Published: 01/12/67

Identification

Tape Formats for Backup
S. H. Webber

Purpose

The tape formats which backup uses are based on the assumption that Multics will likely change with time, and that there will therefore be the need of a generalized tape format which allows the insertion and deletion of certain parameters. That is, it is necessary that any tape formed on a Multics system for backup purposes be compatible with any subsequent Multics system. This section explains how this is to be done.

Introduction

The organization of all backup tapes will be the same. The contents are as follows:

1) Header
2a) Record a
2b) Record b
.
.
.
2n) Record r

The header format will never change. It is possible that record formatting may change as is explained later.

The Tape Header

The contents of the tape header must be complete enough to uniquely specify the tape. This includes information such as the date/time the tape was created and why the tape was created. More specifically the header contains the following:

1) tape type
2) date/time first made known
3) the reload list

    a) user checkpoint labels
    b) system checkpoint labels
    c) incremental reel labels
    d) other

4) destroy date.

An explanation of each of these follows:

1) tape type

The tape type refers to the type of backup tape being formed. There are two types; incremental and checkpoint. The incremental tapes will be kept "forever"; there may be consolidation of the information - but, it will be kept. "Forever" may be 10 years or more. Checkpoint tapes are kept for a smaller length of time and are generally reestablished in full at specified times. The system and user checkpoint dumps create checkpoint tapes.

2) date/time first made known

This time refers to the time this tape reel was first made known to the backup system.

3) the reload list

The reload list is a list of tape reel labels to be used should it become necessary to reload the hierarchy. Each backup tape contains such a list. The list itself specifies which tape reels must be loaded, and in which order, when a reload is called for. Each tape reel then specifies how reloading should proceed if the system goes down while that tape is being written. This list is updated whenever a new incremental reel is created and at the end of system and user checkpoint dumps.

4) destroy date

The destroy date is that date after which the tape is no longer of interest to Multics. For checkpoint tapes this may be weeks after the tape is first made known. For incremental tapes it may be decades after the tape is first made known.

The Record Format

Each logical record of every backup tape is independent of all other logical records (excluding header records). There is sufficient information in the header of each logical record to identify it completely. There are no assumptions made about information contained in other logical records (although there are means for making use of redundant information). This design allows for a minimum loss when the system goes down as well as logical and simple reload algorithms.

Each logical record on tape contains 1 or 2 sections. The first section contains a record header and a preamble. The second section, which is optional, contains the data segment or directory pointed to by the last (deepest) entry of the preamble.

The record header contains the following information:

1) dump type
2) terminal entry type
3) tape type
4) number of preamble entries
5) unique identification of the deepest entry
6) date/time-last-modified of the deepest entry
7) slot name of the deepest entry
8) slot number of the deepest entry

These are explained in more detail as follows:

1) dump type

The dump type can be any of 5 (at present) different types. These are 1) incremental dump, 2) user checkpoint dump, 3) system checkpoint dump, 4) multilevel dump and 5) auto-dump. The design of the system insures that all of these dumps form tapes which look alike in form and differ only in content. The multilevel dump, incremental dump, and auto-dump processes all write on the current incremental tape. Any retrieval required by a user can refer to any of the incremental tapes (to get the most recent version) and hence they must all appear to the reloader to be identical.

2) terminal entry type

The terminal entry type is a multiple setting switch which at the outset will have 5 settings. These are distinguished by the first 3 bits of the string. The first bit of the string is the directory switch (of the branch) and is hence ON only when the branch is a directory. The second bit of the string specifies whether a segment (directory or data segment as the case may be) follows in the logical record. This switch is known as the "segment-follows-switch". The third bit of the string indicates whether or not the current segment being dumped is a secondary copy by the incremental dumper. (This switch can be set only if the terminal entry is a non-directory branch.) The following possibilities therefore exist:

"110"b The terminal entry is a directory branch with a directory following as a second segment in the logical record.

"100"b The terminal entry is a directory branch but no directory segment follows.

"010"b The terminal entry is a non-directory branch with a data segment following as a second segment in the logical record. The logical record is the result of a primary dump.

"011"b Same as above except the logical record is the result of a secondary dump by the incremental dumper.

"000"b The terminal entry is a non-directory branch - no data segment follows.

The slot number specifies if the entry is a link, a cacl, or a branch. If the entry is a branch the terminal entry type is relevant.

3) The tape type

The tape type specifies what type of tape is being formed, i.e. incremental or checkpoint.

4) Number of preamble entries

This count specifies how many entries follow. It is a direct measure of the current depth in the hierarchy tree when the preamble was dumped.

