

***TS-473***

***Processing Tapes with the SAS® System in the UNIX® Environment***

***Wayne A. Barnes***

***Technical Support Division  
UNIX Systems Interface***

***SAS Institute, Inc.  
SAS Campus Drive  
Cary, N.C. 27513***

## **Abstract**

Due to the infrequent use of tapes in the UNIX environment, this topic has been somewhat of an enigma for a number of SAS users. This paper is meant to document and demystify the issues involved when dealing with tapes on UNIX platforms.

## **Introduction**

By their nature, tape devices are inherently slow. Due to this characteristic, tapes should only be used on a regular basis for archival purposes or for one-time data transfer situations when moving data from one system to another.

When dealing with tapes on UNIX, there are four UNIX commands frequently used. They are:

- `mt` (`tctl` on AIX®)
- `dd`
- `cat`
- `tar`

To get the desired effects when processing tapes in a UNIX environment, the use of a `norewind` device and the SAS system option `TAPECLOSE=LEAVE` are (virtually) always required. The discussion and examples that follow were run on HP-UX under the Korn shell, using SAS 6.09 TS042 and show these two requirements being met.

## **Working with External Files**

As is the case when dealing with any external file in SAS, the `FILENAME` statement is used to establish a fileref; all other SAS statements in a given program remain the same whether the external file resides on disk or tape. The only differences in the `FILENAME` statement when referencing a tape are which device-type keyword is used and the specified 'pathname' for the tape device:

```
FILENAME INTAPE  TAPE  '/dev/rmt/0mn';  
FILENAME OUTTAPE TAPE  '/dev/rmt/0mn';
```

or

```
FILENAME INTAPE  PIPE  'dd if=/dev/rmt/0mn 2> /dev/null';  
FILENAME OUTTAPE PIPE  'dd of=/dev/rmt/0mn 2> /dev/null';
```

The advantage of the second pair of `FILENAME` statements is that the use of your UNIX system's `dd` command will more efficiently process the tape. The disadvantage is that the code is specific to UNIX platforms and would have to be modified if ported to a non-UNIX system.

Here's a simple `DATA` step that writes an external file to tape:

```
OPTIONS TAPECLOSE=LEAVE;  
X 'mt -t /dev/rmt/0mn rewind';  
FILENAME OUTTAPE PIPE 'dd of=/dev/rmt/0mn 2> /dev/null';  
DATA _NULL_;  
  FILE OUTTAPE;  
  PUT '0 Zero';  
  PUT '1 One';  
  PUT '2 Two';  
  PUT '3 Three';  
  PUT '4 Four';  
  PUT '5 Five';  
  PUT '6 Six';  
  PUT '7 Seven';  
  PUT '8 Eight';  
  PUT '9 Nine';  
RUN;
```

Here's a simple DATA step that reads the external file from tape created above:

```
OPTIONS TAPECLOSE=LEAVE;
X 'mt -t /dev/rmt/0mn rewind';
FILENAME INTAPE PIPE 'dd if=/dev/rmt/0mn 2> /dev/null';
DATA NUMBERS;
  INFILE INTAPE PAD;
  INPUT DIGIT WORD $8.;
RUN;
```

Another (powerful) advantage of using the FILENAME statement with the PIPE device-type is that a remote tape drive may be used. To create an external file on a remote tape device, the FILENAME and X statements would be changed to:

```
OPTIONS TAPECLOSE=LEAVE;
X 'remsh host mt -t /dev/rmt/0mn rewind';
FILENAME OUTTAPE PIPE
      'remsh host dd of=/dev/rmt/0mn 2> /dev/null';
DATA _NULL_;
  FILE OUTTAPE;
  PUT '0 Zero';
  PUT '1 One';
  PUT '2 Two';
  PUT '3 Three';
  PUT '4 Four';
  PUT '5 Five';
  PUT '6 Six';
  PUT '7 Seven';
  PUT '8 Eight';
  PUT '9 Nine';
RUN;
```

To read an external file from a remote tape device, the FILENAME and X statements would be changed to:

```
OPTIONS TAPECLOSE=LEAVE;
X 'remsh host mt -t /dev/rmt/0mn rewind';
FILENAME INTAPE PIPE
      'remsh host dd if=/dev/rmt/0mn 2> /dev/null';
DATA NUMBERS;
  INFILE INTAPE PAD;
  INPUT DIGIT WORD $8.;
RUN;
```

(Of course, in both of the above examples "host" would be replaced by the name of the remote machine at your location.)

## **Working with External Files Created on the Mainframe**

There are three main points to remember when dealing with tapes on UNIX that were created on a mainframe:

- UNIX does not support IBM® Standard Label tapes
- UNIX does not support multi-volume tapes
- The DCB characteristics of the file must be known.

Each one of these issues will be discussed followed by an example that deals with a real-life situation which involves all three.

