GDMR

Department of Computer Science

Edinburgh University

THE FILESTORE

Hamish Dewar
Vicki Eachus
Kathy Humphry
Paul McLellan

October 1977

Revised September 1981 DRAFT

Introduction

The filestore described in this document is a stand-alone filing system which provides services to a number of computers within the Department of Computer Science at Edinburgh University. The system was introduced in 1977 and the software was modified in 1981 to accommodate communication with the client computers by means of the Departmental Ethernet-type network, as an alternative to the original form of connection utilising individual filestore-to-client inter-processor links. The storage medium is replaceable magnetic disks.

The filestore software is designed so that a fairly simple client system can make use of it, and thus provide a more comprehensive filing system to its users than would otherwise be possible. In particular the maintenance of directories and the allocation of space on the disks are entirely handled by the filestore.

As well as providing file storage services, the filestore also has the capability to spool files for subsequent transmission to attached peripheral devices.

The first section of the document describes the configuration of the filestore and the network of which it is a part.

The second section presents an overview of the facilities which the filestore provides.

The third section describes in detail how the filestore is used by clients: the protocol of requests and responses and their formats, the restrictions and defaults. It is intended mainly for those implementing systems and programs which communicate with the filestore rather than the users of such systems (for whom the first two sections together with the details of the particular implementation should be sufficient).

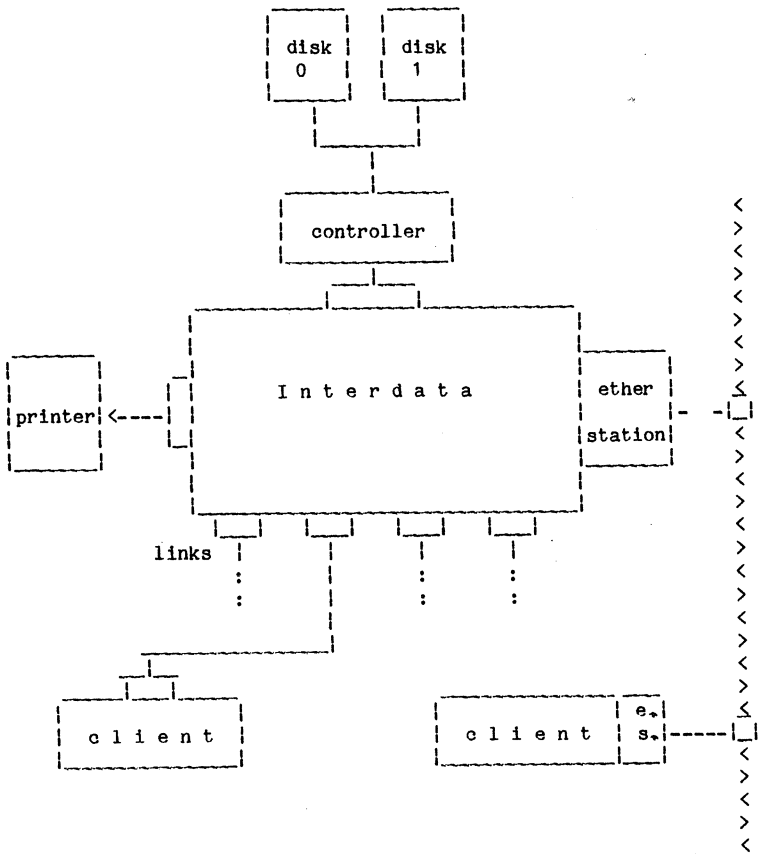
The fourth section describes the system control commands which are provided for system management.

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

There is an appendix consisting of an example dialogue indicating how the facilities are used in practice.

Hardware configuration

The processor for the filestore is an Interdata 70 with at least 32 Kb of store. The storage is on one or more disks using CDC 9762 drives with a formatted capacity of 67.4 Mb each, attached to a Systems Industries Kahili controller. The disk controller is interfaced to the Interdata by a specially built interface board. Attachment to the communicating machines is either by Departmental inter-processor Link, one receiver and transmitter pair being required for each connection, or on the Departmental Ethernet. The only spool-device currently attached is a Data Products 2260 line printer, using a standard Link interface.



