Published: 09/28/67

Identification

Multics System Tape Generator
T. H. Van Vleck

Purpose

The Multics System Tape Generator (MSTG) is a collection of programs used to generate a Multics System Tape (MST; see section BL.1). The MSTG is callable by the Multics command system by typing

```
mstg name1
```

where name1 is the name of a system tape segment list. The MSTG can also be used on the 6.36 system. This usage will be discussed in BL.1.05.

Usage

Input to the MSTG is an ASCII character-stream file known as an MST Segment List. This file contains an entry for every segment to be written on the MST. The MSTG makes a single pass down the segment list, writing a "segment unit" on the tape for each segment specified in the segment list. (See Figure 1.) A segment unit on the MST consists of the segment to be loaded, preceded by a header, which contains the information used to set up the Segment Loading Table entry for the segment. The header is filled in from statements in the segment's entry in the MST segment list.

Contents of the MST Segment List

The MSTSL is composed of a sequence of entries, one for each segment to be written on the MST. The simplest entry is shown in this example:

```
name: xsegment1;

path_name: >dir1;

end;
```

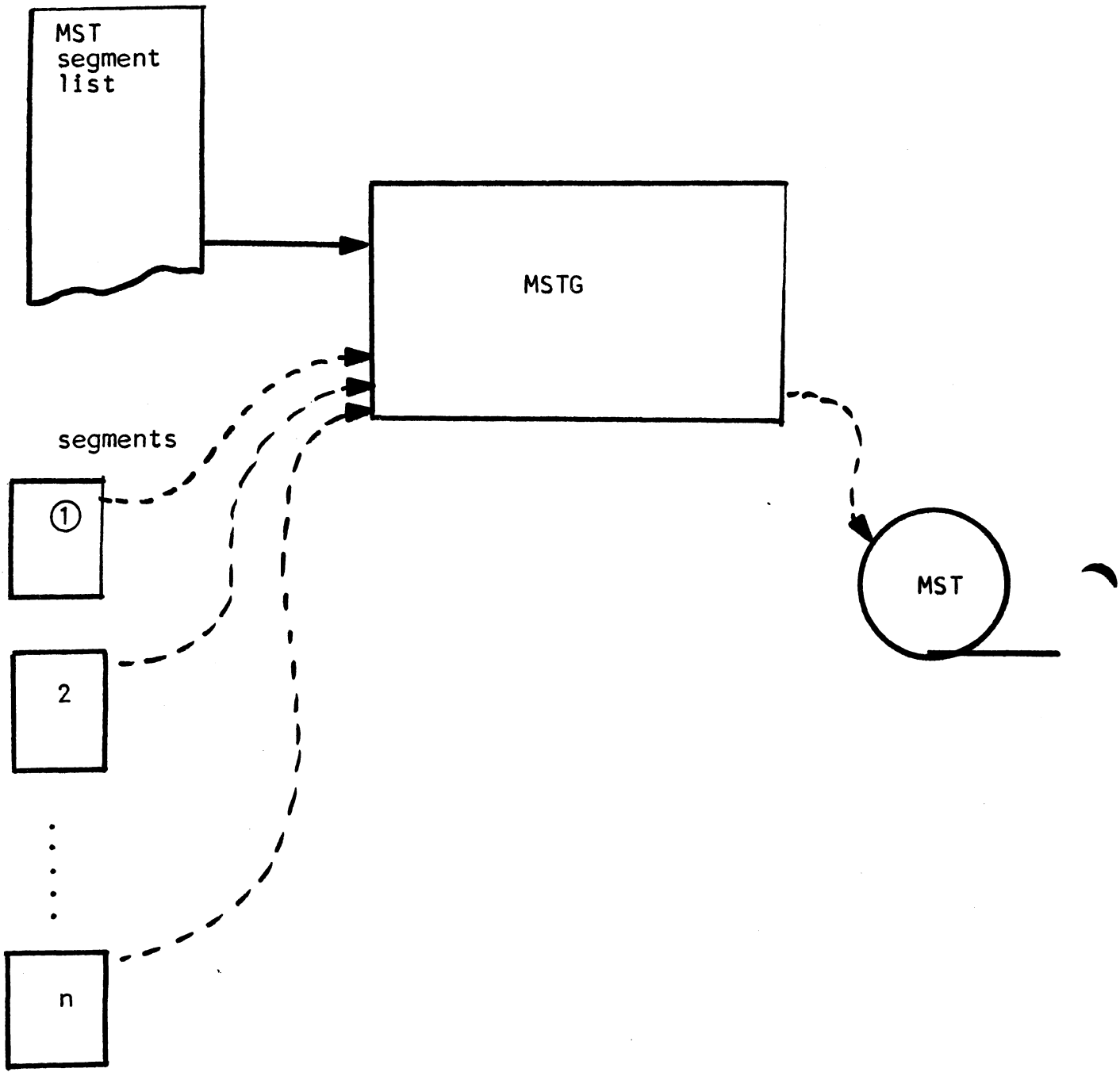


Figure 1

This entry consists of three items, each one of which contains a keyword, followed by arguments if necessary. The example directs the MSTG to copy the segment "xsegment1", which resides in directory "dir1", onto the MST. Since no attributes are specified for the segment, all attributes will take their default values.