5) and 6) Unique identification and date/time-last-modified for the deepest entry

The "uid" and "dtm" are used by the reloader to decide quickly and accurately whether or not the preamble on tape should be reloaded as well as whether or not the data file or directory which may follow should be reloaded.

7) Slot name of the deepest entry

Is the character string consisting of a concatenation of all superior slot numbers. This is needed primarily by the secondary storage reloader which would otherwise not have access to it in that it does not look at the preamble.

8) Slot number of the deepest entry

The slot number of the deepest entry is used by the secondary storage reloader (which does not examine the preamble).

Following the record header comes the preamble. This is a complete series of consecutively inferior entries copied from the hierarchy. This preamble then defines the position of the last entry (and data file if present) in the hierarchy. Furthermore, this complete list of entries allows the hierarchy to be reestablished during the hierarchy reconstruction process and for data files to be loaded before a complete hierarchy is present by reestablishing at least that part of the hierarchy necessary to make that data file accurately established in the hierarchy.

The user checkpoint tapes, which are used only after a complete and up-to-date version of the hierarchy has been established, do not technically need this preamble of entires. However all backup tapes are to appear identical in form and for this reason the user checkpoint tapes include a complete preamble with each data file dumped.

Note that because of the organization of the hierarchy all of the preamble entries, except possibly the last entry, are directory branches. The last entry as explained earlier may be a link, a cacl, a directory branch, or a non-directory branch. Only the latter two of these may be accompanied by a second section on tape.

The preamble format is designed to be general enough so that tapes made for one version of Multics can be used on later versions. This is done by having a fixed identifier-variable relationship which will not change. This allows variables to be deleted and others to be added, but this must be done in the following way. Each variable to be output onto tape must have associated with it a unique identifier. This identifier always refers to that variable and it is written onto tape with the variable. If the variable is subsequently deleted from the Multics system, then that variable is ignored at a subsequent reload. If new variables are needed, new identifiers will be assigned. Old identifiers are not to be used again. With this organization the reloader, by having a list of identifiers referring to variables currently being used by Multics, can decide what information found on tape is meaningful to Multics. The identifier always precedes its associated variable and hence order of variables is generally not important. This also allows subsequent dumpers to dump different, possibly less complete, information. The reloader only operates on that information found on tape.

The preamble string is a bit string consisting of all the entry data superior to the terminal entry. This bit string is formed by first packing each entry in a specified form and then concatenating, in order, all of the entries superior to and including the terminal entry.

Each entry is itself, composed of several variables. At present there are six variables for each entry. These are:

1) items
2) names
3) acl
4) count
5) date
6) slot number

The information called "items" varies depending on whether the entry is a link, a cacl, or a branch. In any case "items" is a bit string representation of certain structured data relevant to the entry. If the entry is a link, "acl" is empty (i.e. is a bit string of length 0). If the entry is a cacl "names" is empty and has length 0. If the entry is a branch (as all except possibly the terminal entry are) "items" contains all the information in the branch which is meaningful to dump. In that the entire storage configuration may be different at reload time, relative pointers and similar transcient quantities are not included.

1) items

The "items" string is always packed with a specific format. This is: all items are preceded by a count, right justified in a 17 bit field. If any count is zero either that item is no longer meaningful to Multics or it was not included on this particular tape. Any new items to be added to the "items" structure must be added at the end.

2) names

This is a list of names (and lengths of names) which are currently associated with this entry. "names" will be empty and of length 0, if the entry was a cacl.

3) acl

The contents of the access control list of an entry is saved in "acl".

4) count

This quantity specifies the maximum slot number for that type of entry. If the entry is a link, the largest negative slot number is saved in "count".

5) date

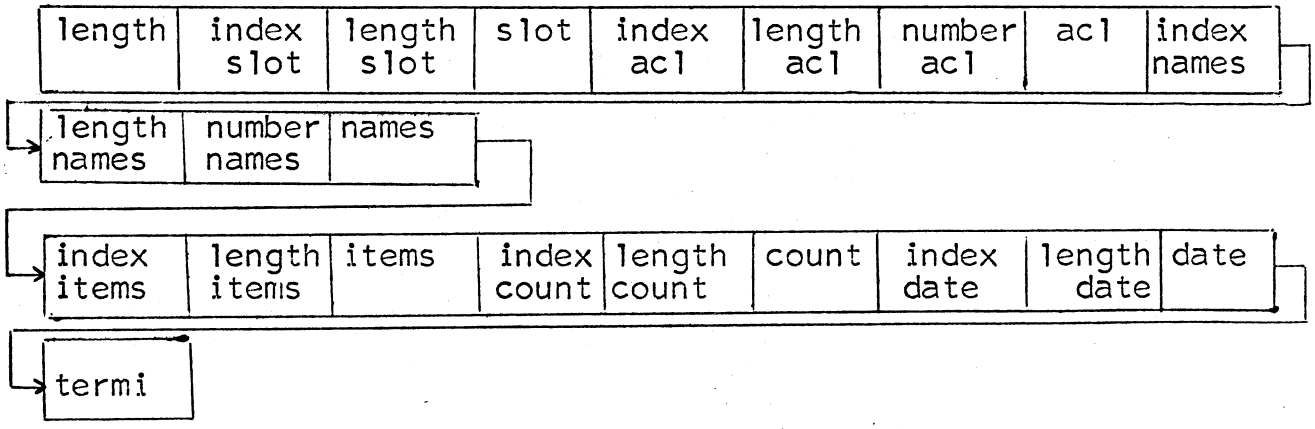
This date is the last time that "count" was updated. This is saved primarily for the reloading process.