SECTION 2 -- OVERVIEW OF FACILITIES

Mode of operation

The other machines to which the filestore is connected are known as clients. Some clients are in the nature of wholesalers handling the file transfers of many users simultaneously. 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.

Note the distinction between client and user. Throughout this document, clients are machines and users are people.

The various functions that the filestore can perform have been broken down into atomic operations, each of which is invoked by a single request from a client. Every request provokes a response from the filestore which is directed to the requesting client. Once a request has been received, no further requests are accepted from that client until the related response has been transmitted, so that the request/response sequence forms the basic unit of client-filestore communication. Note that all activity is initiated by clients; the filestore is a passive server.

Each request consists of a command line, followed, in the case of Write-block requests only, by a sequence of data bytes. The command line specifies the operation which the client requires and whatever parameters are needed for that operation. Responses are either successful responses or failure responses. A failure response is a line containing an error-code and a text message indicating the reason for the failure. A successful response consists of an acknowledgement line, followed, in the case of Read operations only, by a sequence of data bytes. In the simplest case the acknowledgement line is a blank line, since all that requires to be signalled is the satisfactory completion of the requested operation. In other cases it conveys one or two result values to the client. The characters making up command and acknowledgement lines are printable ISO characters, in the interests of ease of transmission and reproduction, but the formats of commands and responses are designed for machine generation and would not be particularly convenient for direct use. The characters occurring in sequences of data bytes are arbitrary 8-bit bytes.

Organisation of disk space

The basic unit of disk space is the block of 512 bytes. The total available space on each disk (less a small amount used by the filestore system itself) is divided into fixed areas called partitions. There are currently two partitions on each disk of 64640 blocks each. All allocated disk space is in the form of files. Each allocated block is in exactly one file, and each file is totally contained in a single partition.

The space allocated to a file consists of a number of areas of adjacent blocks called extents. Files are created in only one way: by opening a file for sequential output (Openw) and writing data to it. An initial extent is allocated when the file is opened, and further extents are allocated as the need arises. Any unused blocks overallocated in the final extent are released when the file is closed. The data is always written into newly allocated blocks; any existing file of the same name is not overwritten, but is deleted when (and if) the new file is successfully closed.

Owners and directories

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 disk space should not exceed. Each file is owned by exactly one owner.

For each owner the filestore maintains a directory giving details of all the files for that owner. There is a one-to-one correspondence between owners and directories.

Note the distinction between owners and users; a user is anyone currently using the filestore and may or may not also be an owner. However, it is not possible to log on to the filestore except as one of the owners.

Security and permissions

It is fundamental to the filestore design that it should in principle be possible for any user to have access to any file in the system, so that flexible sharing of data is possible. Equally it is essential that controls should be provided so that an owner may exclude or restrict access to some or all of their files by other users, and sometimes also, for peace of mind, by themselves.

The protection mechanism is based on two concepts: the level of authority exercised by a user and the access permissions associated with an individual file. Authority is established in only one way: by presenting a password to be matched against the password which is associated with each directory. This matching is performed whenever any use is to be made of a directory. If the match succeeds, the user involved is said to have owner-authority with respect to that directory; if not, to have only public-authority. Some operations require owner-authority in themselves, for example creating and deleting files in a directory and in particular logging-on as the owner, so that in these cases a failure in password matching results in rejection of the request.

However access to individual files may or may not be permitted to all and sundry (users with public-authority), depending on the access permissions set for the files individually. Each file has associated with it two permissions corresponding to the two levels of authority. These can each be at one of four levels, though they are constrained so that the owner-permission is no stricter than the public-permission. The four levels of permission are Free, Read, Obey and No. Free permission allows the file to be read, modified, 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 not to be accessed from outside. No permission allows no access at all.

