Identification

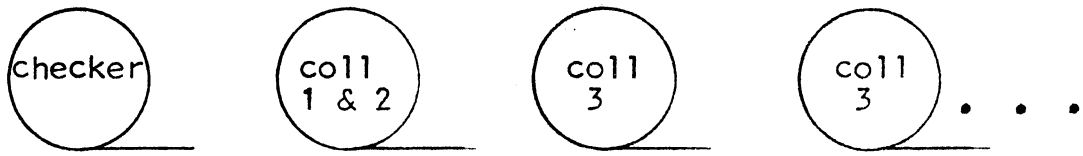
MST checker
P. Schicker, T. H. Van Vleck

Purpose

The MST checker is a tool for finding errors in a set of Multics system tapes to be used for a bootload. The checker also produces a summary describing the system which would be created if a set of MST's were loaded.

Overview

The MST checker is a stand-alone job which can be run on any 645 capable of running Multics. It does not use any storage except core. The checker is on a separate tape which is bootloaded in before the set of MST's to be checked; it reads the tapes in and uses the on-line printer to produce its output.



Tape configuration for checker run.

The checker produces 3 kinds of output: duplicated-segment comments, per-collection error messages, and a final summary.

The checker determines how many tapes to read and how many collections to check by reading the processor data switches.

How to use the MST checker

1. mount checker tape on 3, then your tapes on 4,5,6,3,...
2. bootload
3. it will stop after reading checker tape with 77 in Q. Set switches:

1-6 contain the number of physical tapes to read as a binary number.

7-12 indicate when to check for missing segments. 7=after coll 1, 8=after coll 2, etc. The highest collection to be read must be indicated.

4. Then flip switch 0. It should take off and read tape very rapidly.

5. Run ends with 3 ejects of the printer and stops with a DIS.

Output from the checker

If a segment is read which has the same segment name as a previously-loaded segment, whether the directory path name is the same or not, the checker will print a line of the form

<segname> duplicated.

After reading those collections selected in switches 7-12, the checker calls subroutine "crossref" to print every unsatisfied link, and to check for other logical errors.

The following errors are detected:

1. segment not found

2. symbol not present in segment (this check is suppressed if there is a trap-before-link)

3. use of type 2 link (ITB)

4. status conflict. (wired segment calling non-wired, etc.)

5. supervisor segment calling temporary or initialization segment.

6. bad ring brackets on reference.

Other errors will be defined and code added to detect them from time to time.

After all tapes are read, the program "loading_summary" is called to print a list of all supervisor and initialization segments loaded. For each segment, a line of the form

segno name acc status flags ring cur max path

is printed, where the items have the following contents:

segno- segment number. Blank for collection 4.

name- segment name. If there are additional names, extra lines blank except for name will follow

acc- SDW access field in octal

stat- segment status as follows

W= wired

L= loaded

A= active

blank= normal

flags; a string of letters marking special attributes of the segment, as follows:

U- no link points to this segment

T- seg. is temporary

K- seg. has copy switch on

P- seg. is per-process

E- seg. has enforced access outside ring 0

L- seg. has linkage

C- linkage may be combined

ring- segment ring brackets

cur- current segment length

max- maximum segment length

path- directory path name of segment

How it works

The checker operates in a "limited EPL" environment similar to that available to part 1 of Multics initialization. The main program, "driver", is called by a special version of bootstrap 2. "driver" in turn calls "ignore$init" to initialize fault and interrupt handling, and then calls "getcol" to get a collection and "crossref" to check it as requested by the switches. After all tapes are read, "loading_summary" is called and then the run terminates.

