PDP-1 COMPUTER
ELECTRICAL ENGINEERING DEPARTMENT
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS 02139

PDP-42

MICROTAPE FILE SYSTEM

12 September 1973

1. General Features

The file system provides storage for programs and data on magnetic tape (microtape). Each tape can hold approximately 36 drum fields (4096 words each) of text or binary information. File names are arranged in a hierarchical structure, allowing files to be specified by a succession of up to 102 names, each of which may be as long as nine characters.

Tape space is conserved by storing only non-zero parts of files. Extensive checking is done to avert the possibility of accidentally destroying the directory on a tape.

The file system is not stored as part of the operating system, as are most other system programs. Instead, it occupies the first thirteen blocks of each tape which is to be used for file storage, and bootstraps itself and its directory off of the tape with a simple loader stored on block 0. This allows the same system interface to be used for tapes of different formats, such as the Fortran tape or the Chess tape.

Any new tape which is to be used for file storage should have a file system put on it first. Also, due to changes in the operating system, old tapes may sometimes need to have their file systems updated. This can be done quite simply without affecting the directory on the tape, and users with their own tapes may find it occasionally necessary.

Encoded in the directory on each tape is a tape name, which may contain up to nine characters, with the same restrictions as for a file name. This name may be changed only when a new file system is being put on a tape. The rest of the directory will be unaffected.

2. Using the File System

The most common way to enter the file system is by typing the command nF to ID or Expensive Typewriter. This causes the file system on transport n to be loaded and started. This command will assign unit n as capability 16, read block 0 of the tape into locations 100-477 of core 0, and start it at location 102. This is the normal entry to the file system, which will then listen to the typewriter for commands.

If a typing error is made in a command to the file system, there are two simple ways to recover. Typing centerdot (•) will delete the entire command line, and typing backspace will delete the last character typed in, except for space or comma, which cannot be deleted. Any error condition other than a write permit error will terminate execution of the command.

The microtape will be dismissed upon leaving the file system unless sense switch 6 is up.

3. Commands

Syntax

Commands to the file system consist of a word (a single letter of which is sufficient) describing the action to be taken, followed by the file name(s), field number(s), and a starting address where appropriate. Commands requiring arguments must be terminated with a carriage return. Commands not requiring arguments may be terminated with either a carriage return or a comma.

File Names, F

All files are subfiles of larger files, except for top level files, which are subfiles of the directory. The directory itself may be thought of as a file with no name. The complete name of a file F is a list of the names, in order, separated by spaces, of all files of which the named file is a subfile, e.g., the names "foo barf bletch" and "foo mumble barf" imply that "foo" is a top level file with subfiles "barf" and "mumble". "foo barf" has subfile "bletch", and "foo mumble" has subfile "barf". There is no confusion of the two files named "barf". All characters are legal in a file name except comma, carriage return, centerdot, and space. The maximum length of each part of a file name is nine characters. Longer names will be truncated. Case shafts are characters, and are not filtered.

For use with unsave, edit, text, append, loadgo, yank, nightmare, print, binary, and <octal 16> commands, file names may be abbreviated on type in to as many characters as makes them unambiguous. Also, subfile names need be entered only to a depth which uniquely identifies a single bottom level (data containing) file.

Field Number, N

Any drum field capability index (1-77 with PRL, 1-17 without) or absolute field number (200-237, 300-337) is legal.

Starting Address, S

Any octal number is legal, unless it would cause the file system to try to violate its PRL. Only the low twelve bits are significant (i.e., the starting location must be in core 0).

In the following examples, the command is shown as it would be typed, with these changes --

F, G, and H represent complete file names.

M and N represent field numbers.

S represents a starting address.

When part of a command is shown in parentheses, that part is optional, and the parentheses should not be typed in the command string. If N is not given, it is assumed to be 1. If S is not given, it is assumed to be 102 with PRL or 0 without.

4. Commands for Handling Text

The following commands are used to load and store Expensive Typewriter's text buffer.

file F

The current contents of Expensive Typewriter's text buffer are copied onto the tape under file name F. Any previous file of the name F is deleted.

edit (F(,G(,H ...)))