### *UNIX does not support IBM Standard Label tapes*

In addition to user data files, IBM SL tapes contain labels which themselves are physical files on the tape. To process IBM SL tapes on UNIX, use a norewind device (e.g., `/dev/rmt/0mn`) and the `mt` command with the `fsf` subcommand to position the tape to the desired user data file. The formula for calculating the `fsf count` value is:

```
fsf_count_value = (3 x user_data_file_number) - 2
```

### *UNIX does not support multi-volume tapes*

To process multi-volume tapes on UNIX, the contents of each tape must be copied to disk via the `dd` command. After all of the tapes have been unloaded, the `cat` command can be used to concatenate all of the pieces (in the correct order). After this procedure is completed, the resultant file can then be processed from disk.

### *The DCB characteristics of the file must be known*

Since files coming from the mainframe do not have records that are delimited, the original DCB parameters must be used on the `INFILE` statement. In particular, `LRECL` and `RECFM` must be specified. Also, the block size must be specified on the `INFILE` statement (as `BLKSIZE`) or when using the `FILENAME` statement with the `PIPE` device-type and the `dd` command (as `ibs=`).

### Putting it all together...

For the sake of discussion, let's say we were given a two-reel multi-volume, standard label tape set containing a mainframe (EBCDIC) external file with the same record layout as the external file used in the previous examples. The creator told us that the logical record length is 7 and the record format is fixed.

The first thing we would need to do is use the `mt` command with the `fsf` subcommand to skip over the label on the first tape. Using the formula introduced earlier, where `user_data_file_number` is 1 (because we want the first (and in this case only) user data file), `fsf_count_value` is 1. After putting the tape in the drive and ensuring it is rewound, we issue the `mt` command with `fsf 1`, thereby positioning the tape to the user data file portion. Then we would use `dd` to unload the data portion of the tape to disk.

We would then repeat this process for the second tape, followed by the use of the `cat` command to concatenate the two pieces together, creating the resultant file. The final step would be to run a DATA step referrencing the disk file using the appropriate EBCDIC informats to convert the data.

Here's what the UNIX session would look like:

```
$ mt -t /dev/rmt/0mn rewind
$ mt -t /dev/rmt/0mn fsf 1
$ dd if=/dev/rmt/0mn of=/tmp/tape1 ibs=7
$
$ mt -t /dev/rmt/0mn rewind
$ mt -t /dev/rmt/0mn fsf 1
$ dd if=/dev/rmt/0mn of=/tmp/tape2 ibs=7
$
$ cat /tmp/file1 /tmp/file2 > /tmp/ebcdic.numbers
```

Here's what the DATA step would look like:

```
FILENAME EBCDIC '/tmp/ebcdic.numbers';
DATA NUMBERS;
  INFILE EBCDIC LRECL=7 RECFM=F;
  LENGTH DIGIT 8 TEMP $ 1 WORD $ 6;
  INPUT TEMP $EBCDIC1. WORD $EBCDIC6.;
  DIGIT=INPUT(TEMP,8.);
  DROP TEMP;
RUN;
```

## **Transport Formats**

Transport formats on tape are handled in a manner similar to external files. Therefore, if the tape(s) were created on a mainframe, the three previously discussed mainframe issues must be addressed.

Here is code that will create a transport format on tape:

```
X 'mt -t /dev/rmt/0mn rewind';  
LIBNAME TRANFILE XPORT '/dev/rmt/0mn';  
PROC CPORT LIBRARY=SASUSER FILE=TRANFILE;  
RUN;
```

Here is code that will import a transport format into a SAS library:

```
X 'mt -t /dev/rmt/0mn rewind';  
LIBNAME TRANFILE XPORT '/dev/rmt/0mn';  
PROC CIMPORT INFILE=TRANFILE LIBRARY=WORK;  
RUN;
```

## **SAS Data Sets**

SAS data sets can be written directly to tape by using a libref assigned via a LIBNAME statement with the TAPE engine specified:

```
LIBNAME SASLIB TAPE '/dev/rmt/0mn' ;
```

Multi-volume tape libraries are supported when the SAS system option TAPECLOSE=LEAVE is in effect. Unfortunately, remote tape devices can not be used when establishing the libref.

While SAS data libraries can be written directly to tape, a more efficient approach would be to use a staging directory so the data sets can be processed directly from disk. The UNIX command `tar` can be used to move the SAS data sets to/from the staging directory from/to tape.

## **Conclusion**

This paper discussed the various aspects of dealing with tapes on UNIX platforms. This discussion included tapes created on the mainframe and specific issues involving these "foreign" tapes.

For detailed information on the `FILENAME` and `LIBNAME` statements, the `TAPE` and `PIPE` device-types, the `RECFM`, `LRECL` and `BLKSIZE` host-options and the `TAPECLOSE` SAS system option, please refer to the "SAS Companion for UNIX Environments: Language."

For detailed information on the UNIX commands `mt` (`tctl` on AIX), `dd`, `cat`, `tar` and `remsh`, please refer to your UNIX system's man pages.