To: Distribution

From: T. H. Van Vleck

Date: January 18, 1977

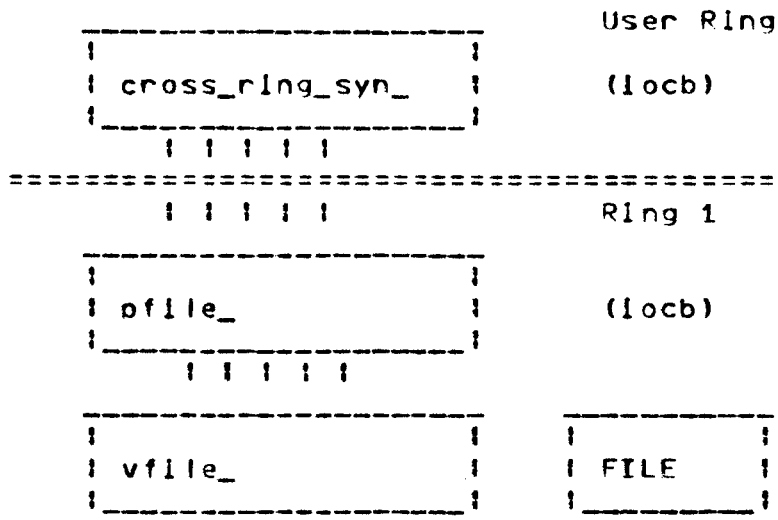
Subject: Protected Files

INTRODUCTION

This memorandum describes a new item, the "protected file," which can be implemented with modest effort. The important characteristics of protected files (pfiles) are:

- The file is organized into records.
- Per-record and per-field access control is provided.
- The file has keyed organization.
- Operations on the file are compatible with those allowed by the vfile_ module.

Pfiles are implemented as ring-1 segments.



IMPLEMENTATION

The actual operations on pfiles are done by the standard vfile_ module running in ring 1. A special splicing module is interposed in the loopath to the vfile_ attachment to perform the per-record access checking interpretively. This program and the vfile_ IOCB reside in ring 1 in order to protect them from interference.

The pfile_ I/O module could itself be a gate; but it is more general to allow pfile_ to run in any ring and to construct a cross-ring synonym attachment from the user ring to ring 1. Such a facility has been provided for in the design of iox_, and has been desired in order to simplify the I/O environment of the Initializer process; by implementing it now we provide the bases for additional system improvements.

Access Control on Each Record

The most natural method of per-record access control for Multics users is an access control list. Each record in a pfile has its own ACL, consisting of pairs of access name and mode in the standard fashion. The modes which can be specified for pfile records are different from those provided for other system objects: instead of single bits indicating permission to execute certain functions, the modes will consist of masks which define the readable and writeable fields of the record.

For example, a record in a personnel file might look like this:

ID #	Name	Title	Home phone

A: rrrrrrr wwwwwwww rrrrrrrr wwwwwwwwwwww
B: rrrrrrr rrrrrrrr wwwwwwww

The r's and w's shown above indicate hypothetical field access controls for two different modes, A and B.

A control operation allows the user to define new modes, which may then be used in ACL entries in the pfile. Each mode has a symbolic name, an ACL propagation bit, and a pair of field specifications.

The field specification describes the record as a bit string. Each field is specified by its offset from the beginning of the record, and its length. For each field, the user may have read permission and/or write permission. The description of the record is used to generate two masks: a read mask and an update

mask. When the user reads a record, those fields he does not have read access to are returned zero. When the user updates a record, any fields he does not have permission to write will retain their old values.

The ACL propagation bit in each mode controls whether a user may add ACL terms to the ACL of a record. If this bit is OFF, the user may not change the ACL. If the bit is ON, the user may add acl terms; but he may not use modes which provide access he does not have himself.

Access to the Whole File

The pfile itself will have an extended ACL providing the following modes to the file as a whole:

o open
l list
m modify

These modes may be thought of as moces to the file index.

Open access is required in order to make any use of the pfile.

List access allows a user to list all the keys in the file or to read the file sequentially.

Modify access permits the user to add or delete records, and to change all ACL's.

File Structure

The per-record ACL is pointed to by a record extension at the end of each record. The pfile_ portion of the record extension will have only a locator value which finds some space allocated in the vfile_ which is part of no record. The ACL is stored in a manner similar to that used by directory control: that is, ACL elements will consist of relative pointers to names which are separately allocated. The ACL will also contain a pointer to a mode definition structure which is also allocated.

Control Operations

The pfile_ module will accept several special control operations to manage the access control features.

create mode This operation defines a new mode

list modes This operation lists the defined modes

add ACL entry This operation adds an entry to the ACL of the current record

delete ACL entry This operation deletes an entry from the ACL of the current record

list ACL This operation lists the ACL of the current record.

set initial ACL This operation defines the initial record ACL which will be used when a record is appended to the pfile.

These control operations will be available as commands through the locall command.

Commands

We will need to have commands to support pfiles just like the commands for mailboxes and message segments.

```
pfile_add_name
pfile_create
pfile_delete
pfile_delete_acl
pfile_delete_name
pfile_list_acl
pfile_rename
pfile_set_acl
pfile_set_max_length
```

also, we will need commands which deal with the vfile_ structure of the pfile

```
pfile_adjust
pfile_status
```

and the name suffix ".pfile" must be recognized by various service routines such as delete_.

Applications

Several useful applications can be envisioned for pfiles. A few of these are described briefly below.

- Database manager. The database manager needs per-field security, controlled by a set of database administrators. Let these administrators have modify permission on the

file, and let them define modes for each record as necessary, without using the ACL propagation bit.

- Mail table. The administrative package can provide a per-system public pfile which has a record for each user of the system. These records would provide the user's mailbox pathname, dprint banner, office mailing address, etc. The user would have the ability to modify any field in his record, and to set the ACL of his record to anything he wanted.

- Tax records. A more complicated example would be the use of pfiles to store tax records. The taxpayer's records would be writeable only by the revenue service. Most of the record would be readable by the taxpayer, although the records of audits, for example, might be unavailable to him. The taxpayer, in turn, might wish to allow his attorney to see some portion of the tax return. If the taxpayer has the ACL propagation bit ON for his tax records, he may give his attorney access to all of the record except that part unavailable to the taxpayer himself. The attorney might wish to consult a tax specialist; if the taxpayer gives the attorney the ACL propagation bit, the attorney may provide access to the tax specialist without bothering the taxpayer.

Future Extensions

AIM access could be interpreted on a per-record basis. The initial implementation will not provide such a facility, because nobody has come up with a use for it.

Other access control prescripts could also be implemented for the records of a pfile, such as per-record passwords. The initial implementation will not provide these either, but the code will be written so that experiments and new features can be added later.