file - Multics
Re-Design
Proposals

TO: C. T. Clingen
F. J. Corbató
J. W. Gintell
A. Kobziar
M. Meer
N. I. Morris
J. H. Saltzer ✓
E. Stone
D. Vinograd
S. Webber

FROM: V. Voydock, R. Feiertag

DATE: January 12, 1972

SUBJECT: Work List for File System Changes

Enclosed, for your information, is a work list outlining the steps we are taking to implement a number of changes to directory control. Your comments, if any, are welcome.

Work List for File System Changes

The following is a plan for implementing a number of changes to the file system, including simplifications and improvements to the access control mechanism, and variable length file maps. These changes are described in detail in a document dated 10/29/72, entitled "New Directory Format". The reader is assumed to have read that document.

This document is an internal working document and, therefore, does not attempt to justify the order or nature of the various steps in the implementation plan. Another document is being prepared which describes for users the differences between the current file system and the file system as it will be when the plan is completed, and also explains all steps in the plan which are visible to users.

The planned changes require that the format of directories be changed. To accomplish this, the entire file system hierarchy must be reloaded (i.e., a cold boot must be performed). To lessen the probability of having to back up the system after the cold boot, file system changes which are not essential to the cold boot will be postponed until after the cold boot. An exception to this rule might be a change which requires that minor modifications be made to a large number of programs. If these programs are open for some other reason, the change may be made.

The implementation plan is divided into four parts: those changes which must (or should) be made before the cold boot, those which must be made in the system which performs the cold boot, those which will be made shortly after the cold boot, and those which will be made a long time after the cold boot.

Before going on to the plan we need to make the following definition. We say that an ACL is consistent if all of its entries have the same ring brackets. This obviously only makes sense with old style ring brackets. For example, the ACL

```
Jones.Project.*  rw 4,5,5
*.Project.*       r 4,5,5
```

is consistent. While the ACL

```
Jones.Project.*  rw 4,5,5
*.Project.*       r 4,4,4
```

is inconsistent. Let is now proceed to the details of the plan.

I. Before Cold Boot

A. Make the system libraries have consistent ACLs and change the library updating tools so that they will keep them consistent.

B. Change act_proc to put consistent ACLs on the PDS and KST.

C. Change append_branch to set default ring brackets of v,v,v (rather than v,5,5) for segments.

D. Remove all human users from the SysDaemon project. For example, Repair.SysDaemon. (See "Access Control Proposals" document for motivation.)

E. Create a new system daemon which is used only to perform the complete dump which needs to be done before a "directory reformatting" reload. (See "Access Control Proposals" document for motivation.)

F. Remove all calls to branch_info and delete it.

II. In the System Which Performs the Cold Boot

A. Write new access control primitives which know about the new directory

format. We need four sets of primitives: directory (ACL and ring brackets), segment (ACL and ring brackets). The primitives for manipulating directory ACLs will use the extended access field. The primitives for manipulating directory ring brackets will do nothing and return (this is step 1 of implementing directory ring brackets). The primitive which deletes entries from an ACL will not have an "all" option. The primitive which replaces the current ACL A of a segment S with a new ACL B will have a switch argument (call it "dsw"). If dsw = "0"b then the primitive will delete the ACL A from S, replace it with the entry "*.SysDaemon.* M" where M = REW for segments and SMA for directories and then add (i.e., merge) the ACL B to S. If dsw = "1"b the primitives will merely delete A from S and replace it with B.

B. Write write-arounds for the old acl primitives. The ACL listing and deleting write-arounds are straightforward. The others (acl_add and acl_replace) must map multiple sets of ring brackets into a single set and then call the appropriate new access control primitive. They perform the following mapping:

1. acl_replace: Let A be the ACL which is to replace the ACL currently on the segment. If A is consistent the mapping is obvious. Otherwise, let M be the minimum of the r1's of all the entries of A. If M>3 then map all the ring numbers of all the entries of A into 4. If M≤3 return an error code.

2. acl_add: Let A be the ACL which results from adding the entries passed to acl_add to the ACL currently on the segment. Treat A as described in acl_replace above.

C. Change the CACL primitives to know about the new CACL format (i.e., no ring brackets).

D. Change the appending primitives to know about the new directory format. Items changed include per segment ring brackets, allocated file map, and author. Items added include name list backpointer and tree depth. Any items in the entry which will not immediately be implemented should be set to the proper default initial value (e.g., the max length should be set to 256K even though the file system will not interpret this item until sometime after the cold boot). The primitives must place access for directories in the extended access field, and set the extended directory ring brackets to 7, 7. Finally, the entry "*.SysDaemon.* M" must be put on the ACL of every branch where M = REW for segments and SMA for directories.

E. Change the programs that know about file maps to go through the relative pointer. (segment control, salvagers)

F. Change the salvagers (online and offline) to know about the new format.

G. Change all "status" primitives (star_, status, list_dir, get_author, etc.) to know about new format.

H. Change the programs that know about current length and records used.

I. Change chname to set the pointer to the end of the name list and have name entry contain a relative pointer to branch.

J. Change the programs that reference link pathnames to know about the new format.

K. Change the programs that know about the hash table relative pointer to know the absolute location of the hash table.

L. Change hashing procedures to allow for directories larger than 128K and make hash table point directly to name instead of branch.

M. Change the programs which know about the branch count in the directory header, to keep separate counts for segments and directories.

N. Change appropriate primitives to keep metering statistics.

O. Implement variable length file maps and 256K segments.

P. Change all programs that call lock to have a cleanup procedure.

Q. Change all programs that call lock and intend to modify the directory to call a special locking program (prelude to having separate read and write locks).

R. Move various directory control data items (e.g., link depth) out of sys_info and into active_hardcore_data.

S. Make primitives check access properly and return correct error codes, especially those primitives currently checking for E permission.

T. Use new primitive to calculate directory access.

III. Immediately After Cold Boot (before users are allowed on the system)

A. Run a program which places the ACL "*.SysDaemon.* M" on the ACL of every branch, where M = REW for segments and SMA for directories.

IV. After the Cold Boot (access control)

A. Implement directory ring brackets.

B. Implement the "maximum length" segment attribute.

C. Implement the safety switch (i.e., change delentry to honor the safety switch).

D. Implement initial ACLs (the old directory mode meanings still apply).

E. Change ACL commands to use new characters for modes (SMA).

V. Long After Step IV

A. Change the system to ignore append permission on non-directory segments.

B. Change delentry to not require that the process have "w" access on the segment being deleted.

C. Turn off CACLs.