5. MEDLEY FILE SYSTEMS

This chapter discusses the conventions for using files from Medley.

File Naming Conventions

In Lisp, a file name (pathname) consists of a collection of fields: the *host*, *directory*, *name*, *extension* and *version*. These fields are optional. The standard Lisp syntax for these fields is:

{*host*}<*directory*>*name.extension;version*

The *directory* field can be a directory path consisting of a sequence of directory and subdirectory components. Slashes (/) and right angle brackets (>) can be used to delimit a directory name; there is no distinction made between them. Square brackets ([]) are not acceptable as directory delimiters.

Duplicated directory delimiters are treated as a single delimiter. Thus, the following two file names specify the same file:

```
{DSK}<LISP>USERS>FOO.;1
{DSK}</LISP/USERS/>FOO.;1
```

Hosts that Medley Supports

{CORE}	Creates "files" in memory; useful for quick temporary files
{LPT}	Creates files that are automatically sent to your printer
{NULL}	Creates a file that does nothing
{DSK} and {UNIX}	Give you access to the Sun's file systems; the rest of the chapter concentrates on them.

The above hosts are described in more detail in the IRM.

Using SunOS Files from Medley

You can access any mounted SunOS file system directly from Lisp. The mounted file system is available as an I/O device of the Lisp environment. This file system appears as the local disk of Lisp, even though it may be a remotely mounted file system of networked Sun file servers.

Many of the file devices to which the Medley environment can talk, including PUP, XNS file servers, the {CORE} device, and others, have facilities that are not directly supported by SunOS. For example, many file systems have file version numbers and case insensitive file search conventions.

Medley on the Sun Workstation has two distinct "host" names that can be used to access the SunOS file system. These host names are provided for compatibility with existing applications and tools. They also simultaneously allow natural interaction with the SunOS file system. The names are:

{DSK} On the Xerox workstation, {DSK} gave you access to your local hard disk; to use {DSK}, you had to create a directory on each disk partition you wanted to use. On the Sun Workstation, in contrast, the {DSK} device lets you access the file system using similar conventions to those used for {DSK} on the Xerox workstation local disk devices. In particular, {DSK} files have version numbers; {DSK} file name recognition also ignores the case of letters.

{UNIX} The {UNIX} device lets you use the mounted file systems with the normal naming conventions of the SunOS file system. {UNIX} files do not have version numbers, and the file name recognition treats lowercase letters as distinct from their uppercase equivalents.

File streams can be opened or closed on both devices. The reason for having both devices is to more easily support the running of applications that were originally developed on a Xerox workstation, while still allowing new applications to interact more naturally with UNIX.

NOTE: Both {DSK} and {UNIX} work as filters. They act as pointers to a device. On 11xxs, {DSK}foo is the same as {DSK}<lispfiles>foo. On the Sun, {DSK}foo is the same as $HOME/foo ~*user*/foo.

Common {DSK} and {UNIX} Naming Conventions

- To include a special character (e.g., > or ;) in a file name, precede it with a single quote (’). To include a single quote in a file name, precede it with another single quote. You can quote any of these characters: <, >, ;, ~, and a period (.). The following examples show how the single quote notation on {DSK} and {UNIX} is used.

{DSK} Name From Lisp	**File Name From SunOS**
foo’>bar.baz;1	foo>bar.baz
foo’;bar.baz;1	foo;bar.baz
foo’’bar.baz;1	foo’bar.baz

- {DSK} and {UNIX} do not allow you to use either the slash (/) or the NUL character in file names. Thus, you cannot name files containing these characters.

- Both {DSK} and {UNIX} can handle the following characters, which were defined as special characters in Medley Release 1.1: backslash (\) and tilde (~).

- {DSK} and {UNIX} can distinguish between a file name with a period at the end (e.g., foo.) and a simple file name (e.g., foo). The final period is preceded with a single quote, as shown in the following example:

{DSK} Name From Lisp	**File Name From SunOS**
foo.;1	foo
foo’..;1	foo

- On {DSK} and {UNIX}, the C-Shell and SunOS directory notations (~, ., and ..) are supported in the Lisp directory specification. The tilde character (~) is allowed at the very beginning of the directory specification of a pathname. A combination of relative path specifiers (~, ., ..) is supported. The tilde character corresponds to the

user's home directory at login. The period (.) corresponds to the current working directory. Two periods (..) indicates the parent of the current working directory.

