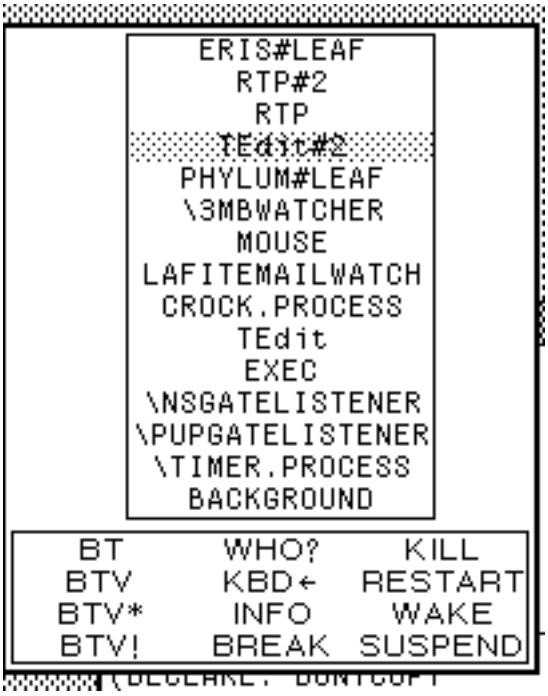
F.G.H.
3/26/85

LispCourse #16: Files and Directories ž Part 2

Update on Processes

When dealing with processes, I neglected to mention the following situation: For each formatting menu that is used in TEdit, a new TEdit process is started. Thus if you have one TEdit running with the Paragraph-Looks Menu open above that window, you will see 2 Tedit processes running in your PSW. The process called *TEdit* will refer to the TEdit itself, and the process called *TEDIT#2* will refer to the process operating the Paragraph Menu. This situation can sometimes be confusing!!



Dealing with Directories

The *Connected* Directory

At any given time you are *connected* to some device and directory. Connecting to a device and directory means that that device/directory is used as the default whenever a file name is specified with no device and/or directory.

> Whenever a file name is specified without a device and path name, the device/pathname is assumed to be the current *connected directory*.

Essentially, the connected directory is appended to the beginning of every file name that you type in without a device and path name.

If a file name is specified with a path name but without a device, then the device part of the current connected directory is used as the device.

Examples:

Connected Directory: *{Phylum}<halasz>lisp>*

File Name: *init == {Phylum}<halasz>lisp>init*

File Name: *new>init == {Phylum}<halasz>lisp>new>init*

File Name: *<jones>lisp>init == {Phylum}<jones>lisp>init*

File Name: *{Eris}<jones>lisp>init == {Eris}<jones>lisp>init*

Form: *(DELFILE 'ABC) ==*
(DELFILE '{Phylum}<halasz>lisp>ABC)

Form: *(DELFILE '{Eris}<jones>lisp>init) ==*
(DELFILE '{Eris}<jones>lisp>init)

LOGINHOST/DIR

The value of the variable LOGINHOST/DIR is the default connected directory. GREET sets your connected directory to this value (e.g., whenever you load a new sysout). CONN and CNDIR also use this variable (see below).

Usually LOGINHOST/DIR is set in your INIT file to your "home" directory. If not set in your INIT file, it will default to {DSK}.

Connecting to a directory:

There are two equivalent ways to change the current connected directory**:**

CONN *device/pathname* ž a P.A. command to change the currently connect directory. *device/pathname* is the directory to connect to.

(CNDIR *device/pathname*) ž a function to change the current connected directory. *device/pathname* is the directory to connect to. CNDIR returns the full path name of the directory being connected to.

Examples:

10_ CONN {phylum}<halasz>

{phylum}<halasz>

11_ (CNDIR '{phylum}<halasz>lisp>)

{phylum}<halasz>lisp>

12_ (CNDIR '<jones>lisp>)

{phylum}<jones>lisp>

Notes:

1. If *device/pathname* is NIL, the value of the variable LOGINHOST/DIR will be used.

2. If pathname is NIL but there is a device specified, then for any device that supports directories, the directory will be set to the user's login name.

