Permanent Files

Although all file I/O was accomplished through local files, there was a separate, orthogonal system that kept track of permanent files (PFs). PFs were disk files that persisted between jobs. The relationship between local files and PFs was not intuitive, and was a frequent source of confusion and complaint from users.

PFs were managed by the permanent file system, which was largely distinct from the portions of the OS that did I/O to and from disk files. The PF system gave each permanent file a system-wide unique name containing 1-40 characters. Unlike local filenames, PF names could contain any Display Code character, though most filenames contained only alphanumerics. Because the PF namespace was systemwide, most users prefixed their permanent filenames with their initials or some other identifying string.

Cataloging files

Disk files were deleted from the system at job termination unless they were first entered into the PF system. This was done via the statement:

`CATALOG, local_filename, permanent_filename [,optional parameters].`

This made local_filename permanent by creating an entry for it in the PF system under the name permanent_filename. CATALOG did not make a copy of the data; the data still resided in the same sectors on the same disk drive. CATALOG simply made a copy of the pointers to those sectors.

To access an existing permanent file, a user issued an ATTACH statement of the form:

`ATTACH, local_filename, permanent_filename [,optional parameters].`

This created a local file which was associated with the named permanent file. For the rest of this job, I/O to local_filename was done to the file previously cataloged under permanent_filename.

Permanent file attributes

In addition to the 40-character file name, permanent files had these attributes:

- A 9-character ID. This field was pretty worthless. Its main function was to help group files; you could list permanent files with a given ID.
- A cycle number, from 1-63. The default was 1. File cycles were like versions in DEC's VMS operating system (which post-dates SCOPE/Hustler, of course). You could have up to 5 different files with the same file name, with each file having a different cycle number. When you attached a permanent file, you got the highest-numbered cycle on the system unless you explicitly requested a specific cycle. The contents of the different cycles need not be related, but in practise different cycles typically were different versions of the same program or data.
file. The concept of cycles was intended to allow the easy maintenance of commonly-used files. If you had a program that many people used and you wanted to provide a new version of it, you might not be able to modify the file if it was currently in use. So, you would create a file with the same name but a higher cycle number. People currently using the older version would be unaffected, but people subsequently attaching to that filename would get the new version. The problem came when you got to cycle 63.

- Retention period. This was the number of days before the file would automatically be purged from the system. The default was 15. 999 meant the system should never purge the file.

- Permissions. There were 5 different permissions associated with a file. Each was controlled by a 9-character password. The owner of a file had very few special privileges with respect to his or her own files; only the permission to purge them, as I recall. Thus, the use of passwords on files was much more common in SCOPE/Hustler than on other operating systems. By default, permanent files were cataloged with no special permission control, which meant that anyone on the system could do anything to the file. Nevertheless, it was rare for users to mess with each other's files, and when they did, it was usually accidental.

The following password-protected permissions were available, listed with their two-character CATALOG and ATTACH parameter name:

- Read permission (RD). This allowed the user to read the file.
- Extend permission (EX). This allowed the user to permanently add data to the end of a permanent file. Oddly, you could append to a file without EX permission. However, the appended data would not be recorded in the permanent file system until an Extend operation was performed on the file. You needed EX permission to perform this Extend. It's surprising that this ghastly feature didn't cause more confusion than it did. However, most operating system utilities worked by making a modified copy of a file rather than modifying the file in place, so it was relatively uncommon to actually need to extend or even overwrite a file.
- Modify permission (MD). This allowed the user to overwrite (“rewrite”) portions of the permanent file that were already part of the permanent file system. I don't think you needed MD permission to rewrite data that had been appended to a file but not Extended. I believe that Modify permission also allowed a file to be truncated.
- Control permission (CN). This allowed the user to purge (delete) the file. I believe it was also necessary to create a new cycle of an existing file.
- Turnkey permission (TK). This redundant feature was a separate permission that essentially combined all the features of RD, EX, MD, and CN. If there was a TK password on a file, you needed to submit the TK password to attach the file. Even if you knew the RD, EX, MD, and CN passwords, or if there weren't these passwords on a file, you needed to submit the TK password to gain any access to the file.

- Passwords were used for more than just security reasons. If a file was attached with only Read permission and no other user currently had the file attached with anything more than Read permission, then multiple users could attach the file. This was called Multi-Read access. Placing CN, EX, and MD passwords on a file, but no RD password was therefore common. This allowed multiple users to use the file simultaneously.

### Permanent file operations

#### Modification

It was relatively uncommon for permanent files to be modified. Many SCOPE/Hustler programs that maintained data in files worked by making a copy of the file; the new copy was then typically

https://codex.retro1.org/ Printed on 2023/12/10 02:20
cataloged as a higher cycle number. A major exception to this was EDITOR, the standard system text editor. Of course, database programs also worked by modifying files directly. But traditional database software became available late in SCOPE/Hustler's career and was never used much.

On those few occasions in which a permanent file was appended to after being cataloged, it was necessary to perform an “extend” operation on the file in order for the appended data to become a permanent part of the file. The control card for doing this was:

```
EXTEND,local_filename.
```

However, the EXTEND control card was rarely used, because those few programs that appended to permanent files generally performed the EXTEND operation programmatically.

Overwriting existing sectors in a permanent file did not require an EXTEND-like operation. The changed portions of the local file would immediately become part of the permanent file.

### Purging

An attached permanent file could be removed from the permanent file system by means of the PURGE control card:

```
PURGE,local_filename.
```

Note that the local file, not the permanent file, was specified. The local file remained available after the PURGE card, and could be recataloged if desired. If it was not recataloged by the end of the job, the file disappeared at end-of-job.

### Backup

There were different programs to backup permanent files to tape: SELDUMP and PFDUMP. (The word “dump” was used for this operation; unfortunately, the word “dump” was also used to mean other things.) I believe that SELDUMP and PFDUMP used the same tape format.

SELDUMP was available only to Operations staff. It was run daily to backup files for all users. There was a regular backup cycle. Each day, all files since the last “dump” were backed up to tape. Periodically, a dump of the entire system was also made. Since SELDUMP (and PFDUMP) would not backup a file that was attached with anything other than read access, care had to be taken to take periodic dumps of the system when it was not available to users, to ensure that at least occasionally, all files would be backed up.

SELDUMP was a sophisticated system that managed a tape library database and decided which reel of tape to use at any time. SELDUMP could also write to multiple tape units simultaneously.

PFDUMP was used by ordinary users to back up their own (or someone else's) files. The PFDUMP control card had a complex syntax:

```
```

Retro1.org - https://codex.retro1.org/
Both programs recorded the metadata for each permanent file (including passwords, owner, etc.) in addition to the file's data.

To restore files, the PFLOAD card was used. PFLOAD could read either system dump tapes (written by SELDUMP), or user dump tapes (written by PFDUMP). PFLOAD restored not only the program data, but all other information, such as owner, passwords, etc.

Tapes written to by SELDUMP had Visual Reel Numbers (VRNs) starting with the letters “PF”(?). Tapes written to by PFDUMP had VRNs starting with “UP”. Jobs were not allowed to access tapes with VRNs starting with these special letters unless it was via PFLOAD. This was a security precaution that was necessary because you could dump someone else's password-protected file to your own tape. If you were allowed to read a dump tape with your own program, you could read someone else's file without knowing the password.