- File names are returned by the system (e.g., INFILEP) in more canonical form. The function which returns the full file name returns it in the canonical form , as in {DSK}<usr>etc> rather than {DSK}/usr/etc/. This change will make some tools which depend on the conventional file name representation described in the *Interlisp-D Reference Manual* work correctly on the Medley file system (e.g., COPYFILES).

{DSK} Naming Conventions

The {DSK} device performs the following file name transformation when actually accessing the SunOS file system:

- Mixed case letters are read as such.
- File name searches are done case-sensitive first; if a match is not found, the system does a case-insensitive search.
- The left angle bracket character (<) is translated to a slash (/), the delimiter for the root directory.
- {DSK} supports relative pathnames. You can specify relative pathnames by omitting a slash (/) or left angle bracket (<) as the first character in the directory field. For example:

  {DSK}foo.fee and {DSK}~/foo.fee are relative to the user's UNIX home directory (~/foo.fee).

  {DSK}./foo.fee is relative to the user's current working directory (SunOS./foo.fee).

  {DSK}../foo.fee is relative to the parent directory of the user's current UNIX working directory (../foo.fee).

The Medley 2.0-S {DSK} device supports the notation in which the three meta characters (', .., and ~) are used together, as shown in the following example:

```
{DSK}~/../tom/foo.c
```

In this example, the {DSK} device interprets tom as one of the subdirectories of the parent directory of the user's home directory.

{DSK} also supports the tilde-name (~name) convention. {DSK} interprets {DSK}~tom/foo.c as a file named foo on tom's home directory. In this notation, the user name is case-sensitive (e.g., ~tom and ~Tom are treated as different users).

Version Numbering

The UNIX file system does not support version numbers in file names; {DSK} emulates versions with a naming convention. (GNU Emacs also uses this convention.) This section explains how {DSK} version numbers are represented in the SunOS file system.

- When you create a completely new file, it appears in the SunOS file system without a version number.

{DSK} Name From Lisp	**File Name From SunOS**

`bar.baz;1`	`bar.baz`

- When you create (from Medley) a file with a version other than 1, Medley adds version numbers to that file name, as a trailing number between tildes, e.g., "`myfile.~12~`" for the twelfth version of myfile.

  The following shows some examples of equivalent file names in Lisp and SunOS.

{DSK} Name From Lisp	**File Name From SunOS**
`bar.baz;1`	`bar.baz.~1~`
`bar.baz;2`	`bar.baz.~2~`
`bar.;23`	`bar.~23~`

- Medley always maintains a versionless file which is hard-linked to the highest extant version of the file (i.e., they are two names for the very same file). This file name does not appear in the {DSK} directory listing.

From {DSK}	**From SunOS**
`foo.c;15`	`foo.c` (hard linked with `foo.c.~23~`)
`foo.c;23`	`foo.c.~15~`
	`foo.c.~23~`

  Similarly, a file created in UNIX with no version number is treated by {DSK} as the highest version.

- When you create a new version of a file, the versionless–file link is broken, and the versionless file is hard–linked to the new highest version.

From {DSK}	**From SunOS**
`foo.c;15`	`foo.c` (hard linked with `foo.c.~24~`)
`foo.c;22`	`foo.c.~15~`
`foo.c;24` (new file)	`foo.c.~22~` (no link with `foo.c`)
	`foo.c.~24~` (new file, link from `foo.c`)

- When you delete the highest version of a file, the versionless file is also deleted. If any older versions of the file remain, a new link is created from the versionless name to the highest version extant. For example, if you have the files

From {DSK}	**From SunOS**
`foo.c;1`	`foo.c` (linked to `foo.c.~2~`)
`foo.c;2`	`foo.c.~1~`
	`foo.c.~2~`

  and you delete `foo.c;2` from {DSK}, the resulting files are:

From {DSK}	**From SunOS**
`foo.c;1`	`foo.c` (linked to `foo.c.~1~`)
	`foo.c.~1~`

- When you rename a file, it works the same as deleting the file under the old name then creating it under the new name. For example, if you have the following {DSK} files

