Identification

Reading the SNT of another process

list_snt

D.B. Wagner

Purpose

There are instances in which it is necessary for some process A to examine another process B's address space. This may happen, for example, if a debugging aid is running in A and a program under test is running in B.

For process A to examine process B, it must be able to translate B's segment numbers into its own. Furthermore A will not always be dealing with segment numbers in B: in some cases it is dealing with "call names" in B.

To facilitate translation of segment numbers and call names across processes, the administrative ring primitive list_snt is provided. Operating in one process, it makes a copy of all the non-sensitive information in another process's Segment Name Table. Then unprivileged procedures can read this copy and freely use the information in it.

Insert: "The standard access control list on a sut guarantees that the only other processes in the same process group will be able to successfully execute this primitive or of something.

Usage

The call is:

```
call list_snt (process_id, sntp, place);
```

The parameters are declared:

```
dcl process_id bit (36),
```

```
sntp ptr,
place area ((t));
```

Process_id is the process id of the process to be examined. List_snt makes a copy of that process's SNT in the area place. It stores a pointer to this copy into sntp.

The declarations for the copy are exactly the declarations for the SNT itself as it exists at this ~~time~~ writing. However if the format of the SNT changes, the format of the copy will probably remain as it is.

[Therefore the SNT declarations, the last two pages of BD.3.01, should be inserted at this point in the final published version of this document.]

Implementation

1. The name of the process directory is obtained by feeding the given process id into the unique_chars procedure.

2. The SNT is in segment "SNT" in this process directory.

3. [At this writing it would be rather difficult to get at the SNT header, since the SMM just keeps a pointer to it in its own static storage. The SNT header could be found heuristically, but, in keeping with the Spirit of Multics, such tricks must be eschewed.]

4. Now it is only necessary to go through the name list and segment list, making copies of all Name Headers and Segment Headers, making sure that the (18-bit) pointers used in the SNT are properly relocated.