The original permissions set on a new file are normally those established as the defaults for the directory in which it is created, but they may subsequently be altered (see Permit and Openw below). Only a user with owner-authority may alter the permissions of a file or the default permissions.

By logging on successfully, a user establishes owner-authority with respect to their own directory. The logon password is set as the active password to be used in determining authority with respect to other directories, so that the user may also enjoy owner-authority with respect to these. There is a command (Quote) to alter the active password to something other than the user's own logon password; this does not involve loss of owner-authority with respect to the user's own directory.

A password consists of up to six alphanumeric characters, starting with a letter. A null directory password is matched by any password whatsoever.

File naming and file types

Owners give files filenames, which must be unique amongst their own files. Hence an ownername together with a filename completely identifies a file within the system.

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 owner names:

```
ABC          KLM2                MAINT          IMP
```

A filename consists of up to twelve characters of which the first is a letter or a dollar-sign, and the remainder any combination of letters, digits or colons. In a number of client systems, the colon may have the significance of separating an extension from a root filename, but to the filestore, colon is just an additional digit. Also, some clients may in various ways restrict the range of filenames available to the user. For certain commands, a null filename is also valid. The following are valid filenames to the filestore:

```
A            LONGFILENAME        LAYOUT:XI      CAT:DOG:COW
$LIST        $T                  $WORK:1
```

A full filename consists of an ownername, a period and a filename. However there is always a default ownername in force for each user logged on to the filestore, so that in the majority of cases when a file is being identified, the ownername (and the period) can be omitted. Typically the default is the ownername of the logged-on user, but it can be changed. The following are all valid full filenames:

```
SYS.LOADER:B
ABC.IMP030277
KLM2.$LIST
KLM2.
WILSON.DIRECTORY:D
```

Files are either permanent or temporary, the choice being open to the user, as indicated by the form of filename selected. A permanent filename is one beginning with a letter; a temporary filename is one beginning with a dollar-sign. 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 remain in existence only as long as the owner is 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. It is an important feature of the filestore design that an owner may be logged-on to the filestore several times over; temporary files are only deleted at the final logoff, when the owner ceases to be logged-on at all.

Directory information

The directory associated with each owner contains details of each of the files belonging to that owner. These details are recorded in file-slots within the directory and details of each extent occupied by the file are recorded in extent-slots. The maximum number of files (F) and extents (E) is limited according to the formula: 4F + E <= 500. Thus for an average of two extents per file, the directory can accommodate about 80 files.

Associated with each file-slot are the file-name, some usage information, the date-stamp, and the attributes. The date-stamp gives the date and time (to the nearest minute) that the file was created. The attributes comprise the file permissions (described above) and an archive-status which is either Archive or Vulnerable. Files with Archive set are automatically stored on archive if they have been altered since the last archiving period. Files which have not been used for a period (to be decided) will be automatically deleted. When attributes are specified in commands or in file-information responses, they are given in the order: Owner permission, Public permission and Archive status, using the initial letter of the appropriate attribute value (F,R,O,N for permissions and A,V for archive status). Thus, for example, "FRV" denotes: Owner permission - Free; Public permission - Read; Archive status - Vulnerable.

A directory is not a file, and cannot be read by the ordinary file transfer operations. Information about the contents of directories is made available by the Finfo (file information) command.

Transient and subliminal files

A file which is in the process of being created by sequential writing is termed a transient file. When and if written successfully, it ceases to be transient. If for any reason, such as a crash in the client system, writing is not completed normally but the transaction is abandoned, the file remains in the transient state, with the content being whatever data had been written to it before abandonment. A file in the transient state may be deleted or renamed only, but is otherwise inaccessible. A transient file never supersedes an existing file of the same name, so that the user's options are left open after a non-standard termination to retain either or both. The user is cautioned that in specifying a file for deletion or renaming, a transient file is not specially identified as such, the filename simply being that specified when the file was opened. When a transient and non-transient file of the same name are both in existence, the Delete and Rename operations are guaranteed to operate on the transient one, but beware the hazard of forgetting that a transient file has been deleted and giving the same command again. Note also that some utility programs like editors buffer several blocks of a file locally, so that a transient file may not contain as much data as might be assumed.