From {DSK}	**From SunOS**
`foo.c;1`	`foo.c` (linked to `foo.c.~2~`)
`foo.c;2`	`foo.c.~1~`
`fee.c;1`	`foo.c.~2~`
`fee.c;2`	`fee.c` (linked to `fee.c.~2~`)

```
fee.c.~1~
fee.c.~2~
```

and you rename "foo.c" to "fee.c", your renamed {DSK} files and the linked SunOS files would appear as:

From {DSK}	From SunOS
foo.c;1	foo.c (linked to foo.c.~1~)
fee.c;1	foo.c.~1~
fee.c;2	fee.c (linked to fee.c.~3~)
fee.c;3	fee.c.~1~
	fee.c.~2~
	fee.c.~3~ (renamed file)

- When a file has a name suffix that is not a valid version number (e.g., myfile.~12x~), the suffix is regarded as part of the file name.

From {DSK}	From SunOS
myfile.~12x~;1	myfile.~12x~

Pathnames

A pathname on {DSK} is always case insensitive. When the user specifies a file, the {DSK} device handler first searches for the file with the specified name. If no such file is found, it then searches for a file with the same spelling but different case.

Most Lisp functions, such as FINDFILE and INFILEP,which return pathnames return them with the original case when they are applied on files on {DSK} when IL:*DSK-UPPER-CASE-FILE-NAMES* is NIL. If IL:*UPPER-CASE-FILE-NAMES* is not NIL, these functions return only uppercase pathnames. The only exception is the function DIRECTORY , which returns a list of pathnames. The case of the pathnames is controlled by the global variable IL:*UPPER-CASE-FILE-NAMES* in a similar manner to IL:*DSK-UPPER-CASE-FILE-NAMES*.

If a pathname on {DSK} has no directory specification, a tilde-slash combination (~/) is used, i.e., the Lisp directory specification {DSK}foo is the equivalent of {UNIX}~/foo.

{UNIX} Naming Conventions

For the {UNIX} device, file name translation takes place only on the directory. An initial left angle bracket (<) is treated as if it were an initial slash (/); both signify a path relative to the SunOS file system root directory; if there is no initial left angle bracket or slash, the directory is relative to the current working directory. Initially this is the working directory where Lisp was started; you can change it using the CHDIR function, described below. Tilde (~) is translated to the user's home directory.

For example, {UNIX}myfile/abc means the file abc on the ./myfile directory.

The {UNIX} device does not recognize version numbers, does not return them, and ignores them for recognition.

No case translation or recognition is done; upper- and lowercase letters are treated as distinct.

Examples:

{UNIX} Name From Lisp	File Name From SunOS
<foo>fee>bar.baz;1	/foo/fee/bar.baz;1
<foo>fee/bar.;1	/foo/fee/bar.;1
<foo/fee>	/foo/fee/
</foo/fee/>	/foo/fee/
/foo/fee/bar.~1~	/foo/fee/bar.~1~
/foo/fee/	/foo/fee/

In the first two examples the `;1` is treated as part of the file name, not the version number. In the last two examples that translation is not done.

Directories

In places where Lisp expects a directory name, {UNIX} paths must end with a slash (/).

Directory Enumeration

You cannot use the wildcard character, asterisk (*), in subdirectories for either {DSK} or {UNIX} devices. For example

```
>(DIRECTORY '{DSK}/users/x*/foo)
NIL
```

Enumeration of files in directories differs between {DSK} and {UNIX} devices. On the {DSK} device, a versionless file which has a link to the highest version file is not enumerated in a directory.

On the {UNIX} device, all files are enumerated in a directory. For instance, if the following SunOS files linked with `foo.c.~2~` exist

```
foo.c
foo.c.~1~
foo.c.~2~
```

the {DSK} directory enumeration would look like this:

```
>(DIRECTORY '{DSK}/users/venue/*)
({DSK}/users/venue/foo.c;1
{DSK}/users/venue/foo.c;2)
```

The {UNIX} directory enumeration, on the other hand, would look like this:

```
>(DIRECTORY '{UNIX}/users/venue/*)
({UNIX}/users/venue/foo.c
{UNIX}/users/venue/foo.c.~1~
{UNIX}/users/venue/foo.c.~2~)
```

Directory Creation

{DSK} When you write a new file on {DSK}, if the directory named in a pathname does not exist, the {DSK} device handler creates the directory automatically. This feature is provided for compatibility with other Interlisp-D implementations.

