Published: 04/25/67

Identification

Multics System Tape Generator
V. B. Nguyen

Purpose

The Multics system tape generator is used to produce the Multics system tape. It takes all segments needed for the Multics system tape from the Multics segment library and writes the output tape in the Multics Standard Magnetic tape format. (This format is described in MSPM section BB.3.01.) It obeys the conventions of the logical format of the Multics system tape (MSPM Section BL.1.01).

This section describes the Multics System Tape Generator (MSTG) program and its first implementation on the 6.36 environment.

Discussion

All segments needed by the Multics Initializer must be placed on the Multics System Tape (MST), and a list of these segments must be provided for the MSTG. This list is called the Multics System Tape Segment List (MSTSL).

Contents of the MSTSL

The MSTSL is a character stream file containing an entry for each segment to be put on the MST; the order of the entries is in respect to the order of the segments on the MST.

This entry is called header entry and must contain 18 items which are described in MSPM section BL.2.01, Segment Loading Table. As the MST is divided into collections, an entry must be provided in the MSTSL to mark the end of each collection, this entry is called collection entry.

The format of the MSTSL is the following:

<MSTSL> ::= <list of collections>

<list of collections> ::= <collection> | <collection>
<list of collections>

<collection> ::= <list of header entries> <collection entry>

<list of header entries> ::= <header entry> |
<header entry>
<list of header entries>

Collection entry

The collection entry contains:

collection N;

where: N= a 6 digit number.

The end of the MSTSL must be a collection entry as follows:

collection 777777;

Header entry

The header entry contains items which are information about the segment. The general format of any items of the header entry is the following:

Syntax:

<item> ::= <keyword> :<item body>; | end;

<keyword> := names | pathname | maxlength | curlength |
access | status |

pagesize | hyperpage | initseg | perprocseg |
descsegment |

linkprovided | linkseg | comblink | linkstatus |
enf acc provide |

enfaccess | temposeg

<item body> := <list of segment names> | <pathname> |
<access> | <status> |
<linkage section status> | <integer> |
yes | no | undefined

<list of segment names> ::= <segment name> |
<list of segment names>
<segment name>

A comment of the following type may appear anywhere in a header:

<comment> ::= <empty>|/*<character string without*/>*/

Semantics:

This first two items must be the segment names followed by the pathname, other items can be given in any order. Each header entry consists in fact of 19 items, one for each 18 components of the SLT followed by the END item.

Spaces and new line characters are not significant.

Note:

the <item body> will be discussed in more details in the 6.36 implementation section.

The algorithm for making segment units of the SNT is the following:

1. Get next entry from the MSTSL
2. If entry is a collection mark, go to step 6
3. Search for the segment
4. Manufacture logical header and call procedure to write onto tape
5. Call procedure to write onto tape for output of the segment itself, and go to step 1
6. Manufacture logical mark with the collection number and write onto tape
7. If collection number is 777777, go to step 9
8. Go to step 1
9. End sequence

The Implementation of the MSTG in the 6.36 Environment

Under the 6.36, the MSTG program must take into consideration the memory space; any segment taken from the Multics segment library tape and written onto output tape must be released from core by an explicit call release (segment).

The magnetic tape writing facility must be provided in absence of the magnetic tape device interface module. This facility specification is attached at the end of this section.

The mechanism for manufacturing the Multics System Tape Segment List is the following:

1. For any segment of the MST, a header must be provided. This header is established by the author of the segment in order to minimize errors. The header consists of an EDA format file; it must contain the 18 items required by the SLT. The next section will describe the content of each item and the edition of it. For any segment which has a linkage section segment, 2 consecutive headers must be provided, one for the text segment and one for the linkage segment.

2. The MSTSL is obtained by combining in a desired order all header files together, header files stand now for entries of the MSTSL. The CTSS "join" command performs this function, it is described in MSPM section BE.5.12.

In order for the MSTSL file to become a 6.36 process segment, it has to be converted into the text or link format. A subroutine is needed to perform the conversion of the MSTSL EDA file into the absolute binary text file. This subroutine is MAKETL and documented in MSPM section BE.5.13.

