Identification

Multics System Tape Format
A. Bensoussan

Purpose

The Multics system tape (MST) is the tape which contains
all information needed by the Multics Initializer. It
is manufactured by the Multics system tape generator command,
running under Multics (initially it will be written by
a 6.36 program), and it is read successively by the bootload
program, the bootstrap initializer, the segment loader
and, if the file system hierarchy must be reloaded, by
the file system initializer.

General Discussion

When a user creates a tape under Multics, the tape contains
the information that the user wants to record (logical
information) plus some information added by the I/O system
(physical information). When the tape is read under Multics
the I/O system returns to the user the logical information,
in such a way that the physical information is transparent
to the user.

The MST is a special tape in the sense that, although
it is written in Multics Standard tape format using the
I/O system, it is read without using the I/O system.

This section describes the content of the MST with respect
to both aspects, logical format and physical format with
the main idea of making them completely independent so .
that it is possible to isolate the code which has to deal
with physical format in order to return only the logical
information.

The following items are defined in this section:

    a.   Logical record
    b.   Segment unit
    c.   Collection
    d.   Syntactical description of logical MST format
    e.   List of the MST collections
    f.   Physical format of the MST

Logical record

A logical record consists of one control word followed
by a variable number of words that contains the information
to be recorded.

The variable length information which follows the control
word may be of 3 different classes: segment, header associated
with a segment, or mark.

The control word contains the <u>class</u> and the <u>length</u> of
the information which follows it.

Therefore we have 3 classes of Logical records:

      a.   Logical header
      b.   Logical segment
      c.   Logical mark

0.   Control word

The content of the control word is the following:

Bits 0-17:   0 if header control word
            1 if segment control word
            2 if mark control word
Bits 18-35:   length in words of the header, segment or mark

1.   Header

The header keeps track of information needed to make up
an entry in the Segment Loading Table (SLT).   Therefore
it contains the same items that are found in the SLT entry
(see BL.2.01), with the same format, except that:

In the SLT, segment names and path names are provided
through pointers to the names themselves located in the
name segment associated with the SLT; in the header, segment
names and path names are written explicitly as character
strings with the format they have in the name segment
(see BL.2.01).

2.   Segment

The segment, of course, does not have any conventional
format.   It consists of text, link or data.

3.   Mark

The mark consists of only one word: Bits 0-17 of this
word contain the mark number; this mark number is an integer
starting from 1 and increasing by 1 from one mark to the
next one.

Segment unit

A logical segment is always preceded by a logical header.
A logical header followed by a logical segment is called
a "segment unit".

Collection

It is possible to group a certain number of segment units
to form a collection.

A "collection" consists of a list of n segment units ($n \geq 0$)
followed by a logical mark.

If n = 0 the collection is empty.

The logical MST is a list of collections. The end of the tape
is detected by an empty collection the mark number of which
is 777777 octal.

Syntactical description of logical MST format

1.   Syntax

```
<logical record      >:: = <logical header>|<logical segment>
                                      |<logical mark>
<logical header      >:: = <header control word><header>
<logical segment     >:: = <segment control word><segment>
<logical mark        >:: = <mark control word><mark>
```

Logical headers, logical segments and logical marks are·
the only logical records that can be found on the MST.
They are used below to define a syntax describing rules
of their occurence on the MST.

```
<         segment unit>:: = <logical header><logical segment>
< list of segment units>:: = <empty>|<segment unit>|
                             <list of segment units><segment unit>
<            collection>:: = <list of segment units><logical mark>
<    list of collections>:: = <collection>|<list of collections>
                                      <collection>
<            logical MST>:: = <list of collections>
```

2.   Semantics

The syntax does not allow detection of the end of the
MST; one could have defined two types of logical marks,
one for the end of collection, one for the end of the
MST. It appears to be simpler to do it using the following
rule: The end of the MST is detected by an empty collection
having the mark number 777777 octal.