Examples:

CONN {phylum} => CONN {phylum}<halasz>

CONN {dsk} => CONN {dsk}

3. CONN is UNDOable, but CNDIR is not.

4. As described above, certain devices require that a directory be created before you can connect to it. Other devices further require that all of the sub-directories in a path name exist before you can connect to the directory specified by that path name.

Examples:

CONN {phylum}<foo> == error

CONN {phylum}<halasz>foo == okay

CONN {FLOPPY}<FOO> == okay

Asking "what is the current connected directory?"

The function **DIRECTORYNAME** can be used to find out about LOGINHOST/DIR and about the current connect directory.

(DIRECTORYNAME) returns the value of LOGINHOST/DIR

(DIRECTORYNAME T) returns the current connected directory.

The DIRECTORIES List

Whenever you specify a file name that can't be found, DWIM tries to "correct" the spelling of the name.

If there is no device/pathname specified and the file does not exist in the currently connected directory, then DWIM will consult the value of the variable DIRECTORIES.

DIRECTORIES is a ***ordered*** list of device/pathnames for DWIM to look on for any file that it cannot find. DWIM will "temporarily connect" to each device/pathname on this list until it finds one that contains the file name it is looking for.

Example:

14_ (SETQ DIRECTORIES '({phylum}<halasz> {phylum}<halasz>lisp> {phylum}<notecards> {eris}<lispusers>))

({phylum}<halasz> {phylum}<halasz>lisp> {phylum}<notecards> {eris}<lispusers>)

15_ CONN {eris}<lisp>

{eris}<lisp>

16_ (TEDIT 'INIT)

={PHYLUM}<HALASZ>LISP>INIT

{process}#6,24304

[Since there was no file called INIT on the connected directory, {eris}<lisp>, DWIM tried to find the file. Using the DIRECTORIES list, it first looked for the file {phylum}<halasz>init. Since that file doesn't exists, it looked for the file {PHYLUM}<HALASZ>LISP>INIT, which does exist. So it used that as the "corrected" file name.

17_ (TEDIT '{ERIS}<lisp>INIT]

FILE NOT FOUND

{ERIS}<lisp>INIT

[DWIM did not use the DIRECTORIES list here because the Device/Pathname was already specified.]

DIRECTORIES should be set in your INIT file to a list of the directories where you typically keep files of general interest. The DIRECTORIES list is initialized in the system INIT file to include directories like *{eris}<lispusers>* and *{eris}<lisp>harmony>library>*. Therefore, your INIT file should use the ADDVARS file package command.

My INIT file contains the following clause:

```
(ADDVARS
    (DIRECTORIES
        {DSK}
        {DSK2}
        {PHYLUM}<HALASZ>LISP>
        {PHYLUM}<HALASZ>
        {PHYLUM}<NOTECARDS>RELEASE1.2>LIBRARY>
        {PHYLUM}<NOTECARDS>RELEASE1.1>))
```

Finding out what files are in a directory ž DIR, FILDIR etc.

One of the most common operations in using files is finding out what files you have in a given directory. For example, "what files do I have on {phylum}<halasz>?".

The functions FILDIR, DIR, and DIRECTORY can be used to get this information.

(FILDIR *FileNamePattern*) ž a function that returns a list of all of the full file names matching the the *FileNamePattern.* The *FileNamePattern* can be any part of a full file name. It may contain the * and ? wildcards, standing for *any number of any character* and *any one character*, respectively.

The parts of the full file name are defaulted as follows:

If the device name is left out, the device name in the current connected directory will be used.

If the device/pathname is left out, then the current connect directory will be used.

If no file name is specified, then the name is assumed to be *.*;*.

If the extension is left off, then the extension and version is assumed to be *;*. For example, *FOO == FOO.*;**

If the version is left off, then the version is assumed to be *. For example, *FOO.BAR == FOO.BAR;**

Examples:

(FILDIR) == all files in the connected directory

(FILDIR '.DCOM)* == all files in the connected directory that have a DCOM extension

(FILDIR '{ERIS}<LISP>LIBRARY>) == all files on {eris}<lisp>library>

(FILDIR '{phylum}<notecards>library>.DCOM)* == all files in {phylum}<notecards>library> that have a DCOM extension.

