PFS REFERENCE MANUAL

W. W. Lichtenberger

THE ALOHA SYSTEM
University of Hawaii

Document No. M-3
May 1, 1974

Contract NAS2-6700
Department of Defense
Advanced Research Projects Agency
ARPA Order No. 1956

1.0 INTRODUCTION

This is a users' guide for a subsystem designed for long-term storage of files. The system was originally intended for temporary use during the early development of the BCC 500. Because of the turn of events, however, it is still with us and probably will be throughout the life of the 500. In some ways the use of PFS has distinctly positive benefits; the user has good control over his files, their various back-up copies, and/or previous versions of them. Its principal disadvantages are that it requires the user to remember to write out his files before logging out of the system and its slightly different file-naming conventions from that of the basic 500 system. These are really minor, however, and the user should be able quickly to adapt to the use of PFS.

Any large computing system making use of on-line access requires a means for storing and retrieving large quantities of information which must be accessed during computations. A common example of this is the storage of programs and data files which they require or generate. These files must be readily accessible, since it frequently costs much time to read them into the system over and over again from paper tape or card media. This is especially true for the remote user.

Large operating systems facilitate on-line access by providing mechanisms for naming and handling files and for storing them, usually on disk. Unfortunately, even well-developed operating systems are subject to failure, and it is likely that valuable information can be lost at such times. The reasons for such loss mostly stem from the design of the operating system's directory structures. If they are damaged by the failure, then the system may become totally confused about its files. Approaches all requiring some type of redundancy in the directory structures can be used to remedy this serious problem, but most of them can lead to system inefficiencies, and, in any event, to added system complexity.

The basic idea of PFS is to remove the permanent storage of files from the duties of the operating system and assign the file storage to a user-level sub-system. PFS is designed in a very simple, straightforward manner, removing much of the flexibility of a fancier system, but providing a much higher degree of integrity for user files.

2.0 PFS DESIGN AND USE

To accommodate PFS a portion of the disk file has been removed from the domain of the 500's operating system. PFS accesses this portion of the disk file as a classical I/O device. That is, only PFS makes decisions concerning the allocation of the storage and the absolute addresses of the records being written or read from disk. Each system user is allocated a fixed area of this storage. The user's space is divided into a number of pages, the first one of which is used for a file directory. The PFS directory page has a number of entries which are in one to one correspondence with the other pages in the user space. As files are copied into the PFS user space, the (non-directory) pages are filled on an as available basis.

When the user logs in to the 500 system, he usually finds that his files are not known by the system. Before proceeding, then, the user enters PFS and instructs it to copy the requisite files from the permanent storage area into the operating system's file storage area. He then works with his files, and before leaving the system, he reenters PFS and instructs it to copy his updated files from the operating system back onto the permanent area. Because this operation is entirely manual and requires the user's attention and consideration, and because the copying operation is relatively fast, the user's files are thus reasonably secure from system malfunctions.

WARNING

> The user must remember that it is his responsibility to copy back to PFS those files which he has changed during his computation session and that all his work will be lost if he forgets. Good practice requires that the user occasionally write out files which have changed even during a session -- in case the system should go down after some reasonable amount of work.

Satisfactory evidence of the merit of this approach is that in over seven years of operation of PFS-like systems, only a very small number of files have been lost by operating system or hardware malfunction. This should be contrasted with other systems in which the approach is to provide protection within the system itself; in such cases the losses have been much greater. Of course, the user himself can foul up, and the hardware can fail; but these situations are tolerably low in frequency.

As a matter of good practice, of course, users are reminded that it may be wise to write out a file onto the permanent storage area at regular intervals as their console sessions progress (assuming the files are changing) so that in case of a system crash a relatively small amount of their work will be lost.

Users can have access to the directories of other users in reasonablly controlled ways and can share files. They can change the accessibility of their files to themselves and to other users and can do a number of other actions related to maintenance of their own directories.

The following is a simple summary of PFS commands. The reader can easily learn to use PFS by reading it.

3.0 PFS COMMANDS

Upon entering PFS, the user sees the PFS herald, "&". The user can then type one or more commands. The commands, except for arguments like file names, are generally one-letter commands, i.e., the user types the first letter of the command and the rest of it is supplied by PFS. Most of the commands must be confirmed by typing a ".". In the following, the characters which must be typed by the user are underlined; the characters "<" and ">" are meta-symbols to designate the type of object which PFS expects.