If no file name is specified, control is given to Expensive Typewriter with no data transfer occurring. Otherwise, Expensive Typewriter's buffer is cleared, and the files are appended to it in the order specified. The tape is rewound and Expensive Typewriter started.

text F(,G(,H ...))

The action is the same as edit except the tape is not rewound and Expensive Typewriter is not started.

append F(,G(,H ...))

The named files are appended to Expensive Typewriter's buffer.

nightmare (F(,G(,H ...)))

The specified files are loaded into Expensive Typewriter's buffer as in the edit command, and then the assembler is started in normal (i.e., Nightmare) mode.

5. Commands for Handling Binary Data

The following commands operate on drum fields.

save F(,N)

The contents of drum field N are written on the tape as file F. Any previous file named F is deleted. If N is not specified, field one is assumed.

save F,M,N

M, a field number, is the same as the standard single argument of save. N is a second field number. All fields between M and N, inclusive, are saved as file F.

Examples --

```
s drum,300,337      /saves entire B drum contents
s fields,1,4        /saves fields 1, 2, 3, and 4
```

write F(,N)

The action is the same as save (with one argument), except if more than one core is assigned, all contiguous storage above core 0 is considered as part of the file, following drum field N. This command is used to store multiple field programs in a format to be retrieved by the loadgo command. Attachments will not be saved, even if they are adjacent to assigned core.

unsave F(,N)

The file F is copied onto drum field N. If the file is more than one field in length, consecutive fields are used, starting with N.

For protection, the file system will not unsave a text file, since it may exceed one field without the user realizing it. If the file was saved as binary, unsave will work regardless of its length.

yank F(,N(,S))

The file F is copied onto drum field N, then brought into core 0 and started at location S. The tape is rewound. If the file is more than one field in length, consecutive fields are used, starting with N, then field N is brought into core and started. When the loaded program is started, I(0-5) contains the number of drum field N.

loadgo F(,N(,S))

The action is the same as yank, except if the file F is more than one field in length, additional core will be assigned as necessary and the file loaded directly into it. Only one drum field will be used. When the loaded program is started, it will all be in core. The file will not be loaded into attachments. Any that get in the way will be replaced with assigned core.

6. Commands for Handling the Directory

delete F(,G(,H ...))

The file(s) F (,G(,H ...)) are deleted from the tape. When more than one file is being deleted, a file will go away as soon as the comma is typed after its name. If the given file name does not represent a bottom level file, it and all of its subfiles will be deleted.

rename F,G

The file G is created, having the same subfiles and data as the file F. F is then deleted. The effect is to change the name of F to G; however, any previous file G will be deleted (except the contents of F will not be deleted if F happens to have been a subfile of the original G).

binary F

The mode of the file F is changed to binary, so that the unsave, yank, or loadgo commands will accept it. This command is only necessary if, for some reason, the file system thinks F is a text file.

print (F)(,)

The names of the subfiles of file F are printed out. If F is not given, the entire directory is printed. If the command is terminated with a comma, the complete hierarchy of F is printed. If it is terminated with a carriage return, only the immediate subfiles of F are printed. The number of blocks in each file is given in octal.

If F is partially matched, the lowest level component of F is typed out immediately following the command. If F is a bottom level file, the number of blocks in F is typed out.

totals

The numbers of free tape blocks and directory entries are printed out. These numbers are in octal.

you are who

The tape name encoded in the directory is printed out.

rewind

The tape starts to rewind, and will continue until either it is entirely rewound, or another command requests a data transfer. The command completes immediately.

write directory

If the in-core directory differs from the tape directory (i.e. some command has changed it), it is rewritten onto the tape. Normally, this command is not needed, since the file system will automatically rewrite the directory if necessary before giving up control to any other program.

7. Commands for Entering and Leaving the File System

back

The tape is rewound and control is returned to ID. Do not use the call button to return to ID (see the warning below).

go (N)(,S)

Drum field N is brought into core 0 and started at location S. I(0-5) contains the number of drum field N, as in loadgo and yank. The tape is rewound.

