3. THE FILE SYSTEM

Introduction

The 1108 hard disk file system is designed to provide Interlisp-D users with a flexible mechanism for storing and accessing files. Like the file systems for the 1100 and the 1132, the 1108 file system supports such features as random access and version numbers on files. In addition, the 1108 local file system supports a hierarchical naming structure for files.

Disk Partitions

The hard disk used with the 1108 may be partitioned into up to 10 regions called *logical volumes*. Logical volumes are like directories on the disk device: they may be used to hold Interlisp virtual memories, Interlisp files, Mesa files, or Star files. You can partition the disk with the *Installation Utility* floppy or with *Othello*. Because partitioning the hard disk erases all its contents, you are advised to partition the disk appropriately before storing anything on it. Otherwise, you will have to off-load all files from the disk, repartition it, and then copy the files back to the disk.

Although an Interlisp virtual memory file can coexist on a logical volume with other files, it is generally advisable to give each virtual memory file a logical volume that it does not share with anything else. Otherwise, the resulting fragmentation adversely affects swapping performance. A logical volume intended to contain an Interlisp virtual memory should be between 8,000 disk pages (4 megabytes) and 64,000 pages (32 megabytes) long. The closer to 64,000 pages you can make your virtual memory volume the better, provided there is still enough

space on the disk for any other volumes you might need.

The Intermezzo file system (unlike its predecessors) allows Interlisp user files to coexist on a logical volume with Mesa or Star files. So it is no longer necessary to have a special logical volume given over to Lisp user files, though you can still have one if you like. Note that to store Interlisp files on a logical volume, you must create a Lisp directory on that volume (see below for instructions).

File System Utility Functions

So long as there is a logical volume with a Lisp directory on it, you will have a local disk device called {DSK}. This device can be used from within Interlisp-D just like the {DSK} device on the 1100 and the 1132, except that it supports a hierarchical naming structure for files.

If you do not have a logical volume with a Lisp directory on it, Interlisp emulates the {DSK} device by a core device. (A core device is a file device whose backing store is entirely within the Lisp virtual memory.) However, this causes three problems. (a) The core device provides limited scratch space for some system programs; (b) when running GREET, Interlisp fails to find {DSK}Init.Lisp and has to prompt you for an init file; and (c) since the core device is contained in virtual memory, it (and the files stored on it) can last only as long as you keep your virtual memory image.

To create a Lisp user file directory on a logical volume, call

(CREATEDSKDIRECTORY *VolumeName*) [Function]

CREATEDSKDIRECTORY affects only the specified volume, and (unlike previous versions of the file system) does not affect Mesa or Star files in the

specified volume. CREATEDSKDIRECTORY returns the name of the directory created. You should install an Interlisp directory only the first time the logical volume is used. After that, the system automatically recognizes and opens access to the logical volumes that have Interlisp directories on them.

Should you ever want to get rid of a Lisp directory (and all the files in it), call

(PURGEDSKDIRECTORY *VolumeName*) [Function]

PURGEDSKDIRECTORY affects only the Lisp files on the specified volume; it does not tamper with Mesa or Star files. An alternative but cruder way to get rid of a Lisp directory is to use *Othello* or the *Installation Utility* to Erase the entire logical volume.

To find out if a particular logical volume already has a Lisp directory on it, call

(LISPDIRECTORYP *VolumeName*) [Function]

To find out what logical volumes you have on your local disk, call

(VOLUMES) [Function]

To find out the total size of a logical volume in disk pages, call

(VOLUMESIZE *VolumeName*) [Function]

To find out the number of free pages left on a volume, call

(DISKFREEPAGES *VolumeName Recompute*)
[Function]

And to find out which logical volume contains the virtual memory you are currently running in, call

(DISKPARTITION) [Function]

There is a display window that can keep track of the information provided by LISPDIRECTORYP, VOLUMES, VOLUMESIZE, and DISKFREEPAGES.