Additional items may be specified in an entry for a segment to set attributes of the segment (in other words, to specify values of items in the SLT entry for the loaded segment). The function of the SLT is discussed in BL.2, MSPM. Table 1 gives a list of all item keywords and their default values if any.

Keyword	Default
name	REQUIRED
path_name	REQUIRED
status	normal
page_size	64
hyperpage	1
initseg	no
per_process	no
temp_seg	no
link_status	normal
enforced-access	no
combine_link	no
max_length	=seg
cur_length	=seg
access	=seg
link_seg	=seg
link_provided	=seg
loadname	*
first_name	*
linkage	*
end	REQUIRED

TABLE 1

Four types of keywords are specified in Table 1: those with constant default values, those with default values determined by inspecting the segment (indicated by "=seg" in the table), those which must be present (and so have no default value), and special keywords (indicated by "*" in the table). A description of each keyword is given below.

There are two control statements which may appear in place of a segment entry. The statement

```
collection n;
```

causes the MSTG to generate a collection mark for collection n; n is interpreted as a decimal number.

The end of the MST Segment List is indicated by the statement

```
fini;
```

When this statement is encountered, the MSTG closes out the system tape and returns.

Figure 2 is a flow chart of the main program of the MSTG.

Syntax

The following characters are considered "break characters", and may not occur in any keyword, name, number, etc.:

space	comma
new_line	colon
tab	semicolon
end-of-text	
comment string	

Multiple occurrences of those in the first column are ignored. A comment string, as in PL/I, begins with "/*" and ends with the next "*/".

