installation, the customer specifies the Internet Domain Name to be associated with the namespace in which local hosts are registered, by editing the **internet domain name** attribute of the namespace object that represents the local namespace itself. All hosts that are named within that namespace then inherit the Internet Domain Name that is entered in the namespace object.

For example, the SCRC namespace object might have this attribute:

Editing SCRC Namespace Object

```
Internet Domain Name:  SCRC.Symbolics.COM
```

SCRC|JUNCO is a host named Junco in the SCRC namespace. Junco inherits the Internet Domain Name of its namespace, so its Internet Domain name is:

```
Junco.SCRC.Symbolics.COM
```

In some cases a host in that namespace is not in the same Internet domain. An individual host can override the Internet domain of its namespace by entering a value in the **internet domain name** attribute of its host object.

Editing SCRC|GRACKLE Host Object

```
Internet Domain Name:  Grackle.MIT.EDU
```

In summary, the **internet domain name** attribute of the host object is used solely to override the attribute of the namespace object.

14.3 Installing Dialnet

Here is an overview of the procedures for installing Dialnet.

- Install the Domain Name Server software.

- Choose a machine to be your Dialnet host.

- Install the hardware that makes Dialnet possible.

- Update the local namespace so that the host knows about the Dialnet hardware.

- Create the private Dialnet registry.

14.3.1 Installing the Domain Name Server for Dialnet

This installation step has two parts: loading the domain name server software, and updating the namespace database.

Loading the Domain Name Server Software

The instructions for loading the domain server software are included elsewhere in the documentation: See the section "Internet Domain Names Capability" in *Genera 7.0 Release Notes.*

Updating the Namespace

Dialnet hosts need to have an Internet Domain name associated with them. See the section "Internet Domain Names Namespace Attribute", page 97.

Most Dialnet sites belong in subdomains of the DialNet.Symbolics.COM domain. Use the namespace editor to edit the local namespace object. Enter a value for the **internet domain name** attribute. For example, a site named Acme would have this entry in its namespace object:

```
    Internet Domain Name:  Acme.Dialnet.Symbolics.COM
```

Some sites are already using the Internet domain names and have such names assigned to their hosts. Those sites should continue to use the Internet domain names that they are currently using, and enter the domain name of the namespace to the **internet domain name** attribute of the namespace object.

Any host that needs to override the default **internet domain name** stored in the namespace object can enter a different value for the **internet domain name** attribute of their host object. The **internet domain name** attribute of a host object is used solely to override the attribute of the namespace object.

14.3.2 Choosing a Machine to Be Your Dialnet Host

The machine you choose to be your Dialnet host should be one that is on the dial network most of the time. This is important since this machine receives mail for later distribution to other Dialnet hosts and to other local hosts. If the host is only occasionally connected to the dial network, it cannot do a reliable or speedy job of delivering mail to Dialnet hosts. The Dialnet host must be a Symbolics computer, since the Dialnet software currently only runs on this machine.

We recommend that your Dialnet host be the same machine as your namespace/file/mail server, since that host is supposed to be up most of the time as well. Also, you should choose a machine that has a local file system (LMFS), since sending and receiving mail over Dialnet requires use of the Mailer on your Dialnet host and the Mailer requires a local file system.

14.3.3 Installing the Dialnet Hardware

The first step in connecting your Symbolics computer to the dial network is to configure a Symbolics machine at your site with a modem. Modems may be ordered from Symbolics.

The mechanics of the physical connection of your host to the dial network depend on the model of the processor.

- If you have a Symbolics 3600 processor, follow these instructions:

  Many Symbolics 3600 processors contain a Vadic 3450 modem mounted inside the processor cabinet. Bring both a modular jack and a male EIA connector out to the I/O bulkhead. The modular jack (labeled **MODEM TELCO**) accepts a standard modular plug from the data circuit provided by the telephone company. Connect the male EIA connector (labeled EIA 4) (via a short cable, which you can make yourself or order from Symbolics, see below) to any one of the three female EIA connectors that provide access to the serial lines (labeled EIA 1, EIA 2, and EIA 3). EIA 1 corresponds to serial unit 1, EIA 2 to serial unit 2, and EIA 3 to serial unit 3.

  The cable between EIA 4 and EIA 1, EIA 2, or EIA 3 should convey the following signals on the pins given below:

  - Pin 2 (TXD ; Transmitted Data)
  - Pin 3 (RXD ; Received Data)
  - Pin 4 (RTS ; Request To Send)
  - Pin 5 (CTS ; Clear To Send)
  - Pin 6 (DSR ; Data Set Ready)
  - Pin 7 (SG ; Signal Ground)
  - Pin 8 (CXR ; Carrier Detect)
  - Pin 20 (DTR ; Data Terminal Ready)

  Terminate this cable with one male connector and one female connector. If you prefer not to build such a cable yourself, you can order it from Symbolics.

  If your 3600 has been upgraded to support audio and phase-encoded video [via UPGR-SY70], then the gender of the serial ports is male. Construct the cable as described above except that both connectors on the cable should be female. Again, if you prefer not to build such a cable yourself, you can order it from Symbolics.

  Earlier versions of the FEP proms installed in 3600 processors do not support all the features of the 3600 serial lines. Dialnet requires FEP version 127 or higher. The easiest way to determine what version of FEP proms is installed is to use the Show Herald command with the keyword :detailed.

```
Show Herald :Detailed Yes
```

One line of the resulting display shows the Fep version.

- If your Symbolics computer is any machine model, aside from a 3600, or does not contain a Vadic 3450 modem, follow these instructions:

  Symbolics computers, other than the 3600 model, have no internal modem, but instead expect an external Vadic 3451 or CDS-224 modem to be connected to one of the three serial ports.

  Each modem has two electrical connections (excluding the power cord). One of the electrical connectors is similar to the connector the telephone company puts on telephones. Plug this into the telephone jack. The other electrical connector has a 25-pin connector which is standard for RS-232 serial lines. Plug this into one of the serial line ports on the back of your Symbolics computer. For example, if you have a Vadic 3451 modem, which has a pre-installed phone line, you simply connect this to the telephone jack. If you have a CDS-224 modem, you first connect the modular jack to the **line in** on the modem and then to the telephone jack.

  Bring the serial lines out to the I/O bulkhead and terminate them in male EIA connectors. On the 3640 and 3645 processors, these connectors are labeled SERIAL 1, SERIAL 2, and SERIAL 3. On the 3670 and 3675 processors, these connectors are labeled EIA 1, EIA 2, and EIA 3. On the 3610, 3620, and 3650 processors, these connectors are labeled SERIAL 1 and SERIAL 2.

  The cable connecting the modem to the serial port conveys all the signals described above for 3600 cabling; only the gender of the serial port connector differs. If you prefer not to build this cable yourself, you can order it from Symbolics.

14.3.4 Updating the Local Namespace to Know About the Hardware

You must record the physical connection of your machine to the dial network in the namespace database so that the generic network system can decide how best to establish connections over the dial network. You can represent this connection by adding a **peripheral** attribute to the *local view* of the Dialnet host object.

To edit the namespace database use the Edit Namespace Object command, choosing to edit the host object.

The term *local view* is used here in contrast to the dial namespace view of the object. If you are editing a host that has *already* been identified as being in both the local and the dial namespace view namespaces, you are asked to choose (via a small pop-up menu) between the local and the dial namespace view of the object before you actually begin to edit the object. Choose the local view; there are no servers for the dial namespace, so you would have no way to save your changes.

When editing the host object, use the **peripheral** attribute to represent the modem that connects this host to the dial network. There are five relevant indicators for the peripheral attribute: **unit, model, baud, phone-number**, and **autoanswer.**

unit Corresponds to the serial port number on the I/O bulkhead of the machine. The value should be a number between 0 and 3, inclusive. The serial port for the console is numbered 0.

model Corresponds to the type of modem attached to this serial port. The value should be one of the following: va212, va3450, va3451, or cds-224.

baud Should be 1200.

phone-number Corresponds to the telephone number of the telephone trunk to which this modem is attached. The phone-number associates this modem with a particular dial network address. (The address, in turn, associates this peripheral with a host in the dial namespace. This latter mapping is done via the Dialnet registry.)

autoanswer Corresponds to the ability of this modem to receive incoming calls from other sites. If you wish to receive calls from other sites, then this indicator should have a value of **yes.** If it has a value other than **yes,** incoming calls are not answered and communication with other sites can only be initiated by the local site.

In general, if two sites wish to communicate over the dial network, at least one of the sites needs to enable **autoanswer.**

An example peripheral attribute might be:

```
peripheral modem unit 2 model va3450 baud 1200 phone-number 16175777348 autoanswer yes
```

14.3.5 Creating Dialnet Registries

The private Dialnet registry contains information describing hosts with which you intend to communicate, plus your own local Dialnet host. You declare the phone number of your local host here, as well as in the host's namespace object. This file lives in sys:site;private-dialnet-registry.lisp.

Dialnet *registries* are files that represent the shape of and possible connections on the dial network. These files really contain namespace information in a form suitable for periodic distribution by separate administrative groups. The following registries exist:

The public dialnet registry

This registry is maintained and distributed by Symbolics. It contains information on publicly accessible Symbolics hosts and domains, on Telenet PADs (the GTE Telenet equivalent of the Arpanet TIP), and on dialing conventions for the international dial network. For example, information in this registry might be the addresses of Symbolics software support groups, all the GTE Telenet PAD access numbers, and enough information about the international phone system to help you find the cheapest way to place calls to the Symbolics hosts described in the registry. This registry might also contain the domain of and address for the users' group, so that new customers could easily contact the group.

This file is sys:site;public-dialnet-registry.lisp.

The users' group dialnet registry

This registry is maintained and distributed by the Symbolics users' group, and contains whatever host and domain information the group's members see fit to enter and distribute.

For example, the users' group might distribute the addresses and domains of all members that wanted to share information via the dial network.

This file is sys:site;users-group-dialnet-registry.lisp.

The private dialnet registry

You maintain this registry at your site. It contains local dialing conventions and any private host and domain information for the site.

A private dialnet registry might contain the names, addresses, and domains of the following:

- Other sites in the organization
- Other organizations that did not want to be published by the users' group
- Common carriers (for example, Tymnet)

- Subscriber data services (for example, Dow Jones Information Services)

- Gateways to other domains (for example, .ARPA)

This file is sys:site;private-dialnet-registry.lisp.

Information moves into the users' group registry when someone at the local site contacts the group, and the group distributes a registry. Information moves into the public registry when someone at the local site contacts someone at Symbolics, and Symbolics next distributes a registry.

Because of delays in the distribution of registries, the private registry may be useful as a repository for advance copies of public or users' group information. When two registries contain differing information about the same object, any conflict is resolved as follows:

1. The public registry is assumed to be least current.

2. The users' group registry is assumed to be more current than the public registry.

3. The private registry is assumed to be more current than either the public or users' group registry.

14.3.5.1 Example of a Private Dialnet Registry

Here is a sample private Dialnet registry:

```
;;;The public dialnet registry contains the common case where the
;;;  phone is just a standard outside line.
;;;ACME is on a Centrex-system phone for which you have to dial 9
;;;  to get an outside line.
(:SUBNET "1901482yyyy>1zzzwwwwwww" :DIAL "yyyy" :cost "0")
     ;;Local extensions
(:SUBNET "1901482yyyy>1zzzwwwwwww" :DIAL "yyyy" :COST "0")
(:SUBNET "1901482yyyy>1800zzzzzzz" :DIAL "91800zzzzzzz" :COST "2")
(:SUBNET "1901482yyyy>1xxxzzzzzzz" :DIAL "9zzzzzzz" :COST "1")

;;; Acme's Dialnet host
(:HOST "ACME-GENIUS" :ADDRESS "19014823002" :LOCAL-NAME "ACME|GENIUS")

;;; Acme's Dialnet domain
(:DOMAIN "Acme.DialNet.Symbolics.COM" :host "ACME-GENIUS")
```

```
;;; Acme needs to communicate with Coyote Enterprises.
(:HOST "COYOTE-LISPM-1" :ADDRESS "19035492047")

(:DOMAIN "Coyote.Dialnet.Symbolics.COM" :host "COYOTE-LISPM-1")
```

14.3.5.2 Contents of a Dialnet Registry

A Dialnet registry contains lists of alternating keywords and values. The first keyword in a form (the car of the list) indicates the type of information being conveyed and is one of the following:

:subnet Specifies the cost of and dialing information for a subnet of the dial network. The keywords and values for this form are described elsewhere. See the section "Dial Network Addressing" in *Networks.*

```
(:subnet "1xxxyyyyyyy>1800zzzzzzz" :dial "1800zzzzzzz" :cost "1")
```

:host Specifies a host on the dial network and its address. The name specified by the **:host** keyword is the name of the host in the dial namespace. To avoid duplicate names in this namespace, we use the site name as a prefix. As an example, consider a site named Trilogy, one of whose hosts with a dial network address is named Cerberus. The unique dial namespace name of this host is then trilogy-cerberus.

```
(:host "trilogy-cerberus" :address "14151515151")
```

The address for the host is a string representing the address of the host on the international dial network. In our examples, drawn typically from hosts in the United States, the country code is 1, the area code is a three-digit number whose second digit is always 0 or 1, and the rest of the address is the familiar local seven-digit number. The address is the concatenation of these three fields.

If a host is present in a local namespace other than the dial namespace, its local name should be noted with the :local-name keyword. For example, the following specifies a host whose name (in the dial namespace) is USMC-Gomer, whose address on the dial network (that is, the network named "dial" in the dial namespace) is 14155551212, and whose name in the local namespace (here, the namespace named USMC) is Gomer.

```
(:host "USMC-Gomer" :address "14155551212"
 :local-name "USMC|Gomer")
```

Each host in the dialnet registry is assumed to support the following mail-related services. When the dialnet registry is loaded, host objects are automatically created or updated to contain the appropriate service attributes for these services.

- STORE-AND-FORWARD-MAIL (over the DIAL medium, using the SMTP protocol)

- MAIL-TO-USER (over the DIAL medium, using the SMTP protocol)

- MAIL-PROBE (over the DIAL medium, using the MAIL-PROBE protocol)

:domain

Specifies an Internet mail domain and the name of the associated gateway host. Internet mail domains are used by Zmail and the store-and-forward mailer to direct mail around the dial network, in the absence of other namespace information. For example, if the following forms were present in the users' group registry

```
(:host "CSNY-Young" :address "12121234567")
(:domain "CSNY.DialNet.Symbolics.COM"
 :host "CSNY-Young")
```

Then, mail to the following addresses would be routed via DialNet to Young.CSNY.DialNet.Symbolics.COM for further distribution:

```
Neil@Young.CSNY.DialNet.Symbolics.COM
Graham@Nash.CSNY.DialNet.Symbolics.COM
```

For a complete discussion of Internet Domain Names and installation instructions: see the section "Internet Domain Names Capability" in *Genera 7.0 Release Notes.*

:telenet-pad

Specifies the name and address of a GTE Telenet PAD. These gateways to the Telenet network can be an economical way of routing traffic across the dial network and are also useful in their own right as access to higher level GTE Telenet services such as TeleMail. The Symbolics Dialnet implementation uses Telenet automatically when it determines that by doing so it can make a cheaper connection.

```
(:telenet-pad "boston-telenet-pad"
 :address "16172920662")
```

14.3.5.3 Loading a Dialnet Registry

All the information in the three Dialnet registries (public, users' group, and private, if present) can be loaded into the local host with the **dial:load-dialnet-registry** function. You might use this function to load the names and addresses of all Telenet PADs, for example.

```
(dial:load-dialnet-registry)
```

Note that some system programs also load the Dialnet registries. The Store-and-Forward mailer does this when the mailer is started, and the Internet Domain Name Server program does this when that server is started.