While they are in the process of being created, 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 normally concern the ordinary user.

Subliminal files are files which are due for deletion. A file becomes subliminal if it is in use when it is deleted; a nameless file is subliminal throughout its existence. They are automatically destroyed when they cease being used. Subliminal files are inaccessible to anyone who is not already using the file.

Spool devices

Spool devices like the lineprinter attached to the filestore are not directly accessed by clients. Rather files of data intended for such a device are written to a special directory associated with the device. The ordinary methods for writing files are used for this purpose. Whenever a complete file is present in the directory of a spool device, the filestore transmits it to the device and then deletes it. When more than one file is waiting to be output, the selection of the next to be dealt with gives some preference to the shortest in the queue.

Files to be printed on the lineprinter are written to the directory LP. Any filename specified is ignored at present, so that conventionally a null filename is used, ie "LP." is specified. 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.

Control terminal

The local terminal attached to the filestore is used to enter data at system startup time and subsequently for logging and monitoring functions, and to input special control and administrative commands. Most of these functions can also be exercised from a remote client, since there is a format for presenting terminal commands from a client, although many of them require special authority and are not available to ordinary users.

The terminal is used as a monitoring device to record certain internal filestore events; e.g. disk errors, device errors and logging information. There is a command (Monitor) to invoke monitoring of commands sent from clients to provide a record of the communications between the filestore and a particular client; this can be useful when developing new client systems. The content of data packets is not monitored.

SECTION 3 - PROTOCOL

This section presents the information needed to implement a client system or utility which communicates with the filestore. As already described, the filestore provides services on receipt of requests, and always produces a response. Although only ordinary ISO characters are used, the protocol is designed for convenience of implementation rather than readability and is intended to make available to the system designer a set of primitives from which a powerful filing system capability may be constructed and presented in an appropriate form to the system user.

Requests may be regarded as falling into three categories, in terms of the context they pre-suppose; these will be distinguished as zero-level, user-level and transaction-level.

1. A command at zero level pre-supposes no context. There is no default ownername and no owner authority. This is the level at which someone who has not (yet) logged on to the filestore operates. Most requests which are valid at user level are also in principle valid at zero level, but the restrictions just noted mean that this level is normally used solely to issue a Logon request which, if successful, returns a user-number (uno). The user-number is a token for that logged-on owner which enables the user to operate at user level.

2. At user level, all requests must be accompanied by the user-number of the user requesting service. Most requests belong to this level, such as Delete, Logoff and Quote. In particular, there are three requests, Openr, Openw and Openmod, which initiate a file transfer on a block-by-block basis (from now on, a transaction) and return a transaction-number (xno).

3. The requests at transaction level are a specific set of commands (such as Readsq, Writeda and Close) which further or terminate a transaction opened by one of the Open commands just mentioned. Each request of this type must be accompanied by the transaction-number of the transaction to which the requested service relates.

A request and a response are both indivisible units for communication purposes and the sequence of the two is not interruptable by another interchange for the same client. All the data making up a request must be transmitted without significant delay and the client must be ready to accept the complete response, also without significant delay. The filestore times out any client which offends in this respect and abandons the operation.

