THE FILESTORE

Hamish Dewar
Vicki Eachus
Kathy Humphry
Paul McLellan

October 1977

Introduction

The filestore is a stand-alone filing system. It communicates through links with other machines for which it provides service. The storage is disc based, although a magnetic-tape unit is to be added. The filestore system is designed so that a fairly simple system can make use of it, and thus provide a much more comprehensive filing-system to its users than would otherwise be possible.

The first section of this document presents an overview of the filestore, together with a description of the facilities which it provides.

The second section describes in detail how to use the filestore. It contains all the command formats, responses, restrictions and defaults. It is intended mainly for people implementing systems and programs which communicate with the filestore and not for people who use such systems (for whom the first section together with the details of the particular implementation should be sufficient).

The third section gives details of the error-messages which the filestore produces and the actions taken.

There is an appendix giving an example script showing how the facilities described can be used in practice.

SECTION 1 - FILESTORE FACILITIES

Hardware configuration

The processor for the filestore is an Interdata 70 with 64 Kb of store. The disc is a single (possibly to be raised to two) CDC 9762 drive with a formatted capacity of 67.4 Mb, attatched to a Systems Industries Kahili controller. The disc controller is interfaced to the Interdata by a specially built interface board. Attatchment to the communicating machines is by Departmental Link (Mark II), one receiver and transmitter pair being required for each connection. The line-printer is a Data Products 2260, using a standard link interface. At a later stage a Racal Thermionics T7000 magnetic-tape drive (1600 bpi, 9 track phase-encoded) may be added. It is also intended to replace the link interfaces with a DMA link multiplexor.

Organisation of devices

The other machines to which the filestore is connected are known as clients. These are classified as major or minor according to the maximum load they are allowed to impose on the filestore. There is a further important distinction between types of client. Some clients, called wholesalers, handle the file transfers of many users over a single line. These are normally multi-user operating systems relying in whole, or in part, on the filestore for file capability. Other clients only support a single user. It is not intended that the filestore should support intense demand from a single client, such as with paging.

The filestore has a protocol which enables several file transactions to take place over a single line, by transferring files on a block by block basis. However, there is a mode of transfer in which a whole file is transferred without protocol, requiring a dedicated line for the duration of the transfer; such transfers are terminated by a character with the ninth bit set.

The various functions that the filestore can perform are invoked by commands directed from any of its link receivers or the console keyboard. Paired with each of these input devices is an output device, known as its twin; link receivers are paired with their transmitters, the console keyboard is paired with the console teleprinter. Every command provokes a response which is directed to the twin of the command source. When used in this way, the receiver and transmitter are known as the control-device and the report-device respectively. When a command is received, no further commands are accepted from that control-device until the command has been completed (whether successfully or unsuccessfully) and a response has been transmitted to the report-device. A control-device and its twin report-device are known as a command-pair.

Each command or response consists of a series of ISO characters terminated by a newline (linefeed). Commands contain a command-word followed by a sequence of parameters, all separated by commas. In commands, leading spaces only are ignored and lower-case letters are translated to upper-case. Most command-words may be abbreviated to a single letter. A failure response is indicated by a minus sign followed by two hex digits known as the errorcode; the error message in text form follows, preceeded by a colon and terminated by a newline. Depending on the command, the successful response is either an item of data or a null response (but note that the terminating newline is still sent). All characters in both commands and responses are printable to make the filestore usable from keyboards; however, the formats of commands and responses are designed for machine generation and are not particularly convenient for direct use.

When the filestore receives a file (or block) through a device, the device is known as an inlet; when it transmits a file (or block) through a device, the device is known as an outlet. When a single transaction requires both an inlet and an outlet, then the inlet must be the twin of the outlet.

The filestore protocol is designed so that the inlet and control-device, and the outlet and report-device may be a single pair of devices. Indeed, this is the usual way of working, however control of a device from another device pair is allowed. When transmitting and receiving files, no assumptions are made about the data in the file and full eight-bit transparency is provided.

All devices have a device-name by which they are known to the filestore system. These names refer to a pair of devices, the transmitter or receiver being deduced from context. Some devices are protected; in general, a protected device may not be referenced in commands from other devices.

The lineprinter is not regarded as a device in the above sense, but as a file. Any data written to the file will eventually be printed.

There is no restriction on who may issue commands to the filestore; they must, however, warn the filestore by logging-on and terminate their session by logging-off. Anyone logged-on to the filestore at any particular time is known as a user.

Organisation of disc space

The basic unit of disc space is the sector of 512 bytes. The total available space (less a small amount used by the filestore system itself) is divided into fixed areas called partitions. There are currently two partitions of 64640 sectors each. All allocated disc space is in the form of files. Each allocated sector is in exactly one file.

There is a class called owners, who may request disc space for

There is a class called *owners*, who may request disc space for files. Most owners are simply the people who are allowed to keep files on the filestore, but some owners represent libraries of related files. It is not necessary to be an owner to use files (only to create them). Each owner is identified by a unique *ownername* and the filestore maintains a *register* of these ownernames and their *quotas*, fixed personal limits which their total allocation of disc space may not exceed. Each file is owned by exactly one owner. Owners give files *filenames*, which must be unique amongst their own files. An ownername and a filename together constitute a *complete-filename* which identifies the file uniquely.

The filestore maintains a *directory* giving details of all the files for a particular owner. There is a one-to-one correspondence between owners and directories. Each owner *resides* in one of the partitions, and all files of that owner are constrained to be in that partition. In this way, each partition is shared out among a number of owners from whose point of view the partition represents the total available disc space.

Note the distinction between owners and users; a user is anyone currently using the filestore and may or may not also be an owner.

Files

All files have one of two *organisations*. They are either *linked* or *contiguous*. Linked files consist of a number of areas of adjacent sectors, called *extents*. Linked files may only be read and written *sequentially*; that is, they are interpreted as a stream of characters. They are created automatically whenever written. Space is allocated by obtaining an extent called the *initial-allocation* and repeatedly extending the file by linking on further extents called *subsequent-allocations* should it prove necessary. In fact, the filestore extends each extent if the following sectors are free. Any unused sectors overallocated in the final extent are returned. The directory of each owner contains default values for the sizes of these allocations. When a linked file is successfully written, then any file having the same name is destroyed.

Contiguous files consist of a single extent. They may be read and written by *direct-access* (on a random block basis); they may also be written sequentially, in which case they may be read sequentially. Contiguous files must be created explicitly before they can be used since the same physical file is used every time the file is accessed. Hence, unlike linked files, it is not possible to read and write files of the same name, nor is information in an old file preserved until a new file of the same name is written successfully; thus, a linked file consisting of a single extent is not a contiguous file.

The filestore maintains details of the files in directories. Each file occupies one *file-slot* in the directory, and each subsequent-allocation occupies one *extent-slot*. There is a fixed maximum number of these slots, currently 63 files and 192 extents. There is also a directory *header* containing information pertinent to the whole directory.

There are four different types of file; *permanent*, *temporary*, *transient* and *subliminal*.