14.3.6 Testing Your Dialnet Installation

After you have set up your private Dialnet registry, you should make sure Dialnet is working. To test it, you must load your Dialnet registry. Do this by using the function **(dial:load-dialnet-registry)**. Then, try to connect to a host whose dial-up number is known to you, over the dial network, using the terminal program:

Select the Terminal window by pressing SELECT T. Then, try to dial a host whose phone number you know by typing:

Connect to host: **dial|dial|13125678901**

You are told that the given host does not know how to support LOGIN, and then asked if you meant any of the possible protocols the Symbolics Computer knows how to use over a dial-up line. Answer *yes* when it asks about a protocol named TTY-LOGIN, which uses the DIAL-RAW medium. It should then dial up and allow you to log in as if you were sitting at a normal dial-up' console. You can customize your console somewhat by pressing NETWORK X and click on the appropriate terminal emulation mode.

14.4 Using the Terminal Program with the Dial Network

Once you have set up the hardware and namespace information that describes how your host is connected to the dial network, you can use that host to dial up other hosts. This is an excellent test of the hardware and software configuration, even if you don't usually dial up other hosts. And, of course, it can be very useful in its own right, providing access to hosts accessible only via dial-up lines.

Press SELECT T to get to the Terminal program, then type a host name at the Connect to Host: prompt. Host names are of the form dial|dial|16175777348. To break that down a bit, that's the host at address 16175777348 on the network named dial|dial, which in turn is the network named dial in the dial namespace. If you need to make the same call frequently, you can add hosts to your own local namespace (not the dial namespace) with addresses on the dial|dial network. In addition to an address attribute, you will probably want to give such a host a service attribute of:

```
    Service:  LOGIN DIAL-RAW TTY-LOGIN
```

If this host is in some other domain (this is likely if it is at some other site), you may want to give that host its own Internet Domain Name which corresponds to the Internet Domain Name established by that host's administrator. Use the namespace editor to add a value for the **internet domain name** attribute of the host object for that host.

Telenet PADs have names in the dial namespace such as dial|boston-telenet-pad. Telenet PADs are listed in the public dialnet registry, so to dial up a PAD you first have to load the dialnet registry. See the section "Loading a Dialnet Registry", page 107.

14.5 A Sample Dialnet Installation

This section details a sample installation at site NYC. The goal of this installation is to bring up this site on the dial network, both to exchange electronic mail with other sites and to use the Terminal program to access other hosts over dialup lines. There are five machines at site NYC, named Bronx, Queens, Manhattan, Brooklyn, and Staten-Island. Bronx (a 3600) was the first machine installed at the site, so it bears the title of "site support system" and has an inboard Vadic modem. In this example it also has an attached Kanji tablet for Japanese language work, and supports an LGP1 printer. When Manhattan (a 3670 with a large disk) later arrived, it was made the SYS host and the namespace server, and Bronx was given over to being a file server for user files, in addition to its continued use as a Japanese workstation and spooler for the LGP1 printer.

1. **Install the Domain Name Server.** This installation step has two parts: updating the namespace database to include the **internet domain names** attribute of the namespace object for the local namespace, and loading the domain server software. See the section "Installing the Domain Name Server for Dialnet", page 98.

2. **Find the modem.** In this example, Bronx has an internally mounted Vadic VA3450 modem by virtue of being a 3600 machine model.

3. **Connect the modem to a serial port.** The serial port, of course, should be otherwise free. In this example, serial port 1 is already in use (for example, it could be supporting a tablet or a printer), so you will be attaching the modem to serial unit 2. A short cable should be used to connect EIA 2 and EIA 3, where EIA 2 is the connector for serial port 2 and EIA 3 is the

connector for the inboard modem. For details on how to make this cable (if such a cable wasn't shipped to you by Symbolics) or on how cabling should be done for all machine models except the 3600: See the section "Installing the Dialnet Hardware", page 99.

4. **Connect the modem to the phone company's data circuit.** This requires a telephone line with modular connectors on both ends, of sufficient length to reach from the telephone jack to the **modem telco** jack on the 3600 I/O bulkhead. (Models other than the 3600 use the same type of cable but the connection is from the telephone jack to the **line** jack in the CDS modem). In this example, the data circuit is on a private branch exchange (PBX) that supports direct inward dialing, allows other extensions to be called by dialing their 4-digit trunk numbers, but requires that a 9 be dialed first when making calls outside the PBX extensions. The sample phone number for Bronx is area code 212, phone number 765-4321.

5. **Register the modem in the namespace database.** Use the namespace editor to add a **peripheral** attribute to the local view of the host to which the modem is attached. In this example, you are attaching the modem to serial unit 2, the modem is a Vadic VA3450, the phone number is 1-212-765-4321 (1 (US) country code, 212 area code, 765 exchange, 4321 trunk). Furthermore, this example supposes you want other sites to dial your site and assume some of the communications costs themselves, so you want to enable the autoanswer feature. Thus the peripheral attribute you add and save looks like:

   ```
   peripheral modem unit 3 model va3450 phone-number 12127654321 autoanswer yes
   ```

   The **peripheral** and **modem** indicators must be first and second. Past the **modem** indicator, the order of indicator/value pairs is irrelevant.

   For further information on registering the modem in the namespace database: See the section "Creating Dialnet Registries", page 102.

6. **Create a private dialnet registry.** You need to enter into the private dialnet registry information concerning:

   - Local dialing conventions

   - Hosts connected to the dial network and their addresses

   - Locally supported Internet mail domains

   You do this by creating a private dialnet registry. [You should edit the file sys:site;private-dialnet-registry.lisp and enter the following information. To create the mode line, use this sequence of commands:]

```
Lisp Mode (m-X)
```

`Set Package (m-X)` *enter Dial to the prompt*

```
Set Lisp Syntax (m-X)

Update Attribute List (m-X)
```

After you use these commands you are asked the following question. Answer Y.

```
Set attribute for the file and attribute list, too?
```

Or, you can manually type the attribute line as it appears in this example and then use command Reparse Attribute List (m-X).

```
;;; -*- Mode: LISP; Package: DIAL; Base: 10 -*-

;;; one PBX number to another: just dial the extension.
;;; local external phone number: dial 9, then the number.
;;; different area code: dial 9, then the number.
;;; WATS number: dial 9, then the number.
;;;      Note that WATS is cheaper than long-distance.
(:subnet "1212765ssss>1212765dddd" :dial "dddd" :cost "0")
(:subnet "1212765ssss>1212xxxdddd" :dial "9xxxdddd" :cost "1")
(:subnet "1212765ssss>1aaaxxxdddd" :dial "9aaaxxxdddd" :cost "2")
(:subnet "1212765ssss>1800xxxdddd" :dial "91800xxxdddd" :cost "1")

;;; modem is on DIAL|NYC-Bronx, phone (212) 765-4321,
;;;  and the local name of the host is Bronx.
(:host "NYC-Bronx" :address "12127654321" :local-name "NYC|Bronx")

;;; DIAL|NYC-Bronx will run a store-and-forward mailer,
;;; servicing the domain named NYC.DialNet.Symbolics.COM.
(:domain "NYC.DialNet.Symbolics.COM" :host "NYC-Bronx")
```

For related information:

See the section "Dialnet".
See the section "Creating Dialnet Registries", page 102.
See the section "Dialnet and Internet Domain Names", page 97.
See the section "Overview of the Mailer" in *Communicating with Other Users.*

7. **Load the dialnet registry.** Type the following form:

```
(dial:load-dialnet-registry)
```

8. **Test the dial network connection with the Terminal program.** This step is optional, of course, since you might not have a dial-accessible host handy. This example assumes a timesharing host named NYC-Tammany, with a single dialup line at (212) 666-1040. You select the terminal program with `SELECT T` and enter the address of NYC-Tammany, in this case dial|dial|12126661040. The number is dialed, the connection is made, the screen clears and any subsequent communication takes place with NYC-Tammany. After the session with the foreign host is complete, the connection can be broken by pressing `NETWORK L`. For more information on using the Terminal program to access hosts via dialup lines: See the section "Using the Terminal Program with the Dial Network", page 107.

15. Installing and Configuring the Mailer

When you want to offer store-and-forward mail service on a Symbolics host (or on several hosts), you must update the host object(s) corresponding to the host(s) that will offer mail service.

For an overview on changing objects in the namespace database: See the section "Maintaining the Namespace Database", page 81.

1. Use the Edit Namespace Object command, with a namespace class of Host, and answer the prompt with the name of the host that is to provide mail service. In this example, for a host named Red, what you type is underlined:

   ```
   Edit Namespace Object <space> (a namespace class or Any [default Any]) <space>
   Host (the name of host) Red
   ```

2. If this host is going to send and receive mail over telephone lines, and if you have loaded a Dialnet registry that included this host, the namespace editor asks whether you want to edit the DIAL or the local namespace view of this host. Choose the local view. For more information on Dialnet: See the section "Symbolics Dialnet" in *Networks.*

3. Add the following service triples to the host object definition:

   ```
   MAIL-TO-USER CHAOS CHAOS-MAIL
   STORE-AND-FORWARD-MAIL CHAOS CHAOS-MAIL
   ```

4. If the host has an address on the DIAL network, also add the following triples; otherwise, skip this step:

   ```
   MAIL-TO-USER DIAL SMTP
   STORE-AND-FORWARD-MAIL DIAL SMTP
   ```

5. Click on [Save] to save your entries.

6. Using the Namespace Editor, add the following property pair to the site object definition:

   ```
   All-MAIL-ADDRESSES-FORWARD YES
   ```

7. After you have added the services in the previous steps, click on [Save] to save the changes.

8. Wait for confirmation that the changes have been saved, and then click on [Quit] to exit from the namespace editor.

The mail services have now been registered in your namespace database, for the host you specified.

The next procedure installs the Mailer software onto the machine that is going to be the mail server, which is the same host as specified above.

This installation is actually a further customization of the new release world for your site; that is, the particular world that is run on the host providing mail service is different from the worlds running on the other machines at the site. The difference is that the host which provides the mail service should also be a namespace server and file server. You should backup the files on this machine, since its file system is being used.

1. You should free up enough disk space in the FEP file system of the server machine to accommodate the following material:

   - There should be enough space for a world about 10% larger than the world already there. (The world with the Mailer loaded is saved back onto the disk at the end of this procedure.) You may also want to load the mailer into a world and save it with Incremental Disk Save (IDS). For information about IDS worlds: See the section "Using Incremental Disk Save (IDS)", page 64. If mail service is to be provided by a Symbolics 3640: See the section "Instructions for Managing Disk Space on the 3640 with a 140 Megabyte Disk", page 71.

   - The Mailer sources and binaries occupy about 200 LMFS blocks.

   - The Mailer databases each require a minimum of 20 LMFS blocks. The size of a database increases approximately linearly with the number of messages stored by the mail server (the host); an "average" piece of mail occupies about 2 LMFS blocks.

   You can use the Dired (m-X) Zmacs command, the Delete File command, or SELECT F (the File System Editor) to delete unused or excess files in the same way that you freed up enough space for the customized new release world.

2. Using the Create Directory command, create the following directories on the mailer host:

   ```
   >Mail>
   >Mail>Static>
   >Mail>Archive>
   ```

3. Create the two local customizations files, Mailboxes.Text and Options.Text, which are both in the directory >Mail>Static> on the mailer host.

a. The Mailboxes.text file contains information which allows the Mailer to know where to deliver mail. This file defines all the local recipients of mail, including individuals and mailing lists.

   Using the editor, edit the file >Mail>Static>Mailboxes.text. An example follows:

```
; -*- Mode: Lisp; Package: Mailer; Lowercase: Yes -*-

;; This file belongs on White only.

;; The mail addresses "Postmaster" and "File-Server"
;; are resolved with respect to individual hosts.  Edgar
;; gets all mail addressed to these "non-people" users,
;; will be the person who takes care of dead letters, etc.

(define-local Postmaster Edgar)
(define-local File-Server Edgar)
(define-local Mail-Server Edgar)
(define-local Lisp-Machine Edgar)

;;; People who get mail delivered on White
(deliver-local Robert Edgar Henry Malcolm Peterson Smiley Nelson)

;;; The basic mailing list for bugs.  Each person listed
;;; gets a copy of each bug report, and a copy goes to the files.
(define Bug-LispM
    (list Edgar Davies Henry
            "white:>Mail>Archive>Bug-Reports.text"))

;;; Miscellaneous mailing lists

;;; Good to have a mailing list that allows you to reach everyone
(define seventh-crisis
    (list Robert Edgar Henry Malcolm Peterson Davies Richard Smiley Nelson))

;;; Simple mailing list, commented to show particular preferences
(define sports-followers
        (list Richard       ;; golf
              Smiley        ;; gymnastics
              Nelson)       ;; tennis
```

```
;;; Another simple list.
;;; Put yourself on this list if you want to hear about tape bargains
(define tapers
        (list Richard Henry))
```

Write out the file, and stay in the editor for the next step.

Here are some notes about the mailboxes.text file above:

- It is necessary to appoint someone to be the *Postmaster* at your site. This is the person who receives any mail messages that cannot be delivered and cannot be returned. Also, if anyone outside your site needs to communicate with you about the operation of your mailer, it is conventional for them to send mail to Postmaster. In the example above, the

  ```
  (define Postmaster Edgar)
  ```

  tells the Mail Server to place mail addressed to Postmaster in Edgar's inbox.

- The *deliver-local* list is the list of people who receive mail on this mail server. For example,

  ```
  (deliver-local Edgar)
  ```

  means that mail addressed to Edgar at your site is put into the file Local:>Edgar>Mail.text, which is where Zmail expects to find it.

- Use *define* to make synonyms. For example, *(define Edgar Hoover)* delivers mail addressed to Edgar to whatever mailboxes "Hoover" implies. The second "argument" to define may also be a *list* form, which is how you create a mailing list.

- You probably do not want to use the *define-local* list. Names on this list do not go into the the forward files distributed to the rest of the site.

b. The Options.Lisp file contains various customizations for the operation of the Mailer. This file contains Lisp forms which load when the mailer is started or when you use the function **(mailer:update-options)**. If you make a change to this file while the Mailer is running, type **(mailer:update-options)** to a Lisp listener on the Mailer host, to update the mailer.

Using the editor, edit the file >Mail>Static>Options.lisp. Set it up to look like this:

```
;;; -*- Mode: Lisp; Package: Mailer; Base: 10; Syntax: Common-Lisp -*-

(setq mailer:deferred-delivery-times '("7:45am" "5:15pm" "11:15pm"))
(setq mailer:deferred-receipt-times  '("7:45am" "5:15pm" "11:15pm"))
(setq mailer:deferred-receipt-hosts
      '("Riverside.SCRC.Dialnet.Symbolics.COM"
        "LispM-1.Coyote.Dialnet.Symbolics.COM"))
```

After you finish editing the file, save it.