*(FILDIR '{phylum}<halasz>*outline*)* == all files on {phylum}<halasz> and its subdirectories that contain the string "outline".

Result is:

```
({PHYLUM}<HALASZ>LISPCOURSE>OUTLINE.FORM;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE01.TED;7
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE01.TED;8
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE01.TED;9
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE02.TED;2
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE02.TED;3
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE02.TED;4
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE03.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE04.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE04.TED;2
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE05.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE06.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE07.TED;13
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE07.TED;14
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE07.TED;15
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE08.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE09.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE10.TED;3
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE10.TED;4
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE10.TED;5
```

```
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE11.TED;7
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE11.TED;8
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE11.TED;9
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE12.TED;9
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE12.TED;10
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE12.TED;11
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE13.TED;12
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE13.TED;13
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE13.TED;14
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE14.TED;1
{PHYLUM}<HALASZ>LISPCOURSE>OUTLINE15.TED;1)
```

DIR *FileNamePattern Commands* ž a P.A. command that returns a list of all of the full file names matching the the *FileNamePattern.* The *FileNamePattern* is exactly as in FILDIR.

The *Commands* consists of one of more atoms that specify the additional attributes that should be printed for each file AND/OR an action that should be carried out on each file listed. The possible atoms are:

READDATE ž print the date/time each file was last read

WRITEDATE ž print the date/time each file was last written on

CREATIONDATE ž print the date/time each file was created

SIZE ž print the size (in pages) of each file

LENGTH ž print the length (in bytes) of each file

AUTHOR ž print the author of each file

DELETE ž delete each file

PROMPT *msg* ž print *msg* then wait for the user to type "Y" or "N". If the suer types "N" skip the rest of the commands for the current file.

Examples:

Print the names and creation dates of all the DCOM files in the connected directory: *DIR *.DCOM CREATIONDATE*

Result:

```
   {PHYLUM}<notecards>RELEASE1.2>LIBRARY>
INTERVALTEST.DCOM;1       25-Mar-85  0:10:21 PST
NCCHAIN.DCOM;1             2-Jan-85  3:00:34 PST
NCCLUSTER.DCOM;1          12-Feb-85  3:11:14 PST
```

```
NCCLUSTER.DCOM;2          25-Mar-85  0:13:02 PST
NCFILECARD.DCOM;9         12-Feb-85 23:32:47 PST
NCFILECARD.DCOM;10        12-Feb-85 23:47:14 PST
NCFILECARD.DCOM;11        12-Mar-85 18:49:23 PST
NCFILESUBSTANCE.DCOM;1    12-Feb-85 23:33:04 PST
NCFILESUBSTANCE.DCOM;2    12-Feb-85 23:48:39 PST
NCFILESUBSTANCE.DCOM;3    12-Mar-85 18:48:46 PST
NCKEYS.DCOM;1              7-Feb-85 20:33:25 PST
NCKEYS.DCOM;2             22-Mar-85 11:06:40 PST
NCSCREEN.DCOM;1           11-Feb-85 14:38:00 PST
NCSCREEN.DCOM;2           25-Mar-85  0:15:05 PST
```

Print the names, creation dates and authors of all the files in {phylum}<notecards>release1.1>library>:

DIR {phylum}<notecards>release1.1>library> CREATIONDATE AUTHOR

Result:

```
   {PHYLUM}<notecards>RELEASE1.1>LIBRARY>
NCCHAIN.;6          2-Dec-84 13:23:32 PST trigg.PA
NCCHAIN.;7         10-Dec-84 17:37:45 PST trigg.PA
NCCHAIN.;8          3-Jan-85  3:00:17 PST trigg.PA
NCCHAIN.DCOM;4      3-Jan-85  3:00:34 PST trigg.PA
NCCLUSTER.;11       3-Jan-85  3:01:00 PST trigg.PA
NCCLUSTER.;12       3-Jan-85 19:58:09 PST trigg.PA
NCCLUSTER.;13       3-Jan-85 23:52:03 PST trigg.PA
NCCLUSTER.;14      12-Feb-85  3:10:55 PST trigg.PA
NCCLUSTER.DCOM;9    3-Jan-85 23:52:55 PST trigg.PA
NCCLUSTER.DCOM;10  12-Feb-85  3:11:14 PST trigg.PA
NCKEYS.;3           4-Feb-85 13:08:12 PST Halasz.PA
NCKEYS.;4           4-Feb-85 13:10:46 PST Halasz.PA
NCKEYS.;5           4-Feb-85 13:12:35 PST Halasz.PA
NCKEYS.;6           4-Feb-85 19:32:46 PST Halasz.PA
NCKEYS.;7           4-Feb-85 22:16:11 PST Halasz.PA
NCKEYS.DCOM;3       4-Feb-85 19:33:05 PST Halasz.PA
NCKEYS.DCOM;4       4-Feb-85 22:16:34 PST Halasz.PA
NCKEYS.TED;2        4-Feb-85 18:05:06 PST Halasz.PA
NCKEYS.TED;3        4-Feb-85 18:14:39 PST Halasz.PA
NCKEYS.TED;4        4-Feb-85 18:15:39 PST Halasz.PA
NCKEYS.TED;5        4-Feb-85 18:24:10 PST Halasz.PA
NCKEYS.TED;6        4-Feb-85 18:26:46 PST Halasz.PA
NCSCREEN.;3         3-Jan-85  3:02:39 PST trigg.PA
NCSCREEN.;4         3-Jan-85 20:00:18 PST trigg.PA
NCSCREEN.;5         4-Jan-85  2:40:02 PST trigg.PA
NCSCREEN.;6        11-Feb-85 14:37:29 PST Trigg.PA
NCSCREEN.DCOM;4     4-Jan-85  2:40:59 PST trigg.PA
```

```
NCSCREEN.DCOM;5   11-Feb-85 14:38:00 PST Trigg.PA
VIDEOTAPE.;1      11-Feb-85 16:24:01 PST Halasz.PA
```

DELETE all of the files on the local disk's partition 5: *DIR {DSK5} DELETE*

Result:

```
   {DSK5}
ALTOD1MC.EB;1 c      deleted
Com.cm;1             deleted
DiskDescriptor.;1    deleted
DORADOLISPMC.EB;1    deleted
Dumper.boot;1        deleted
Executive.Run;1      deleted
FONTS.WIDTHS;1       deleted
HOLD.NOTEFILE;1      deleted
INIT.LISP;1          deleted
LISP.RUN;1           deleted
LISP.SYMS;1          deleted
LISP.VIRTUALMEM;1    deleted
REM.CM;1             deleted
Swat.;1              deleted
Swatee.;1            deleted
Sys.Boot;1           deleted
SYS.ERRORS;1         deleted
SysDir.;1            deleted
SysFont.Al;1         deleted
User.Cm;1            deleted
```

Go thru all the files on {DSK5} and ask the user if they should be deleted: *DIR {DSK5} PROMPT "Delete? " DELETE*

Result:

```
   {DSK5}
ALTOD1MC.EB;1        Delete? Yes     deleted
Com.cm;1             Delete? Yes     deleted
DiskDescriptor.;1    Delete? No
DORADOLISPMC.EB;1    Delete? No
Dumper.boot;1        Delete? No
Executive.Run;1      Delete? Yes     deleted
FONTS.WIDTHS;1       Delete? Yes     deleted
HOLD.NOTEFILE;1      Delete? Yes     deleted
INIT.LISP;1          Delete? Yes     deleted
LISP.RUN;1           Delete? Yes     deleted
LISP.SYMS;1          Delete? Yes     deleted
```

```
LISP.VIRTUALMEM;1    Delete? No
REM.CM;1             Delete? Yes     deleted
Swat.;1              Delete? Yes     deleted
Swatee.;1            Delete? Yes     deleted
Sys.Boot;1           Delete? Yes     deleted
SYS.ERRORS;1         Delete? Yes     deleted
SysDir.;1            Delete? Yes     deleted
SysFont.Al;1         Delete? Yes     deleted
User.Cm;1            Delete? Yes     deleted
```

Note: The Italics indicate the users response.