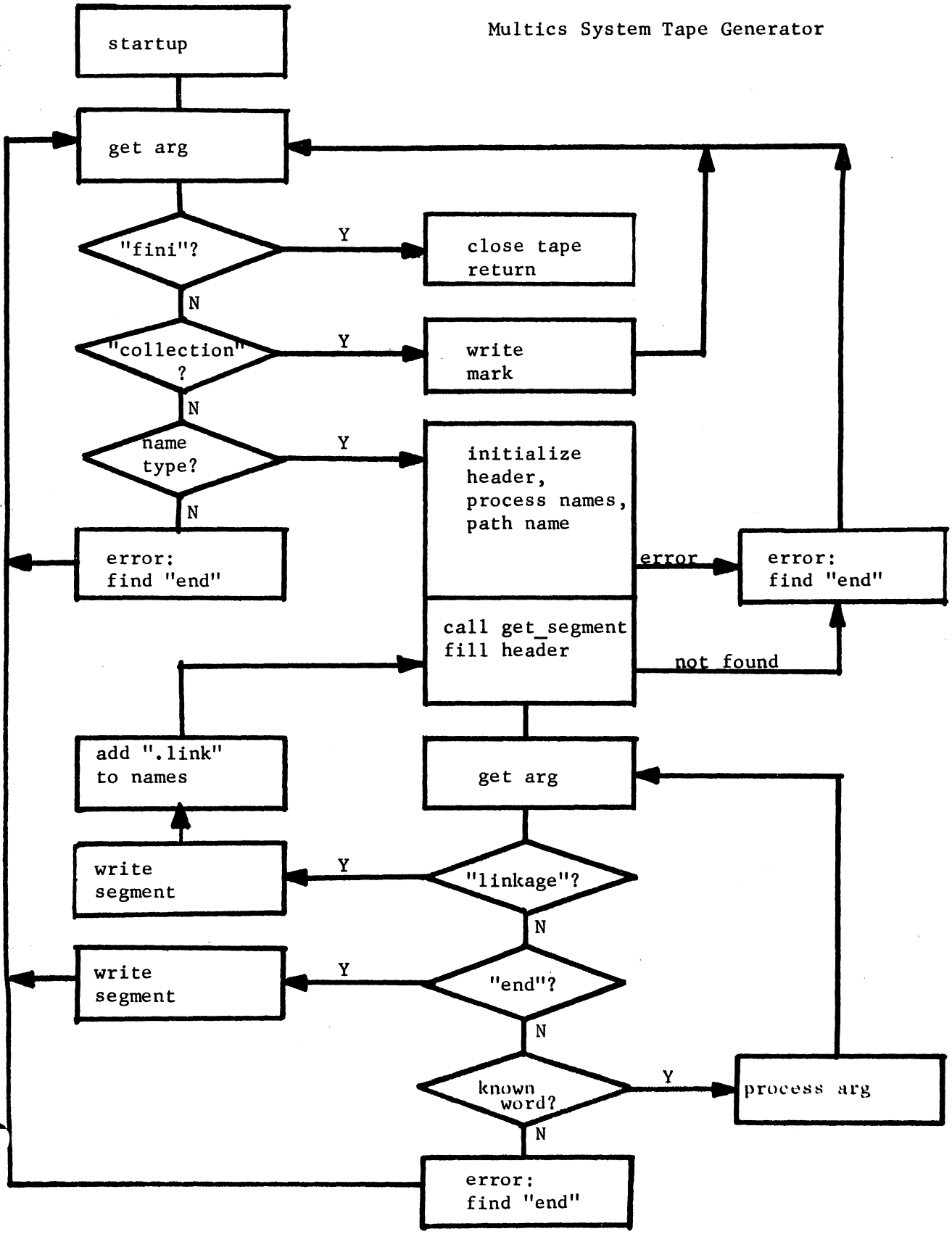
All keywords should be followed by a colon and an argument (which is sometimes a list separated by commas) or by a semicolon, if the keyword has no argument.

Keywords

I. Required keywords

A. "name" - this keyword is followed by a list of names for the segment, separated by commas. The first name in the list is used by MSTG to reference the segment. (See "loadname" and "first_name", below.) Additional names will also go into the SLT entry for the segment when the system tape is loaded. Names may be up to 32 ASCII characters long.

Multics System Tape Generator



B. "path_name" - this keyword is followed by the directory path name for the segment (up to 511 characters).

C. "end" - this keyword must be the last in each segment entry. It has no argument.

II. Keywords with constant default values

A. "status" - this item indicates the status of the segment after initialization. The values of the argument represent one of four statuses as follows:

1. "normal"
2. "wired_down"
3. "active"
4. "loaded"

the default value is "normal".

B. "page_size" - this keyword is followed by the page size as a decimal integer:

1. "1024"
2. "64"

The default value is 64.

C. "hyperpage" - this keyword is followed by an integer giving the hyperpage size in units of the page size. The default value is 1.

D. "initseg" - this item specifies whether the segment is an initialization segment or a supervisor segment. (initialization segments have segment numbers above 2048). Possible values are:

1. "on" or "yes" - initialization
2. "off" or "no" - supervisor

Default value is "off".