The variables **mailer:deferred-delivery-times** and **mailer:deferred-receipt-times** tell the mailer how often to try to deliver and pick up mail from the Dialnet hosts. The value of these variables may be **nil**, **t**, a specific time, or a list of times. If you specify **nil** for the value of one of these variables, it means *do not do this*; if you specify **t**, it means *do this as frequently as possible.* If you specify a time interval, such as **"three hours"**, this is the frequency with which to pick up and deliver mail, in this case, every three hours. If you specify a list of times to pick up and deliver mail, for example, *("7:45am" "5:15pm" "11:15pm")*, this specifies exact delivery and pick up times.

The variable **mailer:deferred-receipt-hosts** is a list of Mailer hosts from which the local Mailer attempts to retrieve mail. This variable is only useful on Dialnet mailers. You should probably always include "Riverside.SCRC.Dialnet.Symbolics.COM" on this list, since Riverside does not dial out for any mail delivery.

When you test the mailer, you may want to set **mailer:deferred-delivery-times** to **t** until you receive your first mail message.

For a description of the options: See the section "Files and Directories Used by the Mailer" in *Communicating with Other Users.*

4. To install the Mailer system, you should create a special world that includes the Mailer. As is true when creating any other world, it is recommended that you do as little as possible to the environment prior to world creation. In this situation, you should do all of the steps listed above, including editing the namespace and creating the Mailer files, and then cold boot the machine. Once you are in the process of building a new world, do not switch windows or do anything that causes an unnecessary allocation of storage.

First, cold boot the host. After this, disable services, log in to the sys host, and then use the Load System command to load the Mailer system. After using the Load System command, save the world with the Save World command.

The procedure for installing the mailer system is shown in the following example:

```
Command: Halt Machine RETURN
Fep Command: boot RETURN
         .
         .
         .
Command: Disable Services RETURN

Command: (si:login-to-sys-host)

Command: Load System Mailer

Files to be loaded:

.
.
.

Command: Save World (Complete or Incremental) name-of-file-to-save-it-in
```

When you save the world, you may want to use the Save World Incremental command, which allows you to have an incremental world built on the site-customized distribution world containing the Mailer. For information about this command: See the section "Using Incremental Disk Save (IDS)", page 64. You have now configured the newly saved world to be a mail server for the site.

5. Boot the new world using the FEP Boot command. When services are enabled to the mail server, the mailer starts automatically. Shortly thereafter, you can press SELECT O to bring up the Mailer Log window. Then test the mailer by sending messages to and from various machines at the site.

15.1 Testing and Registering the Mailer

The best way to test the Mailer is to attempt to send mail to various people. Send mail to Postmaster at your local site, to make sure that works. Then, you should send mail to Symbolics so we know about your site and are able to communicate with you. Send mail to "HOSS@Riverside.SCRC.Dialnet.Symbolics.COM" that contains the following information:

- The name of your site, for example, "Acme"
- The name of your mailer machine (Fearless), for example, Fearless.Acme.Dialnet.Symbolics.COM
- How you can be contacted such as your name, telephone number, and hours during which we can call you, in case we receive your mail and cannot respond.

If you are on Dialnet, then include these steps:

- The Dial network address of your mailer machine, for example 16175371234
- Whether you wish to have your telephone number published in the Users' Group or Public Dialnet registries (See the section "Creating Dialnet Registries", page 102.) Or, you can specify if you only want Symbolics to send you mail.

To follow what the Mailer is doing, press SELECT O on the Mailer host and watch the log output.

15.2 Configuring Large Sites for Multiple Mail Servers

The Store-and-Forward Mailer supports *forwarding tables* to help coordinate mail delivery at sites with several mail servers.

One particular mail server is configured to be in charge of forwarding-table maintenance. The forwarding tables themselves are written by the this host to the file systems of all the other mail servers at the site. This asymmetry is, in a sense, a further customization of the particular mail server that writes the forwarding tables. The customization is usually done by placing a **setq** of the variable **mailer:forwarding-table-hosts** in the options.text file. For example:

```
(SETQ MAILER:FORWARDING-TABLE-HOSTS
   '("MANFRED" "NATASHA" "BORIS"))
```

Here, the hosts Manfred, Natasha, and Boris receive new forwarding tables from the host to which this init file belongs. The forwarding table for a given host is written in the file >Mail>Dynamic>Forwarding.text on that host's local LMFS.

If you want to run the identical init file on all the server machines at a site, the following example may be instructive. Here, a SYS host (Fearless) runs the Mailer and is responsible for writing out the forwarding tables. The File-Server init file, which all file servers use, includes the following lines:

```
(DEFMACRO FILE-SERVER-ONLY-ON (HOSTS &BODY BODY)
  `(WHEN (OR ,@(LOOP FOR HOST IN HOSTS
     COLLECT `(SEND NET:*LOCAL-HOST* :PATHNAME-HOST-NAMEP ,(STRING
       HOST)))) ,@BODY))

(FILE-SERVER-ONLY-ON (FEARLESS)
  (SETQ MAILER:FORWARDING-TABLE-HOSTS
     '("MANFRED" "NATASHA" "BORIS")))
```

The file Mailboxes.text on Fearless contains the names of all the mailing lists for this network. In addition to the usual forms defining mailing lists, the file contains forms like the following:

```
;; What follows is a global table of mail addresses for the network.
;; There is one entry for each host, listing all of the mail addresses to be
;; forwarded to that host.  Each entry is broken into two sublists,
;; one for mailing lists and one for individuals.  This is the only
;; place in which this table should be edited.
;; The forwarding tables for all other hosts are generated from this one.

(DELIVER-TO NATASHA

  ;;; The mailing list file on Natasha is >mail>static>mailboxes.text.

        ;; Individuals
        Andy Bob Charles David Edgar

        ;; Lists
        ASAS Audio Audiophiles
        BBoard Bikers Bleeding-Hearts Bridge
        ...)
```

Similar **deliver-to** forms are supplied for Boris and all other hosts with store-and-forward-mailers.

When the Mailer on Fearless is booted, (that is when **mailer:start-mailer** is called, either automatically when you enable services, or minimally, by the user) or when the Mailer notices that the local mailboxes.text file has changed and has been stable for at least 10 minutes, it reads in its mailboxes.text file (Fearless will never have a forwarding.text file) and then writes out forwarding.text files on all the other Mailer hosts. Those hosts eventually read in the new forwarding.text files and their own mailboxes.text files, merge the two sets of definitions, and carry on.

The forwarding.text file that Fearless generates for Boris includes forms for hosts with Store-and-Forward Mailers, such as Natasha :

```
; Mailbox forwarding table for BORIS.
; Written 6/10/86 15:33:54 by Mail-Server running on FEARLESS.
; From F:>Mail>Static>Mailboxes.text created on 6/10/86 15:25:24.
; This table is automatically generated by a program.  Do not edit it.

(DELIVER-TO NATASHA

        Andy Bob Charles David Edgar
        ASAS Audio Audiophiles
        BBoard Bikers Bleeding-Hearts Bridge
```

If Boris gets incoming mail for these individuals or lists, the mail is forwarded to Natasha. There is no entry for Boris in this list, since those entries come from the mailboxes.text file on Boris.

16. The Front-End Processor

16.1 Introduction to the FEP

Symbolics computers use a front-end processor known as the FEP. The FEP is a small computer inside the processor cabinet, based on a microcomputer chip. It plays several roles in the operation of the system, the most visible being booting (loading and starting the software of) the Lisp system.

This discussion corresponds to FEP EPROM version 127 or higher. Use the ShowVersion command to determine the FEP version of your machine. See the section "FEP System Commands: General Usage", page 133. If you have FEP EPROMS of version lower than 127, contact Customer Service.

The FEP system has two parts:

- EPROMS containing the kernel of the FEP system
- Loadable overlay files (with the extension .FLOD) containing the rest of the FEP software, in the FEP file system

The FEP system provides a kernel that is independent of all changeable knowledge of the 3600 family of machines. This kernel remains constant and does not need to be modified to support new features. Instead, support for new features can be provided by overlay files that are read from disk or cartridge tape. New FEP features are distributed as part of software releases.

16.2 Using the FEP

The FEP system software implements a small number of commands in EPROM and the rest in the *overlay files*, which have an extension of .FLOD. To be used, a command must be defined in an overlay file that has been scanned. Additionally, the overlay file must be resident in memory. Only one overlay file can be resident in memory at any time.

Scanning an overlay file makes all the commands it contains known to the FEP. Scanning inserts all the commands defined in that file into the the FEP's list of valid commands. This list is the command tree; it indicates which commands are available and in which overlay file they reside. Commands remain in the command tree unless the FEP is reset or the machine is powered down. After an overlay file has been scanned, it can be loaded into memory.

When you type a command, the FEP system either dispatches directly to the command (if the command is supported in the kernel, or if the correct overlay is resident) or automatically loads the overlay file first and then dispatches to the command. In the latter case, prompting for the next command argument is delayed a short time while the overlay file is read.

Here is a list of the overlay files and what they contain:

Overlay File: *Contains:*

v127-info.flod commands that give information about the machine, for example, the Show Configuration command.

v127-loaders.flod commands to load the machine, for example, the Load Microcode and Load World commands.

v127-lisp.flod commands for manipulating Lisp, for example, the Start, Continue, and Show Status, Set LMFS FSPT Unit, and Show LMFS FSPT Unit commands. The last two commands allow you to specify a disk unit for the location of the LMFS's FSPT and to show you this location. This overlay file also contains FEP-based support for the UNIBUS option.

v127-debug.flod the FEP debugger, which is invoked by the Debug command.

v127-tests.flod the Test commands.

v127-disk.flod the Disk Restore and Disk Format commands.

The last two overlay files listed do not belong in the hello.boot file since they are only used during software installation or testing. Use the Scan command on the -tests and -disk overlays when necessary, by typing the following to the FEP prompt before issuing the contained commands:

```
Scan v127-disk.flod
Scan v127-tests.flod
```

16.2.1 FEP System Hello.Boot File

In the Lisp world, you create a hello.boot file; this file is used to perform a bootstrap operation on the FEP. Each time the machine is reset, the commands that are included in the overlay files must again be made available. Using the FEP command, Hello, to load the hello.boot file makes these commands available.

In order to use the FEP system, you must create a hello.boot file in the editor, with a pathname of FEP*n*:>hello.boot. FEP*n* refers to the disk unit number, in the case where a computer has more than one disk. This file normally contains a sequence of Scan commands that scan the overlay files containing the standard commands and the Initialize Hardware Tables command.

In Zmacs, create the hello.boot file in the FEP directory with a pathname of fep0:>hello.boot.

Put the following sequence of Scan commands in the hello.boot file:

```
    Scan FEP0:>v127-info.flod
    Scan FEP0:>V127-loaders.flod
    Scan FEP0:>v127-lisp.flod
    Scan FEP0:>v127-debug.flod
    Initialize Hardware Tables
```

Make sure you press RETURN after `Initialize Hardware Tables`, and then save the file. For an explanation of the Scan commands: See the section "Using the FEP", page 123.

16.2.2 Lisp Utility for the FEP System

The following function writes the distributed overlay files to a cartridge tape in a format acceptable to the FEP's Scan command. Use this function from Lisp:

tape:write-fep-overlay-flods-to-cart *overlay-prefix* &optional (*tape-spec* "**local:cart**") &rest *private-flods* *Function*

overlay-prefix is a string such as "v127", indicating the FEP EPROM version. *tape-spec* defaults to "LOCAL:CART"; if you supply NIL, *tape-spec* prompts for a tape specification, which must specify a cartridge tape. Each item of *&rest private-flods* is a string, such as "LOADERS"; you can use these to specify individual overlay files that you want to write to tape. The file sys:n-fep;*overlay-prefix*-private-flod.lisp is among the additional flods written to tape. The pathname of each file is displayed as it is written to cartridge tape.

For information about what each overlay file contains: See the section "Using the FEP", page 123.

To use **tape:write-fep-overlay-flods-to-cart**, type the following form at a Lisp Listener. In this example, we want to copy FEP EPROM version 127 to tape.

```
    (tape:write-fep-overlay-flods-to-cart  "V127")
```

This writes all the flod files for FEP Eprom version 127 from disk to tape.

We recommend that you use this function to make a backup tape containing the overlay files. Then, this tape will be of use to you if you ever have a disk without enough overlay files on it to boot Lisp. If you are in this situation, and have a tape of the overlay files, then load the tape into a tape drive and type the following at the FEP prompt:

```
FEP Command: Mount cart:
FEP Command: Scan Cart:
FEP Command: Scan
.
.
.
```

Repeat the last command, Scan, until you get an "End of File". Now type boot to activate the boot file and boot Lisp, or type each command from a boot file manually if you do not have a boot file. If you type each command from a boot file manually, this is the command sequence:

```
Clear Machine
Load Microcode microcode-file-name
Load World distribution-world-file-name
Set Chaos-Address this-machine's-chaos-address
Start
```

Once you have booted Lisp, copy the overlay files from sys:n-fep; onto the FEP file system. Use the Copy File command to do this. For example:

```
Copy File SYS;n-fep;v127-*.flod.newest FEPn:>
```

16.2.3 Hints on Using the FEP

The FEP command prompt is displayed when you are at FEP command processor level. It looks like this:

```
FEP Command:
```

The FEP command processor provides defaults and documentation where appropriate. When using it, remember these hints:

- You need type only enough of a FEP command to identify it uniquely, as shown below:

Input	*Completes to*
b RETURN	Boot
l w RETURN	Load World >World1.load
st RETURN	Start

- You can press the HELP key for a list of all FEP commands. For example:

```
HELP
```

Prints out:

```
Add ...more...
Attach ...more...
Boot  --  Execute an indirect command file
Clear ...more...
Compute ...more...
Continue  --  Continue running the machine after a halt
Copy ...more...
Debug  --  Enter the Fep Debugger
Declare ...more...
Detach ...more...
Disable ...more...
Dismount  --  Dismount a device
Enable  ...more...
Find ...more...
Fsm ...more...
Hello  --  Execute a "hello" file (default FEP:>Hello.boot) to initialize the FEP
Initialize ...more...
Load ...more...
Mount  --  Mount a device
Reset ...more...
Scan  --  Scan a file (module) for commands
Set ...more...
Show ...more...
Shutdown  -- Halt the machine
Start
```

Some of these commands are used in ordinary booting; others exist primarily for system maintainers, to help them debug unusual problems.

- You can press the HELP key after typing a command name, for a list of all possible completions to that command. For example:

```
set <SPACE> HELP
```

prints:

```
Set Chaos-address --  Set the chaos address
Set Default-disk-unit  --  Sets the default disk unit
Set Display-string --  Set the NanoFep's display string
Set Ethernet-address  --  Set the Ethernet address
Set Lisp ...more...
Set LMFS ...more...
Set Monitor-type  --  Sets the monitor type to Philips or Moniterm
Set Prompt  --  Sets the top level command prompt
Set Wired ...more...
```

Note that you must press SPACE after typing a command name and before pressing HELP, to receive a list of the command's arguments.

- You can insert parenthetical comments in any white space within or after FEP commands. Such comments make useful documentation for boot files. For example:

```
load world >World1.load (contains geological survey programs)
set chaos-address 401 (Koala)
```

load world >World1.load and set chaos-address 401 are FEP commands, and the parenthetical phrases are user-supplied comments.