(DIRECTORY *FileNamePattern Commands)* ž is a function that acts just like the DIR P.A. command. The only difference is that DIRECTORY doesn't automatically list the file names. There is an additional command (**P**) to do this.

> Example: *(DIRECTORY '{PHYLUM}<HALASZ> 'P 'AUTHOR)* is equivalent to *DIR {PHYLUM}<HALASZ> AUTHOR*

Documentation of FILDIR, DIR, and DIRECTORY ž can be found in Section 14.3 of the IRM.

Manipulating Files

Basic Lisp File Manipulation Functions

There are a number of functions that allow you to manipulated files in Interlisp, e.g., to copy, move, and delete files.

These functions work on the file *per se* and not on the *information in the file*. Therefore, they apply to all flavors of files.

Each function takes one or two file name arguments. These arguments can *not* contain wildcard characters. They can however, leave off the device/pathname of the full file name. In this case, the connected directory and the DIRECTORIES list will be used as described above. If the version number is not specified, all function assume the HIGHEST version is intended (except DELFILE, which assumes the lowest version).

(DELFILE *FileName*) ž deletes *FileName* (i.e., gets rid of it forever and ever!!!)

Example: (DELFILE '<HALASZ>LISP>INIT)

(COPYFILE *FromFile ToFile*) ž makes a copy of *FromFile* and places on *ToFile*. If a file named *ToFile* already exists, the copied file becomes the next higher version number.

Example: (COPYFILE '<HALASZ>LISP>INIT
'{ERIS}<JONES>LISP>INIT)

(RENAMEFILE *OldName NewName*) ž renames the file *OldName* to be *NewName*. If a file named *NewName* already exists, the copied file becomes the next higher version number. On some devices, RENAMEFILE actually does a rename (and hence is relatively fast). On other devices, RENAMEFILE actually does a COPYFILE from *OldName* to *NewName* and then a DELFILE of *OldName* (and hence is relatively slow). RENAMEFILE from one device to another device always does a COPYFILE followed by a DELFILE.

Example: (RENAMEFILE '{phylum}<halasz>lisp>init;5 '<halasz>init)

Example: (RENAMEFILE '{phylum}<halasz>lisp>init
'{eris}<jones>init)

(SEE *FileName*) [Note: SEE is an NLAMBDA function] ž prints *FileName* to the TTY window so that you can examine it. SEE does no formatting, it just dumps it on your screen character by character.

Example: (SEE '{phylum}<halasz>lisp>test)

Result printed in TTY window:

```
69_SEE <HALASZ>LISP>TEST
(FILECREATED "21-Feb-85 15:50:09" {PHYLUM}<HALASZ>LISP>TEST.;3 1096
   changes to: (FILEPKGCOMS ANNOUNCEDFILES)
          (VARS TESTCOMS)
   previous date: "21-Feb-85 15:48:38" {PHYLUM}<HALASZ>LISP>TEST.;2)
**COMMENT**
(PRETTYCOMPRINT TESTCOMS)
(RPAQQ TESTCOMS ((ANNOUNCEDFILES ONE TWO THREE)
         (FILEPKGCOMS ANNOUNCEDFILES)))
(PRINT "Loading ONE")
(FILESLOAD ONE)
(PRINT "Loading TWO")
(FILESLOAD TWO)
(PRINT "Loading THREE")
(FILESLOAD THREE)
(PUTDEF (QUOTE ANNOUNCEDFILES) (QUOTE FILEPKGCOMS) (QUOTE
                            ((COM
                             MACRO
                             (X
                              (P
                               *
                               (FOR File in (QUOTE X)
                                  join
                                  (LIST (BQUOTE (PRINT ,
                                                (CONCAT

"Loading "
                                                        File)))
                                     (BQUOTE (FILESLOAD , File))))))
                             CONTENTS
                             (LAMBDA (COM NAME TYPE)
                                 (AND (EQ TYPE (QUOTE ANNOUNCEDFILES))
                                    (SUBSET (INFILECOMTAIL COM)
                                        (FUNCTION LITATOM)))))))
(PUTPROPS TEST COPYRIGHT ("Xerox Corporation" 1985))
(DECLARE: DONTCOPY
 (FILEMAP (NIL)))
STOP
NIL
70_
```