Three types of file names are mentioned below: <file name>, <disk file name>, and <system file name>. <system file name>s use the 500 System file naming conventions. <disk file name>s have a slightly different convention: names may not be longer than thirteen characters and must consist of letters, digits, "-", and ":". The term <file name> is used in those situations where the name is both a <disk file name> and a <system file name> and must therefore conform to an intersection of both naming conventions.

It is possible under certain rules to refer to files belonging to other users, i.e., files in other users' directories. These files may be designated by their usual file name, preceded by the directory-owner name; e.g.,

@<user name>:<disk file name>

Thus, this document may be retrieved on-line from WWL's directory by referring to it as

@WWL:PFS-DOCUMENT

PFS requires that user names -- when they are supplied -- be typed perfectly. If a user makes a typing error during a user name, he can abort the command by typing control-K. The file name can have preceding characters removed by the use of control-A. As each control-A is typed, PFS echoes "^" to signify backing up one character. If the user feels it is more convenient, he can also abort the command with a control-K. Typing two control-K's in succession will cause the user to leave PFS and return to the executive language processor.

3.1 Leaving PFS

The user may get out of PFS and return to the executive language processor by typing two control-K's in succession. Alternatively he may use the PFS command:

&FINISHED.

3.2 Transfer Commands

Only two commands actually do the transfers between disk and the system required for long-term storage and retrieval: READ and WRITE. Occasionally error diagnostics may accompany a transfer command. PFS is quite conservative about disk errors at the present time and reports any errors which occur without automatically retrying. Diagnostic messages are discussed in Section 4. All such messages should be reported to hardware maintenance, and the command should be immediately repeated.

3.2.1 Copying a file from the system to PFS storage

&WRITE DISK FILE <file name>; (<reminder>)

or &WRITE DISK FILE <disk file name>. (<reminder>)
FROM <system file name>.

Copies a file from the system onto the permanent storage area. The date & hour are recorded, and the file's type is preserved. The reminder is one of two messages, (NEW) or (OLD), and serves to remind the user that he is creating a new file or writing over an old one. Either situation can occur as the result of an error on the part of the user. Note that if the user utilizes the short form of the command (i.e., terminates it with a ";") the reminder message occurs after the fact; the file will be written anyway. If the user utilizes the long form of the command (i.e., uses two "." terminators), the second "." serves to confirm the data transfer after the reminder shows the user what he is about to do. Beginning users should use the long form

of the command until they have more confidence with PFS.

3.2.2 Copying a file from permanent storage into the system

&READ DISK FILE <file name>;

or &READ DISK FILE <disk file name>.
TO <system file name>.

Inverse of the write disk file command.

3.3 File Directory Commands

The user must be able to inspect the contents of his directory from time to time. Occasionally he may forget a file name, or he may want to see when he created it or last updated it. Files also have associated a number of other parameters which he may want to inspect or to verify that a change he has just made is correct. Too, users may want to look at other users' directories. The following commands facilitate the inspection and modification of directories and items contained within them.

3.3.1 Listing the contents of a file directory

&LIST.

or &LIST FOR USER @<user name>:.

The PFS directory for the indicated user is listed. All information concerning the various files for the user is printed for those items not invisible under access protection.

3.3.2 Listing only the file names within a directory

&BRIEF LIST.

or &BRIEF LIST FOR USER @<user name>:.

Same as above except only the file names are printed. This command is frequently used when a user has forgotten the exact name of a file or is looking for the presence of a file in a directory. It is faster than the LIST command.

3.3.3 Looking at the parameters for a given file

&EXAMINE FILE <file name>.

or &EXAMINE FILE @<user name>:<file name>.

Prints all the information that LIST would print except only for the indicated file.

3.3.4 Changing a file name

&CHANGE FILE NAME <old name>.
TO <new name>.

or &CHANGE FILE NAME @<user name>:<old name>.
TO <new name>.

The use of the command is obvious. Users cannot normally change the names of files in other users' directories as the second form of the command implies above. Section 3.3 discusses file sharing and protection.

3.3.5 Deleting a file from the directory

&DELETE FILE <file name>.

or &DELETE FILE @<user name>:<file name>.

The use of the command is obvious. The space occupied by the file being deleted is available for re-use.

3.3.6 Checking the amount of available disk storage space

&UNUSED SPACE.

or &UNUSED SPACE FOR USER @<user name>:.

Prints the number of unused disk pages in the indicated user's allocation.