Syntax

Each request consists of a command-line, followed in the case of the Write-block commands only (Writesq, Writeda), by a sequence of data bytes. The command line consists of a command letter followed by a reference followed by at most two additional parameters separated by commas, the whole being terminated by a newline (linefeed) character. The command letter defines the operation requested. The reference is a single character code representing either a user-number (or zero) or a transaction-number, depending on the type of command as established by the command letter. The significance of the parameters depends on the specific command but each belongs to one of the following classes:

ownername
filename
password
attributes
byte count
block count
block number

Case distinctions are ignored for the command letter and for letters in names, passwords and permissions. The single symbol used for the reference is a character in the ISO set ranging from '0', representing zero, upwards. So, for example ':' represents ten, '@' sixteen, 'A' seventeen, and 'a' forty-nine. Parameters which are numeric, such as byte and block counts, are represented in a special form of hexadecimal -- high-density hex -- which uses the same encoding of digits onto the ISO collate sequence as just described, and, although the radix remains 16, permits arbitrarily large digits (at least up to the limit of ISO printable characters). Hence 512 (decimal) is representable as P0 in high-density hex, although 200 is also a valid representation. In the narrative which follows (though not in example formats), any reference to a value in high-density hex is flagged with the symbol "#"; numeric values not so flagged are in decimal.

The following are examples of correct commands:

```
L0ABC,SHRDLU
t7WILSON.FILENAME:B,90
X3
p3qwerty
```

Successful responses always start with an acknowledgement line containing one or two high-density hex values (separated by a comma if two). In the case of data request commands only (Read and Finfo, for example) the acknowledgement is followed by a sequence of data bytes, the number of bytes being specified in the acknowledgement line.

In the descriptions below, the term 'packet' is used as shorthand for a byte-count followed by a newline followed by the specified number of data bytes.

Summary of commands

command	let	ref	param 1	param 2	response
Logon	L	0	ownername	password	uno
Logoff	M	uno	---	---	---
Delete	D	uno	filename	---	---
Rename	B	uno	filename	filename	---
Permit	E	uno	filename	permissions	---
Finfo	F	uno	ownername	file-number	PACKET
Datime	G	uno	---	---	PACKET
Pass	P	uno	password	---	---
Quote	Q	uno	password	---	---
Setdir	J	uno	ownername	---	---
Copyfile	O	uno	filename	filename	---
Readfile	Z	uno	filename	--	--- then FILE
Openr	S	uno	filename	---	xno
Openw	T	uno	filename	block-count	xno
Openmod	A	uno	filename	---	xno
Reset	U	xno	block-number	---	---
Close	K	xno	---	---	---
Uclose	H	xno	---	---	---
Readsq	X	xno	---	---	PACKET
Writesq	Y	xno	PACKET		---
Readda	R	xno	block-number	---	PACKET
Writeda	W	xno	block-number	PACKET	---
Readback	I	xno	---	---	PACKET
remote control command	]	uno	system-command		PACKET
Boot	Iso FF	-	---	---	FILE

Description of individual commands

Logon L 0 ownername , password uno

This command logs "ownername" onto the filestore system. The password presented is checked against the directory password of the owner and must match. The successful response is a user-number ("uno") allocated to the user for the duration of the logged-on session. This must be quoted in subsequent commands, and is only accepted as valid in commands received from the same client as the Logon command was received. The default directory (see Setdir) is set to "ownername"; the quoted password (see Quote) is set to "password". For example:

L0PAUL,PASS	Logs Paul on if his directory password is "pass" or null
L0PAUL	Logs on Paul if his directory password is null

Logoff M uno

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

M3	Logs user #3 off the filestore and deletes all temporary files if the same owner is not logged-on elsewhere

Delete D uno filename

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

D3PROG1:L	Delete PROG1:L in the default ownername's directory (i.e. usually that of the owner logged-on as user #3)
D3WILSON.$DOCUMENT	Delete $DOCUMENT from WILSON's directory

Rename B uno filename , newname

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. If a temporary file is renamed as a permanent file, the owner must have sufficient quota. The user must have owner-authority with respect to the directory in which the file is held. The default response is the null response. For example:

B3THIS,THAT	Renames THIS in the default owner's directory as THAT
B3KLM.TAPE,$RHUBARB	Renames TAPE as $RHUBARB in KLM's directory

Permit E uno filename , attributes

This command alters the attributes of "filename" as specified by "attributes". Either the two permissions or the archive status or both may be specified; if either is omitted, the relevant attribute is unchanged. The user must have owner-authority with respect to the directory in which the file is held. If a null filename is specified, the default attributes on the directory are altered. The successful response is the null response. For example:

E7MYFILE,FFV	Alters the permissions of MYFILE in the default ownername's directory to have Owner permission and Public permission both Free and makes the file vulnerable (so that it will not be automatically archived)
E7SID.ILIAD:XII,RR	Gives read-only access to SID.ILIAD:XII to anyone and leaves the archive-status unchanged
E7SID.,FR	Alters the default permissions on the directory SID so that files subsequently created in it will have permissions FR.