Most files are permanent files; once in existence they remain until *deleted*. The space which they occupy is deducted from the owner's quota.

Temporary files are only available to owners while they are logged-on to the filestore; they are deleted automatically on logoff. However, the space that they use is not deducted from the owner's quota and so there is an essentially unlimited amount of workspace available. Because temporary files only exist whilst their owners are logged-on to the filestore, only the owner may create them, although they are accessible to other users while they continue to exist. For convenience, an owner may be multiply logged-on to the filestore; temporary files are only deleted at the final logoff, when the owner ceases being logged-on at all.

Transient files are linked files which are in the process of being written sequentially. If written successfully, then they will become permanent or temporary as appropriate. If not written successfully then they remain transient and may only be deleted or renamed, and are otherwise inaccessible. While they are in the process of being written, transient files may not be accessed in any way, although it is still possible to read any old file of the same name. Except when written unsuccessfully, transient files should not concern the ordinary user and are handled automatically by the filestore.

Subliminal files are permanent files which have been deleted while in use, or have been created without a name. In either case they will be destroyed when they cease being used. The space that they occupy is not restored to the quota of the owner until the file is destroyed. Subliminal files have no name and thus are inaccessible to anyone who is not already using the file; in particular, all files written without a name are distinct subliminal files.

Associated with each fileslot are the data (such as disc addresses), which the filestore maintains for its own use, and some backup and age information. Every file has an *archive-indicator*, set by the owner, which is either *archive* or *vulnerable*. Files with archive set will (when the magnetic tape drive is added) be automatically backed up if they have been altered since the last time a backup was taken. Files which have not been used for a period (to be decided) will be automatically deleted. Each file has a *datestamp*, giving the date and time (to the nearest minute) that the file was last written. The information available about a file thus consists of the filename, the file organisation, the archive indicator, the permissions, the size in sectors and extents, and the datestamp.

Every directory conceptually contains a *pseudo-file*. This is not a file in the normal sense but it may be read exactly as if it

were, and contains the above details (in text form) about each were, and contains the above details (in text form) about each file in the directory.

Security and permissions

Protection of files is done partly on a directory basis, and partly on a file-by-file basis. Users demonstrate their right of access with respect to the whole directory, but what type of access these rights give can be varied from file to file.

For a user to access a file or alter a directory he must have a right to do so. These rights are regulated by authority which is gained by quoting passwords. There are three levels of authority.

The lowest level of authority is termed public-authority. Anyone has public authority. Public authority gives one the right to read and write files which have been permitted to the public by their owners.

The next level of authority is password-authority and this is gained by quoting a password which matches the directory-password. This gives one a right to read and write files which have been permitted to password-quoters by their owners. This is the usual way of allowing users access to some of the files of a given owner.

The highest level of authority is owner-authority. This is obtained by quoting the logon-password. This gives one a right to read and write all files in the directory, create new files, delete old files and alter permissions. Normally, this authority will be gained at logon by quoting the logon-password, but it is not necessary to be logged-on as the owner to gain owner-authority. However, only the owner has the right to alter the passwords. Some owners will require more fileslots than is possible (due to the size of a directory); they can be given automatic owner-authority with respect to another owner, known as an alias.

If the logon-password or the directory-password is null then it is matched by any quoted password. If a quoted password is null, then it only matches null passwords. This gives a way to unquote a password.

Note that authority is demonstrated by a user with respect to a particular directory, but passwords are quoted for matching to any directory. It is thus possible to have owner-authority with respect to one directory and password-authority with respect to another if the passwords in the directories are set up appropriately. Once quoted, a password remains quoted until another password is quoted, so that the password does not have to be requoted every time a file is accessed. Also, since the passwords apply to whole directories, it is possible to gain access to a group of files by quoting a single password.

Each file has associated with it three permissions corresponding to the three levels of authority. These can each be at one of four levels, though they are constrained so that the password-permission is no stricter than the public-permission and the owner-permission no stricter than the password-permission. The four levels of permission are _free_, _read_, _obey_ and _none_. Free permission allows the file to be read, written, and, if the user has owner authority, deleted. Read permission allows the file to be read. Obey permission allows the file to be used within the filestore system itself, but it may not be accessed from outside. None permission allows no access at all. Only a user with owner-authority may alter the permissions of a file.

This system of passwords and permissions gives a fairly versatile, if imprecise way of permitting and barring file-access to other users.

Implementation

Certain constructs are universal to all implementations which use the filestore and these are described here. Some implementations may enforce stricter conditions than these, such as shorter filenames. Indeed, some systems may not make all the filestore's facilities available.

An ownername consists of up to six alphanumeric characters, the first of which is a letter. Ownernames are allocated administratively and are not chosen by the owner. The following are valid ownernames:

```
GEORGE
WILSON
TOM
```

A password has the same syntax as an ownername, though it may be null. Passwords are, however, chosen by owners and are alterable at any time. The following are all valid passwords:

(nothing)

```
QWERTY
A
MVEMJS
```

A permanent filename consists of up to twelve characters, of which the first is a letter, and the remainder are letters, digits or colons. This gives the user the freedom to choose how long an extension to add to a filename. The following are all valid permanent filenames:

```
A
LONGFILENAME
LAYOUT:OBJ
CAT:DOG:COW
```

A temporary filename consists of a dollar, followed by up to nine characters of which the first is a letter and the remainder are all letters, digits or colons. The following are all valid temporary filenames:

```
$T
$TEMPFILE
$WORK:1
```

When a file is created for sequential writing then it is transient. If written successfully it becomes permanent or temporary as appropriate. There is nothing to distinguish a transient filename; the filestore handles them automatically.

A subliminal file has no name. For very temporary workfile use they can be created by omitting the filename from a complete-filename. They may then be written sequentially and reread. All subliminal files in a directory are distinct, so using subliminal files for temporary workspace can never lead to clashes on names. The following is the valid filename when creating subliminal files:

```
(nothing)
```

The lineprinter spool file has a special name consisting of a single asterisk. No ownername may be present. Thus the following is the only valid filename to write to the printer:

```
*
```

The pseudo-file containing the directory information is called DIRECTORY and may only be read sequentially. The information provided can be varied by appending a colon and a single letter. DIRECTORY:U gives usage information, such as how may extent slots are in use. DIRECTORY:A and DIRECTORY:D give full information about the files with the files sorted in alphabetic or reverse date order respectively. DIRECTORY:E gives everything. If no extension is given, only the filenames are listed.

