TO: Distribution

FROM: Peter Haber

DATE: May 30, 1974

SUBJECT: Proposal for a Subroutine to Find Matching ACL Names

The purpose of this document is to describe a primitive the implementation of which will simplify the access control commands. PLEASE NOTE: Handling of the star convention in access names will be changed as follows: presence of a star means specific entry, absence of component means general case. Example: "*.Multics.*" means the literal access name "*.Multics.*", but ".Multics.*" means any access name with a second component of "Multics" and a third component of "*". (This convention is currently different for different acl commands).

Current Scheme

An acl manipulating command determines matches between an acl structure and user arguments in the following manner.

1) Perform a listacl of the segment in question.

2) For each access_name argument:

a) "expand" the argument by a subroutine call to determine its type, and then perform the expansion: e.g.
argument = ".Multics."
call to subroutine, find type = 2
therefore, expanded argument =
"*"||argument||"*" =
"*.Multics.*"

b) call another subroutine with a pointer to the acl list and the expanded argument. This second subroutine turns on bits in an argument bit string to indicate which acl entries match the expanded argument.

The proposed replacement subroutine allows the same information to be gathered by a single call, with matching names

This entry returns information giving the names on an acl corresponding to the access name argument.

Usage

```
dcl find_common_acl_names_ entry (char (32) aligned,
 fixed bin, ptr, bit (1) aligned, fixed bin (35));

call find_common_acl_names_ (access_name,
count, names_ptr, identity_found, code);
```

1) access_name
is the "unexpanded" access name argument; e.g. ".Multics." (Input).

2) count
is the count of names on the acl which match access_name. Important: each entry name is returned only once. See example. (Output).

3) names_ptr
is a pointer to an array of (count) 32 character names which match access_name (Output)

4) identity_found
is ON if a precise match is found between the expanded access_name and an acl entry name (Output).

5) code
is an error code (Output).

Example

Assume the acl of >udd>m>Jones>foo, an ordinary segment, contains acl entries for

```
Jones.Multics.*
Jones.*.*
*.*.*
*.Multics.a
*.SysDaemon.*
```

the call

```
call find_common_acl_names_$find_common_acl_names_init
(">udd>m>Jones", "foo", 1, 0, code);
```

would initialize the data for subsequent calls.

the call

returned in a list structure.

Proposed Replacement

Name find_common_acl_names_

This subroutine is used to make correlations between user arguments and names on the acl of a given segment.

Entry find_common_acl_names_init

This entry is called first to initialize internal data.

Usage

```
dcl find_common_acl_names_$find_common_acl_names_init
(char (168) aligned, char (32) aligned, fixed bin, fixed
 bin, fixed bin (35));

call find_common_acl_names_$find_common_acl_names_init
(dn, en, type, ex_acl_type, code);
```

1) dn
   is the directory portion of the pathname of the segment in question (Input).

2) en
   is the entry portion of the pathname of the segment in question (Input).

3) type
   indicates the type of the segment in question
   1 => segment
   2 => multisegment file
   3 => directory
   4 => library segment
   5 => extended access segment

4) ex_acl_type
   indicates the type of extended access segment in question. (Unused if type ¬= 5)
   1 => queue message segment
   2 => mailbox message segment
   3 => vfd segment (Input).

5) code
   is an error code (Output).

Entry find_common_acl_names_

```
call find_common_acl_names_ ("Jones", ....);
```

would cause

```
count <= 2
names_ptr => "Jones.Multics.*"
             "Jones.*.*"
identity_found <= "1"b
```

the subsequent call with ".Multics" as access_name, would cause

```
count <= 1
names_ptr => "*.Multics.a"    (note: no Jones.Multics.*)
identity_found = "0"b;
```

This subroutine will allow the orderly collection of a set of names associated with an acl of any segment which acl is to be manipulated, and will eliminate some confusing and unnecessary code in the system acl commands.