Datime G uno packet

This command is provided out of courtesy. The data part of the response packet gives the current time and date in the form:

23/12/81 19.30

Finfo F uno ownername , file-number packet

This command is used to obtain file information about a single file in the directory specified by "ownername" (default the logged-on owner). The file-number indicates which file, counting from the most recently created as 1 up to the total number of relevant files. For a user with owner-authority, all the file entries are relevant; for a user with public-authority, only those where some access to the file itself is permitted. The data part of the response packet takes the form: filename, permissions, date, time and size of the file in blocks and extents. For example,

NNTEST2:OBJ FN 25/12/81 09.45 56(3)

A null response is provided for a file-number greater than the number of (relevant) file entries in the directory.

By convention, file-number zero elicits information about the complete directory. The data part of the response packet takes the form: ownername, partition and owner-number within partition, current time and date, the number of files and extents, and the total number of blocks occupied by the files, together with the quota. For example,

ABC (1,5) at 13.12 on 24/12/81
Files: 9 Extents: 15 Blocks: 212/500

Pass P uno password

This command alters the directory password of the logged-on user to "password". If no password is specified (null), it will be matched by any quoted password, making public-authority equivalent to owner-authority. The successful response is the null response. For example:

P8QWERTY	Sets the password in the logged-on owner's directory to QWERTY
P2	Sets the logged-on owner's password to null giving anyone access to all files at owner-authority

Quote Q uno password

This command sets the quoted password for the user to "password". This password is used to match against those in directories when accessing other owners' files. If "password" is omitted (null), it will match only null directory passwords. The successful response is the null response. For example:

Q3SHRDLU	Sets the quoted password to SHRDLU to gain authority with respect to directories with password set to SHRDLU
Q3	Sets the quoted password to null.

Setdir J uno ownername

This command sets the default directory name for the user to "ownername". The default ownername is used whenever a filename is presented by the user without an explicit ownername. If "ownername" is omitted in the command, then the default ownername is reset to that of the logged-on owner. The successful response is the null response. For example:

J3ABC	Sets the default ownername for user #3 to ABC, so that, for example, filename TEST3 means ABC.TEST3
J3	Resets the default ownername to that of the owner logged on as user #3.

Copy O uno filename-1 , filename-2

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

O3ABC.TESTDATA,JIM	Copy TESTDATA owned by ABC (if permitted in read mode) into the directory of user #3
O5HAL.TESTDATA,LP.	List HAL.TESTDATA on the lineprinter.

Readfile Z u filename

This command is used to read "filename" without protocol. The successful response is a null acknowledgement followed by the content of the file. For example:

Z5MYFILE	Reads MYFILE without further protocol. It is up to the client to decide when it has received the last byte.

Boot Iso FF (12) FILE

The Boot command is provided for down-line initial loading of client systems. It consists of the single character Iso Form Feed (12), without newline. There is no acknowledgement line, the response consisting simply of a complete file designated as the boot-file for the client concerned. Before sending the file, the filestore closes any transactions open for the client (using Uclose rather than Close) and logs off any users, so that the client comes up with a clean slate.