Finally, be careful! If you make a mistake when giving FEP commands, you might leave the machine in a state from which it cannot be cold booted.

16.3 FEP System Features

The following sections describe features of the FEP system.

16.3.1 Pathname Completion Is Supported

Commands support two kinds of pathname completion, one of which you activate by pressing the COMPLETE key and the other by pressing the HELP key.

If you press the COMPLETE key the system attempts to complete the pathname supplied so far. It replaces your input with as much of a completed pathname as it can without running into a conflict between two similar pathnames in the file system. For example, if you type

```
Load Microcode (default is FEP0:>3640-mic.mic) tm COMPLETE
```

the FEP system might respond with

```
Load Microcode (default is FEP0:>3640-mic.mic) FEP0:>3640-
```

if the possibilities are:

```
FEP0:>3640-mic.mic
FEP0:>3640-fpa-mic.mic
```

If you press the HELP key, the system displays the possible completions of the pathname. For example, if you type

```
Load Microcode (default is FEP0:>3640-mic.mic) tm HELP
```

the FEP system might respond with:

```
FEP0:>3640-mic.mic.389 ...
FEP0:>3640-fpa-mic.mic.389 ...
```

Similarly, if you type

```
Load World (default is ...) Inc HELP
```

the FEP system would show all the files that begin with Inc, such as the Incremental Worlds, created by Incremental Disk Saves.

16.3.2 Pathname Merging is Supported

Pathname merging is supported.

Pathnames given to FEP commands are merged against the default in much the same way as in Lisp. Therefore, you need to specify only those fields that are different from the default. If either the name or the type is given, the default version is the newest version, not the version of the default. Pathnames are not case sensitive. Examples of pathname merging are:

Default	*Input*	*Merged*
FEP0:>3600-MIC.MIC	3600-FPA-MIC	FEP0:>3600-FPA-MIC.MIC
FEP0:>3600-MIC.MIC	fep1:	FEP1:>3600-MIC.MIC
FEP0:>3600-MIC.MIC	..389	FEP0:>3600-MIC.MIC.389
FEP0:>3600-MIC.MIC.389	3600-FPA-MIC	FEP0:>3600-FPA-MIC.MIC
FEP0:>3600-MIC.MIC	3600-mic..389	FEP0:>3600-MIC.MIC.389

16.3.3 Show Directory Command Understands Simple Wildcards

Examples of the use of simple wildcards are:

Specification	*Possible use*
`*.load`	to list all world loads
`*.boot`	to list all boot files
`v127*.flod`	to list all .FLOD files for FEP version 127
`*sys*.*`	to list all files with SYS as a substring of their name
`*.mic.389`	to list all version 389 microcode files

This does not include either directory wildcards or handling a version of 0 (meaning newest) correctly. Show Directory does not support relative pathnames.

FEP pathnames have a four-character type-field limitation, thus, you cannot use .*lo*d to find all .LOAD and .FLOD files.

16.3.4 The FEP Determines Microcode Default From the Hardware Configuration

The initial default for the Load Microcode command is determined from the hardware configuration. Here are some of the hardware configurations:

Hardware Configuration	*Default*
3600, FPA	FEP:>3600-fpa-mic.mic
3600, FPA, XSQ	FEP:>3600-fpa-xsq-mic.mic
3640, FPA	FEP:>3640-fpa-mic.mic
3640, FPA, XSQ	FEP:>3640-fpa-xsq-mic.mic

This is only the initial default; using the Load Microcode command sets the default to the appropriate pathname if the command completes successfully. You can always reset the default to the initial (computed) default with the Compute Microcode Default command.

16.3.5 Show Directory Shows Detailed Information

The Show Directory command shows detailed information about each FEP file system directory entry in much the same way as the Dired facility does in Zmacs. In addition to showing the pathname, it also shows the following information about each file:

- The number of blocks allocated

- The number of bytes and the byte size (or DIRECTORY if the file is really a directory)

- Flags (such as don't delete, deleted, don't reap)
- Creation time (in Greenwich Mean Time)
- File comment
- File author

16.4 Cold Booting

Cold booting completely resets Lisp. When you are finished using the computer, you can cold boot it to put it into a fresh state for the next user. Avoid cold booting a machine that someone else may be using, though, since the other person might be expecting the machine to remain in its current state. Here is the procedure for cold booting:

1. Go to a Lisp Listener by pressing `SELECT L`.

2. Log out, if you are logged in, by typing the Logout command.

3. Halt the machine by typing the Halt Machine command. (The function **(sys:halt)** can also be used.)

   If you cannot obtain a Lisp Listener window, or if no Lisp Listener is responding to keyboard input, press `h-c-FUNCTION`. However, the Halt Machine command is preferred over `h-c-FUNCTION` for stopping Lisp, because `h-c-FUNCTION` might interrupt disk I/O operations.

   Pressing `h-c-FUNCTION` does not immediately stop Lisp. Instead, the FEP asks Lisp to stop itself cleanly. If Lisp does stop itself, the FEP prints the message "Lisp stopped itself." If Lisp does not stop itself after about three seconds, the FEP prints, "Waiting for Lisp to stop itself..." If after another three seconds Lisp does not stop itself, the FEP forcibly stops Lisp and prints, "Halting execution of Lisp." The purpose of this behavior is to reduce the chance of halting the computer during a disk write, which might cause ECC errors.

4. When control has returned to the FEP (you will see the `FEP Command:` prompt, type the following FEP command to cold boot the machine:

   ```
   boot file-name RETURN
   ```

   where *file-name* is a boot file, with an extension of .boot. Its default value is the last file name given the Boot or Show File command. Its initial default value is Boot.boot on the current default disk unit. The following is a typical boot file:

```
Clear Machine
Load Microcode microcode-file-name
Declare Paging-files paging-file-names
Load World distribution-world-file-name
Set Chaos-Address this-machine's-chaos-address
Start
```

Alternatively, if the microcode is already loaded and the Chaosnet address is set, you can type these FEP commands manually:

```
load world distribution-world-file-name
start
```

File names specified in a command file refer by default to the disk unit containing the command file. For example, the FEP command Load World >World1.load, if contained in the file fep0:>Boot.boot, loads the world file fep0:>World1.load. If one of the files required for booting resides on a different disk unit than the command file, you must explicitly give its full pathname in the Load command. For example, if >World1.load resides on fep1, the corresponding Load command in fep0:>Boot.boot must be Load World fep1:>World1.load.

Cold booting takes approximately one minute. It takes another minute for Lisp to start. During this time, the machine might print a message asking you to enter date and time information, if it has no other way to find it. (This behavior is site dependent.) If so, type it in the following format:

```
09/21/86  15:04
```

Be sure to enter the date and time correctly, as it is important that the file system know exactly when files are created and modified. If the calendar clock has been set, the machine uses the calendar clock reading as the default time for you to type in. If the calendar clock has not been set, the machine offers to set it to the time you specify.

16.5 Resetting the FEP

Resetting the FEP restarts the FEP system, thereby discarding knowledge of the FEP's free storage area. Resetting might be necessary if you unplug the console video cable from either end or turn the console off and on. You also need to reset the FEP if you receive the error message: `No More Memory`. [You can reset the FEP from either the keyboard or the processor front panel. Note that when the FEP is being reset the fault light (located on the front panel of the processor box) is turned on by the hardware. Then, when the FEP finishes initializing itself the FEP turns the fault light off.]

- To reset the FEP from the keyboard:

  1. Type the Halt Machine command at a Lisp command prompt to stop Lisp and give control of the keyboard to the FEP.

     If no Lisp Listener is responsive, press h-c-FUNCTION to stop Lisp.

  2. Type the command Reset FEP to the FEP prompt.

  3. Press Y to answer the confirmation prompt.

  4. Type the command Hello to the FEP prompt to initialize the overlay files.

- To reset the FEP from the processor front panel:

  1. Push the red RESET button on the processor front panel.

  2. Press the spring-loaded YES switch to answer the "Reset FEP?" question (This question is asked only if you have a 3600 machine model).

After you reset the FEP, the keyboard is connected to the FEP, not to Lisp. Type the Hello command to the FEP prompt, and then give the Start command and press RETURN to warm boot the machine and Lisp, and return control of the keyboard to Lisp.

16.6 FEP Commands

Some FEP commands are involved in normal use of the computer. These include Boot, Show Directory, Show Version, and Start. Other commands are used primarily by system maintainers to debug unusual problems. Among these are Disk Restore and Disk Format. Be careful when giving these latter commands. If you make a mistake, you might destroy the state of the loaded or saved Lisp system.

16.6.1 FEP System Commands: General Usage

Add Disk-type Lets you declare an arbitrary disk type to the FEP. Use this command if you need to format and restore a disk which is not yet supported by Symbolics. You can declare up to four disk types before you have to give the Clear Disk-types command. Add Disk-Type is needed only to format and restore disks. It is not needed for normal operation of any validly formatted disk with a FEP file system.

Add Disk-type has the following arguments, for which it prompts with the argument names in parentheses:

name	The textual name by which this disk type is known
cylinders	The number of cylinders supported by the drive
heads	The number of heads on the drive
sectors	The number of sectors
gap1	The length of "gap1"
gap2	The length of "gap2"
gap3	The length of "gap3"
fast	0 for slower disks, 1 for faster disks

These numbers require careful computation and involve some restrictions of the computer hardware. The calculation should be done by Symbolics Customer Service.

Add Paging-file *file-name*

Adds the data pages of *file-name* to the list of pages Lisp uses. This command does not add the paging file to the list of declared paging files (declared by the Declare Paging-files command or the Declare More Paging-files command).

The Declare Paging-files and the Declare More Paging-files commands replace the Add Paging-file command in boot files. You can use Add Paging-file command when typing boot file commands to the FEP prompt manually, but this command is now obsolete in boot files.

file-name defaults to >.page on the currently selected disk unit.

This command is implicitly executed by the Load World command for each declared paging file (declared by the Declare Paging-files command), or for the single file FEP*n*:>page.page if there are no declared files.

The Start command notifies you if no paging file has been established. It prompts for confirmation to boot the machine before starting the processor.

Attach Graphics Tablet *<serial-port-number>*

Allows a graphics tablet to be connected to a serial port and the FEP to interpret the motion commands; this avoids the delays involved in having Lisp do the interpretation. This command supports software distributed by the Symbolics Graphics Division.

Boot *file-name* Executes the commands specified in *file-name*. *file-name* is the name of a boot file; it defaults to the last file name given the Boot or Show File command. Its initial default is >Boot.boot.

Clear *subcommand*

This command has the following subcommands:

Clear Disk-types Clears all disk types declared with the Add Disk-type command.

Clear Machine Clears the internal state of the registers and memories.

Clear Screen Clears the console's screen.

Clear Paging-files Clears the list of remaining pages that Lisp uses, and clears the FEP's list of paging files added by the Add Paging-file command. This command does not clear the list of declared paging files (declared by the Declare Paging-Files command or the Declare More Paging-files command). To clear the list of declared paging files, you must give the Declare Paging-files command without listing any files.

Compute Microcode Default

Computes the default microcode name from the hardware configuration and resets the Load Microcode command default to the computed name. Some examples:

Hardware Configuration	*Default*
`3600, FPA`	`FEP:>3600-fpa-mic.mic`
`3600, FPA, XSQ`	`FEP:>3600-fpa-xsq-mic.mic`
`3640, FPA`	`FEP:>3640-fpa-mic.mic`
`3640, FPA, XSQ`	`FEP:>3640-fpa-xsq-mic.mic`

Continue Continues the computer's operation from where it left off. However, if you have stopped the world and loaded new microcode, Continue does not work. Instead, you must warm boot by using the Start command.

Copy File from *pathname* to *pathname*
Copies a file from disk to tape on a machine at the same site.

Declare Paging-files *file-names...*

Declares *file-names...* to be the paging files for all subsequent Load World commands until a new Declare Paging-files command overrides it. *file-names* is a list of files separated by **spaces, not commas**. The default pathname (directory and file extension) for the first file is always FEP0:>.page. The default for subsequent files is the previous pathname without the filename. For example, if you specify FEP1:>abc.xyz, the default for the next file is FEP1:>.xyz.

The Declare Paging-files command is the same as the Declare More Paging-files command, expect that the latter command does not clear previous declarations.

For example, this command:

```
Declare Paging-files fep0:page aux fep1:page2 page3
```

declares the files: FEP0:>page.page, FEP0:>aux.page, FEP1:>page2.page, and FEP1:>page3.page.

The command does not declare duplicates. For example:

```
Declare Paging-files fep0:page aux page
```

does not declare fep0:>page.page a second time.

The command checks to see if each paging file actually exists. If a file does not exist, a warning is issued. The file is still added to the list of declared files in case the file is created some time in the future.

To undeclare all paging files, use Declare Paging-files without specifying any files.

If there are no declared paging files, the Load World command simply loads the paging file called FEP:>Page.page. If there are declared paging files, it adds them in the order in which they were declared. If there is a problem with a file, it warns you and goes on to the next file.

In .boot files, the Declare Paging-files command should be before the Load World command. Here is a sample boot file with the Declare Paging-files command:

```
Clear Machine
Load Microcode FEP1:>3640-mic.mic.389
Declare Paging-files FEP0:>Page More-Page Even-More-Page
Load World FEP1:>release-7-0.load.1
Set Chaos 401
Start
```

You can also put the Declare Paging-files command in the hello.boot file, after Initialize Hardware Tables. The Declare Paging-files command and the Declare More Paging-files command do not have to appear in the same boot file, either hello.boot or Boot.boot.

Declare More Paging-files *file-names*

Declares *file-names* to be the paging files for all subsequent Load World commands and adds the files to the list of previously declared paging files; it is the same as the Declare Paging-files command except that it does not clear previous declarations.

This command is of use when you decide to declare additional paging files after you have already used the Declare Paging-files command. It is also useful when you want to declare many paging files, but cannot do so with a single Declare Paging-files command because doing so on a single line can overflow the FEP's line input buffer.

Here is a sample boot file with the Declare Paging-files command and the Declare More Paging-files command:

```
Clear Machine
Load Microcode FEP1:>3640-mic.mic.389
Declare Paging-files FEP0:>Page More-Page Even-More-Page
Declare More Paging-files FEP0:>Yet-More-Page Still-More-Page
Load World FEP1:>Release-7-0.load.1
Set Chaos 401
Start
```

You can also put the Declare More Paging-files command in the hello.boot file, after Initialize Hardware Tables. The Declare Paging-files command and the Declare More Paging-files command do not have to appear in the same boot file, either hello.boot or Boot.boot.

Detach Graphics Tablet

Allows a graphics tablet to be disconnected from a serial port. This command supports software distributed by the Symbolics Graphics Division.

Disk Format

Formats the disk. This command overwrites all data on the disk. When you give the command, the FEP asks several questions; it expects answers in the following form:

Questions	*Valid answers*
Of type	M2284, T306, M2284, M2294 M2312, M2351A, XT1140, XT1150 XT2190, D2257, P807
On unit	Disk unit number
With pack id	0
From cylinder	Cylinder number; inclusive lower bound
Through cylinder	Cylinder number; inclusive upper bound

This answers to these questions, such as pack id, can be obtained by using the FEP command Show Disk Label.

