Overview of the Implementation

Jack Freeman
June 14, 1974

Technical Memo

TM.74-21

Overview of the Implementation

Goal

Our goal is to add to TENEX, with a minimum of disruption to the current code, a facility by which users may safely be allowed to run programs which use capabilities not available to the users directly. A simple example to think about is SNDMSG. We would like to be able to allow users to run this program without themselves having to have APPEND access to their correspondents' MESSAGE.TXT files, as is now necessary.

Requirements

The following things are required in order to achieve this goal.

1. A means for associating capabilities (e.g., access rights to files) with the programs to be run in this way.
2. A means for bringing a program's capabilities with it when it is activated as a fork.
3. A means for protecting a program during its execution so that any "private" capabilities it has cannot be usurped by other programs.

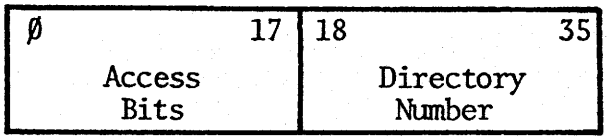
Associating Capabilities with a Program

This requires a significant modification to the sharing and protection mechanisms in the File System. Currently it is not possible to associate capabilities with any single system entity in a generally useful way. This needed facility can be gotten by the addition of an Access Control List

mechanism to the File System.

For files, we:

1. Add to the FDB data structure for each file a variable length list of Access Control Words (ACWs), each with the following format.



The Access Bits are the same 6 bits (READ, WRITE, APPEND, etc.) TENEX now uses, and the Directory Number is a regular old TENEX directory number.

2. Change the TENEX access determination algorithm to take the ACLs into account. A reasonable choice is to first compute access according to the current scheme, then compute the access conveyed by the ACL, and merge these two to get the final value. Computing the access conveyed by the ACL means searching the ACL for a directory number equal to the "current directory" or "LOGIN directory" of the fork making the access and returning the left half of the matching ACW if there is one. Note that the notion of "current" and "LOGIN" directories per fork implies additional changes to TENEX. A proposed implementation of this notion is given in an attached document. (See the memo on "Management and Protection of Directory Numbers"

3. Add a JSYS (called, say, SETACL) by which ACLs may be modified. Protection on the use of SETACL should be analogous to the protection currently enforced for the operation of changing file protection fields.

With the additions just described, it becomes possible to assign capabilities to individual "users" of TENEX. This can be used to get our required assignment of capabilities to programs by the expedient of creating a user to correspond to each program (or closely related group of programs) we're interested in.

To see how this can work, consider how we would set things up for SNDMSG. We create a "user" and a directory named <SNDMSG>. In this directory we put the SAV file for SNDMSG plus any other files it may need as it executes. We set the regular old file protection field on the SAV file to allow EXECUTE access by "the public". (Any other files in <SNDMSG> would probably be made accessible only to <SNDMSG> itself.) In every user's directory we set the regular old file protection field on MESSAGE.TXT to give whatever access is desired to the user himself and no access via the Group and Public mechanisms. We use SETACL to put a single ACW on the ACL of MESSAGE.TXT. This ACW will specify APPEND access and have the directory number corresponding to <SNDMSG> in its right half.

Bringing a Program's Capabilities with It when It's Activated as a Fork

We have seen now how to give programs access to files (and by generalization to other system objects).

The embodiment of a program in execution is a fork. Therefore, we must provide a means by which a program can bring along its capabilities when it is invoked as a fork.

This means that we must allow the fork (an active entity) to go about picking up the accesses which the program will need in order to perform its job. The program is entitled to these accesses by virtue of having its name on the ACL of various objects, but it is the fork which needs them.

Hence, we must give the fork the ability to use the program's "name" so that it may acquire the needed accesses. One way to provide for this requirement is discussed in the memo on directory numbers.

We will need a new JSYS, called, say, PGET, which will create a fork and initialize it from a SAV file. PGET should be a fairly simple variation on the current GET JSYS. The variation we're interested in at the moment is that PGET will take the "name" of the directory from which it gets the program and give it to the process.

It should be clear that we are getting close to our goal. There are more things to be done, but we stop momentarily for a survey from the point of view of our SNDMSG example. With things setup as described earlier, suppose we now invoke SNDMSG using PGET. After PGET finishes, SNDMSG will be running and will be "named" by the directory number corresponding to <SNDMSG>. It will therefore be able to append stuff to anybody's MESSAGE.TXT file because the ACLs of these files convey APPEND access to <SNDMSG>.

The remaining changes have to do with insuring that other forks running in the same job with our program aren't able to usurp, or force our programs to misuse, their private capabilities.

Protecting Processes from Other Processes

There are two things to worry about in this area.

Other forks, including superiors, must be restricted from diddling with forks. There is already fork/fork protection in TENEX in the sense that most fork JSYSes can only be used to diddle inferior forks. What we have to do is extend this protection. An implementation of this protection

extension is described in the fork protection memo.

The second thing to worry about is protecting the private capabilities used by a private fork from use by other forks in the same job. For example, when SNDMSG uses its own powers to open somebody's MESSAGE.TXT file, we would like to make sure that only SNDMSG is able to reference this file. Protection of JFNs is discussed in the JFN protection memo.