A complete-filename consists of an ownername, a period and a filename. Normally the ownername will be omitted (together with the period) and a default assumed. This default is normally the logged-on owner, but it can be changed. If the ownername is omitted, but the period is not, then the owner is assumed to be PUB. The following are all valid complete-filenames:

```
SYS.LOADER:B
TOM.IMP030277
HARRY.$LIST
HENRY.
*
$TEMP5
HARRY.DIRECTORY:D
IMP:70
MYFILE
```

A permission consists of the owner, password and public permissions. Each may be any of F, R, O or N for free, read, obey and none. This may be followed by the archive indicator which may be A or V for archive and vulnerable. If any (or all) permissions are omitted, then the remainder are assumed unchanged from any old file of the same name. If the archive indicator is omitted, then it is assumed unchanged. If no old file of the same name exists, then the permission is interpreted as if there was an old file with permission FRNV. The following are all valid permissions:

```
(nothing)
```

FFFA
V
FNN
ODV
R

Device-names consist of up to four alphanumeric characters. The following are all valid device-names:

ISYS
7502
ISYN
P15
PDP9
LEGO
PUB
TTY

Partitions are named by a single letter, commencing with A. There are currently two partitions so that the following are the valid partition names:

A
B

Initial and subsequent allocation sizes must be between 1 and 255 sectors, and the subsequent allocation size must be no more than the initial. The following are thus valid initial and subsequent allocation sizes:

1	1
255	255
30	3
10	5
100	20

Use of the lineprinter

The lineprinter is not a directly accessible device as are all other filestore devices in that it does not have a name which may be used in transfer commands.

Files to be printed are sent to a special file (complete-filename *). Such files, after closure, become subliminal files owned by the printer spooler, which sends any file in its directory to the lineprinter device as soon as possible and then deletes it.

The lineprinter is essentially a client but it does not have the capability to control its interaction with the filestore, the operator (user) controls the device, using special lineprinter commands, from the console. These commands inform the filestore wether the lineprinter is able or unable to print anything.

The printer-spooler sends a form-feed (FF) character to the lineprinter after every file (unless the last character in the file was a FF) to ensure that each file starts on a new page.

Device errors

If an error is detected on a link during a command/acknowledgement/data transfer, the transfer is suspended immediately and action is taken to inform the client of the error. Intervention from the console is then required to continue the transfer or to kill it. The transmitter and receiver of a device-pair are treated as separate devices since either one may have an error, although receiver errors are rare.

The lineprinter should be considered as a client in the following description.

The reaction of the filestore to a device error is as follows:

1) a reset is sent down the offending link to inform the client of the error,

2) the display light which represents the device is set to flash

3) a message of the form:

device:T or R NOT READY

is monitored.

The operator should then:

1) cause the client system to recover from the fault so that it may

(a) continue the transfer with no loss of data

or

(b) kill the transfer and return to command status

2) type the corresponding command

(a) RETRY

(b) KILL or ABORT

at the console to clear the filestore error status and stop the display from flashing.

Console

The filestore console represents three devices

The console keyboard
The console printer

which are a normal filestore device pair, and

The console monitor

As a keyboard/printer the console may be used to input normal filestore commands/data and output filestore responses/data. Certain commands may be input only from the console since they control filestore operation and are not for use by clients, some require privilage.

The console is used as a monitoring device to record certain internal filestore events; e.g. disc errors, device errors and logging information. There is a command (QUERYDEV) to invoke monitoring of commands sent from clients to provide a record of the communications between the filestore and a particular client, this is useful when developing new client systems. (Data tranfers are not monitored).

These three functions are interleaved to use one physical device.

The console may be in one of 4 states - _listen_, _read_, _print_, monitor. It is normally in the listen state. Pressing a key on the keyboard sends it into the read state as it inputs the command/data, if it was not waiting for input then no characters typed are accepted or echoed. The console may otherwise change from the listen state to the print state in order to output data/responses invoked by a command, it may also enter the monitor state to output filestore information. A newline in input or output returns the console to listen state, followed by further changes. This means that nothing may be typed at the console unless the filestore is ready for it, monitor information is not held up by excessive output or vice versa. Command input need not be held up by monitor output; the monitor state may be changed to the read state by pressing any key on the keyboard enough times to break through the output.

Character echoing on input may be suppressed by pressing the STX key, all following characters on the line are not printed, CR cancels the echo-suppression. This may be used to input passwords. Other non-printing characters are ignored.

SECTION 2 - COMMANDS AND RESPONSES

SECTION 2 - COMMANDS AND RESPONSES

This section is intended to give enough information to implement a system which communicates directly with the filestore. As already described, the filestore provides services on receipt of commands, and always produces a response. There are three different levels of command. At the highest level, the only acceptable command is LOGON, which (if successful) returns a user-number, abbreviated to userno. At the second level, all commands are accompanied by the userno of the user requesting service. Most commands are at this level, such as DELETE, LOGOFF and QUOTE. In particular, there are three commands, OPENR, OPENW and OPENDA, which initiate a file transfer (from now on, a transaction) and return a transaction-record-number, abbreviated to xno. All third level commands are accompanied by the xno on which the requested service acts. The commands at the third level are those such as READDA, WRITESQ and CLOSE which further or terminate a transaction.

Additionally, there are some commands only available at the system keyboard, such as USERS or ABORT. These commands do not always observe the strict syntax rules given in this section since they are neither transmitted nor received by programs.

For convenience, there is a user, with userno zero, who is always logged-on but only has public-authority.

Syntax

All commands have a similar format. They all start with the command-name and are followed by a comma and a number of parameters, all separated by commas, and a terminating newline. The command-name is either a word relevant to the required service (such as LOGON) or a (usually unconnected) letter (such as L). Most commands have single-letter short forms. The parameters vary depending on the type of command, but are all drawn from single-letter, permissions, passwords, filenames, ownernames, device-names or hexadecimal (up to 16-bit) unsigned numbers. It is often allowable to omit a parameter (so that two commas are adjacent) and allow a default to be understood; a trailing run of commas may, however, be omitted. Leading spaces are ignored and all lower-case letters are translated into their upper-case equivalents. Spaces within commands are not allowed. For example the following are all syntactically correct commands:

```
LOGON,HENRY,SHRDLU
CREATE,1F,HENRY.FILENAME:B,FV,A2
PASS,3
PASS,3,PP1
PASS,3,PP1,PP2
PASS,3,,PP2
PASS,3,PP1,
PASS,3,,
```

Available commands

Detailed specifications of all the available commands are provided later in this section, in alphabetical order of command. A brief summary is given here.

Before anyone can issue commands to the filestore, they must first log on to obtain a userno; this is done by the LOGON command. A userno obtained by logging-on remains valid until the user logs off using the LOGOFF command.

Users who are also owners may alter their passwords and default allocation sizes by means of the PASS and DEFALL commands.

Existing files may be deleted, renamed or have their permissions altered by the DELETE, RENAME and PERMS commands respectively. New contiguous files may be created by the CREATE command.

Transactions may be started by the OPENR, OPENW, OPENDA READFILE and WRITEFILE commands. These open the file for sequential-access read, sequential-access write, direct-access and transfer without protocol respectively. When opening for writing, with either OPENW or WRITEFILE, a new copy of the file is used as explained in section one. Blocks may be read from and written to transactions open for direct-access by the READDA and WRITEDA commands. The next sector may be read from or written to a file open for sequential-access by the READSQ and WRITESQ commands. Any transaction with protocol may be closed by the CLOSE or UCLOSE commands, the latter terminating the transfer unsuccessfully. A file open for sequential-access may be reread by the RESET command.

