LocalFile
1108 Local Hard Disk File System
Intermezzo Release

Stored as [eris]<lisp>intermezzo>doc>localFile.tedit

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 1132, the 1108 file system supports features like random access and version numbers on files. In addition the 1108 local file system supports a hierarchical naming structures for files.

Partitioning the Disk

The hard disk used with an 1108 may be partitioned into up to ten 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*. Since 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 offload all files from the disk, repartition, and then copy the files back to the disk.

Although an Interlisp virtual memory file could coexist on a logical volume with other files, it is generally adviseable to give each virtual memory file a logical volume which it does not share with anything else. Otherwise the resulting fragmentation would adversely affect swapping performance. A logical volume intended to contain an Interlisp virtual memory should be between 16,000 and 64,000 disk pages long.

The Intermezzo file system (unlike its predecessors) allows Interlisp user files to coexist on a logical volume that contains Mesa or Star files. So it is no longer necessary to have a special logical volume given over to Lisp user files, though you still can 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 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 will emulate the {DSK} device by a coredevice, which is fine except: (a) The coredevice provides limited scratch space for some system programs; (b) when running GREET, Interlisp will fail to find {DSK}INIT.LISP and will have to prompt the user for an init file; and (c) since the coredevice 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 the function

(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. Installing an Interlisp directory is something that should have to be done only the first time the logical volume is used. After that, the system will automatically recognize and open 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 the function

(PURGEDSKDIRECTORY volumeName) [function]

PURGEDSKDIRECTORY affects only the Lisp files on the specified volume; it will not tamper with Mesa or Star files on the same volume. 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 the function

(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 the function

(DISKPARTITION) [function]

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 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}<*logical-volume-name*>*file-name*

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 users to group files to a finer degree of granularity. Files with subdirectories are written

{DSK}<*logical-volume-name*>*subdir1*>...>*subdirN*>*file-name*

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 will default to the next logical volume which has a Lisp directory on it including or after the volume containing the currently-running virtual memory*. Thus if your disk has 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 will refer 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 which create scratch files on the device {DSK} without specifying a directory (logical volume).

Displaying File System State

DSKDISPLAY is a library package which provides a display window for 1108 local file system. The window keeps track of what logical volumes you have on your local disk, which ones have valid Lisp directories on them, and how much space is left on each volume.

The file system display 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 the user left-buttons it with the mouse. CLOSED means that the display window is closed and never updated. When the DSKDISPLAY package is loaded, the display mode is set to CLOSED.

The functional interface to the file system display is provided by

(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 will be changed to newState.

The mouse interface to the file system display is as follows: left-buttoning the display window will update it, and middle-buttoning the window will bring up a menu which allows you to change the display state.

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, or lower-level (Pilot) failure. Scavenging service for Lisp directories is provided by the library package SCAVENGEDSKDIRECTORY; scavenging for Pilot is provided by either *Othello* or the *Installation Utility*.

Pilot failures manifest themselves as a "HARD DISK ERROR" break within Lisp. To fix such a failure, return to 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 aberrent behavior while doing a directory search or enumeration. To repair the directory, return to top level, load the package SCAVENGEDSKDIRECTORY, and call

(SCAVENGEDSKDIRECTORY volumeName) [function]

Should you have any doubt which logical volumes to scavenge, scavenge them all. Should you not be sure which scavenger to use on a volume, use them both, Pilot first, then Lisp directory. Neither scavenger will harm an intact volume.