Openr S uno filename xno , block-count

This command is used to initiate a transaction to read "filename" on a block-by-block basis. The file must already exist and the permission at the appropriate authority level must be at least R. The successful response is the transaction-number ("xno") of the transaction, and the size in blocks of the file. The "xno" returned must be quoted in all commands which further or terminate the transaction, and these are only accepted from the client from which the file was opened. For example:

S4ODESSA	Opens a transaction for reading of ODESSA

Openw T uno filename , block-count xno

This command is used to initiate a transaction to write "filename" sequentially on a block-by-block basis. The user must have owner-authority with respect to the directory involved, and, if a file of the same name already exists in that directory, the permission at owner level must be F. To write a temporary file the owner must be the logged on owner.

The permissions set on the file being created are the default permissions for the directory, unless there is an existing file of the same name in which case the permissions are inherited from the old file.

The block-count, if specified, indicates the known or estimated size of the file in blocks, which may be of assistance in allocating space for the file.

The successful response is the transaction-number ("xno") of the transaction, which must be quoted in all commands which further or terminate the transaction. Commands quoting the "xno" returned are only accepted at the client from which the file was opened. For example:

T6DATA17	Opens a transaction to write DATA17
T8ABC.TESTFILE,90	Opens a transaction to write ABC.TESTFILE with initial allocation #90

Openmod A uno filename xno

This command is used to open an existing file for modification. Both read and write operations are permitted but the size of the file cannot be altered. The successful response is the transaction-number ("xno") of the transaction which must be quoted in all commands which further or terminate the transaction. Commands quoting the "xno" returned are only accepted from the client which opened the file. For example:

A4SORT:1	Opens a transaction for modification of SORT:1

Reset U xno block-number

This command is used to reset transaction "xno" to the specified block-number, counting from zero as the first block. The successful response is the null response. For example:

U4	Resets transaction #4 to the beginning of the file.
U4A2	Resets transaction #4 to block #A2

Close K xno

This command cancels the validity of the "xno" concerned. If the transaction had been sequential output then the file concerned becomes the permanent file and any other existing file of the same name is deleted.

K4	Closes transaction #4

Uclose H xno

This command is used to terminate a transaction unsuccessfully. It differs from Close only in the case of a transaction open for sequential writing. In this case an old file of the same name is not deleted and the destination file is left transient. The successful response is the null response. For example:

H4	Closes transaction #4 unsuccessfully

Readda R xno blockno packet

This command reads block "blockno" from transaction "xno", which must have been opened for reading or modification. The byte-count in a successful response is the value #P0; and this is followed by 512 bytes of data. If the command is unsuccessful then the data block is not transmitted. For example:

R9C1	Reads block #C1 from transaction #9

Writeda W xno blockno , packet

This command writes block "blockno" to transaction "xno", which must have been opened for modification. Byte-count should be #P0, and the command line is followed immediately by 512 bytes of data. The successful response is the null response. For example:

W74,P0	Writes block #4 to transaction #7

Readsq X xno packet

This command reads the next sequential block from transaction "xno", which must have been opened for reading. The acknowledgement line of a successful response gives the number of bytes of information to follow. This is usually less than 512 for the last block. A request to read the next block when the end of the file has been reached, generates a byte-count of zero and no data. For example.

XC	Reads the next block from transaction #C

Writesq Y xno packet

This command writes the next sequential block to transaction "xno", which must have been opened for writing or modification.

A value of less than 512 for the byte-count implies that the last block in the file is being sent, prior to closing the file. After such a short block no further transfers are valid; only Close or Uclose. The successful response is the null response; for reasons of efficiency, this is sent as soon as the data has been received and the validity of the request checked, but before the block of data is written to disk. Any disk transfer errors are notified at the time the file is closed. For example:

YCP0	Writes the 512 bytes supplied as the next block on transaction #C.
Y482	Writes the 130 bytes supplied as the first 130 bytes of the next block on transaction #4 and prevents further transfers to that transaction.

Readback I xno packet

This command reads back the last block written to transaction "xno" which must be open for sequential writing. The block read back is removed from the file and subsequent Readback commands retrieve successively earlier blocks. This command is invalid once the final (short) block has been written to a sequential output file (see above Writesq). A request to read back the previous block when the file is empty generates a byte-count of zero and no data. For example:

I2	Reads the next block back from transaction #2

remote control command] uno system-command packet

Commands of this form provide a method of injecting system control commands (see next section) from a client, rather than the filestore terminal.