0
1
2
3

The tape on drive 0, 1, 2, or 3 is assigned and the file system thereon loaded and started.

WARNING: The call button should not be used to leave the file system. Neither should the console be turned off nor the microtape removed while the file system is running. This is because the directory may have changed, but not yet been rewritten onto the tape. The file system will automatically rewrite the directory if necessary if left properly.

8. Directory Handling

Along with the body of code which is the file system, and files, a directory is stored on the tape. This contains all the information which is needed to associate file names with actual tape blocks. If this directory were changed accidentally, either by a hardware malfunction or by a program writing on the wrong tape, all files might become inaccessible -- they would still exist on the tape, but the file system would not know what blocks each file occupied. A number of steps have been taken to guard against such problems. First, before each command is executed, the directory is carefully checked for validity. Second, the file system will not write out a bad directory on a tape. Third, each tape has a name encoded in the directory. The directory will not be written on a tape whose name differs from the name in the in-core directory.

The message 'anomalous directory' will be printed if, at any time, the in-core directory is found to be improperly formed. No commands will be executed in this state. Someone who understands the format and construction of the directory should be consulted immediately. Also, the circumstances surrounding the lossage should be noted in the log.

9. Using the File System From Other Programs

There are two ways in which the file system may be called from a user program. The first is the common way of starting the file system and giving it control of the typewriter, as is done by (e.g.) Expensive Typewriter. The second is a way of starting the file system so that the calling program continues running, and must, in some way, account for the file system's I/O, as is done by (e.g.) FORTRAN.

The normal way of starting a file system from a program is to assign the tape unit, then read block 0 into locations 100-477 of core 0, put the index of the tape capability into X(6-17) (if X(0-17) is 0, capability 16 will be assumed), and start at location 102. The file system will read the rest of itself into core. The calling program will be wiped out.

The second way of starting a file system is to create a sphere (see memo PDP-35, parts 4 and 5), and run the file system in a separate computation. In this way the calling program may continue to run. The file system must be started in the sphere in the normal way, but in addition, its capabilities (in particular, the tape and the "typewriter") must be created for it beforehand by its superior. Any of the capabilities given the file system in this way may be entry capabilities, if desired. Capability 0 will be used as if it is a typewriter, and the capability indicated by X (as above) when the file system was started, as the tape.

When an entry is used on capability 0, the following two commands may be used, in addition to the regular ones. These special commands do not transfer data to or from the drum. They work only through the 400 word core buffer in the file system, and must be accessed with read/write sphere ivk's. The address of the core buffer is given in A whenever capability 0 is invoked for typein.

<octal 15> F

This command appends a new block to the end of the (binary) file F. F will be created, if it does not already exist. The data must be placed in the buffer immediately before this command is executed.

<octal 16> F,B

This command reads block B (starting at 0) of file F, without undue tape twiddling. If there are fewer than B blocks in F, a question mark is typed. Blocks containing all zeros are included in the count, so it is possible for a file to contain more blocks than is indicated by the print listing (which tells how many physical tape blocks were used). The data from this command will be found in the buffer immediately upon its completion.

10. Preparing a Tape for Use With the File System

These procedures, while not difficult, should nonetheless be performed only by users who are very sure of what they are doing.

The file system is stored as a binary program on public tape pdp-0, under the file name "filesys". This version is always kept up to date with the latest memos and system changes.

The normal use of this program will be to replace the file system on tapes that already have a directory and files on them. Load the file system program into core 0 and start it at location 0. It will ask for the tape drive number, and then allow the tape name to be changed. If no change is desired, type a carriage return when it asks for the new name. The file system will be replaced on the tape, and all files will be saved.

If the file system is to be put on a new tape that has never before been used with a file sytem, that tape should have its timing and mark tracks written and a tape test run before getting the file system. When the tape is properly marked, load the file system program into core 0 and start it at location 1. The tape drive must have been previously assigned, and the program will ask for verification of the drive number. The procedure from there is the same as for the entry at 0, except that a completely empty directory will be put on the tape. If the tape had previously contained files and a directory, they will be lost.