A user may quote a password by the QUOTE command. The default ownername may be altered by the OWNER command.

The date and time is available by the DATIME command. The number of free sectors in the partition is available through the FREE command.

Responses

A response is always sent on completion of servicing a command. This is either the null response (a newline), or a line containing an item of data; in particular, the response is always a single line. In the case of the READDA and READSQ the response is sent before any data is transmitted from the filestore to the client; if an errorcode is sent, indicating that the command failed, then no data is transmitted at all. In the case of WRITEDA and WRITESQ, the response is sent before any data is received at the filestore from the client; if an errorcode is sent then no data is accepted. In the case of WRITEFILE being used to transfer a file to the filestore without protocol, then the whole file is received even if an error occurs (no indication of the error is sent and the remainder of the file is thrown away); in the case of READFILE being used to transfer a file from the filestore, then end-of-transmission (byte with bit nine set) is sent should an error occur (again, no indication of the error is sent). In all the commands which transfer data (READFILE, WRITEFILE, READDA, WRITEDA, READSQ and WRITESQ), the response is not transmitted until the inlet or outlet is actually ready to receive or transmit the data; the transfer can then take place immediately. Errorcodes are explained in detail in section three.

Conventions

The rest of this section is a detailed description of the commands. In the line showing the format of each command, all upper-case words stand for themselves, while lower-case words should be replaced by a value of the appropriate type. The details about filenames, passwords, devicenames and so on are in section one. All numeric quantities in commands are hexadecimal; to avoid confusion, all hex numbers appearing in plain text are preceded by a hash, so that, for example, 257 is written #101. Numbers in plain text not preceded by a hash are in decimal.

Details of filestore commands

COPY

```
COPY,userno,filename-1,filename-2,perms
```

This command causes "filename-2" to be a copy of "filename-1" and to have permissions "perms". The successful response is the null response. The response is sent when the filestore has determined that the copy is valid, but before the file has been copied. Thus errors occuring during the copy operation are not signaled.

For example: 3, ?

COPY,HENRY.FRED,JIM	Copy FRED owned by HENRY (if permitted in read mode) into the directory of user #3 with the same permissions.
COPY,5,HENRY.FRED,*	List HENRY.FRED on the lineprinter.

CLOSE

```
CLOSE,xno
```

This command is used to terminate transaction "xno". The successful response is the null response. For example:

CLOSE,14	Closes transaction #14

CREATE

```
CREATE,userno,filename,permission,size
```

This command creates a contiguous file, "filename" in the appropriate directory (either the ownername specified, or the default ownername) with permissions "permission" and size "size" sectors. "userno" must have owner-authority with respect to the directory, and the owner must have sufficient quota remaining if the file is to be permanent. The command will fail if a contiguous block of "size" sectors is not available, or a file of the same name already exists in the directory. The successful response is the null response. For example:

```
CREATE,4,SMALLFILE,,4    Creates a file of #4 sectors in
                         the default ownername's directory
                         with permissions FRNV
CREATE,5,HENRY.HUGEFILE,FRRA,1000
                         Creates a file HENRY.HUGEFILE of
                         #1000 sectors with permissions
                         FRRA
```

DATIME

```
DATIME
```

This command gives the date and time. The successful response is a line containing the date and the time in the form 'DD/MM/YY HH.NN' (with leading zeros present) where DD,MM,YY,HH and NN are the day, month, year, hour and minute respectively. For example, at 2.30pm on November 8th 1977:

```
DATIME                   Returns the line
                         08/11/77   14.30
```

DEFALL

```
DEFALL,userno,diasize,dsasize
```

This command sets the default initial-allocation and subsequent-allocation sizes in the directory of the logged-on owner to "diasize" and "dsasize" respectively. The sizes must be between #1 and #FF and the subsequent-allocation size must be no bigger than the initial; if either is omitted it defaults to #1. The successful response is the null response. For example:

DEFALL,4,C0,40	Sets the default initial and subsequent allocation sizes of the owner logged-on as user #3 to #C0 and #40
DEFALL,4,80,80	Sets both default sizes to #80

DELETE

```
DELETE,userno,filename
```

This command deletes "filename". The file must have F permission and the user must have owner authority with respect to the directory in which the file is held. If the file is in use, then it becomes subliminal and will be deleted when all transactions using it are closed (see CLOSE). The successful response is the null response. For example:

DELETE,3,PROG1:L	Delete PROG1:L in the default ownername's directory (probably that of the owner logged-on as user #3)
DELETE,3,HENRY.$DOCUMENT	Deletes $DOCUMENT from HENRY's directory

FREE

FREE,userno,partition

This command gives a breakdown of the freespace in partition "partition" (partitions have a single-letter name). If "partition" is omitted, it defaults to the partition in which the logged-on owner resides. The information is presented as '***** sectors in **** extents (largest *****)' All the numbers are in decimal. For example, if in partition A where the owner logged-on as user #3 resides, there are 9083 sectors free in 117 extents, the largest of which is 2000 sectors:

```
FREE,3                        Returns the line
                   9083 sectors in  117 extents (largest  2000)
FREE,3,B                      Returns a similar line  describing
                              the freespace in partition B
```

LOGOFF

LOGOFF,userno

This command logs "userno" off the filestore system. "userno" ceases to be valid and may be issued to someone else. A user is not allowed to log-off if he has any files open. If the logged-on owner is not logged-on at any other command device, then all temporary files are automatically deleted. The successful response is the null response. For example:

```
LOGOFF,3                      Logs user #3 off the filestore and
                              deletes all temporary files if the
                              same   owner   is   not   logged-on
                              elsewhere
```

LOGON

```
LOGON,ownername,password
```

This command logs "ownername" onto the filestore system. If "ownername" is omitted, then it defaults to ANON. "password" is checked against the logon-password of the owner and must match (that for ANON always does). The successful response is the userno allocated to the owner for the duration of the logged-on session. This must be quoted in most commands, but is only accepted as valid in commands received from the same control-device as the LOGON command was received. The default ownername (see OWNER) is set to "ownername"; the quoted password (see QUOTE) is set to "password". For example:

LOGON	Logs ANON on
LOGON,PAUL,PASS	Logs PAUL on if his logon-password is PASS or null
LOGON,,PASS	Logs ANON on and sets his quoted password to PASS
LOGON,PAUL	Logs on PAUL if his logon-password is null