Disk Restore

Restores the file system or files from cartridge tapes to disk. **Note:** Before using Disk Restore, ensure that memory is clear and the appropriate microcode is in place, by giving the commands Clear Machine and Load Microcode Tape:. Note the trailing colon (:) after Tape.

Disk Restore displays two questions:

1. `Have you used Set Disk-type for all units that do not have valid label blocks?`

   The disk type must be known before the first reference to it, in case the label block is not yet written.

   A tape can contain information in either *image restore* (*block restore*) format or *file restore* format. In both cases, up to 1152 characters of information are displayed, describing the contents of the section of the tape. Usually the information is supplied by the person producing the tape.

   - *image restore*: Data recorded in this format can be either an initial file system or raw disk blocks from a source disk. The tape-generating program writes out block numbers normalized to block 0 (and writes the number of the original starting block into the header information) so that they can be written to the new disk at a different location if desired. File systems *must* be restored to block 0; the header information reminds you of this.

- *file restore*: Data recorded in this format must be a file. The header includes the name of the source file, its length, and a comment supplied by the writer of the tape. You are asked for a destination pathname for the data; the default disk unit is assumed unless another is specified. Both a file system and the specified destination file must already exist on the unit, and the destination file must be large enough to hold the tape data. If the data pages of the destination file are not contiguous (because of bad blocks, say, or lack of contiguous space), then the restored data is fragmented also.

  When the file restoration is completed, a special restoration block is read, containing the length of the file, the author, the creation date, and a comment.

  Large files (for example non incremental worlds) cross tape boundaries. But since all block numbers are relative to the beginning of the file, the second reel of tape is logically continuous with the first, and file restoration proceeds as for single-reel files.

2. `Do you want to restore it?`

   You are prompted with Y (Yes), N (No), S (Skip microcodes), and F (Find microcode). If you answer "no", it skips the current tape restore section and searches for the next one. If you answer "skip microcodes" it stops when it reaches the world load, if you answer "find microcode", it asks what microcode to find. The default is the current microcode.

Dismount *pathname*

Forcibly dismounts the device (and unit) that is the device field of the pathname.

Hello

Takes a pathname (defaulting to FEP*n*:>hello.boot) that normally contains a sequence of Scan commands and the Initialize Hardware Tables command. The Scan commands scan the files containing the standard commands. Scan the hello.boot file each time the FEP is reset or the machine is powered up by typing Hello to the FEP prompt. When all the commands in the hello.boot file are completed, you can boot your machine with the Boot command.

Initialize Hardware Tables

Initializes hardware tables inside the FEP. This command reads the ID proms of the boards in the machine. Before this command is executed for the first time, the FEP has no knowledge of the type or location of the memory boards in the machine. This command is not strictly necessary, because any command that requires that the hardware tables be initialized automatically executes it. However, as the initialization function might have to print diagnostic messages on the console, it is useful to run it from the Hello.boot file; that way,any problems are identified at a predictable time, rather than spontaneously in the middle of some other activity.

Load Fep *file-name*

Loads and starts loadable FEP programs. The names of the FEP programs are usually of the form V127-*name*, where V127 is the number of the FEP version on which the program runs and *name* is the name of the program.

Load Microcode *file-name*

Loads microcode memory and other high-speed memories from the specified file. The default value of *file-name* is the last file name given to the Load Microcode command. Its initial default is >Microcode1.mic, which is determined by the hardware configuration (see the Compute Microcode Default command). Give the Clear Machine command before the Load Microcode command, if the computer was just powered on.

```
FEP Command:  Load Microcode FEP1:>3640-mic.mic.389
```

Load Sync-program *file-name*

Loads the specified file (of type .SYNC) into the sync program memory of the I/O board and clears the screen. This is used for machines with monitors that require different sync programs than the one that is preprogrammed into the FEP. The default value of *file-name* is the last file name given to the Load Sync-program command. Its initial default is >Sync.sync.

Load World *file-name*

Restores enough of the saved world in the computer so that you can start up the machine. It prints both the desired microcode version for this world and the currently loaded microcode version. The default value of *file-name* is the last file name given to the Load World command. Its initial default is >Released-World.load.

```
FEP Command:  Load World FEP1:>genera-7-0.load
```

Mount *pathname* Mounts the device (and unit) specified in the device field of the pathname. (This replaces the Mount <disk-unit-number> command of the old FEP system.) To mount disk unit 2, use the following:

```
Mount FEP2:
```

Whenever a device is supplied in any new FEP system command (for example, CART: or FEP3:), the device is mounted automatically. The Lisp system expects the FEP to inform it about all usable disks at the time the FEP starts Lisp running; thus, this command is necessary only when a disk that Lisp needs to know about has not been mentioned in any other interaction with the FEP.

Reset *subcommand*

Resets various parts of the computer. Subcommands are:

Reset Device CART:

Resets the cartridge tape drive.

Reset Device *pathname*

Performs a device-dependent reset of the device (and unit) specified in the device field of the pathname.

```
FEP Command: Reset Device FEP1:>
```

Reset Fep Restarts the FEP program, discarding the FEP's knowledge of what microcode was loaded, and so on. After the FEP is reset, you must initialize the overlay files by typing Hello to the FEP command prompt. Then, boot the machine by typing Boot to the FEP command prompt. If the FEP is running in RAM, asks whether to switch back to PROM.

Reset Most Resets the processor clock, the Lbus, the sequencer, the video, and the disks. If you think that the internal state of the computer is inconsistent, try Reset Most before power-cycling.

Reset Sequencer Resets the sequencer data paths.

Reset Video Reloads the console screen's sync program.

Set *subcommand* This command has the following subcommands:

Set Chaos-address *octal-value*

Sets the Chaosnet address. The default value of *octal-value* is the previous Chaosnet address. It is set to zero when the FEP is started.

The FEP checks for an acceptable Chaosnet address before starting Lisp. If none is specified as argument to this command, it warns you, asks whether the current setting is acceptable, and allows you to change it if necessary. Here is what you type to the FEP prompt:

```
FEP Command:  Set Chaos-Address 401
```

Set Default-disk-unit *unit*

Sets the default disk unit. *unit* becomes the default for most subsequent disk references. However, within a command file executed by a Boot command, the default disk unit is the one on which the command file is located. Here is what you type if you want the default disk unit to be FEP1.

```
FEP Command:  Set Default-disk-unit  FEP1:>
```

Set Disk Type *unit type pack-id*

Tells the FEP that disk *unit* is of type *type* and has pack id *pack-id.* Disk Restore might need this information if the disk has no label block or if the label block contains incorrect information. Give Set Disk-type after an implicit or explicit Mount command.

Set Display-string *string*

Displays the string in the nanofep display of machines that have a nanofep display. The length of the string is limited to 12 characters, the number of characters in the nanofep display. If more characters are used, the string is truncated. This command can be used in a .boot file.

Set Monitor-type *monitor-type*

Specifies the monitor type. The Set Monitor-type command ensures that the sync program used is for the monitor type requested. *monitor-type* can be either **Moniterm** or **Philips**; the types can be abbreviated to their first letter, **m** for **Moniterm** and **p** for **Philips**.

Set Monitor-type is used if the monitor is changed at a site and the ID prom is not changed accordingly. This command can be used in a boot file.

The following examples show two valid uses of the same command.

```
Set Monitor-type Moniterm

set mon m
```

Set Wired Addresses *%wired-virtual-address-high*

Sets values for wired addresses. This solves the following problem. If there are local or Symbolics-distributed patches to the wired system, and if these patches cause an internal limit to be exceeded, an error is signalled stating that the variable **sys:%wired-virtual-address-high** needs to be increased and gives a suggested new value. This command makes it easy to set the necessary variables.

Note: This command must be executed after the Load World command and before the Start command.

Show *subcommand*

Show has several subcommands:

Show Configuration Displays the hardware configuration, scans the backplane, and describes the boards that exist on the Lbus.

Show Directory *wildcard-directory-spec*

Displays the contents of a directory matching the wildcard in the FEP file system and the associated file comments. *directory-spec* must end in >* and can be preceded by fep: or fep*n*:. For example, Show Directory >* is acceptable, and shows the contents of the directory on the default disk. The default is FEP:>*.* or previous spec. For information about simple wildcards: See the section "Show Directory Command Understands Simple Wildcards", page 130.

Use Show Directory to check whether a file is in the FEP file system directory. For example, to see a directory listing of the contents of the files on FEP1, type the following to the FEP prompt:

```
FEP Command:  Show Directory  FEP1:>*.*
```

Show Disk Label *unit*

Displays the label of *unit*, the specified disk unit. This is done independently of the unit's being mounted, so you can tell what the label contains. The default for *unit* is the current default disk unit. For example, to see the disk label of FEP1, type:

```
FEP Command:  Show Disk Label 1
```

Show File *file-name* Displays the contents of *file-name*, a file in the FEP file system. *file-name* defaults to the last file name given to the Show File or Boot command. Its initial default is >Boot.boot. For example, to look at the contents of a boot file, type:

```
Show File  FEP0:>boot.boot
```

Show Paging-Files Shows two lists of files. The first set is the list of declared files (declared by using the Declare Paging-files command or the Declare

More Paging-files command) added by the Load World command. The second list is the files that have been added for paging. This is useful when you run out of swap space and need to avoid adding a paging file that is already being used.

Show Status

Displays the internal status of some machine registers. For information on interpreting the output of this command: See the section "Finding Out Why Your Machine Crashed", page 162. See the section "FEP Show Status Command Output", page 163.

Show Version

Displays the version number of loaded FEP software.

Shutdown

Halts the FEP. To restart (and reset) it, push the RESET button on the processor front panel. The preferred way to turn off the machine is:

1. Halt Lisp, using the Halt Machine command.

2. Halt the FEP, using the Shutdown command.

3. Power off the processor and console.

On the 3600, the Shutdown command asks "Do you really want to halt the FEP?" Answer Y to confirm. It then displays the message "FEP Halted" in the nanofep display.

On all machine models other than the 3600, the Shutdown command asks the question "Do you really want to power down the 3600?" Answer Y to confirm. It then lights the fault light on the switch panel.

Note: The Shutdown command replaced the Halt command.

Start

Starts the loaded Lisp world. If the world has just been loaded, this is a cold boot; if the world had been loaded previously, this is a warm boot. The Start command checks for an acceptable network address; if none was set, it is read from the computer and you are asked to confirm it. Test
Performs simple tests of main memory, A memory, and the disks.

Note: The Test command is chiefly for use by Symbolics developers.

16.6.2 FEP System Commands: Command Tree Maintenance

The command tree is a list of commands known to the FEP. When you execute the Hello command, all the files named in the hello.boot file are put in the list. This list is the command tree.

Clear Command Tree
Removes all commands defined by the overlay files, retaining only the commands native to the EPROM.

Scan *pathname* Adds (or updates) the commands in the specified file to the command tree.

Show Command Modules
Displays the currently active overlay files, whether or not they are currently loaded, the commands in the overlay files, and the command descriptions. For example, if the overlay file FEP0:>v127-loaders.flod is not currently loaded in your environment, the display looks like this:

```
Pathname FEP0:>v127-loaders.flod, not loaded
```

Show Command Tree
Displays the current command tree. Show Command Tree shows whether a command came from an overlay file or a module. It also shows the address within FEP memory at which the command resides (or would reside if the overlay file were loaded).

16.6.3 FEP System Commands: IDS

For more information on Incremental Disk Save: See the section "Incremental Disk Save (IDS)", page 62.

Enable IDS Informs Lisp that the Incremental Disk Save facility should be enabled. This command must be issued after the Load World command and before the Start command. This command is pervasive; you do not need to execute it for a world that has been saved with IDS enabled.

Disable IDS Informs Lisp that the Incremental Disk Save facility should not be enabled. This must be issued after the Load World command and before the Start command. Disable IDS is pervasive. It is included for completeness; Symbolics does not at this time know of any situations that would warrant its use.

Find IDS Files *pathname*

Examines all files that are incremental worlds in the specified FEP directory. Files that appear to be valid incremental worlds cause the internal database of IDS files to be updated. As the IDS files are found, the names of the files and generations of the disk save are displayed.

Add IDS File *pathname*

Explicitly adds the world specified in the pathname, provided it is a valid IDS world, to the internal IDS database. For example, type the following to add an IDS world named Inc-Release-7-0-from-Rel-7-0.load, which resides on FEP0:

```
FEP Command:  Add IDS File FEP0:>Inc-Release-7-0-from-Rel-7-0.load
```

Show IDS Files

Shows the internal IDS database in an understandable format. It starts by showing each file that does not have a parent. This usually means full disk saved worlds, but can also mean incremental worlds whose parents have been deleted. It then displays the incrementally saved worlds in depth-first fashion. Each description line shows the generation number, the file, the timestamps (IDs) of the file, and the timestamps of the parent (parent IDs).

Clear IDS Files

Clears the internal IDS database. You can use this, for example, if the FEP has been running continuously for a long time and the IDS database is cluttered with worlds that no longer exist.

16.6.4 FEP System Commands: Load-to-Paging Migration

Enable Load-to-Paging Migration

Enables the migration of referenced pages from the world load file to the paging file. This copies each page that has been read from the world load file to the paging file. All future reads of that page then come from the paging file rather than from the world load file.

Disable Load-to-Paging Migration

Disables the automatic migration of world load file pages to the paging file. When Load-to-Paging Migration is disabled, pages are written out to the paging file only if they are modified. Unmodified pages remain in the world load file.

These commands should be executed after the Load World command and before the Start command.

The advantage of having Load-to-Paging Migration enabled is that it provides better paging performance. The advantage of having it disabled is that the roughly 80 percent of the world load that is never modified does not need to have paging space allocated for it, so the effective available paging space is increased by roughly 80 percent of the size of the world load. The default in Release 6.1 and Genera 7.0 disables Load-to-Paging Migration. This default may change in the future.

16.7 FEP File System

The Symbolics computer disk has a file system called the *FEP file system.* The entire disk is divided up into *FEP files* (that is, files of the FEP file system). FEP files have names syntactically similar to those of files in the Symbolics computer's own local file system. However, the FEP file system and the Lisp Machine File System (LMFS) are completely distinct.

The *FEP file system* manages the disk space available on a disk pack, grouping sets of data into named structures called *FEP files.* All the available space on a disk pack is described by the FEP file system. A single FEP file system cannot extend beyond a single disk pack; each disk pack has its own separate FEP file system.

The FEP file system supports all of the generic file system operations. It also supports multiple file versions, soft deletion and expunging, and hierarchical directories.

Although "FEP" is an acronym for *front-end processor,* the FEP file system is managed by the main Lisp processor. It is called the FEP file system because the FEP can read files stored in the FEP file system. For example, the FEP uses the FEP file system for booting the machine and running diagnostics.