Header Editing

A. All header files must have the second name "HEADER".

B. The list of items contained in the header is identical to the one in BL.2.01, it is repeated below for convenience.

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

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 after 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 in words is 64 (ON) or 1024 (OFF).

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 than a per-system segment.

11. Descriptor segment switch - This switch in 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 loaded 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. Enforces access switch - If on, this switch indicates that the segment has the access given in item 17 to references 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 hardcore 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).

C. The general format to declare any item of the header entry is the following:

<item>::= <keyword>:<item body>;

The first two items must be "names" and "pathname". The other items may follow in any order.

1. names : <list of segment names>;
e.g. names: name1, name2, name3,...,namen;
/*segment names*/

2. pathname: <pathname>;
e.g. pathname: ABC; /*pathname*/

3. maxlength: <integer>;
<integer>::= maximum size in unit of 1024 words

4. curlength: <integer>; |undefined;
<integer>:: = current size in units of 64 words "undefined" when the size is not known by the user

5. access:<access>;
<access>:: = Any combination of letters A,D,S, M,E,W

A Slave Access

D Data

S Slave Procedure

M Master Procedure

E Execute Only

W Write Permit

i.e. DW stands for Data Write Permit

SAW stands for Slave Procedure Slave Access Write Permit

6. status: <status>;
<status>:: = one of the 4 letters W,L,A,N

W = Wired down
L = Loaded
A = Active
N = Normal

7. pagesize: <integer>;
<integer>:= 64/1024

8. hyperpage: <integer>;
<integer>:: = Hyperpage in units of page size

9. initseg: Yes;|No;
Yes = initialization segment

10. perprocseg: Yes;|No;
Yes = per-process segment

11. descsegment: Yes;|No;
Yes = Descriptor segment

12. linkprovided: Yes;|No;
Yes = segment has a linkage section segment

13. linkseg: Yes;|No;
Yes = segment is a linkage section

14. comblink: Yes;|No;
Yes = this linkage has to be combined

15. linkstatus: <status>;|undefined;
<status>::= one of these 5 letters N,W,L,A,O

N = Normal
W = Wired down
L = Loaded
A = Active
O = Out reference

16. enfaccprovide: Yes;|No;
Yes = enforces access is provided for this segment

17. enfacc: <access>;|undefined;
<access> See item 5
"undefine" when the enforced access is not provided

18. temposeg: Yes;|No;
Yes = segment can be deleted after Part 1 of Multics Initializer
The last item should be as follows:

19. End;

6.36 Multics standard tape format writing facility.

A 645 library subroutine is provided which accepts calls to write on a magnetic tape. This subroutine escapes to the 635 GECOS environment and performs calls to GEFRC and GEIOS to write a magnetic tape in the Multics standard magnetic tape format. This format is described in detail in MSPM section BF.6.01. No more than one such tape may be written on any one 6.36 job.

The library subroutine accepts the following four calls:

```
call attach_tape(err);
```

This call performs any system functions not already accomplished by GECOS control cards (e.g., allocating a drive, mounting a tape, etc.) It also writes on the beginning of the tape the standard label sequence.

```
call write_tape(buff, n, err);
```

The n words located in buff(1) through buff(n) will be copied from buff and added to the stream of words destined for this tape. Subroutine write_tape will break up this stream into blocks of the appropriate size, add header and trailer information, and write the blocks onto the tape. Write tape will insert EOF records where appropriate for the standard format.

```
call detach_tape(err);
```

The final physical record is written onto the tape and a standard end-of-tape sequence is written; the tape is rewound and the drive deallocated.

```
call get-err-count(count);
```

The tape writing subroutine will keep track of the number of error records which it has written; this call, which may be made at any time between the "attach_tape" and the "detach_tape" calls, will return the current value of this number.

In each of the first three calls, err is a return argument which, if zero, means that the call was handled successfully, and if non-zero means that the call failed for some reason. A non-zero return is accompanied by a diagnostic message in the error file.