Published: 09/13/68
(Supersedes: BX.99.02 05/21/68)

Identification

Interim Facility to Read a Tape
tape_in
K. J. Martin

Purpose

As an interim facility, the tape_in command causes a tape to be read placing specified segments into the user's working directory.

Usage

```
tape_in  reel_number  1name1  1name2  2name1  2name2
                      ...  nname1  nname2
```

where reel_number is the number of the tape reel to be read and 1name1 1name2 through nname1 nname2 is the list of CTSS name-pairs (in capital letters) of segments to be input into the user's workng directory. If segment_list is one element long and that element is "all", all segments on the tape will be input.

Naming Conventions

A user who wishes to input a Multics segment must know the following naming conventions regarding CTSS name_pairs if the segment was placed on tape by the tape_out command. The CTSS first name is the Multics first component (if six characters or less) or a concatenation of the first three and the last three characters of that component. The CTSS second name is the Multics second component truncated to six characters. Any CTSS name of less than 6 characters is padded with blanks on the right. A single-component Multics name has CTSS second name "TEXT". Thus the Multics name "alpha" becomes "ALPHA TEXT" and "jolly_roger.flag" becomes "JOLGER FLAG".

Implementation

The tape_in command calls smm$set_name_status to create a control segment in the user's working directory. The name of that segment is determined by calling unique_bits to obtain a unique 70-bit string, then calling unique_chars to convert it to a character string and concatenating it with the character string ".control".

The tape_in command then fills the control segment with appropriate control lines. The tape reel is specified in the mount control line. The files control line is inserted giving the path name of the user's working directory as the place to put the segments. If segment_list consisted only of "all", a dismount control line immediately follows. In this case the tape daemon will input all segments on the tape. Otherwise, consecutive lines are pairs of names from segment_list concluded with a ndfiles control line. A dismount control line is then added. The last line is a history control line giving the name of the user's mail box. Errors or appropriate comments are mailed to the user by the tape daemon.

The tape_in command creates a link in the tape daemon's working directory to the control segment.

It then looks in the segment td_wakeup and signals an event over the channel whose id is found there. The channel and process id's are td_wakeup$channel and td_wakeup$process_id. The event id signalled is the same 70-bit unique bit string obtained earlier, so that the tape daemon may construct the name of the link in its working directory.

The tape_in command is finished, so it returns to command level.

Possible errors arise from smm, unique_bits, append_link and the interprocess communication facility. Any of these errors which indicate failure to perform properly (as opposed to status indications) are fatal to the command. Tape_in types a comment to the user describing the failure and returns.

If any names are left over from segment_list (i.e., there were an uneven number) tape_in comments and ignores the left-over name. Other pairs of names are not affected.

Example

The command line

```
tape_in  144  quark epl  quark text  quark link
```

would generate the control segment:

```
          mount     144 r
          files     >my_dir
               quark epl
               quark text
               quark link
          ndfiles
          dismount
          history   personal_name.project
          stop

tape_in   scr       all
```

would generate the control segment:

```
          mount     scr r
          files     >my_dir
          all
          dismount
          history   personal_name.project
          stop
```