3.4 File Accessibility Commands: File Sharing and Protection

Files written on the disk are provided with one of several levels of access to each of three classes of users: its owner(s), the public, and to that subset of the public who may know a special password associated with the file. These accesses are called respectively, owner access, general access, and password access.

Each access class may have one of four levels: invisible (-), inaccessible (N), read-only (R), and read-write (W). When a file is first written it is given the initial accesses:

owner: W
password: - (- means here "no access")
general: N

Thus the newly-created file is read-write to its owner(s) and inaccessible to all others.

The reader will recall that the users of PFS may list the directories of other users and attempt to read or write their files. The algorithm used by PFS to check an attempted access for validity is straightforward: first a check is made to see whether the user attempting the access is an owner; if so, then is the nature of the access legal? If the ownership test fails, then does the password (if one is typed) match; and if so, is the nature of the access now legal? Finally, is the attempted access legal for general users? If the answer to all these questions is no, the access is denied. When a user lists the contents of a directory which he does not own, all files marked invisible will be skipped.

Ownership of a directory is required to be able to execute several PFS commands. Users cannot, for example, CHANGE or DELETE file names without ownership status. Ownership is also required to do those commands which change the accesses from their initial values. These commands are given below.

Files with passwords associated with them are referred to much as a general user reference is made, i.e., by specifying a user name with the file name, except the password is supplied in parentheses after the user name. PFS does not echo the password. For example, a password reference would be typed as:

@WWL(<password>):SOME-FILE

Because of the non-echoing password the file name would appear on the terminal as

@WWL():SOME-FILE

3.4.1 Changing general access to files

&GENERAL ACCESS TO FILE <file name>.
IS -/N/R/W.

or &GENERAL ACCESS TO FILE @<user name>:<file name>.
IS -/N/R/W.

The construct -/N/R/W means - or N or R or W. - means the file is invisible; general users cannot see the file in a directory listing. N stands for inaccessible; general users can see it in the directory but can't get at it. R is for read-only, and W means read-write.

3.4.2 Changing password access to files

&PASSWORD ACCESS TO FILE <file name>.
IS (<password>)-/N/R/W.

or &PASSWORD ACCESS TO FILE @<user name>:<file name>.
IS (<password>)-/N/R/W.

Same as above, but for password access. In this case the "-" stands for "no such access." It is used to remove existing password access for a file. In this case the password portion is echoed to the user as confirmation of the new password.

3.4.3 Changing owner access to files

&OWNER ACCESS TO FILE <file name>.
IS -/N/R/W.

or &OWNER ACCESS TO FILE @<user name>:<file name>.
IS -/N/R/W.

Same as above, but for owner access. If an owner wants to prevent himself from overwriting an important file, he can, for example, set his owner access to read-only. Of course, he can set it back whenever he wishes.

3.5 Pseudo-ownership and Abbreviations: More File Sharing

Most files are owned by the user creating them; they contain information of interest only to him and remain under his private jurisdiction. Some files, however, are of wider interest. While the owner may wish to retain the privilege of modifying them, others may have the right to read them. Files in this category are easily handled by means of general and password accesses. Some files may need to be written occasionally by others. Again the password access mechanism will suffice for this situation. Finally, however, there is a small number of files which require frequent read-write access by several individuals. The password mechanism becomes cumbersome if it must be used too frequently.

PFS provides a means by which several users may enjoy almost all the privileges of ownership of files in a given directory. This status is called pseudo-ownership, and users having it are called friends of the directory owner. Friends can execute all of the operations and commands the directory owner can do except modify the list of friends of the owner. There may be as many as twelve friends of any directory owner.

The list of friends may be set or modified by the following command:

3.5.1 Equating an item number to a user name

&MODIFY ITEM #<item number>.
TO USER @<user name>:.

> This command permits the owner to change the friend list for his directory. His friends are designated by the numbers 1 through 12. The friend list is given in the directory listing by the LIST command. A user name may be removed from the list by typing "-" in place of the expected user name.

The MODIFY ITEM command permits two additional items to be set to user names. Item 13 determines the identity of a user who may be referred to merely by "<", and item 14 determines who is ">". These two symbols may be used in place of a user name at any point in PFS. Thus if a given user is himself the friend of another user and finds himself doing file operations frequently in that other user's directory, the symbols "<" and ">" can greatly facilitate his references.

For example, it is much easier to type

<SHARED-FILE.

than it is to type

@LICHTENBERGER:SHARED-FILE.

Items 13 and 14 may be returned to null by means of the MODIFY ITEM command, specifying "-" in place of the expected user name.

