Published: 09/29/67

Identification

Segment Loading Table
D. H. Johnson

Purpose

The segment loading table (SLT) is an initialization table maintained by the segment loading table manager (MSPM BL.2.02). The SLT is itself a segment of the Multics Initializer. This table contains an entry for each initialization and hard-core supervisor segment which is currently known during system initialization. Each entry contains many items which describe the segment. The SLT is created by the Bootstrap Initializer (MSPM BL.4) and expanded as segments are loaded from the system tape by the segment loader (MSPM BL.6.01).

Contents of the Segment Loading Table

The SLT contains an entry for each segment that is part of the Multics Initializer or hard-core supervisor. Each entry consists of descriptive information about the segment. The information for each segment is either obtained from the segment's logical header record on the Multics System Tape or it is calculated during initialization. The entry items are given below.

Items obtained from the system tape

1. Segment names - This item is a list of all of the names of the segment.

2. Path name - This item is the directory path name for the segment in the file system hierarchy.

3. Maximum length - The maximum size of the segment is given in units of 1024 words.

4. Current length - The current size of the segment is given in units of 64 words.

5. Access - The descriptor access control field bits are stored in this item.

6. Status - This item indicates the status of the segment <u>after</u> initialization. It is one of the following:

wired down - A segment of this status must be in core at all times.

loaded - A segment of this status must be active and loaded (page table in core) at all times although the segment itself may be read in and out of core as needed.

active - A segment with this status must remain active (AST entry provided only) but need not remain loaded (i.e., the page table may be removed).

normal - A segment of this status requires no special consideration and is handled as a normal Multics segment.

7. 64 word paged switch - This switch indicates whether the page size is 64 (ON) or 1024 (OFF) words.

8. Hyperpage size - The hyperpage size is given in units of the page size.

9. Initialization switch - This switch indicates whether the segment is part of initialization (ON) or the hard-core supervisor (OFF).

10. Per-process switch - This switch is ON if the segment is a per-process segment rather that a per-system segment.

11. Descriptor segment switch - This switch is ON only for the segment loading table entry describing the descriptor segment.

12. Linkage segment provided switch - This switch is ON if the segment has an associated linkage section segment.

13. Linkage section switch - This switch is ON if the segment is a linkage segment.

14. Combine linkage switch - If this switch is ON, the linkage section associated with this segment may be combined with linkage of other segments of the same status.

15. Linkage section status - If the linkage section switch is ON, this item indicates the status of the linkage section. There are five types of linkage segments. They are:

normal - The linkage section was produced by a programming language translator. Most linkage sections are of this type.

combined - This linkage segment is constructed by the pre-link module (MSPM BL.7.02). It is one of the following four:

combined wired down linkage - linkage information for wired down hard-core supervisor segments

combined loaded linkage - linkage information for loaded hard-core supervisor segments

combined active linkage - linkage information for active hard-core supervisor segments

combined out reference linkage - linkage information for references from the hard-core supervisor to segments in outer rings of protection

16. Enforced access switch - This switch indicates if ON, that the segment has the access given in item 17 if referenced in outer rings of protection. If the switch is OFF, the normal protection mechanism is used.

17. Enforced access - This item only has meaning if the enforced access switch is ON. It then contains the descriptor access control field to be used for the segment in all outer rings of protection. The descriptor access control field for the hard-core ring is contained in item 5.

18. Temporary segment switch - If this switch is ON, the core used by this segment may be released at an appropriate time during system initialization. This will be done before the core map is updated (MSPM BL.10.02).

Items set after segment loading

1. Combined linkage section segment number - If the combine linkage switch is ON, this item gives the segment number of the combined linkage.

2. Combined linkage offset - If the combine linkage switch is ON, this item gives the offset in the combined linkage segment of the beginning of this segment's linkage section.

3. Text or linkage segment number - This item connects the text and linkage section segment entries. If the segment has a linkage section segment, this item gives the segment number of the linkage section that was loaded from the system tape. If the segment is a linkage section, this item gives the segment number of its associated text segment.