6) slot number

As the hierarchy reconstruction process restores the hierarchy with a preamble it forms the slot name for the deepest entry by successively concatenating the slot numbers of each of the superior entries. The slot numbers alone at each and every level of the preamble are therefore sufficient to uniquely define any of the entries in the preamble with respect to the hierarchy.

The 6 variables "items", "names", etc., are concatenated together to form the entry bit string for the entry. Each of the 6 quantities is preceded by its identifier and then its length. For "names" and "acl" there is also a count specifying the total number.

A typical entry of the preamble string would then appear as follows:



The index and length for each variable will always be 17 bits long (each) for any version of Multics. The preamble consists of many such strings concatenated together. The finer structure of items, names, and acl is shown in figure '.

Before each entry of the preamble string there is a length which specifies how long that entry is (in bits). With this information and the total number of entries, the reloader can easily decode the bit strings and give the correct information to putentry, the directory control primitive (BG.8).

As a further check in decoding, each separate entry bit string is terminated by a special identifier. The value of this identifier (termi) is sufficiently large so that it will not interfere with future Multics expansion. When the string scanner finds this identifier, it realizes that this particular entry has been completely decoded and that the next data will be an index of the next entry.

The slot number data (i.e. index, length, and actual data) must precede the items, names, and acl data for each entry. This is so unpacking routines can tell the entry type prior to handling the actual entry data.

Each of the 6 items is packed in a specific way which will allow unambiguous unpacking. This is done by preceding each packed quantity with a length (in units of bits). If the length is ever zero, that quantity is not represented on the tape being processed. The order of the quantities so packed must naturally remain unchanged. Deletions of quantities from the system are signaled by a length of zero. All additions must come at the end of the string.

Figure 1

Items	(each item is preceded by its length)	
{itemsi|length}	systsize	size of system trap information
	systrap	system trap
	bupsize	size of backup area
	backup	backup information - retrieval arguments
	vacant	vacant switch
	dirsw	directory switch
	uid	unique id
	dtu	date/time-last-used
	dtm	date/time-file-last modified
	dtd	date/time-last dumped
	dtem	date/time-entry-last modified
	io	io information
	ml	maximum length of segment

Items (each item is preceded by its length) Cont.

cl	current length of segment
copy	copy switch
rd	retention date
actind	activity indicator
actime	activity time
pp1	priority parameter 1
pp2	priority parameter 2
acct	accounting
did	device id
cons	consistent dump switch
dupsw	backup (retrieval trap) switch
systsw	system switch

Names (each name is preceded by its length)

{namesi|length|number} name1
.
.
.
namen

Acl (each ACL item is preceded by its length)

{acli|length|number}

one for each ACL

- job
- name
- count
- mode
- trapsize
- trap
- protectsize
- segsize
- protect list1
- protect list2
- .
- .
- .
- protect listN
- entrynames

Link	(each link item is preceded by its length)	
{itemsi|length}	pathsize	pathname size
	pathname	pathname
	vacant	vacant switch
	uid	unique id
	dtu	date/time-last-used
	dtem	date/time-entry-last-modified
	dtd	date/time-last-dumped

The following PL/I declarations specify the various identifiers used by the backup coding and decoding modules:

```
dcl  (itemsize initial ("000...1"b),        /* items index */
     namesi initial ("000...10"b),          /* names index */
     acli initial ("000...11"b),            /* acl index */
     sloti initial ("000...100"b),          /* slot number index */
     counti initial ("000...101"b),         /* count index */
     datei initial ("000...110"b),          /* date index */
     termi initial ("000...1111101000"b),   /* terminal index =
                                               1000 */
     fixil initial ("000...1011"b),         /* first fix initial */
     headeri initial ("000...1100101"b),    /* header index = 101 */
     recordi initial ("000...1100110"b)     /* record index = 102 */
     bit (17);
```