List of the MST collection

1.   Collection 1: Bootstrap

Contains one segment unit for each segment that has to
be loaded by the Bootstrap Initializer before control
is given to the Initializer Control program. The first
segment unit is the bootstrap initializer itself, the
first physical record of which is loaded by the bootload
diode program.

2.   Collection 2: Configuration Part 1 loadlists

A given hardware configuration may require more than one
segment to describe it. Therefore a loadlist is needed
for each hardware configuration.

Collection 2 contains one segment unit for each loadlist
naming the segments that describe the configuration.
The segment loader selects and reads only one segment
unit in this collection.

3.   Collection 3: Configuration Part 1 Library

Contains one segment unit for each segment that may be
mentioned in any configuration loadlist. The segment
loader reads, in this collection, all the segments which
are named in the loadlist selected from collection 2.

4.   Collection 4: Supervisor Part 1 loadlists

Contains one segment unit for each loadlist naming the
segments that have to be loaded by the segment loader
during part 1 of the Initializer control program.

5.   Collection 5: Supervisor Part 1 Library

Contains one segment unit for each segment that may have
to be loaded by the segment loader during part 1 of the
Initializer control program.

6.   Collection 6: Supervisor Part 2 loadlists

Contains one segment unit for each loadist naming the
segments that have to be loaded by the segment loader
during part 2 of the Initializer control program.

7.   Collection 7: Supervisor Part 2 library

Contains one segment unit for each segment that may have
to be loaded by the segment loader during part 2 of the
Initializer control program.

8.   Collection 8: Supervisor Part 3 loadlists

Contains one segment unit for each loadlist naming the
segments that have to be loaded by the segment loader
during part 3 of the Initializer control program.

9.   Collection 9: Supervisor Part 3 library

Contains one segment unit for each segment that may have
to be loaded by the segment loader during part 3 of the
Initializer control program.

10.   Collection 10: Configuration Part 2 loadlists

Contains one segment unit for each loadlist naming the
segments that make up the second part of the configuration.

11.   Collection 11: Configuration Part 2 library

Contains one segment unit for each configuration segment
that may have to be loaded during part 3 of the Initializer
control program.

12.   Collection 12: Hierarchy

Contains one segment unit for each segment that will have
to be loaded by the file system initializer during part
4 of the Initializer control program, if the file system
hierarchy must be reloaded.

13.   Collection 13: End of MST

This is an empty collection (no segment unit) with a mark
number equal to 777777 octal.

Note 1

The loadists contain only names of text segments; if a
text segment has an associated linkage section segment,
which is indicated in the header of the text segment,
the linkage segment unit is expected to be immediately
after the text segment unit in the library collection.
The linkage section segment name must not be in the loadlist.

Furthermore, the relative order of the segment units in
a library collection is expected to be the same as the
appearance of their names in the corresponding loadist.

Note 2

In the MST used in phase I, collections consisting of
loadlists contain only one segment unit. The segment
loader does not select the loadlist but rather takes the
only one which is in the collection. In addition, collections
consisting of libraries contain only the segments mentioned
in the corresponding loadlist.

MST physical format

1.   Physical record

The way the logical information is recorded on the MST
is the following: the logical information is broken up
into blocks of n words. An h-word physical header is
added in front of them and a t-word physical trailer is
added after them.

A physical record is comprised of

    a.   an h-word physical header
    b.   an n-word block of logical information
    c   a t-word physical trailer.

A physical record is located between 2 gaps.

Every k physical records, the MST contains a physical
end-of-file mark. The values of n, h, t, and k are specified
in BB.3.01.

2.   Label

A label is added at the beginning of the MST; this label
is ended by a physical end of file mark.

The label is not part of the logical MST and it is skipped
by the bootload program which reads in the first physical
record that follows the label.

The complete definition of the Multics standard tape format,
including the content of the physical header and trailer,
is described in BB.3.01.

Note that the bootload and the bootstrap initializer program
are organized in such a way that the bootstrap initializer
does not constitute an exception to either the physical
format or the logical format of the Multics System tape.