4. Pre-linked switch - If this switch is ON, the segment has been pre-linked (MSPM BL.7.02).

5. AST pointer - This item is a relative pointer to the AST entry created for this segment by the file system initializer (MSPM BL.10.02).

PL/1 Implementation of the SLT

The segment loading table consists of a table of fixed length entries and a list of symbolic names. A pointer variable 'sltp' is provided in static storage which points to the base of the SLT. The SLT contains a pointer 'name_seg_ptr' which points to the list of symbolic names. The PL/1 declaration is given below.

```
/* PL/1 declaration for the segment loading table */

dcl 1 slt ctl(sltp),        /*SLT header (7 words) */
    2 name_seg_ptr ptr,     /* words 0-1, pointer (ITS pair) to
                               name segment */
    2 entry_length fixed bin(18), /* word 2, number of words
                                     per entry */
    2 first_sup_seg fixed bin(18), /* word 3, first supervisor
                                      segment number */
    2 last_sup_seg fixed bin(18), /* word 4, last supervisor
                                     segment number */
    2 first_init_seg fixed bin(18), /* word 5, first initializer
                                       segment number */
    2 last_init_seg fixed bin(18), /* word 6, last initializer
                                      segment number */
    2 seg(131071),          /* segment entries (5 words each) */

   /* entry word 1  */

    3 names_ptr bit(18),    /* relative pointer to segment
                               names  */
    3 path_ptr bit(18),     /* relative pointer to directory
                               path name */
   /* entry word 2  */

    3 max_length bit(8),    /* maximum segment size */
    3 access bit(6),        /* segment descriptor access control
                               field */
    3 status bit(3),        /* segment status */
                            /*  000 = normal
                                001 = active
                                010 = loaded
                                100 = wired down  */
    3 cur_length bit(12),   /* current segment size */
    3 init_sw bit(1),       /* initialization segment switch */
    3 page_64_sw bit(1),    /* 64 word paged switch */
    3 temp_sw bit(1),       /* temporary segment switch */
    3 process_sw bit(1),    /* per-process switch */
    3 linkage_sw bit(1),    /* linkage segment provided switch */
    3 combine_sw bit(1),    /* combine linkage switch */
    3 access_sw bit(1),     /* enforced access switch */

   /* entry word 3 */

    3 hyperpage_size bit(8),    /* hyperpage size */
    3 ds_sw bit(1),             /* descriptor segment switch */
```

```
      3 pre_linked_sw bit(1),      /* pre-linked switch */
      3 link_sect_sw bit(1),       /* linkage section switch */
      3 link_sect_status bit(3),   /* linkage section status */
                                   /* 000 = normal
                                      001 = combined wired down
                                      010 = combined loaded
                                      011 = combined active
                                      100 = combined out refs */
      3 dummy bit(4),              /* dummy quantity for padding */
      3 text_link_segno bit(18),   /*text - linkage segment number*/

    /* entry word 4 */

      3 link_section bit(18), /* combined linkage section
                                 segment number */
      3 link_offset bit(18),  /* relative pointer
                                 to combined linkage information */
    /* entry word 5 */
      3 enforced_access bit(6), /* descriptor access control
                                   field in outer rings */
      3 asterp bit(18);         /* relative pointer to AST entry */



/* auxiliary segment of SLT for storing of segment names and
    directory path names  */

dcl 1 name_seg ctl(names_ptr),
                              /*name segment header */
      2 next_loc fixed bin(18); /* word 0, next available free
                                   location in name segment */


dcl 1 segnam ctl(namep),    /* segment name block */
      2 count fixed bin(17), /* count of segment names */
      2 name(namep - g segnam.count) char(32);
                           /* 8 word name block for each name */


dcl 1 path ctl(pathp),      /* directory path name block */
      2 size fixed bin(17),  /* size of path name in characters */
      2 name char(pathp - g path.size);
                           /* directory path name */
```