Published: 10 April 1967

Identification

Segment Loading Table Manager
D. H. Johnson

Purpose

The segment loading table (SLT) is accessed directly by Multics Initializer procedures to obtain information about segments. However, there are several frequently used examinations of the SLT which involve more than a simple one item reference. The SLT manager includes utility procedures which provide these queries. Also included is a procedure to add an entry to the SLT.

Segment loading table manager procedures

The segment loading table manager consists of the following five procedures.

1. build_entry - Given a logical header record describing a segment, this procedure assigns a segment number and constructs an SLT entry.

2. get_seg_ptr - Given a segment name, this procedure returns a pointer to the base of the segment.

3. get_seg_name - Given a pointer to a segment, this procedure returns the name of the segment.

4. get_text_seg_ptr - Given a pointer to a linkage section, this procedure returns a pointer to the base of the associated text segment.

5. get_link_seg_ptr - Given a pointer to a text segment, this procedure returns a pointer to the associated linkage section.

These procedures are described in more detail below. The following PL/I statement defines the parameters to all of the SLT manager procedures.

```
dcl seg_ptr ptr,        /* pointers to segments */
    text_seg_ptr ptr,
    link_seg_ptr ptr,
    seg_name char(*),   /* segment name */
    er_ret label,       /* error return */
    header ptr;         /* pointer to logical header array */
```

The logical header array is similar to an SLT entry (see MSPM BL.2.01). It consists of:

```
entry words 1 - 5
number of segment names
segment names (8 words for each name)
size of path name in characters
path name
```

1. call slt_manager$build_entry(header,seg_ptr,er_ret) ;

This procedure builds an entry in the SLT for the segment described by the data at argument header. The information at header is in the format of a logical header record on the Multics System Tape (MSPM BL.1.01). This information indicates whether the segment it describes is an initialization or hard-core supervisor segment. Either the SLT item last_sup_seg or last_init_seg value is incremented by one and used as the segment number (and SLT index) for the segment. The information at header is copied into the SLT entry exactly as it appears except for the segment names and path name. They are placed in the auxiliary segment name_seg and pointers to the names are stored in the SLT entry. The segment number is used to construct and return a pointer to the base of the segment in seg_ptr. If the SLT entry can not be built, control returns to the label er_ret.

2. call slt_manager$get_seg_ptr(seg_name,seg_ptr,er_ret) ;

This procedure attempts to return a pointer to the base of a segment identified by the name given in the argument seg_name. It searches the SLT entries in turn looking for the specified segment name. If the name is found, the SLT entry index is used to construct a pointer to the base of the segment. The pointer is placed in seg_ptr. If the name could not be found in the SLT, control transfers to the label er_ret.

3. call slt_manager$get_seg_name(seg_ptr,seg_name,er_ret) ;

This procedure checks whether the segment number given in seg_ptr has an SLT entry. If it doesn't, control returns to the label er_ret. Otherwise, the SLT entry is examined and the first name of the segment is placed in argument seg_name.

4. call slt_manager$get_text_seg_ptr(link_seg_ptr,
text_seg_ptr,er_ret) ;

This procedure is given a pointer to a linkage section and is requested to return a pointer to the associated text segment. The SLT entry specified by the linkage section segment pointer is examined.

If the entry item link_sect_sw is OFF, the segment is not a linkage section. Control transfers to the error return, er_ret.

If the linkage section segment is not one of the special combined linkage segments as indicated by item link_sect_status, a pointer to the text segment is constructed from the SLT entry item

text_link_segno. Control then returns to the caller.

If the linkage section segment is a combined linkage segment for active, loaded, or wired down hard-core supervisor segments, the definition pointer is obtained from the header in the linkage section pointed to by link_seg_ptr. The definition pointer is assumed to point to either the text segment or the original linkage section segment that was loaded from the system tape. The SLT entry for the segment referenced by the definition pointer is examined. If the entry indicates that this segment has combined linkage, i.e., linkage_sw is set ON, and its link_section and link_offset entry items match link_seg_ptr, then the definition pointer is used to construct a pointer to the base of the text segment. If, however, the SLT entry pointed at by the definition pointer indicates that the segment is a normal linkage section, i.e., link_sect_sw is set ON, the text_link_segno item is used to construct a pointer to the text segment. If the SLT entry pointed to by the definition pointer fails these tests, control is transferred to the error return, er_ret.

5. call slt_manager$get_link_seg_ptr(text_seg_ptr,
link_seg_ptr,er_ret) ;

This procedure attempts to return a pointer to a linkage section given a pointer to its text segment. The SLT entry referenced by text_seg_ptr is examined. If it doesn't have a linkage section associated with it, i.e., linkage_sw is set OFF, the error return, er_ret, is used. Otherwise, the combine linkage switch, combine_sw, is tested to see if the segment's linkage is part of a combined linkage segment. If it is, the link_section and link_offset items are used to construct a pointer to the linkage section. If the text segment only has the original linkage segment, a pointer to it is constructed using the text_link_segno item.