Published: 9/27/66

Identification

The Process Bootstrap Module
R.L. Rappaport

Purpose

The Process Bootstrap Module is the vehicle by which processes restore themselves to core subsequent to having been unloaded.

Introduction

All processes that are not running contain a history, in their respective Process Concealed Stacks, of a call to entry point Swap-DBR, in the Process Switching Module. In the course of events, a process that is not running may be unloaded. (See Section BJ.10.03). When a process is unloaded the Process Concealed Stack is removed from core storage, along with a few other basic parts of the process. At some subsequent time some other process, operating in Swap-DBR, will give control of a processor to the unloaded process, causing the unloaded process to itself begin operating in the second half of Swap-DBR. (See Section BJ.5.01). However the unloaded process will not have available the call history of how it arrived in Swap-DBR originally. The unloaded process will then call the Process Bootstrap Module in order to retrieve this information, along with other needed information removed when the process was unloaded, and the process will not return to the second half of Swap-DBR until it has completely restored itself to core.

Description

The description of the Process Bootstrap Module which follows is only meaningful if the reader is acquainted with the definitions of the unloaded, loaded, and active states of processes. For a complete description of these ideas see Sections BJ.10.00 through BJ.10.04. A brief summary of pertinent facts from these documents is listed here. An unloaded, active process has only four pieces of information in core memory.

1. An entry for itself in the Active Process Table.
2. An entry for itself in the Process Segment Table.
3. An entry for its Known Segment Table in the Active Segment Table.
4. An entry for its hardcore ring pageable stack in the Active Segment Table.

When a process begins the loading operation it has, in addition to the above:

1. A hardcore ring Descriptor Segment with segment descriptor words for standard versions of all hardcore supervisor modules.

2. An initial Interim Process Data Segment.

Both these segments are gifts of the process which preceeded this loading process in the use of this processor.

A completely loaded process has at least the following in core, in addition to the items mentioned for unloaded processes:

1. A hardcore ring Descriptor Segment with segment descriptor words for this process' versions of all hardcore supervisor modules.

2. The complete Process Data Segment which includes the Process Concealed Stack.

3. The page table for this process' Known Segment Table.

4. The page table for this process' hardcore ring pageable stack.

5. Complete wired down copies of all non-standard versions of hardcore supervisor modules used by this process.

The Process Bootstrap Module is called by loading processes, operating in the second half of Swap-DBR. The calling sequence is:

```
call bootstrap (pointer);
```

where pointer is the pointer to the process' entry in the Process Segment Table. The stack used on the call to the Process Bootstrap Module is the Interim Process Concealed Stack, which is contained in the Interim Process Data Segment.

The purpose of the Process Bootstrap Module is to enable a process to restore, to core, the segments listed above that are needed before a process can be considered completely loaded. The Process Load Module in Segment Control (See Section BG.3) provides a group of primitives, which if called, retrieve specific segments belonging to a process. The Process Bootstrap Module, essentially, does nothing more than exercise these primitives.

Upon entering the Process Bootstrap Module, a process calls entry point loadprocess in Segment Control. The stack used in this call is the Interim Process Concealed Stack, which is wired down and is of given finite length. Loadprocess establishes in core, page tables for the process' Known Segment Table and hardcore ring pageable stack. The procedure uses the physical locations of the files for these segments, which are accessible through

their respective Active Segment Table entries, for the retrieval. It is the maintenance of these entries that allows Segment Control to retrieve these two segments using only a given finite length of stack. Return from loadprocess implies that the respective page tables are in core. Therefore the Process Bootstrap Module switches stacks to allow itself to use the hardcore ring pageable stack. Now using this stack the process calls entry point loadsegments, also in Segment Control, which proceeds to load and wire down all the extra segments that this process needs, including its own Process Data Segment. Return from this procedure implies that all these segments are in core. Therefore, the job of the Process Bootstrap Module is nearly complete. It needs only to switch stacks so that it is using the Interim Process Concealed Stack again and to perform a return to Swap-DBR, in order to finish. Figure 1 is a flow diagram of the Process Bootstrap Module.

It should be noted that loading processes should not be unloaded. This is because the Interim Process Data Segment does not have a residence on secondary storage. Hence there is no place to page the information in this segment. The unloading of loading processes is prevented by the existence of the process loading switch in the Active Process Table entry of the respective loading processes. This switch is on during the entire time in which the process cannot be unloaded.

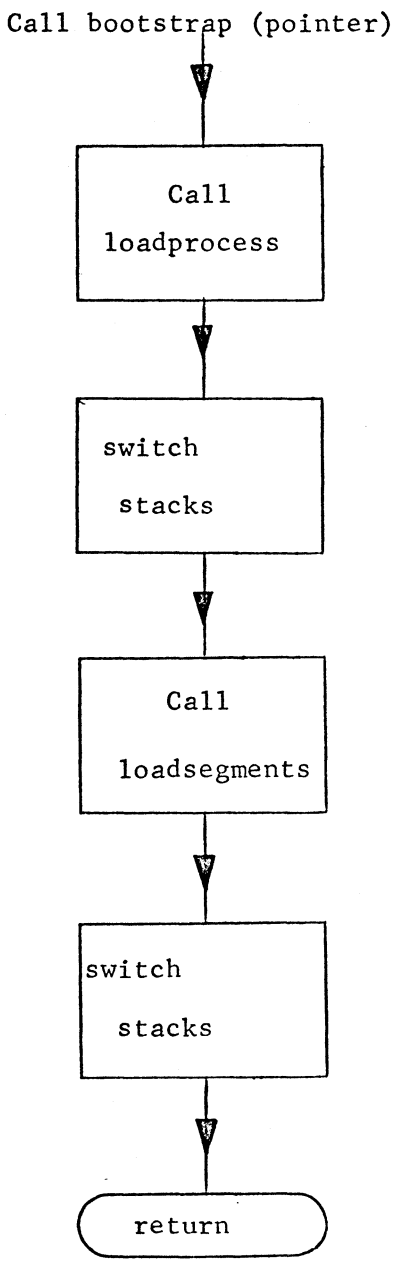

Figure 1. Flow diagram for Process Bootstrap Module.