Published: 03/17/67

Identification

Segment Loader (initial version)
D. H. Johnson

Purpose

This section describes the segment loader which is used by the Initializer Control Program (MSPM BL.5) during Multics system initialization to input information from the Multics System Tape (MSPM BL.1). The segment loader reads both initialization and hard-core supervisor segments from the tape. Each time the segment loader is called it first reads a load list segment, which contains the names of the remaining segments to be loaded during that particular call to the loader. The initial version of the segment loader, which is described in this document, does not provide the flexibility of selecting one of several possible Multics systems from the same tape. Rather, the choice of Multics system modules is made at tape generation time since initially only one system is placed on the tape. However, the design of the initial segment loader is influenced by the goal of eventually allowing the system operator to select the modules on the tape which will comprise the system being initialized. This goal should be kept in mind when considering the load list concept described below.

Introduction

The segment loader is called by the Initializer Control Program several times during system initialization to load segments from the system tape. The initialization environment changes as segments are loaded and various modules of the Multics supervisor become operable. The segment loader must interface properly with the initialization environment each time it is called. In order to assure this the following strategy has been adopted.

1. A single magnetic tape input procedure is used by the segment loader throughout the execution of the Initializer Control Program. This input procedure is responsible for reading physical records from the tape and moving a requested number of words from the physical record buffers to the area specified by the caller (see MSPM BL.6.02).

2. The segment loader calls the tape reader to move information into its work area and also directly into other segments using the standard appending addressing. Segment and page faults are taken by appropriate fault handlers to

create page tables and assign hyperpages. The segment and page fault handlers change during initialization as the file system gets initialized but the segment loader is not aware of the change.

Before the segment loader may be called for the first time the Initializer Control Program must be in the following state. It has been placed in this state by the Bootstrap Initializer (MSPM BL.4).

1. All of the initialization segments that are needed for the segment loader to work have been loaded. The segment loading table (MSPM BL.2.01) contains entries for these segments.

2. The intersegment linkage mechanism is functioning (MSPM BL.7.01).

3. A missing segment and page fault mechanism is working which manages core storage but does not swap pages between core and mass storage (see MSPM BL.6.03).

4. The magnetic tape input procedure to read the Multics System Tape has been initialized.

Segment Loader

The segment loader is called by the Initializer Control Program in the following manner.

```
call segment_loader ;
```

There are no arguments since the loader first reads a load list to determine which segments to input. When the segment loader returns control, all of the segments noted in the load list have been read into core and entered into the segment loading table. Any error discovered during segment loading causes system initialization to stop.

The segment loader is aware of the logical arrangement of information on the Multics System Tape. It uses the magnetic tape input procedure to request logical items. The logical format of the system tape is described in MSPM BL.1.01 and should be referenced for definitions of terminology used below.

Each time the segment loader is called two collections are read from the system tape. The first collection contains a load list while the second collection contains the segments noted in the load list. For the initial version of the segment loader, the collection containing load list information consists of only one segment unit. The logical header provides descriptive information for making the load list a segment. The logical segment following the header contains symbolic names of the other segments to be loaded. The second collection consists of several segment units. There are at least as many segment units as there are segment names in the load list.

The steps taken by the segment loader are outlined below.

1. The tape is assumed to be positioned at the beginning of a load list collection.

2. The load list segment unit is read and becomes a segment of the initialization program. The loader obtains the logical header and calls the procedure slt_manager$build_entry (described in MSPM BL.2.02) to construct an entry in the segment loading table for the load list segment. The number assigned to the segment is returned to the loader. The loader then causes the logical segment to be copied into the numbered segment.

3. The tape is advanced to the next collection by skipping logical records until a logical mark is sensed.

4. Segment units are read and loaded as either hard-core supervisor or initialization segments. For the initial version of the segment loader, the load list simply represents a complete, in the sense described below, table of contents of the segments following in the next collection on the tape. The load list contains only names of text segments. If a text segment has an associated linkage section segment, which is indicated in the header of the text segment, the linkage segment unit is expected to be immediately after the text segment unit on the tape. The linkage section segment name must not be in the load list. Furthermore, the relative order of the segment units on the tape is expected to be the same as the appearance of their names in the load list.

The algorithm for loading segment units is the following:

a. Read a logical header.

b. If the logical header describes a segment named in the load list, go to step d.

c. Read and ignore a logical segment, then go to step a.

d. Call procedure slt_manager$build_entry to construct an entry in the descriptor segment. (The procedure examines the logical header to determine whether it describes a hard-core supervisor or initialization segment and enters the segment accordingly.)

e. Read a logical segment into the segment reserved by the procedure in step d.

f. Test the logical header to see whether the segment has an associated linkage section.

g. If the loaded segment does not have a linkage section, go to step i.

h. Read a logical header and go to step d.

i. If the loaded segment is not a linkage section, go to step k.

j. Update the text-linkage section segment number items in the SLT entries for the last two segments loaded. Set the definition pointer in the header of the linkage section segment just loaded.

k. Test whether all of the segments named in the load list have been loaded.

l. If there are more segments to load, go to step a.

m. Advance the tape to the beginning of the next collection.

n. Return to the Initializer Control Program.