If you try to "connect" to a nonexistent directory (using the CONN Exec command or the function IL:CNDIR), Medley returns the message

```
Nonexistent directory
```

{UNIX} The {UNIX} device does not support such directory creation. An attempt to create a file on a nonexistent directory results in an error.

```
{UNIX}/users/venue/foo.c.~2~)
```

Directory Deletion

Neither {UNIX} nor {DSK} support automatic directory deletion. To delete a directory you must use the SunOS C-Shell command rmdir.

Open File Limit

The number of simultaneously open {DSK} and {UNIX} files must fall within the SunOS limits for a process. For OS 3.4, this number of open files may be configured, with 30 as the maximum permissible number of open files per process. This means that it is not possible to have more than 30 files open for a process, minus whatever files Medley has open for its own use, at any one time in the Medley system. If you try to open too many files, the system call error number 24, Too many open files, appears in the prompt window.

For OS 4.0, the maximum number of files/processes that can be open at one time is 64, unless your kernel is configured otherwise.

Default Pathname

If no path is given, the {DSK} device defaults to the user's home directory, tilde-slash (~/). The {UNIX} device defaults to the current working directory. This current working directory can be changed with the CHDIR function. The current working directory is also used to resolve the interpretation of the period (.) and double period (..) specifications at the beginning of a {DSK} pathname.

(IL:CHDIR *PATHNAME***)** **[Function]**

Changes the current working directory for the current invocation of Lisp. For example,

```
(CHDIR "{DSK}~/subdir/")
(OPENSTREAM "{DSK}./foo" ...)
```

opens the SunOS file ~/subdir/foo.

When *PATHNAME* does not end with a slash (/) or right angle bracket (>), the whole *PATHNAME* is treated as a directory name:

```
(CHDIR '{DSK}<users>local>)
> "{DSK}<users>local>"
(CHDIR '{UNIX} /usr/local)
> "{UNIX}<usr>local>"
```

If *PATHNAME* is NIL, CHDIR tries to change the current working directory to the current connected directory. If the directory is connected to devices other than {DSK} or {UNIX}, the error message

```
Bad Host Name
```

appears, followed by the host name of the current connected directory.

If the *PATHNAME* does not exist, the error message

```
No-Such-Directory
```

appears followed by the system echo of the pathname.

File Attributes

This section describes how the various file attributes are treated by Lisp on the Sun Workstation and what they translate to in SunOS.

GETFILEINFO obtains file attributes and SETFILEINFO sets the attributes.

WRITEDATE and CREATIONDATE **[File Attributes]**

Resets the date to the current time whenever the contents of a file are modified. This only works for the owner of the file. Since UNIX does not naturally support more than one date for file modification, the WRITEDATE and CREATIONDATE are treated identically by Lisp functions OPENSTREAM, OPENFILE, GETFILEINFO, and by the {DSK} and {UNIX} devices.

TYPE **[File Attribute]**

Sets the TYPE property of files; normally either TEXT or BINARY. However, UNIX does not distinguish between TEXT and BINARY files. Normally, programs will infer the type by the file extension, using the Lisp variables DEFAULTFILETYPE and DEFAULTFILETYPELIST. This is the convention used by Medley. If no file extension is given, the value in DEFAULTFILETYPE is used. SETFILEINFO cannot change the TYPE attribute.

EOL **[File Attribute]**

Returns the end–of–line convention. Both the {DSK} device and **{**UNIX**}** use line feed (LF) as the default EOL convention for text. The EOL for binary files is carriage return (CR). EOL uses the TYPE property of files. (The TYPE property of a file depends on the file extension and the DEFAULTFILETYPE and DEFAULTFILETYPELIST variables). If the TYPE property of a file is TEXT, LF (=10) is used as EOL. If the TYPE property of a file is BINARY, CR (=13) is used as EOL.

NOTE: EOL conventions on {DSK} are not compatible with those on Xerox workstations.

AUTHOR **[File Attribute]**

Returns the author of the file, i.e., the login name of the user who created it. This attribute cannot be changed.

PROTECTION **[File Attribute]**

Returns file protection attributes. The file protection attributes of files under the SunOS cannot be directly manipulated from inside Lisp. It is necessary to use the UNIX chmod command to change file protection bits.