SYSTEM CONTROL COMMANDS

There are a number of commands which give information about parts of the filestore system, and permit the operation of the system to be controlled. These commands take the form of a command-word, followed by a space and one or more parameters separated by commas. The whole command is terminated by a newline. Although mainly intended for use from the local terminal attached to the filestore, some of these commands may also be transmitted remotely from clients, in the form described in the previous section. In a number of cases, system privilege is required when the command is transmitted from a remote client, but not when it is input at the terminal. Certain system management commands, for creating and deleting owners for example, are not described here; these always require system privilege.

Kill KILL clientname

This command is used to terminate all transactions involving a particular client.The Uclose operation is performed for each transaction opened by that client.

Boot BOOT clientname

This command has the same effect as the primitive Boot command described in the previous section, but is intended for clients which cannot emit even one character on start-up.

Monitor MON client-name , code

This command is used to control the monitoring of request/response sequences for the client specified by "client-name", on the system control terminal.

"code" has the following values:
0 : no monitoring of the client (default)
1 : monitor all commands received from the client
2 : monitor main commands, excluding data transfer requests

For example:

MON ISYS,1 Report all commands from ISYS

Users USERS

This command lists the owner names of all users and the clients at which they are logged on and the number of open transactions.

Spool-device control commands

These commands allow control to be exercised over the transfer of files spooled within the filestore to peripheral devices. At present the only spool device is the lineprinter. For example they permit files to be held within the filestore if for some (hardware) reason the device is unavailable. They also allow recovery from print 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. Printing automatically resumes after manual intervention.

If the printer does cause a link error then the fault should be corrected in the device before the filestore is told to retry or restart the transfer.

The following commands are specifically for spool-device control.

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

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

RETRY LP
Continue printing as if no fault had occurred.

RESTART LP
start printing the current file again from its beginning, after clearing any error state.

SECTION 5 - ERRORS

In the event of an error, the filestore returns an errorcode and message instead of the successful response. All error responses consist of a line starting with a minus sign followed by one high-density hex digit. These two characters are followed by a space and an error message; the length of the error message 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.

Details of errors

-2 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.

-3 Invalid transaction number

The xno given in the command is not the xno of an active transaction for the client seeking to use it.

-4 Invalid parameter <culprit>

The parameter indicated is of the incorrect form, such as a filename containing two periods.

-5 Too many transactions

There are too many active transactions to open any more.

-7 Invalid user number

The uno in a command is not that of a user logged-on for the client from which the command was received.

-: File <culprit> in use

An attempt has been made to rename or open a file already open for writing.

-; File <culprit> not found

A command has been issued which requires an existing file but none is found.

-< Owner <culprit> not found

A command has been issued which contains an ownername, or an explicit owner-part in a filename, which does not occur in the register.

-= No authority

An attempt has been made to do something for which the user has no authority, such as delete someone else's file. This is also caused by quoting an incorrect password at logon.

-> No quota for <culprit>

Rename - the owner's residual quota was smaller than the filesize when renaming a temporary file as a permanent.
Openw - the owner's quota was inadequate to allocate the initial extent for the file indicated.
Writesq - the owner's quota was seriously overdrawn when an attempt was made to allocate another extent.

-? No slot for <culprit>

The owner's directory has insufficient space to record details for the new file indicated.

-@ Too many extents

While writing a file, there is insufficient space in the directory to record another extent. The file remains open.

-A Partition full

Openw, Writesq - there is no free block available in the owner's partition.

-C File <culprit> already exists

An attempt has been made to create or rename a file with a name that already exists in the directory.

-E Disk transfer error

A disk transfer error has occurred. If this happens on a directory read or write, the filestore takes protective action starting with barring all writes to the disk involved. 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.