(LISTFILES *FileName*1 *FileName2* ...) [Note: LISTFILES is an NLAMBDA function] ž prints hardcopies of *FileName1*, *FileName2*, etc. on the DEFAULTPRINTINGHOST. LISTFILES is not strictly independent of file flavors. LISTFILES actually determines what flavor of file each *FileNamei* is and then calls the appropriate hardcopy function for that flavor of file.

COPYFILES LispUsers Package

The COPYFILES LispUsers package makes it easy to copy or move groups of files from one place to another using wildcard facilities like those in DIR and FILDIR.

(COPYFILES *source destination options*) ž Copies the files designated by *source* to the place designated by *destination*. *source* is a pattern such as given to DIRECTORY or DIR; it can also be a list of file names. *destination* is either a directory name, or a file-name pattern, with a 1-1 match of "*"s in *to* to "*"s in *source*. (The number of *'s in each *source* pattern needs to match the number of *'s in each *destination* pattern.) The argument *options* is a (list of) options (if you have only one and its an atom, you can supply it as an atom), as follows:

You can specify whether COPYFILES should ask before each transfer. Default is not to ask.

ASK ž ask each time before moving/copying a file (default is to not ask).

(ASK N) ž Ask, with default to No after DWIMWAIT seconds.

(ASK Y) ž Ask, with default to Yes after DWIMWAIT seconds.

COPYFILES normally uses COPYFILE to create a new file. It also usually only copies the "highest version", and creates a new version at the destination. Alternatively, you can specify any of the following:

RENAME or *MOVE* ž use RENAMEFILE instead of COPYFILE, i.e., the source is deleted afterwards.

ALLVERSIONS ž Copy all versions, and preserve version numbers.

REPLACE ž If a file by the same name exists on the destination, overwrite it (don't create a new version)

After COPYFILES gets done, it can be instructed to delete some files afterward:

PURGE ž This involves a separate pass (afterwards): any file on the *destination* which doesn't have a counterpart on the *source* is deleted.

PURGESOURCE ž converse of PURGE (and used by it): if the file is on the source and not on the destination, delete it.

Examples:

(COPYFILES '{ERIS}<MASINTER>.MAIL '{PHYLUM}<MASINTER>OLD-*.MAIL)* will copy the any mail file on {Eris}<Masinter> to {Phylum}<Masinter>, renaming FOO.MAIL to OLD-FOO.MAIL.

(COPYFILES '{ERIS}<MASINTER>.MAIL '{PHYLUM}<MASINTER>OLD-*.MAIL 'RENAME)* will use RENAMEFILE instead.

(COPYFILES '({DSK}TEST {DSK}WEST) '{PHYLUM}<MYDIR>) will move the files TEST and WEST from {DSK} to {PHYLUM}<MYDIR>.

COPYFILES({DSK}.; {FLOPPY})* will copy all files on {DSK} with no extension to {FLOPPY}.

(COPYFILES '{ERIS}<LISPUSERS> '{PHYLUM}<LISPUSERS> '(PURGE)) will make {Phylum}<LISPUSERS> look like {ERIS}<LISPUSERS>; bringing over any file that isn't already on Phylum and then deleting the ones that were on {PHYLUM} and aren't on {ERIS} any more.

Still to come about files and directories:

Using the FileBrowser

CHATing to an IFS

Dealing with Devices

Using Floppies on the Dlion

Archiving Files

Opening and closeing Files

References

Connected directories are covered in Section 18.16.6 of the IRM.

The DIRECTORIES list is covered in Section 18.16.6 and page 15.20 of the IRM.

DIR, FILDIR and DIRECTORY are covered in Section 14.3 of the IRM.

The file manipulation functions are documented on pages 6.3 and 18.10-11 of the IRM.

COPYFILES is documented on {eris}<lisp>harmony>library>CopyFiles.TEdit.