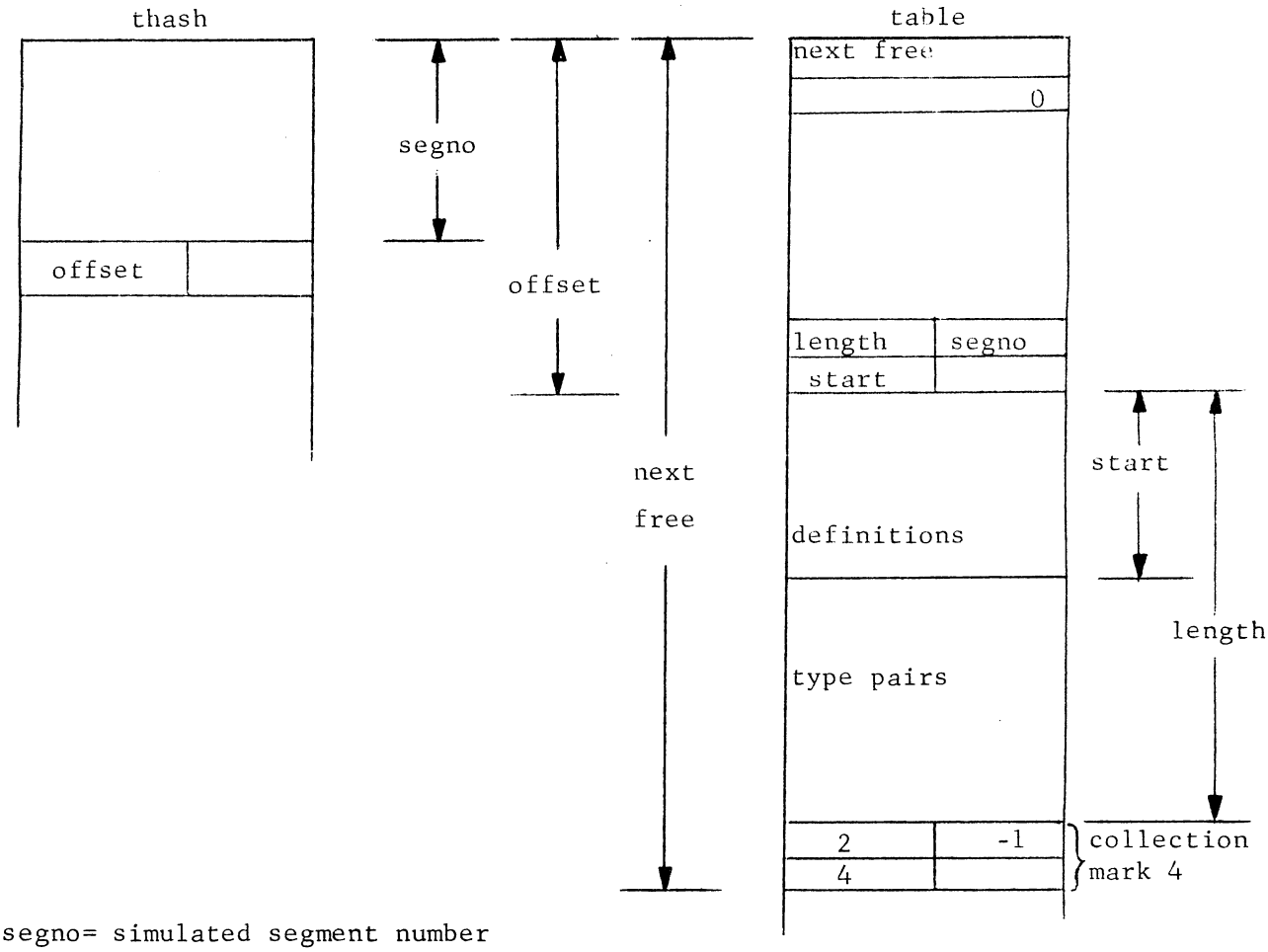
Subroutine "getcol" is quite like the system routine "segment_loader". It reads an entire collection from tape, loading each text segment into a segment called <text> in the MST checker, and loading each linkage section into segment <link>. Then program "extract_def" is called to locate the definitions and type-pair blocks and move them into segment <table>.

Entries are also made for the segment in segments <sslt> and <sname_table>, which serve the functions of an SLT for the tapes being loaded, and in segment <thash>, which is an index into <table> by segment number. This process is repeated for every segment of a collection, and then "getcol" returns.

Subroutine "crossref" checks for missing segments and other errors by making a pass through <table>. The format of this segment is shown in figure 1.

Subroutine "loading_summary" runs through the simulated SLT and prints out the information it finds.

Figure 1



segno= simulated segment number

type pair:

A	B	C
D		E

A: if A=0, then the external reference has not yet been satisfied.
B: type of link ($1 \leq B \leq 5$).
C: trap pointer (if relevant).
D: pointer to segment name.
E: pointer to entry name.

The checker has one master-mode segment-segment "ignore", which serves the purposes of fault handling, interrupt handling, switch reading, and exit to BOS. All faults except page faults are fatal: page faults result in the assignment of a page of core upwards starting from the dummy segment <last>.