OPENDA

OPENDA,userno,dirn,filename,blocksize,command,data

This command is used to initiate a direct-access transaction. Depending on whether "dirn" is R, W or U, the transaction is read-only, write-only or read-write (update). The file specified by "filename" must already exist and be a contiguous file. The file permission at the appropriate authority level must be F if opening for writing or updating, at least R permission if opening for reading. "blocksize" must be either even and less than 512 bytes, or one of 512, 514, 516, 520, 528, 544, 576, 630, 768 or 1024 bytes(512+2**n where 1<=n<=512); this ensures that no block in the file crosses more than one sector boundary. If "blocksize" is omitted then it defaults to 512 bytes. Blocksizes of 512 or 1024 bytes are significantly more efficient than any other. If "command", the command-pair for the transaction is omitted, then it defaults to the command pair at which the command was received and the response sent; if "data", the outlet, inlet or outlet-inlet pair (for R, W and U) is omitted, then it defaults to the response-device, control-device or command pair respectively. See the READDA and WRITEDA commands for details of how these devices are used. The successful response is the xno of the transaction, which must be quoted in all commands which further or terminate the transaction; note that this response is sent to the twin of the device at which the command was received and not to the new report-device for the transaction. Commands quoting the xno returned are only accepted at the control-device specified when opening the file. For example:

OPENDA,4,R,SORT:1

Opens a transaction for direct-access reading of SORT:1 in 512 byte blocks, with the control-device and inlet, and the report-device and outlet the same as the pair of devices at which the command was received and the response sent

OPENDA,4,W,FRED.DATABASE,100,TTY,PUB

Opens a direct-access transaction to update FRED.DATABASE, with commands coming from the teletype and the 128 byte data blocks being received from device PUB

OPENDA,4,U,DECTAPE1,300,P15

Opens a direct-access transaction to update DECTAPE1 in 768 byte blocks, with control-device, report-device, inlet and outlet all being device P15

OPENR

```
OPENR,userno,filename,command,outlet
```

This command is used to initiate a transaction to read "filename" sequentially. The file must already exist and have been written sequentially (irrespective of its organisation). The permission at the appropriate authority level must be at least R. If "command", the command-pair for the transaction, is omitted, then it defaults to the command-pair at which the command was received and the response sent; if "outlet" is omitted, then it defaults to the report-device. See the READSQ command for details of how these devices are used. The successful response is the xno of the transaction, which must be quoted in all commands which further or terminate the transaction; note that the response is sent to the twin of the control-device at which the command was received and not to the new report-device for the transaction. Commands quoting the xno returned are only accepted at the control-device specified when the file was opened. For example:

OPENR,4,ODESSA	Opens a transaction for sequential reading of ODESSA with control-device the same as the control-device at which the command was received, and report-device and outlet the twin of the control-device
OPENR,4,TESTDATA,P15	Opens a transaction to read TESTDATA with control-device, report-device and outlet all being device P15

OPENW

```
OPENW,userno,filename,perms,iall,sall,command,inlet
```

This command is used to initiate a transaction to write "filename" sequentially. The file may already exist, in which case the permission at the appropriate authority level must be F; if it does not exist, then "userno" must have owner-authority, and further, if the file is temporary, must be logged-on as the owner. "perms", the new permissions for the file may only be specified if "userno" has owner-authority. "iall" and "sall" are the initial-allocation and the subsequent-allocation for the file; if they are omitted then the default values are used from the directory (see DEFALL); they are ignored if the file already exists and is contiguous. If "command" is omitted the it defaults to the command-pair at which the command was received and the response was sent; if "inlet" is omitted then it defaults to the control-device. See the WRITEDA command for details of how these devices are used. The successful response is the xno of the transaction, which must be quoted in all commands which further or terminate the transaction; note that the response is sent to the twin of the control-device at which the command was received and not to the new report-device. Commands quoting the xno returned are only accepted at the control-device specified when opening the file. For example:

`OPENW,6,DATA17`
Opens a transaction to write DATA17 with control-device and inlet being the control-device at which the command was received, and report-device the twin of the control-device

`OPENW,8,HENRY.TESTFILE,RNNA,C0,80,P15`
Opens a transaction to write HENRY.TESTFILE giving it permissions RNN, marking it for archiving, with initial allocation #C0, subsequent-allocation #80, control-device, report-device and inlet all being P15

OWNER

```
OWNER,userno,ownername
```

This command sets the default ownername for "userno" to "ownername". The user must be logged-on at the same device from which the command is sent. The default ownername is used whenever a filename is presented by "userno" without an explicit ownername. If "ownername" is omitted, then the default ownername is reset to that of the logged-on owner. The successful response is the null response. For example:

OWNER,3,HENRY	Sets the default ownername for user #3 to HENRY, so that filename ABC means HENRY.ABC
OWNER,3	Resets the default ownername to that of the owner logged-on as user #3

PASS

```
PASS,userno,lpassword,dpassword
```

This command alters the logon-password and the directory-password of the owner logged-on as "userno" to "lpassword" and "dpassword" respectively. If either password is omitted then that password is set to null in the directory and is matched by any quoted password, making public-authority equivalent to password-authority or owner-authority as appropriate. The successful response is the null response. For example:

PASS,A,QWERTY,P254	Sets the logon and directory passwords in the logged-on owner's directory to QWERTY and P254
PASS,2,QWERTY	Sets the logged-on owner's logon-password to QWERTY but gives anyone access to all files at password-authority

PERMS

```
PERMS,userno,filename,permission
```

This command alters the permissions of "filename" to "permission". "userno" must have owner-authority with respect to the directory in which the file is held. The successful response is the null response. For example:

PERMS,7,MYFILE,FV	Alters the permissions of MYFILE in the default ownername's directory to have owner-permission F (password-permission and public-permission are unchanged) and makes the file vulnerable (so that it will not be automatically archived)
PERMS,7,SIDNEY.ILIAD:XII,RRR	Gives read-only access to SIDNEY.ILIAD:XII to anyone and leaves the archive-indicator unchanged

QUOTE

```
QUOTE,userno,password
```

This command sets the quoted password for "userno" to "password". This password is used to match against those in directories when accessing files of other than the logged-on owner (or an alias) If "password" is omitted, then the quoted password is unset and matches nothing. The successful response is the null response. For example:

QUOTE,3,SHRDLU	Sets the quoted password to SHRDLU to gain authority with respect to directories with either password set to SHRDLU
QUOTE,3	Unsets the quoted password

READDA

READDA,xno,blockno

This command reads block "blockno" from transaction "xno", which must have been opened for direct-access reading or updating. The response is sent to the report-device and the block is sent to the outlet, both as specified in the OPENDA command. If the report device and outlet are the same device then the response precedes the data block. No terminator is sent at the end of the data block; the number of bytes sent is determined by the blocksize specified in the OPENDA command. The successful response is the null response; if the command is unsuccessful then the data block is not transmitted. For example:

READDA,1A,C1 — Reads block #C1 from transaction #1A

READFILE

READFILE,userno,filename,outlet,,terminator

This command is used to read "filename" without protocol. The "terminator" parameter defines the method of termination of the file, the values are:

0 - Send only the file (default value of parameter)

1 - Send an EOT symbol after the last symbol in the file.

2 - Send the file, followed by an extra symbol with the 9th-bit set.

If outlet is omitted then it defaults to the report-device. All other parameters are interpreted as for OPENR. The default response is the null response. For example:

READFILE,5,MYFILE — Reads MYFILE out down the report-device without further protocol. it is up to the client to decide when it has received the last byte.