SIZE **[File Attribute]**

Returns the file size. For compatibility with other Lisp environments running on Xerox workstations, the SIZE attribute is computed as the length of the file (in bytes) divided by 512 (rounded up).

NOTE: SETFILEINFO lets you change the SIZE attribute of I/O streams and output streams. However, a file cannot be expanded this way.

File Variables

This section discusses how certain file variables are used by Medley in SunOS.

IL:FileTypeConfirmFlg **[Variable]**

The file–type attribute of a file on {DSK} or {UNIX} is decided from its extension, DEFAULTFILETYPELIST and DEFAULTFILETYPE. Extensions of binary files should be registered in DEFAULTFILETYPELIST. When this rule is broken, a hardcopy of files on {DSK} and {UNIX} may confuse the printers. So when you try to hardcopy a file whose extension is not registered in DEFAULTFILETYPELIST, a menu is invoked to confirm the file type. Text or binary can be selected. The invocation of this menu can be stopped by setting IL:FileTypeConfirmFlg to NIL. The default value of IL:FileTypeConfirmFlg is T.

When extensions of binary files are not registered in DEFAULTFILETYPELIST, copy or rename from a DSK/UNIX device to a non–DSK/UNIX device also may cause file type confusion. This type of copy or rename results in one of the following warning messages in the prompt window, as appropriate:

> **Extension of {DSK}foo.fee;1 isn't in DEFAULTFILETYPELIST. {CORE}foo.fee;1 was copied as TEXT.**
>
> **This message can be stopped by set FileTypeConfirmFlg to NIL.**
>
> **Extension of {DSK}foo.fee;1 isn't in DEFAULTFILETYPELIST. {CORE}foo.fee;1 was renamed as TEXT.**
>
> **This message can be stopped by set FileTypeConfirmFlg to NIL.**

Either of these messages can be stopped by setting IL:FileTypeConfirmFlg to NIL.

IL:DEFAULTFILETYPE **[Variable]**

Initially set to TEXT. Used with the file attribute TYPE.

DEFAULTFILETYPELIST **[Variable]**

A list of accepted file types. Initially set to ((NIL . TEXT)(C . TEXT)(H . TEXT) (LISP . TEXT)(LSP . TEXT)(O . BINARY)(OUT . BINARY)(LCOM . BINARY) (DFASL . BINARY)(DCOM . BINARY)(SKETCH . BINARY) (TEDIT . BINARY)(DISPLAYFONT . BINARY)(WD . BINARY)(IP . BINARY) (RST . BINARY)(BIN . BINARY)(MAIL . BINARY)(SYSOUT . BINARY))

Used with the file attribute TYPE. Binary files, such as Sketch files, InterPress files, or Press files, should have their extensions registered in DEFAULTFILETYPELIST. This is especially important because UNIX does not support file types.

File System Errors

Several types of errors may occur in the Medley file system.

When a remotely mounted file system or NFS service is down, or when network traffic is heavy, any attempt to access a file on that file system results in an error. The following error message is printed in the prompt window:

```
File access timed out
```

Medley will wait until the file system responds or until a timeout occurs. If the file system is mounted with the "hard" option, the timeout is controlled by the value of environment variable `LDEFILETIMEOUT`. If the file system is mounted with the "soft" option, the timeout depends on the NFS file system timeout time, and the value of `LDEFILETIMEOUT`. Medley will wait until the shorter of these two times is exceeded. The NFS file system timeout time, retry times, etc., are controlled by the UNIX command **mount**.

If `LDEFILETIMEOUT` is not set, the default value of 10 seconds is used. The variable is inspected at boot time, and a setting between 1 and 100 seconds is appropriate in most cases.

The following error messages may appear when there are Medley file errors:

```
Not owner
Device error:
Protection-violation
File-won't-open
Too-Many-Files-Open
File too large
File-System-Resources-Exceeded
Connectiom timed out
No-Such-Directory
Bad Host Name
FS-RENAMEFILE-SOURCE-COULDNT-DELETE
```

Another type of error occurs when the user has insufficient access to files. When this happens, Medley will print the following message:

```
File not found
```

The following message then appears in the SunOS prompt window:

```
System call error:  open errno=13  Permission denied
```

See the *UNIX Interface Reference Manual*, Intro (2), for descriptions of all OS system call messages.

[This page intentionally left blank]