Published: 07/08/68

Identification

Stack creation: makestack
D. D. Clark, M. R. Thompson

Purpose

This section describes the segment makestack which is called by the Gatekeeper whenever it is necessary to create a stack in a given ring, and by the process initialization module to create the hard-core paged stack for a new process. The stack referred to here is one of the "call-save-return" stacks which a process uses for calling subroutines and temporary storage. This stack is described in BD.7.00; the important feature here is that there is one of these per ring, and whenever a ring is entered by a process for the first time, a new one must be created for that ring. Segment makestack will create and initialize such a stack.

Calling Sequence

```
     call makestack (ringno);

     dcl ringno fixed bin;
```

where ringno is the number of the ring for which the stack is to be created.

Implementation

Segment makestack will place a pointer to the stack in pdf$stacks + 2*ringno. The location of the stack base can then be found by accessing this location. The location in pdf$stacks for a stack which does not yet exist will contain a null pointer.

The following steps are taken by makestack:

The new stack must have a name. The following convention has been established and declared:

> For a given protection ring, n (0< n < 63), the (call-save-return) stack is named < stack_n >, where n is a character string between 00 and 63.

Having created the name, makestack sets its validation level to ringno by a call to level$set. It is now ready to call the file system primitive appendb to have a branch created for this stack in the Process Directory.

call appendb (dir, name, usermode, optsw,max1, code);

dir is the name of the process directory which is

> >process_dir_dir> concatenated with the result of calling unique_chars with the process id,

name is the stack_n created above,

usermode is the access mode of creator, and is "01011"b, which is read-write-append

optsw = `00'b (see BG.8.02 if you want to know),

max1 is the maximum length of the stack in 1024 word pages and is equal to 255, one less than the maximum allowable, to help catch overflows,

and code is for errors.

After the return from appendb, makestack resets the validation level to its previous value. If an error is returned from appendb, makestack returns without altering the null pointer in pdf$stacks, and when this is discovered by the Gatekeeper, an error is detected.

If the call is successful, estblseg must next be called to make the segment known.

```
          call estblseg (dir,entry,segsw,segptr,uid,optsw,
                         slotlist,code);
```

dir is as before,

entry is the name `stack_n' as before,

segsw =0,

segptr is a pointer to the head of the segment, returned by estblseg. uid, optsw, and slotlist are returned as declared in BG.8.04, code is for errors.

If estblseg returns correctly, makestack will take the pointer returned and load it into pdf$stacks.

The stack must now be initialized. The first frame begins at sb|8, so a pointer to sb|8 is loaded at the base of the stack, sb|0. The forward and back pointers are then set in the first frame: sb|8+16 = null; sb|8+18 = sb|8+32. The validation level (sb|3) is set equal to ringno. We now have an initialized stack for the ring designated by ringno.