The display window can be in one of three states: ON, OFF, or CLOSED. ON means the display window is updated whenever the file system state changes. (This continuous updating can slow down the file system significantly.) OFF means that the display window is open, but updated only when you left-button it with the mouse. CLOSED means that the display window is closed and never updated. The display mode is initially set to CLOSED. To change the state of the display, call

(DSKDISPLAY *NewState*) [Function]

DSKDISPLAY returns the old state of the file system display, and if *NewState* is one of the litatoms ON, OFF, or CLOSED, then the display state is changed to *NewState*. Once the display window is open, you can update it or change its state with the mouse. Left-buttoning the display window updates it, and middle-buttoning the window brings up a menu that allows you to change the display state.

Finally, once an Interlisp directory has been installed on a logical volume, any program running in Lisp has access to the Lisp files on the volume. Access is provided through the usual device-independent file interface: CONN (to connect to any directory or subdirectory on the local disk), OPENSTREAM, CLOSEF, DELFILE, GETFILEINFO, SETFILEINFO, BIN, BOUT, LOAD, etc.

File Name Conventions

Each logical volume with a Lisp directory on it serves as a directory of the device {DSK}. Files are referred to as

{DSK} *<LogicalVolumeName>FileName*

Thus the file Init.Lisp on the volume LispFiles would be called {DSK}<LispFiles>Init.Lisp.

In addition, you can indicate subdirectories using the > character in file names to delimit subdirectory names. Subdirectories allow you to group files to a finer degree of granularity. Files with subdirectories are written

{DSK} <*LogicalVolumeName*>*SubDir1*>...>*SubDirN*> *FileName*

For example, suppose you had a file LRdesign.TEdit on the subdirectory ParserGenerator on the subdirectory Compiler on the directory (logical volume) LispFiles of the hard disk device. Its name would be written {DSK}<LispFiles>Compiler>ParserGenerator>LRDesign.TEdit.

You can default directory names for the 1108 hard disk in an unusual but simple way. That is: if the file does not have a subdirectory and you leave out the directory (logical volume) name, the directory defaults to the next logical volume that has a Lisp directory on it, including or after the volume containing the currently running virtual memory. Thus if your disk has the logical volumes Lisp, Tajo, and LispFiles, and the Lisp volume contains the running virtual memory, and only the LispFiles volume has a Lisp directory on it, then {DSK}Init.Lisp refers to the file {DSK}<LispFiles>Init.Lisp. All the utility functions presented above default logical volume names in a similar way, except for those that can't, such as CREATEDSKDIRECTORY. If you want to find out what the default Lisp directory is, call

(DIRECTORYNAME '{DSK})

This defaulting convention is necessitated by several parts of the Interlisp system that create scratch files on the device {DSK} without specifying a directory (logical volume).

Disk Scavenging

Unlike previous releases, Intermezzo provides full disk-scavenging service to guard against the unlikely event of file system failure. There are two classes of file system failure: Lisp directory failure and lower-level (Pilot) failure. Scavenging service for Lisp directories is provided by SCAVENGEDSKDIRECTORY; scavenging for Pilot is provided by either *Othello* or the *Installation Utility*.

Pilot failures manifest themselves as "HARD DISK ERROR" breaks within Lisp. To fix such a failure, return to the top level, log out of Lisp, get into either *Othello* or the *Installation Utility*, use the Scavenge option on the logical volume in question, and then reboot Lisp.

Lisp directory failures show up as infinite looping or other aberrant behavior while the system is doing a directory search or enumeration. To repair the directory, return to the top level and call

(SCAVENGEDSKDIRECTORY *VolumeName*)
[Function]

If you have any doubt about which logical volumes to scavenge, scavenge them all. If you are not sure which scavenger to use on a volume, use them both, *Othello* (or the *Installation Utility*) first, then run SCAVENGEDSKDIRECTORY. Neither scavenger harms an intact volume.