READFILE,6,MYFILE:PDP,P15,,2 — Reads MYFILE:PDP out down device P15 without further protocol, and sends a byte with the 9-th bit set after the last byte of the file

READFILE,A,FILE1,,1 — Sends FILE1 down the report link, followed by an EOT character. FILE1 should not be a binary file which may contain EOT'S as data.

READSQ

```
READSQ,xno
```

This command reads the next sequential sector from transaction "xno", which must have been opened for sequential reading. The response is sent to the report-device and the block is sent to the outlet, both as specified in the OPENSQ command. If the report-device and outlet are the same device, then the response precedes the data sector. The successful response is the number of bytes of information to follow. This is usually less than 512 for the last sector. A request to read the next block when all sectors have been transmitted, generates a response of zero and no data; further requests generate fault -16:NOT ALLOWED For example.

```
READSQ,11                   Reads the next sector from
                            transaction #11
```

RENAME

```
RENAME,userno,filename,newname,permission
```

This command renames "filename" in the appropriate directory (either the ownername specified, or the default ownername) as "newname" and alters the permissions to "permission". "newname" must not already exist in the directory, nor must it have an explicit ownername since the file must remain in the same directory. When renaming a temporary file as a permanent file then the owner must have sufficient quota. "userno" must have owner-authority with respect to the directory in which the file is held. The default response is the null response. For example:

```
RENAME,3,TWEEDLEDUM,TWEEDLEDEE,V
                            Renames TWEEDLEDUM in the default
                            owner's directory as TWEEDLEDEE
                            and makes the file vulnerable (the
                            access permissions are unchanged)
RENAME,3,PDP9.DECTAPE1,$RHUBARB
                            Renames DECTAPE1 as $RHUBARB in
                            PDP9's directory leaving
                            permissions and archive indicator
                            unchanged
```

RESET

```
RESET,xno
```

This command is used to reset transaction "xno" to reread from the beginning of the file. "xno" must be open for sequential-access. The file is closed and reopened for sequential reading using the same xno. If the file was previously being sequentially written, then the new outlet is the twin of the old inlet. The successful response is the null response. For example:

RESET,4,12	Resets transaction #4 to reread from the beginning.

UCLOSE

```
UCLOSE,xno
```

This command is used to terminate a transaction unsuccessfully. In the case of a transaction open for sequential writing, an old file of the same name is not deleted and the destination file is left transient. Apart from this, it is exactly the same as CLOSE. The successful response is the null response. For example:

UCLOSE,4	Closes transaction #4 unsuccessfully

WRITEDA

```
WRITEDA,xno,blockno
```

This command writes block "blockno" to transaction "xno", which must have been opened for direct-access writing or updating. The response is sent to the report-device and the block is received at the inlet, both as specified in the OPENDA command. The response is sent before the block is read. No terminator is expected at the end of the data block; the number of bytes sent is set by the blocksize specifed in the OPENDA command. The successful response is the null response; if the command is unsuccessful then the data block is not read. For example:

WRITEDA,10,A	Writes block #A to transaction #10

WRITEFILE

```
WRITEFILE,userno,filename,perms,iall,sall,inlet,,terminator
```

This command is used to write "filename" without protocol. If "inlet" is omitted then it defaults to the control-device at which the command was received. The "terminator" parameter describes the method by which the filestore will recognise the end of the file. The values are:

0 - There is no way that the filestore may recognise the end of the file. The user must terminate and close the file by a command from the console (BIT9) the link is hung up until released in this way. (Default)

1 - Take an EOT character as denoting the end of the file; the EOT is not included in the file. Binary files should not be sent in this way as they may contain EOT symbols as data. Bytes sent after an EOT will be assumed to be commands.

2 - Accept all bytes sent down the link as data until a byte with the 9th-bit set is received, then close the file, the termination character is not included in the file.

All other parameters are interpreted exactly as for OPENW. The default response is the null response. For example:

`WRITEFILE,4,$DISCFILE,FRR`

Writes $DISCFILE without protocol, the file being transmitted through the control-device

`WRITEFILE,5,ABCD,,FF,FF,P15,,1`

Writes ABCD with allocations of #FF, the file being received from device P15 and followed by EOT

`WRITEFILE,F,FILE2,,,,,,2`

Receive FILE1 from the command link and write to disc, with default allocations and permissions; close the file and reset the link back to command mode when a 9th-bit is detected as set.

WRITESQ

```
WRITESQ,xno,blocksize
```

This command writes the next sequential sector to transaction "xno", which must have been opened for sequential writing or overwriting. The response is sent to the report device and the data sector is received at the inlet, both as specified in OPENSQ command.

The value of blocksize indicates the number of bytes that will be sent, the filestore reacts to the values as follows

blocksize < 512	: transfers that number of bytes (including 0) and prevents further transfers to the file; CLOSE or UCLOSE only allowed.
blocksize = 512	: transfer 512 bytes
blocksize > 512	: fault -4:INVALID PARAMETERS

For example:

WRITESQ,C,200	Writes the next sector to transaction #C
WRITESQ,C,24	Writes the first #24 bytes of the next sector to transaction #C and prevents further transfers to that transaction.

Single letter synonyms

Most of the commands have a single letter synonym which can be used in place of the complete command word. These are as follows:

A	OPENDA	N	WRITEFILE
B	RENAME	O	COPYFILE
C	CREATE	P	PASS
D	DELETE	Q	QUOTE
E	PERMS	R	READDA
F	FREE	S	OPENR
G	DATIME	T	OPENW
H	UCLOSE	U	RESET
I	unassigned	V	DEFALL
J	OWNER	W	WRITEDA
K	CLOSE	X	READSQ
L	LOGON	Y	WRITESQ
M	LOGOFF	Z	READFILE

Console keyboard commands

There are a number of commands available only at the console keyboard. These either give information about parts of the filestore system, control the line printer or allow errant transactions to be terminated.

The lineprinter may be referred to as a device (instead of a file) from the console only.
Its name is

LP

Commands which may refer to LP are :
KILL, RETRY, STATUS, QUERYDEV.

The command USERS lists the owner-names of all the logged-on users; STATUS will give the status of any device.

RETRY, BIT9, KILL and ABORT are progressively stronger ways of recovering a device.

The following two commands are useful when developing new systems for clients where it is helpful to know what the filestore receives from the client and the state of communications with the client.

QUERYDEV is used to monitor commands and errors for a specified device.

STATUS gives information on the current state and usage of a device pair.

KILL

KILL,device

This command is used to terminate all transactions which uses a device. A call to UCLOSE is generated for each transaction using the device as control-device, report-device, inlet or outlet. If the device (either the receiver or the transmitter or both) is in an error state, then the error state is cancelled; otherwise any transfers actually in progress on either the transmitter or the receiver are abandoned and the device reset. Never KILL TTY or LP.

A reset command is sent only if the device is currently receiving or transmitting data and is sent to the receiver or transmitter as appropriate (but not both unless both are in use). For devices in an error state the reset was sent when the error was detected. The reset should be used by the client to stop its transfer. For example:

KILL,P15	Any transactions using device P15 for anything are cancelled by calls to UCLOSE; any transfers actually in progress are halted and the device is reset.
KILL,LP	interrupt the current lineprinter listing and delete the corresponding file; start on the next listing.