It may be that a user has a number of files which he wants accessed by friends but others he wants to keep private. PFS does not provide a finer access control than those mechanisms previously mentioned. It is possible, however, to remedy the situation just mentioned, especially when a number of users are working on a similar project.

A fictitious user directory can be established. (This cannot be done just to accommodate a few files.) No user of the system is then the true owner of the directory, and a number of users can establish themselves as friends of the directory and interact accordingly. All that remains is to provide a means for these mutual owners to change the constitution of the friend list. To do this they must become, for the moment, the real owner. This is done by tricking PFS -- with the following command -- into assuming a new identity for the user.

The command, which detaches the user from his own PFS directory and attaches him to the new directory, is:

3.5.2 Attaching the user to a new directory

@<user name>:(<password>).

When a user enters PFS, he is given the directory associated with his name. If for some reason he wishes to assume a new identity, i.e., take over the directory of someone else, then he uses this command. But he must know the password of the new user. This password is identical to the new user's TSS password.

3.6 Miscellaneous Commands

PFS contains a few other commands which are occasionally useful. These are listed below.

3.6.1 Verifying the integrity of the user's data

&VERIFY DATA.

or &VERIFY DATA FOR USER @<user name>.

The entire set of disk pages is read one at a time for error checking purposes. If a disk error is encountered, a diagnostic message is typed giving the physical disk address and the nature of the error. Such episodes should always be reported to the hardware people for investigation. Normally, the use of this command should not be required, as all disk pages are reread upon writing, and their contents are compared with that of the original system file.

3.6.2 Typing out correspondence of file pages with physical disk pages

&STATUS.

or &STATUS FOR USER @<user name>:.

Prints the contents of the PFS directory showing the utilization of the various physical disk pages allocated to the indicated user.

3.6.3 Permanently discard physical disk page from user directory

&ABANDON FILE <file name>.

or &ABANDON FILE @<user name>:<file name>.

Ownership access is required. Command works precisely like delete file except that as the file is deleted, each page is read once for possible errors. If an error is encountered, the page involved is marked as bad and is permanently removed from the user's available space so as not to cause further difficulty for the user. The hardware maintenance people should be informed of any disk pages which are deleted as a consequence of the use of this command. The command should be regarded as an extreme measure.

3.6.4 Conversing with another user in PFS

&"<arbitrary text>

If another user links to a given user while he is in PFS, the " command causes PFS to stop interpreting commands or file names and permit the user to carry out a conversation of arbitrary length (any number of carriage returns can be included in the text). The user escapes from this mode with control-K.

4.Ø PFS ERROR MESSAGES

PFS does a considerable amount of checking for various kinds of errors. The integrity of the users' files depends on the various checks that are done. The disk file unit reports several types of data transfer errors, but PFS checks data further. Each directory as it is read is checked for correctness of content; this is also done before writing the directory after updating. Each data block written on the disk is re-read during the writing operation and compared with the correct information.

All these checks and others give rise to the following error messages and their interpretations.

NOT ENOUGH ROOM

This message signifies that the file to be written will not fit into the available space on the disk. The standard remedy is to list the directory contents. This usually reveals some junk files which can be deleted to make room.

?

This message means that the user is attempting an operation which is improper. The user may be denied the attempted access or he may have typed an incorrect file name, user name, or password. Basically the message means to think more carefully and try again.

REREAD ERROR

This message is typed during a WRITE operation if the data is somehow written incorrectly on the disk. The checking operation is similar to the read-after-write checks done by magnetic tape units. The user should repeat the WRITE command. The message includes track, band, sector, and unit information which should be reported to hardware maintenance personnel.

TSU DUMP...

This message is given by any read error detected by the disk hardware. Track, band, sector, and unit information is included in the message, and it should be reported promptly to hardware maintenance personnel. The user should repeat his command -- perhaps several times.

COMPARE ERROR

This message indicates catastrophic failure of the disk controller. It should occur with far less frequency than virtually any other message. Occurrences should be immediately reported to hardware maintenance personnel.

CAN'T WRITE INDEX

The user has performed some strange sequence of actions, or a bug in PFS, has caused the directory to become disarranged. The user should repeat his command. Failing a second time, he should leave PFS and re-enter.

INDEX FORMAT ERROR
CAN'T READ INDEX
INDEX CLOBBERED

These messages result from various checks on the directory as it is read from disk. The user should retry and if he cannot get the directory to read successfully he should contact operations personnel.

DRUM ERROR, BLOCK ...

Catastrophic system error which can never occur.