E. "per_process" - this item specifies whether or not the segment is per process.

Values and default are as for "initseg".

F. "temp_seg" - this item specifies whether the segment may be deleted at a certain stage of initialization. Values and default as for "initseg".

G. "link_status" - this keyword specifies the type of linkage section represented by the segment. Possible values:

1. "normal"
2. "combined,wired_down"
3. "combined,loaded"
4. "combined,active"
5. "combined,out_refs"

The default is "normal".

H. "enforced_access" - this keyword is followed by the access required in outer rings of protection. Values are as for "access", below. Default is null (access not enforced).

I. "combine_link" - this keyword specifies whether the linkage section may be combined with others of the same type. Values and default as "initseg".

III. Keywords with non-constant default values. These default values are set from information gotten by a call to the MSTG subroutine "get_segment", which investigates the current status of the segment. If the items below occur, they over-ride the default settings.

A. "max_length" - the maximum segment length in words.

B. "cur_length" - the current segment length in words.

C. "access" - the descriptor-segment access control bits are set by this item. The argument is a list of one or more of the following items, separated by commas:

1. "slvacc" (slave access)
2. "slvprc" (slave mode procedure)
3. "wpermt" (write permit)
4. "masprc" (master mode procedure)
5. "data"
6. "exonly" (execute only)

D. "link_seg" - on if a linkage section. Values as for "initseg".

E. "link_provided" - on if a linkage section exists for this segment. Values as for "initseg".

IV. Special Keywords

A. "loadname" - This keyword, if used, is followed by a segment name which MSTG will use when looking up the segment. This item appears immediately before the "name" item. ("Name" will still be used on the tape.) It is used to avoid name conflicts in the 6.36 environment; e.g., if <tdope_> is to be written out on the MST but is also being used by the MSTG a fresh copy of the linkage section (i.e., not the currently in-use one) must be put on the tape; this can be accomplished by creating a copy under a name like "xtdope_.link" and using that as the loadname.

B. "first_name" - This keyword is only valid for the first segment of a tape to be bootloaded. It is used instead of the "name" keyword to indicate that special processing must be performed when writing this segment. The special processing consists in writing out a special padding block, and then writing the segment beginning at location 40(8) so that 40 locations from the beginning of the tape record is location 40 of the first segment. That is, we overlay the first part of the segment with the physical and logical header.

C. "linkage" - This keyword has no arguments. It functions something like the "end" statement: in particular, it causes the current segment to be written out. It then re-enters the argument search loop after concatenating the characters

".link"

to all names for the segment, and calling "get_segment" for the linkage section. This means that type II keywords do not revert to their default values, but type III keywords do. (See the example below.)

Example 1

```
name:           my_procedure;
path_name:      >p23;
combine_link:   yes;
access:         masprc;
end;

name:           my_procedure.link;
path_name:      >p23;
combine_link:   yes;
access:         slvprc, slvacc, wpermt;
end;
```

These 10 lines have the same effect as the following 6:

```
name:           my_procedure;
path_name:      >p23;
combine_link:   yes;
access:         masprc;
linkage;
end;
```

Example 2

Suppose that segments "xstat_" and "data" and their linkage sections exist in the address space of MSTG. It is desired to write entries on the tape so that "xstat_" will be used for segment "stat_" when the tape is loaded, and so that "data" will be available to master procedures only. Suppose that MSTG sees "xstat_" and "xstat_.link" as minimum-length segments, whose size we wish to override, so that although one page is written on the tape, many pages may be assigned in the bootload environment. The following statements do the trick:

```
loadname:       xstat_;
name:           stat_;
path_name:      >system_library;
access:         data, sTvacc, wpermt;
cur_length:     4096;
max_length:     8192;
linkage;
cur_length:     1024;
max_length:     2048;
end;

name:           data;
path_name:      >direct_17> direct_17a;
access:         data;
link_provided:  no;
end;
```

Note that the "link_provided" entry is necessary because MSTG would otherwise set the "expect_link" switch. This would lead to an error in bootloading.