Disk streams access FEP files. A disk stream is an I/O stream that performs input and output operations on the disk. (For information about streams: See the section "Types of Streams" in *Reference Guide to Streams, Files, and I/O.* See the section "Stream Operations" in *Reference Guide to Streams, Files, and I/O.* When disk streams are opened with a **:direction** keyword of **:input** or **:output**, the disk stream reads or writes bytes, respectively, buffering the data internally as required. When the **:direction** is **:block**, the disk stream can both read and write the specified disk blocks. Block mode disk streams address blocks with a block number relative to the beginning of the file, starting at file block number zero. This *file block number* is internally translated into the corresponding disk address. The checkwords of all disk blocks contained in the FEP file system are reserved for use by the FEP file system, so block mode transfers should not use the checkwords stored in the disk array. See the section "3600-Family Disk System Definitions and Constants" in *Internals, Processes, and Storage Management.*

The FEP file system is also used by the system for allocating system overhead files, such as the paging file. See the section "FEP File Types", page 150. This section lists some of these files and what they are used for.

The need to allow the FEP to access FEP files, and also to allow the system to use them imposes some constraints on the design of the FEP file system. The internal data structures of the file system must be simple enough to permit the FEP to read them, and a small amount of concurrent access by both the FEP and Lisp must be tolerated. A FEP file's data blocks should have a high degree of locality on the disk to minimize access times. And the FEP file system must be very reliable, since the FEP needs to use the file system for running diagnostics and for booting the machine.

Note: Because of these constraints, the FEP file system is not intended to be a replacement for LMFS. (See the section "Lisp Machine File System" in *Reference Guide to Streams, Files, and I/O*.) Allocating new blocks for FEP files is slow, so creating many files, especially many small files, might impair the performance of the FEP file system, and ultimately the virtual memory system, if paging files or world load files become highly fragmented.

16.7.1 Naming of FEP Files

The FEP filename format is similar to the LMFS filename format. See the section "Lisp Machine File System" in *Reference Guide to Streams, Files, and I/O*. There are differences, however. Here are the format details of a FEP filename:

host

The name of the FEP file system host. The format for a FEP host is *host*|FEP*disk-unit*, where the *host* field specifies which machine's FEP file system you are referring to, and *disk-unit* specifies the disk unit number on the machine. The *host* field defaults to the local machine if you omit it and the terminating vertical bar (|). If you omit both the *host* and *disk-unit* fields, the FEP host defaults to the disk unit the world was booted from on the local machine. For example:

`Merrimack|FEP0`	The FEP file system on Merrimack's unit 0.
`FEP2`	The FEP file system on the local machine's unit 2.
`FEP`	The FEP file system the booted world load file resides on.

directory

The name of the directory. The FEP file system supports hierarchical directories in the same format as in LMFS. Each directory name is limited to a maximum of 32 characters; there is no limit on the total length of a hierarchical directory specification.

name The name of the FEP file, which cannot exceed 32 characters.

type The type of the FEP file, which cannot exceed 4 characters.

version The version number of the FEP file, which must be a positive integer or the word "newest".

FEP files can be renamed. For example, if you save a world containing MACSYMA, you might want to rename the world file to >macsyma.load or >macsymal.load. Be sure to update your boot file if you intend this to be the default world.

16.7.2 FEP File Types

By convention, the following file types are used by the FEP file system for files used by that system.

boot The file contains FEP commands that can be read by the FEP's Boot command. *boot* files are text files, and can be manipulated by the editor. See the section "Configuration Files", page 151.

load The file contains a world load image, or *band*, that is used to boot the system. For example, >release-7.load.newest contains the Genera 7 world load image.

mic The file contains a microcode image, plus the contents of other internal high-speed memories that are initialized when the computer is booted. For example, >tmc5-io4-row-mic.mic.384 contains version 384 of the microcode for version 5 of the TMC.

fspt The file contains a LMFS partition table. It tells LMFS which FEP files to use for file space. For example, >fspt.fspt.newest is the default partition table used by LMFS.

file The file contains a LMFS partition which holds the machine's local file system. The entire Symbolics computer local file system normally resides inside one big file of the FEP file system. For example, >lmfs.file.newest is the default LMFS file partition.

page The file contains disk space that can be used by the virtual memory system. To increase the effective size of virtual memory, you can add additional paging files. See the section "Allocating Extra Paging Space", page 50. For example, >page.page.newest is the default file used by the virtual memory system as storage for swapping pages in and out of main memory.

flod
The file contains a FEP Load file. FEP Load files contain binary code the FEP can load and execute.

fep
The file contains binary information used by the FEP file system. These files should not be written to by user programs. Some examples of these files are:

>root-directory.dir This is the root directory for the FEP file system.

>free-pages.fep Describes which blocks on the disk are allocated to existing files.

>bad-blocks.fep Owns all the blocks that contain a media defect and should not be used.

>sequence-number.fep
Contains the highest sequence number in use. The FEP file system uses sequence numbers internally to uniquely identify files. This is to assist in rebuilding the file system in case of a catastrophic disk failure.

>disk-label.fep Contains the disk pack's physical disk label. The label is used to identify the pack and describe its characteristics.

dir
The file contains a FEP directory. For example, fep0:>root-directory.dir.newest contains the top-level root directory. The directory file for fep0:>dang>examples> would reside in fep0:>dang>examples.dir.1.

16.7.3 Configuration Files

Configuration files contain FEP commands tailored for a particular Symbolics computer configuration. The commands are executed if you specify the file as argument to a Boot command when cold booting the machine. See the section "FEP Commands", page 133.

The configuration file >Boot.boot usually contains FEP commands to:

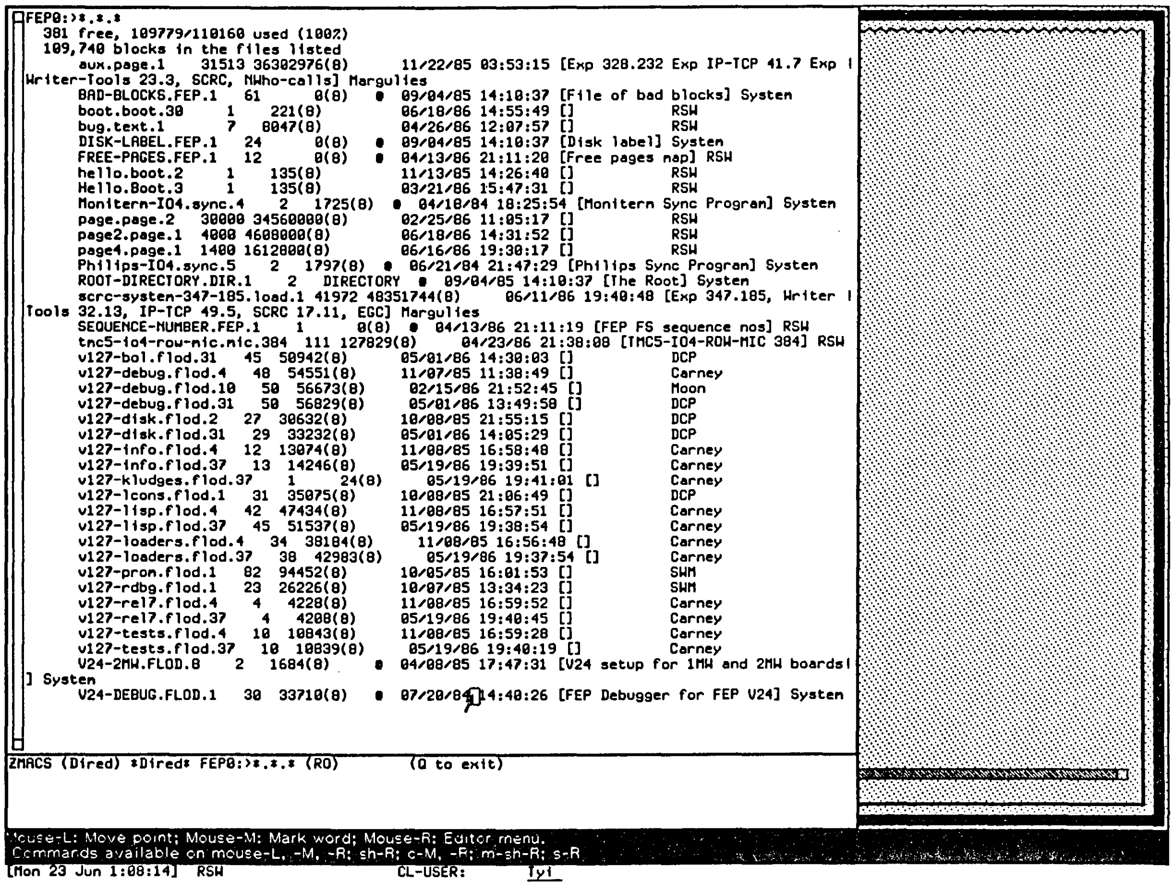
- Clear the internal state of the machine
- Load the microcode
- Load a world
- Set the Chaosnet address

- Start the machine

To change the selection of microcode and world loads that are booted by default, simply use Zmacs to edit the file FEP*n*>Boot.boot, where FEP*n* is the disk unit. Be careful to avoid typographical errors; otherwise, you might have to type in the commands manually in order to boot the machine. Also, be sure that the last command in the file is followed by RETURN.

16.7.4 FEP File Comment Properties

Comment properties supply additional information about the contents of FEP files. In the Dired mode of Zmacs, they are listed inside square brackets, where the reference or expunge date appears for other file systems. You can list the contents of the FEP file system by using the Show FEP Directory command. The Zmacs command Dired (m-X) of fep*n*:>*, or the form (dired "fep:*n*>*") (where *n* is the disk unit) invokes the directory editor on the FEP file system. An example of the Zmacs Dired command output is shown in figure 8.



```
FEP0:>*.*.*
  381 free, 109779/110160 used (100%)
  109,740 blocks in the files listed
      aux.page.1    31513 36302976(8)      11/22/85 03:53:15 [Exp 328.232 Exp IP-TCP 41.7 Exp !
Writer-Tools 23.3, SCRC, NWho-calls] Margulies
      BAD-BLOCKS.FEP.1   61       0(8)   ●  09/04/85 14:10:37 [File of bad blocks] Systen
      boot.boot.30      1    221(8)        06/18/86 14:55:49 []            RSW
      bug.text.1        7   8047(8)        04/26/86 12:07:57 []            RSW
      DISK-LABEL.FEP.1   24       0(8)   ●  09/04/85 14:10:37 [Disk label] Systen
      FREE-PAGES.FEP.1   12       0(8)   ●  04/13/86 21:11:20 [Free pages nap] RSW
      hello.boot.2      1    135(8)        11/13/85 14:26:40 []            RSW
      Hello.Boot.3      1    135(8)        03/21/86 15:47:31 []            RSW
      Monitern-IO4.sync.4    2   1725(8)  ●  04/18/84 18:25:54 [Monitern Sync Progran] Systen
      page.page.2   30000 34560000(8)      02/25/86 11:05:17 []            RSW
      page2.page.1  4000 4608000(8)        06/18/86 14:31:52 []            RSW
      page4.page.1  1400 1612800(8)        06/16/86 19:30:17 []            RSW
      Philips-IO4.sync.5    2   1797(8)  ●  06/21/84 21:47:29 [Philips Sync Progran] Systen
      ROOT-DIRECTORY.DIR.1    2   DIRECTORY  ●  09/04/85 14:10:37 [The Root] Systen
      scrc-systen-347-185.load.1 41972 48351744(8)     06/11/86 19:40:48 [Exp 347.185, Writer !
Tools 32.13, IP-TCP 49.5, SCRC 17.11, EGC] Margulies
      SEQUENCE-NUMBER.FEP.1    1       0(8)  ●  04/13/86 21:11:19 [FEP FS sequence nos] RSW
      tnc5-io4-row-nic.nic.384  111 127829(8)     04/23/86 21:38:08 [TMC5-IO4-ROW-MIC 384] RSW
      v127-bol.flod.31   45  50942(8)      05/01/86 14:30:03 []            DCP
      v127-debug.flod.4   48  54551(8)     11/07/85 11:38:49 []            Carney
      v127-debug.flod.10   50  56673(8)     02/15/86 21:52:45 []           Moon
      v127-debug.flod.31   50  56829(8)     05/01/86 13:49:58 []           DCP
      v127-disk.flod.2   27  30632(8)      10/08/85 21:55:15 []            DCP
      v127-disk.flod.31   29  33232(8)     05/01/86 14:05:29 []            DCP
      v127-info.flod.4   12  13074(8)      11/08/85 16:58:48 []            Carney
      v127-info.flod.37   13  14246(8)     05/19/86 19:39:51 []            Carney
      v127-kludges.flod.37    1      24(8)     05/19/86 19:41:01 []        Carney
      v127-lcons.flod.1   31  35075(8)     10/08/85 21:06:49 []            DCP
      v127-lisp.flod.4   42  47434(8)      11/08/85 16:57:51 []            Carney
      v127-lisp.flod.37   45  51537(8)     05/19/86 19:38:54 []            Carney
      v127-loaders.flod.4   34  38184(8)     11/08/85 16:56:48 []          Carney
      v127-loaders.flod.37   38  42983(8)     05/19/86 19:37:54 []         Carney
      v127-pron.flod.1   82  94452(8)      10/05/85 16:01:53 []            SWM
      v127-rdbg.flod.1   23  26226(8)      10/07/85 13:34:23 []            SWM
      v127-rel7.flod.4    4   4228(8)      11/08/85 16:59:52 []            Carney
      v127-rel7.flod.37    4   4208(8)     05/19/86 19:40:45 []            Carney
      v127-tests.flod.4   10  10843(8)     11/08/85 16:59:28 []            Carney
      v127-tests.flod.37   10  10839(8)     05/19/86 19:40:19 []           Carney
      V24-2MW.FLOD.8    2   1684(8)      ●  04/08/85 17:47:31 [V24 setup for 1MW and 2MW boards!
] Systen
      V24-DEBUG.FLOD.1   30  33710(8)    ●  07/20/84 14:40:26 [FEP Debugger for FEP V24] Systen

ZMACS (Dired) *Dired* FEP0:>*.*.* (RO)        (Q to exit)

Mouse-L: Move point; Mouse-M: Mark word; Mouse-R: Editor menu.
 Commands available on mouse-L, -M, -R; sh-R; c-M, -R; m-sh-R; s-R
[Mon 23 Jun 1:08:14]  RSW                    CL-USER:     Tyi
```

Figure 8. FEP File Comment Properties

16.7.5 Accessing FEP Files

FEP files are accessed by open disk streams. A disk stream is opened by the **open** function. (See the section "Accessing Files" in *Reference Guide to Streams, Files, and I/O*. That section contains more details on accessing files.) If a FEP file system residing on a remote host is referred to, a *remote stream* is returned with limited operations, as specified by the remote file protocol.

In addition to the normal **open** options, the following keywords are recognized:

:if-locked This keyword specifies the action to be taken if the specified file is locked. This keyword is not supported by the remote file protocol.

> **:error** Signal an error. This is the default.
>
> **:share** Open the specified file even if it is already locked, incrementing the file's lock count. This mode permits multiple processes to write to the same file simultaneously. (See the section "FEP File Locks", page 158. That section contains more information on file locks.)

:number-of-disk-blocks
The value of this keyword is the number of disk blocks to buffer internally if the **:direction** keyword is **:input** or **:output**. This keyword is ignored for other values of **:direction** or for files on remote hosts. The default **:number-of-disk-blocks** is two.