ABORT

ABORT,device

This command is used to recover from a complete failure of the system using "device". Firstly all transactions using "device" for control-device, report-device, inlet or outlet are terminated by a generated call to UCLOSE. All users logged-on at the device are logged-off (but temporary files are preserved). Any data transfers actually in progress are halted, the device is reset and (for a control-device) the filestore awaits commands, as for kill. For example:

ABORT,7502	All transactions on device 7502 are cancelled, all users of the device are logged-off and the device is reset, if it was transferring.

Lineprinter control commands

These allow files sent to the printer-spooler to be transferred to the printer device or to be held within the filestore if for some (hardware) reason the device is unavailable. They also allow recovery from printer failures such as paper-jam, end of paper etc., which cause the device to go off-line and thus cause a link error detectable by the filestore.

If the printer does cause a link error then the fault should be corrected in the device. before the filestore is told to restart the tranfer. (See Device Errors - Section 1).

The following commands are specifically for printer control.

LPON

Start printing any existing printer files or print the next and following files that arrive.

LPOFF

When the current file has been printed do not start on the next (if any), save any more printer files until the LPON command is given.

LPRESET

If the printer has gone off-line then remove it from the error state otherwise stop printing the current file; continue printing the current file from its beginning.

The following commands are described elsewhere but their special effect on the lineprinter is as follows:

KILL,LP

Remove the device from the error state or stop the current listing immediately; delete the corresponding file.

RETRY,LP

Remove the lineprinter from the error state and continue printing as if no fault had occurred.

BIT9

```
BIT9,device
```

This command is used to force a termination of a WRITEFILE. Its effect is exactly as if a character with the 9th-bit set were received from the device and should be used for devices which do not have 9 bit capability. For example:

BIT9,7502	The WRITEFILE using the 7502 is terminated normally

As an emergency measure BIT9 may be used from a device other than the console, in case someone tries to WRITEFILE from the console.

QUERYDEV

```
QUERYDEV,device,code
```

This command is used to monitor commands and errors from "device".

"code" has the following values:

```
0 : no further monitoring of the device (default)
1 : monitor error messages sent to the device
2 : monitor all commands received from the device
3 : monitor all commands and error messages for the device
```

For example:

QUERYDEV,TTY	Cancel QUERYDEV for TTY
QUERYDEV,ISYS,1	any errors on ISYS are listed on the console teleprinter.
QUERYDEV,7502,2	Lists all commands sent from the 7502 on the console.

RETRY

RETRY,device

This command restarts a transfer (without loss of data) after a link error. Since only one half of the receiver-transmitter pair specified by "device" will be in the error state, the system looks at each to find an error state and an error message is given if neither are in the error state (see device errors). If both are in an error state then only one is cleared. A transmitter attempts to send the last character (which caused the fault to be detected) again, a receiver waits for the next character to be sent after the reset sent when the error was detected.

If the error still exists on the link then the device returns to the error state.

For example:

RETRY,P15	Clear error on p15
RETRY,LP	Continue the transfer to the lineprinter which stopped when the device became off-line, if the device is now back on line.

STATUS

STATUS,device

This command gives information on the current state of the device which may be of use when developing new client systems or when a link error occurs.

The information printed is of the form:

```
     device name,
     device status word,
     query status (if not 0)
             Q=1 (query commands), E=2 (echo faults)
     transaction  which  posseses  it ( SELF for acknowledge or
             command) or FREE,
     interrupt status ( IMM or AUTO),
     if AUTOmatic then

             the  command  bits  -  I=init,  R=read,   W=write,
             C=command, F=link error,
     + bytes read/written,
     - bytes left to read/write in the block
```

USERS

USERS

This command lists all users and the devices at which they are logged on. For example:

USERS

The ownernames of any logged on users, their control devices, and the number of open transactions (if any) are typed out.

SECTION 3 - ERRORS

In the event of an error, the filestore returns an errorcode and message instead of the successful response. All errorcodes consist of a line containing a minus sign followed by two hex digits. The error message is preceeded by a colon to denote its presence and is terminated by a newline, its length is not fixed. A client system may use the errorcode for its own recovery but pass the error message onto its user as a readable indication of the error. Note the use of the command QUERYDEV, which monitors error messages on the console teleprinter before the unsucsessful response is sent to the client (in case the client is not in a state to receive it).

Details of errors

-01:UNKNOWN DEVICE

A device-name occurring in the command record was not the device-name of any device known to the filestore, or is the name of a protected device. No action is taken.

-02:NOT IMPLEMENTED

An attempt has been made to use a command which has not been implemented, or has been temporarily removed. No action is taken.

-03:INVALID XNO

The xno given in the command record is not the xno of an active transaction, using the device at which the command was received as its control-device. No action is taken.

-04:INVALID PARAMETERS

One or more of the parameters in the command record has incorrect syntax or semantics, such as a filename containing two periods. This is also be caused by omitting a comma when a parameter is omitted, or omitting a parameter which must be present. No action is taken.

-05:TOO MANY TRANSACTIONS

There are too many active transactions to open any more. No action is taken.

-06:BUSY

Trying to logoff with a transaction open. No action is taken

-07:INVALID USER

The userno in a command record is not that of a user logged-in from the control-device on which the command was received. No action is taken.

-08:DEVICE NOT IN USE

The device-name specified in a BIT9 command was not being used for a READFILE or WRITEFILE operation; or the device specified in a RETRY command is not in the error state No action is taken.

-0A:FILE IN USE

An attempt has been made to rename or open a file already open for writing. No action is taken.

-0B:FILE DOES NOT EXIST

A command has been issued which requires an existing file but none is found. No action is taken.

-0C:UNKNOWN OWNER

A command has been issued which contains an ownername, or an explicit owner-part in a filename, who does not occur in the register. No action is taken.

-0D:NO AUTHORITY

An attempt has been made to do something for which the user has no demonstrable authority, such as delete someone else's file. This is also caused by quoting an incorrect password at logon. No action is taken; in the case of logon, the user is not logged on.

-0E:QUOTA EXCEEDED

RENAME - an owner's residual quota was smaller than the filesize when renaming a temporary file as a permanent. No action is taken.

CREATE - an owner's residual quota was smaller than the requested filesize. No action is taken.

OPENSQ, XFER, WRITESQ - the owner's quota was zero when allocating an extent. No action is taken.

-0F:NO SLOT FOR FILE

An owner has 63 files when creating a file or opening a file for sequential writing. No action is taken.

-10:NO SLOT FOR EXTENT

An owner has 192 extent slots in use when allocating another extent. The file remains open.

-11:PARTITION FULL

CREATE - there is no sufficiently long run of free extents. No action is taken.

OPENSQ, XFER, WRITESQ - there is no free sec

-12:TRY AGAIN LATER

An attempt has been made to use a resource which is not queued. These include large disc buffers, and the disc areas that are used when reading the pseudo-file DIRECTORY. This error is also caused if the filestore has shut itself down. No action is taken.