16.7.6 Operating on Disk Streams

All disk streams to a local FEP file system handle the following messages:

:grow &optional *n-blocks* &key **:map-area** **:zero-p** *Message*

This message allocates *n-blocks* of free disk blocks and appends them to the FEP file. The value of *n-blocks* defaults to one. If **:zero-p** is true the new blocks are filled with zeros; otherwise, they are not modified. The return value of **:grow** is the file's data map (the format of the data map is described in **:create-data-map**'s description below). The value of **:map-area** is the area to allocate the data map in, which defaults to **default-cons-area.**

:allocate *n-blocks* &key **:map-area** **:zero-p** *Message*

This message ensures that the FEP file is at least *n-blocks* long, allocating additional free blocks as required. Returns the file's data map (the format of the data map is described in **:create-data-map**'s description below). **:map-area** specifies the area to create the data map in, and defaults to **default-cons-area**. The newly allocated blocks are filled with zeros if **:zero-p** is true. **:zero-p** defaults to **nil.**

:file-access-path *Message*

This message returns the disk stream's file access path.

For example, you can find out what unit number a FEP file resides on as follows:

```
(send (send stream :file-access-path) :unit)
```

:map-block-no *block-number grow-p* *Message*

This message translates the relative file *block-number* into a disk address, and returns two values: the first value is the disk address, and the second is the total number of disk blocks, starting with *block-number*, that are in consecutive disk addresses. *grow-p* specifies whether the file should be extended if *block-number* addresses a block that does not exist. When *grow-p* is true, free disk blocks are allocated and appended to the FEP file to extend it to include *block-number.* Otherwise, if *grow-p* is false, **nil** is returned if *block-number* addresses a block that does not exist.

:create-data-map &optional *area* *Message*

This message returns a copy of the FEP file's data map allocated in area *area*, which defaults to **default-cons-area**. A FEP file data map is a one-dimensional **art-q** array. Each entry in the file data map describes a number of contiguous disk blocks, and requires two array elements. The first element is the number of disk blocks described by the entry. The second element is the disk address for the first block described by the entry. The array's fill-pointer contains the number of active elements in the data map times two.

:write-data-map *new-data-map disk-event* *Message*

This message replaces the file's data map with *new-data-map. disk-event* is the disk event to associate with the disk writes when the disk copy of the file's data map is updated. This message overwrites the file's contents and should be used with caution.

16.7.7 Input and Output Disk Streams

Input and output disk streams are buffered streams. In addition to the standard buffered stream messages, local input and output disk streams also support the messages described elsewhere: See the section "Operating on Disk Streams", page 154.

Input disk streams read bytes of data starting at the current byte position in the FEP file, updating the byte position as the data is read. Output disk streams write bytes of data in the same way.

The bytes of data are stored in buffers internal to the stream. The **:number-of-disk-blocks open** keyword controls how many disk blocks the internal buffers can hold. When the current pointer moves beyond a disk block boundary, the buffered disk block is written to the file for an output stream, or the next unbuffered block is read in from the file for an input stream. Output streams also write out all the buffered disk blocks when the stream is sent a **:close** message without an **:abort** option.

16.7.8 Block Disk Streams

Block disk streams can both read and write disk blocks at specified file block numbers. A file block number is the relative block offset into the file. The first block in the file is at file block number zero, the second is at file block number one, and so on.

Block disk streams do not buffer any blocks internally and are not supported by the remote file protocol.

See the section "Operating on Disk Streams", page 154. In addition to the messages described in that section, block disk streams support the following messages:

:block-length *Message*

The **:block-length** message returns the length of the FEP file in disk blocks.

:block-in *block-number n-blocks disk-arrays* &key **:hang-p** **:disk-event** *Message*

The **:block-in** message causes the disk to start reading data from the disk into the disk arrays in *disk-arrays*, starting with the file block number *block-number*, and continuing for *n-blocks*. *disk-arrays* can be a disk array or a list of disk arrays. The value of *n-blocks* is the number of disk blocks to read. When *n-blocks* is greater than one, each disk array is completely filled before using the next disk array in *disk-arrays*. The checkwords stored in the disk arrays are reserved for use by the FEP file system. See the section "3600-Family Disk System Definitions and Constants" in *Internals, Processes, and Storage Management.* Unused disk arrays or portions of disk arrays remain unmodified.

When the value of **:hang-p** is true, which it is by default, **:block-in** waits for all the reads to complete before returning. If the value of **:hang-p** is false, **:block-in** returns immediately upon enqueuing the disk reads without waiting for completion. In this case, all *disk-arrays* and the *disk-event* must be wired before sending the **:block-in** message, and must remain wired until the disk reads complete.

If the **:disk-event** keyword is supplied, its value is the disk event to associate with the disk reads. Otherwise the **:block-in** message allocates a disk event for its duration. A **:disk-event** must be supplied when **:hang-p** is false.

:block-out *block-number n-blocks disk-arrays* &key **:hang-p** **:disk-event** *Message*

The **:block-out** message causes the disk to start writing the data in the disk arrays in *disk-arrays* onto the disk, starting with the file block number *block-number*, and continuing for *n-blocks*. The arguments to the **:block-out** message are identical to those of the **:block-in** message.

16.7.9 FEP File Properties

In addition to having a name and containing data, FEP files also have properties. These properties store information about the file itself, such as when it was last written and whether it can be deleted or not. File properties are read by the **fs:file-properties** function, and modified by the **fs:change-file-properties** function. The **fs:directory-list** function also returns the file properties of several files at once. (See the section "Accessing Directories" in *Reference Guide to Streams, Files, and I/O.*)

The following file properties can be both read and modified:

:creation-date	The universal time the file was last written to. Universal times are integers. (See the section "Dates and Times" in *Programming the User Interface, Volume B.*)
:author	The user-id of the last writer. The user-id must be a string.
:length-in-bytes	The length of the file, expressed as an integer.
:deleted	When **t** the file is marked as being deleted. A deleted file can then be marked as being undeleted by changing this property to **nil**. The disk space used by a deleted file is not actually reclaimed until the file is expunged.
:dont-delete	When **t**, attempting to delete or overwrite the file signals an error. **nil** indicates the file can be deleted or written to.
:comment	A comment to be displayed in brackets in the directory listing. The comment must be a string.

The following file properties are returned by the **:properties** message, but cannot be modified by **:change-properties**:

:byte-size The number of bits in a byte. The value of this property is always 8.

:length-in-blocks The block length of the file expressed as an integer.

:directory If **t**, the file is a directory; otherwise **nil**.

16.7.10 FEP File Locks

A FEP file is *locked* for the interval from when it is opened for reading or writing until it is closed. If the **:direction** keyword is **:input**, the file is *read-locked*; if the **:direction** keyword is **:output** or **:block**, the file is *write-locked.*

When the **:if-locked** keyword is **:error**, which is its default, a file that is read-locked can still be opened for reading but signals an error if opened for writing; a file that is write-locked cannot be opened for reading or writing. This permits multiple readers to access a file concurrently, while prohibiting writing to the file being read.

When the **:if-locked** keyword is **:share** in an open call for write, it succeeds in opening the file even if it is already read- or write-locked.

An expunge operation on a file that is either read- or write-locked does not expunge the file. If expunging a directory fails to expunge a file, the file must be closed and the directory expunged again.

16.7.11 Installing Microcode

Use the Copy Microcode command to retrieve any new microcode from the file system of the sys host.

Copy Microcode Command

Copy Microcode {*version or pathname*} *destination keywords*

Installs a version of microcode.

version or pathname

Microcode version number or pathname to copy. *version* is a microcode version number (in decimal). *pathname* rarely needs to be supplied. It defaults to a file on FEP*n*:> (where *n* is unit number of the boot disk) whose name is based on the microcode name and version. (The file resides in the logical directory sys:l-ucode;.) The *version* actually stands for the file *appropriate-hardware*-MIC.MIC.*version* on FEP*n*:>. (See the Section "Genera 7.0 Microcode Types" in *Software Installation Guide*) *destination* FEP file specification. The pathname on your FEP*n*:> directory. The default is created from the microcode version.

keywords :update boot file

:update boot file

{FEP-file-spec none}. The default is the current default boot file name.

Initially, the Symbolics personnel who install your system establish these microcode files for you.

16.7.12 Using a Spare World Load for Paging

You can reuse FEP file space for paging files. You may have a spare world load file, which you can transform into a paging file. For example, once you have successfully installed a new software release, you can rename the old world load to be a paging file. **Note:** Do not use the world load you are currently running for a paging file, as this action overwrites the previous contents of the specified file.

If your old world load is Release-6-1.load, is resident on FEP0:>, and is 36,000 blocks in size, and you want to create a new paging file called FEP0:>page2.page (with a block size of 36,000), follow these steps:

1. You should rename the file FEP0:>release-6-1.load to FEP0:>page2.page using the Rename File command. For example, type:

   ```
   Rename File FEP0:>release-6-1.load  FEP0:>page2.page
   ```

   Now the world load has been renamed to a paging file.

2. Use the Add Paging File command to initialize the paging file from the Lisp environment.

3. Edit your FEP*n*:>Boot.boot file to declare the new paging file. Use the Declare Paging-files command in your boot file to do this. For information about the Declare Paging-file command: See the section "FEP System Commands: General Usage", page 133.

You can also create new FEP files and use them for extra paging space: See the section "Allocating Extra Paging Space", page 50.

16.7.13 Adding a Spare World Load as LMFS File Space

Partitions can be added to LMFS by following these steps:

1. Create the partition you wish to add to LMFS prior to entering the File

system editing operations program. In addition, when you add a new partition or a partition on another disk, the disk should be free of errors and properly initialized and formatted.

2. Press SELECT F to select the File system editing operations program.

3. Click on [Local LMFS Operations] to invoke the second level of the File System Maintenance Program.

4. Click on [LMFS Maintenance Operations] to invoke the third level menu, which is a menu of file-system maintenance operations.

5. Click right on [Initialize] to invoke a menu of initialization options, which offers [New File System] and [Auxiliary Partition] as choices. Clicking on [New File System] is similar to clicking left on [Initialize]; it initializes a partition to be the basis of a file system.

6. Click on [Auxiliary Partition] to add another partition.

7. Enter the pathname of the FEP file to be used as the new partition. The default presented, which is correct for [New File System], is never correct for adding a partition.

8. Click on [Do It]. The system then performs much verification and error checking, roughly as much as when initializing a new partition. It must not be interrupted while performing these actions.

9. When finished, the File system editing operations program adds the partition and edits the FSPT automatically.

16.8 Disk Handling

You can include a disk specification of the form FEP*n*: (where *n* refers to disk unit *n*) as the first field of file and directory references to the FEP. A specification of fep: (with no unit number) refers to the disk unit from which the current Lisp world was booted, that is, the unit containing the world load file. If fep*n*: is omitted entirely, the default disk unit, set by Set Default-disk-unit is assumed.

16.8.1 Disk Handling Commands

The following FEP commands manipulate disk units. For a description of these commands: See the section "FEP System Commands: General Usage", page 133.

Add Disk-type

Clear Disk-types

Dismount <pathname>

Mount <pathname>

Reset Device <pathname>

Set Default-disk-unit *unit*

Set Disk-type *unit type pack-id*

Show Directory *directory-spec*

Show Disk Label *unit*

Show File *file-name*

16.8.2 Multiple Disk Units

Each Symbolics computer can access more than one local disk. The following conditions apply:

- The FEP can access any disk at all. Currently, the hardware allows a maximum of eight disks.

- You can boot a Lisp world from any disk by using the FEP command Load World. Also, you can add paging files from any disk by using the FEP command Declare Paging-files in your boot file. To load paging files from Lisp, you can use the Add Paging Files command.

- The form FEP: refers to the disk from which you booted the current world. If this is disk 0, then FEP: is equivalent to the form FEP0:. However, it is also possible to specify another disk explicitly, using such forms as FEP1: or FEP2:.

- World loads are not specific to a type of disk. This means that if one Symbolics computer has a T306 disk and another has an M2284 disk, world loads can be transferred back and forth between the disks.

16.8.2.1 Disk Types

The FEP currently supports the following types of disks:

Century Data T306 (300 megabytes unformatted capacity – removable)

CDC EMD368 (368 megabytes unformatted capacity)

CDC EMD515 (515 megabytes unformatted capacity)

CDC XMD858 (858 megabytes unformatted capacity)

Fujitsu M2284 (168 megabytes unformatted capacity)

Fujitsu M2294 (335 megabytes unformatted capacity)

Fujitsu M2351A (474 megabytes unformatted capacity)

Maxtor XT1140 (140 megabytes unformatted capacity)

Maxtor XT2190 (190 megabytes unformatted capacity)

NEC D2257 (167 megabytes unformatted capacity)

Priam P807 (335 megabytes unformatted capacity)

16.9 Finding Out Why Your Machine Crashed

When your machine crashes, using the FEP Show Status command can give you useful information for diagnosing the cause of the crash. For an outline of the information that Show Status prints: See the section "FEP Show Status Command Output", page 163.

The Show Status output section "3600 program counters" includes the macro PC, the CPC, and the 16 OPCs. The macro PC is the address of the current instruction of compiled Lisp code. The CPC is the address of the current microinstruction. The OPCs are the addresses of the 16 most recently executed microinstructions; OPC+0 is the most recent, OPC+17 the earliest. An arrow points at either the CPC or the first OPC, depending on the error condition that stopped the machine. This is the microinstruction that was executing when the event occurred that was the proximate cause of the machine's stopping itself.

16.9.1 FEP Show Status Command Output

The register contents and program counters displayed by the Show Status command give some information on machine states causing the FEP message "Lisp stopped itself". They are generally not useful for interpreting wired-ferror halts. Show Status merely prints the contents of certain hardware registers, decoding the bits symbolically. The FEP does not interpret these contents, so some output might not be meaningful. The following cautions apply:

- You must interpret some bits depending on the value of other bits.
- Some registers listed below are printed only if they contain "useful" information.

The most important registers are *Sequencer status* and *MC error status.*

FEP buffer status

Bit	*Meaning*
Spy DMA Enb	Spy bus being used by FEP to access disk or net (means spy bus being used for normal functions)
Write to dev / Read from dev	Spy DMA direction
Drive busy	Spy DMA mode (who controls busy line)
Int Enb	Spy DMA enable to interrupt FEP
Count up / Count down	Spy DMA address increment direction
Busy	Spy DMA busy (inside FEP)
Spy DMA busy	Spy DMA busy line (actual line on backplane)
DMA setup	[meaning unknown]

FEP Lbus control

Bit	*Meaning*
ECC Diag	Normal memory error correction logic disabled; instead, FEP can read or write the 8 extra bits of main memory
Doorbell Int Enb	Doorbell (Lisp-to-FEP signal) interrupt enabled
Use Uncorrected Data	FEP unaware of corrected Lbus data if single-bit-error
Ignore Double ECC Error	FEP does not get bus error if uncorrectable Lbus error (either double-bit error or nonexistent memory)
Task 3 Req	FEP trying to wake up microtask 3

Doorbell	Doorbell ringing (Lisp-to-FEP signal)
Lbus Buffer Busy	[self-explanatory]
Lbus Buffer Some Parity Error	[self-explanatory]

FEP Board ID control

Bit	*Meaning*
Continuity	Read-back of random signal that checks board presence
Lbus ID Req	Lbus reading board IDs, not doing normal functions
Half Speed	Main processor clock running at half speed

FEP Proc control

Bit	*Meaning*
Lbus Power Reset	Reset all Lbus devices due to power turn-on or turn-off
Lbus Power Reset (on bus)	Same as above, but actually read back from the bus
Lbus Reset	Reset all Lbus devices
Lbus Reset (on bus)	Reads back the Lbus Reset
Clear Errors	Bit that clears FEP error registers (not an error)
FEP Int Enable	FEP interrupt enable (not an error)
Kept Alive	FEP died and was reset by nanofep
FEP Ram Par Err	Parity error in dynamic ram on FEP board

Sequencer error status

(Status of the SQ board and the main error status bits that can halt the machine)

Bit	*Meaning*
Microcode-halted	A "halt" microinstruction was executed, for one of the following reasons: • A call to the %HALT function (due to a wired-ferror or a call to HALT) • A fatal error, such as an error while entering the error handler or an error in wired code (page fault, disk handlers)

- Executing an undefined macroinstruction (running too old a version of microcode or executing bad macrocode)
- Failure of a microcode consistency check (stack frame too large, stack overwritten)

Self-explanatory hardware errors:

Bit	*Meaning*
Spare-error-bit	[never happens, unless manually wired to some signal]
GC-Map-parity-error	GC MAP ram on DP board
Type-map-parity-error	TYPE MAP ram on DP board
Page-Tag-parity-error	PAGE TAG ram on FEP board
A-memory-parity-error	AMEM ram on DP board
B-memory-parity-error	BMEM ram on DP board
MC-error (map, ifu, or main mem)	Error on MC board; see MC error status
AU-error	Error on AU (FPA) board (if the machine has one)
Task-state-memory-parity-error	TSKM ram on SQ board (doesn't always halt machine)
Control-memory-parity-error	CMEM ram on SQ board (also for microcode breakpoints if an L-Console program is cabled up for debugging)

Hardware "errors" that are not always errors:

Bit	*Meaning*
CTOS-low-parity-error	CSTK ram on SQ board (low half of output register)
CTOS-high-parity-error	CSTK ram on SQ board (high half of output register)

(Note: If CTOS-came-from-IFU is true, above two bits have no meaning.)

Sequencer miscellaneous status

(Status bits that are not errors)

Bit	*Meaning*
CTOS-came-from-IFU	CTOS register holds macroinstruction dispatch

	address from IFU (or TMC) rather than contents of CSTK ram
TSK-STOP (sequencer stopped)	Machine is stopped for some reason
Errhalt-Sync	Some error bit is on (stops machine)
MC Wait	Microinstruction waiting for memory control to allow the instruction to continue
Task Switch	Switching to a different microtask

MC Error status

Bit	*Meaning*
Double bit error	An uncorrectable error in main memory, or a reference to a nonexistent Lbus address. Further information under the heading, ECC syndrome.
Map A parity error, Map B parity error	Parity errors in the map caches on the MC (TMC, IFU) board.
Hit in both map A and map B	Both map caches claiming to map the same address. Could be the map hardware, or some hardware or microcode problem causing map to be written with bad data.

ECC syndrome

(An octal number followed by an address with x's in it)

This register contains the most recent main-memory read-error correction status. The error can be caused by a read by the processor, a read by the FEP, or a read by a DMA I/O device. The events that set this register include nonexistent memory reference, single-bit error correction, and double-bit error detection. Nonexistent memory and double-bit error halt the processor (even if it was the FEP or an I/O device that got the error). Currently, the FEP disables itself from getting a bus error if it references nonexistent Lbus memory or gets a double-bit error in Lbus memory.

One other event that can set this register is a bug in the FEP's code for examining the machine's status. In this case, the first two digits of the address are usually 77.

The address is the physical address of the location referenced. Only bits 23-18 and 1-0 are valid (the rest are x'ed out). These are sufficient bits to determine which Lbus slot (bits 23-19) and which of the 8 banks within a memory board are being referenced. To convert the address to an Lbus slot number, consider the one or two digits at the left of the x's to be an octal value, and divide it by 2.

This is a logical slot number, as printed (in decimal) by the Show Configuration command. It is not related to the numbers printed on the machine chassis. Slot 0 is at the left, as seen from the front of the machine.

The syndrome codes are as follows:

	0	1	2	3	4	5	6	7
000	okay	36	37	2-bit	38	2-bit	2-bit	3
010	39	2-bit	2-bit	6	2-bit	8	9	2-bit
020	40	2-bit	2-bit	13	2-bit	15	16	2-bit
030	2-bit	18	19	2-bit	20	2-bit	2-bit	unused
040	41	2-bit	2-bit	23	2-bit	25	26	2-bit
050	2-bit	28	29	2-bit	30	2-bit	2-bit	31
060	2-bit	33	34	2-bit	35	2-bit	2-bit	unused
070	NXM	2-bit	2-bit	unused	2-bit	unused	unused	2-bit
100	42	2-bit	2-bit	0	2-bit	1	2	2-bit
110	2-bit	4	5	2-bit	7	2-bit	2-bit	10
120	2-bit	11	12	2-bit	14	2-bit	2-bit	unused
130	17	2-bit	2-bit	unused	2-bit	unused	unused	2-bit
140	2-bit	21	22	2-bit	24	2-bit	2-bit	unused
150	27	2-bit	2-bit	unused	2-bit	unused	unused	2-bit
160	32	2-bit	2-bit	unused	2-bit	unused	unused	2-bit
170	2-bit	unused	unused	2-bit	unused	2-bit	2-bit	unused

3600 program counters

Label	*Meaning*
Macro PC	The address of the current instruction of compiled Lisp code. This is prefaced with either (Odd) or (Even) since there are two instructions per word.
Current micro PC (CPC)	The address of the current microinstruction.
Old PCs (OPC)	The addresses of the 16 most recently executed microinstructions. OPC+0 was executed most recently, OPC+17 least recently.

16.9.2 Decoding Micro PCs

Use the function **dbg:decode-micro-pc** to decode the microcode PCs printed by the FEP command Show Status.

dbg:decode-micro-pc *pc* &optional (*name* **sys:%microcode-version**) *Function*
(*version*
(sys:microcode-version-number sys:%microcode-version))

dbg:decode-micro-pc is useful for investigating why a machine crashed. It decodes the octal microinstruction addresses printed by the FEP command

Show Status. To use this function you should first write down the Show Status output. You can then either warm boot the machine using the Start command or call **dbg:decode-micro-pc** on another machine.

pc is an address in the microcode, taken from the CPC or OPC information printed by the Show Status command. Show Status prints these numbers in octal; if your default radix is decimal, precede *pc* by #**o**. Normally the number in the Show Status output with the arrow (→) pointing to it is the relevant number, but sometimes, decoding all of the numbers gives you additional clues.

name and *version* are optional; they specify the version of the microcode that was running at the time of the crash. You can omit these arguments if you call **dbg:decode-micro-pc** while using the machine that crashed and while running the same microcode version as at the time of the crash. You can also omit these arguments if you call this function from another machine that has a software *and* hardware configuration that is *identical* to that of the machine that crashed. To find the microcode version name and number that a machine is running, use the command Print Herald with the keyword :detailed or take the name and version number of the microcode file in the machine's boot file (normally fep0:>Boot.boot). Microcode version numbers are decimal; include a period at the end of the number if your default radix is octal.

Example:

```
    (dbg:decode-micro-pc #o44552 "3640-mic" 389.)
```

dbg:decode-micro-pc prints information that depends on the microinstruction:

Microinstruction	*Information printed*
Halt instruction	The reason it halts the machine. An example is "error in the error handler". These reasons are constant strings in the microcode source program and do not represent any dynamic analysis of the state of the machine.
Signaller of a Lisp error	The internal form of the error message. This is not the same form of error message you would ever see otherwise; normally Lisp software translates these messages into conditions and signals them, and the conditions define more readable error messages. This is useful mainly in decoding OPCs earlier than

the one with the arrow, when the machine halted because of "error in the error handler".

Handler for a macroinstruction in compiled Lisp code
The name of that macroinstruction. A halt here might be caused by running a world together with an incompatible microcode, such as a microcode from an earlier release, that does not implement an instruction used by that world.

If all else fails, the function offers to load the microcode symbol table (from the sys:l-ucode; directory) and then prints the symbolic name of the microinstruction. Loading the microcode symbol table takes a few minutes. Microinstruction symbolic names can sometimes be clues to help in figuring out what the machine was doing at the time it crashed.

Two types of symbolic names exist: those with and without parentheses.

If the name includes parentheses, it is a list of the name of a microcode routine and the path through that routine to reach the microinstruction in question. *Beware of a pitfall!* These names are not unique; the same microinstruction can be reached by multiple paths from different microcode routines. For example, a microinstruction named (FTN-AR-1 3) might also be part of the microcode for the CAR instruction; you cannot assume too much from the name if it contains parentheses. It is only a clue.

If a symbolic name is just a symbol and has no parentheses, it is unique and names the first microinstruction of a microcode routine.

Beware of assuming too much. If the reason Lisp stopped itself is not "microcode halted", the information that **dbg:decode-micro-pc** prints is not likely to be helpful, though it might be useful to people who understand the hardware.

16.9.3 Decoding Macro PCs

To decode the macrocode PC printed by the FEP command Show Status, warm boot or go to another machine running identical software and call the function **sys:%find-structure-header** on the number printed by the FEP. This is an octal number; use #o if necessary. It should return a compiled-function object, which is the function that was executing at the time. To find the exact place in the function that was executing, note the difference between the number printed by the FEP and the address in the printed representation of the compiled-function object. You can use **sys:%pointer-difference** to compute this difference. Multiply this by 2, and add 1 if the FEP said the PC was odd (not even). The result is the instruction number of the current instruction; disassemble the compiled function to see it.

Example:

```
FEP Command:  Show Status
...
3600 program counters:
  Macro PC/ (Odd)1244531
  ...
FEP Command: Start
...
(%find-structure-header #o1244531)
#<DTP-COMPILED-FUNCTION EQUAL 1244530>
(%pointer-difference #o1244531 *)
1
(1+ (* * 2))
3
(disassemble ***)
  0 ENTRY: 2 REQUIRED, 0 OPTIONAL
  1 PUSH-LOCAL FP|0            ;A
  2 PUSH-LOCAL FP|1            ;B
  3 BUILTIN EQL STACK
  ...
```

Instruction 3 (EQL) is the one that halted.

16.10 Debugging in the FEP

The release tapes include some files provided as an extra debugging aid. These files can be used to enter a debugging mode in the FEP. This mode is especially useful for problems that cause control to return to the FEP, making it impossible to use the debugging methods normally used in Lisp.

These files have names of the form: v*n*-debug.flod, where *n* is the FEP version number. The files should now reside on your sys host in the directory with the logical pathname SYS:N-FEP;V127-DEBUG FLOD, where 127 is the version of FEP software.

To use these files, you should copy the appropriate file to the FEP file system *before* you need to use it. To copy the file, first find out which version of FEP software is installed in your machine. You can do this by either using the Show Herald command in Lisp, or by typing the Show Version command at the FEP level. To copy this file to your FEP file system, use the Copy File command.

For example, if you are using FEP version 127 software, you would use the following command to copy the .flod file to the FEP file system:

```
Copy File sys:n-fep;v127-debug.flod fep0:>v127-debug.flod
```

The FLOD file cannot be used on any other FEP version; trying to use one on a different FEP version has no effect.

After you have copied the file to the FEP file system, you can enter the debugging mode by loading the file with the Load FEP command, as shown in the following example:

```
FEP Command: Load Fep >v127-debug.flod
```

This puts you into a debugging mode very similar to the Debugger in Lisp, whereby you can move up and down the stack to examine the state of the machine and determine the source of the problem. The `HELP` key lists the available commands.

One particularly useful command, when the machine has crashed during paging, is `c-m-S`. This command allows you to switch between the auxiliary stack (where paging code runs) and the normal stack (where user code runs). If the machine crashed while executing on the auxiliary stack, user stack frames will not be found until `c-m-S` is executed.

If you need to stop the execution of Lisp and give control to the FEP, use the Halt Machine command.

Halt Machine Command

Halt Machine

Halt Machine stops execution of Lisp and gives control to the FEP. You can now enter Fep commands, for example, to warm or cold boot the machine.

If you need to know the machine model, use the function **(si:machine-model)**.

si:machine-model *Function*

This function returns a keyword symbol designating the model number of the current 3600-family computer.

Possible return values are as follows:

:unknown The model number cannot be determined (usually indicating lack of some ID prom)

:/3600 or :|3600| (The keyword whose print-name is "3600".) The machine is a Symbolics 3600.

:/3670 The machine is a Symbolics 3670.

:/3675 The machine is a Symbolics 3675.

:/3640 The machine is a Symbolics 3640.

:/3645 The machine is a Symbolics 3645.

If you want to look at your machine's hardware configuration, use the Show Machine Configuration command.

Show Machine Configuration Command

Show Machine Configuration *host*

Shows the board-level hardware information about any 3600-family machine on the same network as your machine.

host The name of a 3600-family machine. The default is your machine.

This information is useful for service personnel. You might be asked for the machine serial number (in line 3) if you call Symbolics Software Support. The display from Show Machine Configuration looks like this:

```
:Show Machine Configuration (A host) acme-sling-shot
Chassis (PN 170219, Serial 230) in Chassis or NanoFEP:
    Manufactured on 1/10/85 as rev 1, functions as rev 1, ECO level 0
    Machine Serial Number: 4185
Datapath (PN 170032, Serial 1253):
    Manufactured on 9/20/83 as rev 3, functions as rev 3, ECO level 0
Sequencer (PN 170042, Serial 2576):
    Manufactured on 4/21/85 as rev 4, functions as rev 4, ECO level 0
Memory Control (PN 170052, Serial 1381) in Memory Control or IFU:
    Manufactured on 12/3/83 as rev 5, functions as rev 5, ECO level 0
Front End (PN 170062, Serial 2380) in FEP:
    Manufactured on 2/1/84 as rev 5, functions as rev 5, ECO level 0
512K Memory (PN 170002, Serial 1258) in LBus slot 00:
    Octal Base address: 0
    Manufactured on 3/2/84 as rev 2, functions as rev 2, ECO level 0
512K Memory (PN 170002, Serial 2572) in LBus slot 01:
    Octal Base address: 2000000
    Manufactured on 2/22/85 as rev 2, functions as rev 2, ECO level 0
512K Memory (PN 170002, Serial 140) in LBus slot 02:
    Octal Base address: 4000000
    Manufactured on 1/19/83 as rev 2, functions as rev 2, ECO level 0
IO (PN 170157, Serial 356) in LBus slot 03:
    Octal Base address: 6000000
    Manufactured on 9/22/84 as rev 6, functions as rev 6, ECO level 0
512K Memory (PN 170002, Serial 2932) in LBus slot 04:
    Octal Base address: 10000000
    Manufactured on 4/11/85 as rev 2, functions as rev 2, ECO level 0
FEP Paddle Card (PN 170069, Serial 943) in FEP -- PADDLE side:
    Manufactured on 3/21/85 as rev 1, functions as rev 1, ECO level 0
IO Paddle Card (PN 170245, Serial 3) in LBus slot 03 -- PADDLE side:
    Manufactured on 4/20/84 as rev 1, functions as rev 1, ECO level 0
    Ethernet Address: 08-00-05-03-18-00
    Monitor Type: Philips
```

Index