-13:FILE ALREADY EXISTS

An attempt has been made to create or rename a file with a name that already exists in the directory. No action is taken.

-14:NOT CONTIGUOUS

An attempt has been made to open a linked file for direct-access. No action is taken.

-15:DISC TRANSFER ERROR

A disc transfer error has occurred. If this happens on a directory read or write, then the filestore will stop. Otherwise, the transaction may be continued; in the case of sequential-access, the block is ignored and the next one used on any subsequent attempts.

-16:NOT ALLOWED

An attempt has been made to do something either silly, such as a sequential read from a transaction open for direct-access, or illegal, such as create a temporary file on someone else's filespace. No action is taken.

-17:HAZARD

Due to a catastrophic failure, such as a disc error when writing pert of the free-block bitmap, the filestore will not allow any operation which alters the state of directories. Expert helo should be summoned.

-18:DIRECTORY CORRUPT

An internal consistency check has failed. No action is taken. Expert help should be summoned.

-19:TOO MANY USERS

There are too many users logged-on to the filestore to accept any more. No action is taken.

-1A:BLOCK NOT IN FILE

An attempt has been made to access a block not in the file. No action is taken.

-20:UNKNOWN COMMAND

The command word at the start of the command line was not recognised. No action was taken.

APPENDIX - A SAMPLE SCRIPT

Given below is a sample script produced by a hypothetical wholesaler, running a self-explanatory operating-system. The filenames have been chosen so that there should be no confusion as to which of the two people who log on to the wholesaler's system are issuing which commands.

The column on the left contains the command sequence issued to the wholesaler's system with some comments in lower-case; the column on the right is the traffic between the wholesaler and the filestore. Normally, all commands would be abbreviated to their single-letter forms, but they are (mostly) written in full here for clarity.

Demonstration

```
USER: CHARLY
PASS: OLIVER              >>  LOGON,CHARLY,OLIVER
                          <<  12
Charles Dickens logged
on as user #12
EDIT PICKWICK             >>  OPENR,12,PICKWICK
                          <<  2
                          >>  OPENW,12,PICKWICK
                          <<  4
Transactions #2 and
#4 opened
                          >>  READSQ,2
                          <<  200
                          <<  512 bytes of data
>%N
>M200K7                   >>  WRITESQ,4
                          <<
                          >>  512 bytes of data
                          >>  READSQ,2
                          <<  200
                          <<  512 bytes of data
                          >> .writesq,4
                          <<
                          >> 512 bytes of data
>M50                      >>  READSQ,2
130 bytes real info       <<  82
                          <<  130 bytes of data
                          >>  WRITESQ,4
                          <<
                          >>  512 bytes of data
>E5
>%C                       >>  READSQ,2
End-of-file               <<  0
#A8 bytes data left       >>  WRITESQ,4,A8
                          <<
                          >>  #A8 bytes of data
                          >>  CLOSE,2
                          <<
```

```
                              >>   CLOSE,4
Editing complete              <<
IMP PICKWICK/$OBJ             >>   OPENR,12,PICKWICK
                              <<   3
Imp workfile                  >>   OPENW,12
                              <<   A
                              >>   READSQ,3
                              <<   200
                              <<   512 bytes of data
                              >>   READSQ,3
                              <<   200
                              <<   512 bytes of data
                              >>   READSQ,3
                              <<   200
                              <<   512 bytes of data
                              >>   WRITESQ,A
                              <<
                              >>   512 bytes of data
LOGON: WILLY
PASS: HAMLET                  >>   LOGON,WILLY,HAMLET
                              <<   13
William Shakespeare
logged on as user #13
                              >>   READSQ,3
                              <<   A8
                              <<   #A8 bytes of data
                              <<   READSQ,3
                              <<   0
                              >>   WRITESQ,A,17C
                              <<
                              >>   380 bytes of data
                              >>   CLOSE,3
                              <<
Re-input workfile             >>   RESET,A
                              <<
Open object file              >>   OPENW,12,$OBJ
                              <<   B
TOD                           >>   DATIME,13
                              <<   12.03.77     13.30
                              >>   READSQ,A
                              <<   200
                              <<   512 bytes of data
WRITESQ = Y                   >>   Y,B
                              <<
                              >>   512 bytes of data
READSQ = X                    >>   X,A
                              <<   17C
                              <<   380 bytes of data
                              >>   Y,B
                              <<
                              >>   512 bytes of data
T LK/MACBETH(30,8)            >>   OPENW,13,MACBETH,,1E,8
                              <<   D
                              >>   Y,D
                              <<
                              >>   512 bytes of data
                              >>   X,A
                              <<   0
```

```
                        >>  Y,B
                        <<
                        >>  512 bytes of data
                        >>  CLOSE,A
Workfile deleted        <<
                        >>  Y,D,83
                        <<
                        >>  131 bytes of data
                        >>  CLOSE,B
Compilation done        <<
RUN OBJ                 >>  OPENR,12,OBJ
Wrong filename          <<  -08:FILE DOES NOT EXIST
RUN $OBJ                >>  OPENR,12,$OBJ
                        <<  E
EDT LK                  >>  Y,D
                        <<
                        >>  512 bytes of data
                        >>  CLOSE,D
                        <<
                        >>  X,E
                        <<  200
                        <<  512 bytes of data
                        >>  X,E
                        <<  200
                        <<  512 bytes of data
RENAME ROMEO/JULIET     >>  RENAME,13,ROMEO,JULIET
                        <<
                        >>  X,E
                        <<  200
                        <<  512 bytes of data
T DIRECTORY/TT          >>  OPENR,13,DIRECTORY
Pseudo-file opened      <<  11
                        >>  READSQ,11
                        <<  1E6
                        <<  #1E6 bytes of data
                        >>  X,E
                        <<  83
                        <<  #83 bytes of data
                        >>  READSQ,11
                        <<  0
                        >>  CLOSE,11
                        <<
                        >>  X,E
                        <<  0
                        >>  CLOSE,E
                        <<
T CHARLY.COPPERFIELD/TT >>  OPENR,13,CHARLY.COPPERFIELD
No authority            <<  -0D:NO AUTHORITY
QUOTE MUTUAL            >>  QUOTE,13,MUTUAL
                        <<
T CHARLY.COPPERFIELD/TT >>  OPENRSQ,13,CHARLY.COPPERFIELD
                        <<  1
                        >>  READSQ,1
                        <<  200
                        <<  512 bytes of data
DELETE COPPERFIELD      >>  DELETE,12,COPPERFIELD
File becomes subliminal <<
LOGOFF                  >>  LOGOFF,12
```

```
$OBJ deleted             <<
                         >>  READSQ,1
                         <<  12
                         <<  18 bytes of data
                         >>  READSQ,1
                         <<  0
                         >>  CLOSE,1
COPPERFIELD deleted      <<
LOGOFF                   >>  LOGOFF,13
                         <<
```