Appendix A

IBM File and Tape Formats

This appendix is intended primarily for the UNIX user who must understand IBM data sets and their variations. It should also be of interest to IBM users who want to know how IBM data sets are treated in the Connection Machine world.

NOTE: *Data set* is an IBM term for a data organizational entity that corresponds to the UNIX *file*. The differences and similarities between the IBM data set and UNIX file are discussed in Section A.1.

A.1 Data Set/File Formats

Under UNIX, files have no format. They are raw sequences of bytes residing on a disk, tape, or other medium. An application is responsible for appropriate interpretion of a UNIX file. For example, a UNIX text editor interprets a file as sequences of bytes terminated by newlines (carriage returns), while a database program might interpret a file as a series of fixed-length records.

Files are treated differently on an IBM system. The IBM operating system (typically MVS) handles most of an application's file-processing needs. It allows the user to select from a variety of common file formats. The user's application is then able to read and write these files without any responsibility for file processing that is format driven.

A *data set name* (*dsn*) consists of a sequence of uppercase alphanumeric characters, which may be organized into one or more *simple names*. A series of simple names is formed by separating subsets of characters with the period character (.). If the dsn contains no period characters, it consists of

Copyright © 1994 Thinking Machines Corporation

a single simple name. **payroll.july.software** is an example of a data set name comprised of three simple names.

A data set name can have a maximum of 44 characters, including the period characters used to separate simple names. Simple names can have from 1 to 8 characters. A simple name sequence constitutes a hierarchical order in which each simple name represents a category that subsumes all category names to its right. In the example just given, **payroll** is the supercategory that contains the **july** category, which, in turn, contains **software**.

Data sets are subdivided into *records*, which are ordinarily the smallest unit of addressable data in a data set. Records can be grouped into *blocks* for more efficient storage.

There are three categories of records in the IBM world — *fixed*, *variable*, and *unblocked*.

- *Fixed-format family* (F and FB) — Fixed-length records are, as the name implies, all of the same length. All logical records have the same length and all physical records have the same length. For performance reasons, multiple logical records may be bundled into single physical records using the fixed-blocked format.

  Fixed length is the most common format and is widely used in databases. Under UNIX, extracting a specific record in a fixed-length file is done by multiplying the record length by the number of the desired record and then doing a *seek* operation. On an IBM system, the user merely issues a read call with the number of the desired record as an argument.

- *Variable-format family* (V, VB, and VBS) — Variable-length records can be of arbitrary length. Under UNIX, the application is obliged to scan an entire file to locate a particular variable-length record. On an IBM system, the first few bytes of each record specify the length of the record, allowing the operating system to determine where the record begins and ends. Variable-length logical records are often subdivided into *segments*, allowing them to be written to tape in some regular fashion.

- *Unblocked* — All rules on logical and physical record lengths are voided. Logical and physical record lengths can be of different lengths. This approach is usually taken when there is an internal file structure that an applications programmer wishes to use.

Version 7.3, April 1994
Copyright © 1994 Thinking Machines Corporation

NOTE: For a more complete and detailed discussion of these topics, see the IBM publication, *MVS/Extended Architecture Data Administration Guide*, Order Number GC26-4013-2.

A.2 Tape Formats

As with any form of physical media, tapes impose their own structure upon the data that is stored on them. To a UNIX user, tapes are visible simply as files in the file system. When read, they present a sequence of bytes followed by an *end of file* condition. To an IBM user, a tape is an integral extension of the file system. Data sets that have moved to tape have not actually left the file system *per se*; they have been entered into a catalog and are archived. The operating system maintains this catalog and is capable of retrieving the data set when needed. Retrieval usually requires a tape robot or human to mount the tape(s) containing the data set.

When reading from and writing tapes, an IBM user is more aware of the structural features of the data layout than is a UNIX user. Sections A.2.1 and A.2.2 discuss these features in some detail.

A.2.1 Tape Labels

Ordinarily, when an IBM system writes a data set to tape, it also writes a label, which carries various kinds of administrative information about the contents of the tape. These labels are simply the electronic version of the physical label that is pasted on the outside of the tape case. They usually contain the following types of information:

- Volume serial number (vsn) — This is a six-digit number that is assigned to the tape, which the tape keeps for its lifetime.
- Data set name (dsn) — This is the name of the data set that follows the label.
- File format information — This is information about how the data is organized on the tape. It can include the record format (e.g., fixed, variable) and the record length.

Copyright © 1994 Thinking Machines Corporation

- Other — Tape labels can also include passwords, expiration dates, and other administrative information.

There are a number of formats in which labels are written, the most common of which are IBM Standard Label (SL) and ANSI label format. IBM also supports a tape format called *nonlabelled* tape.

A tape label consists of a small file written at the head of the tape and another small file after the last of the meaningful data. See Figure 3.

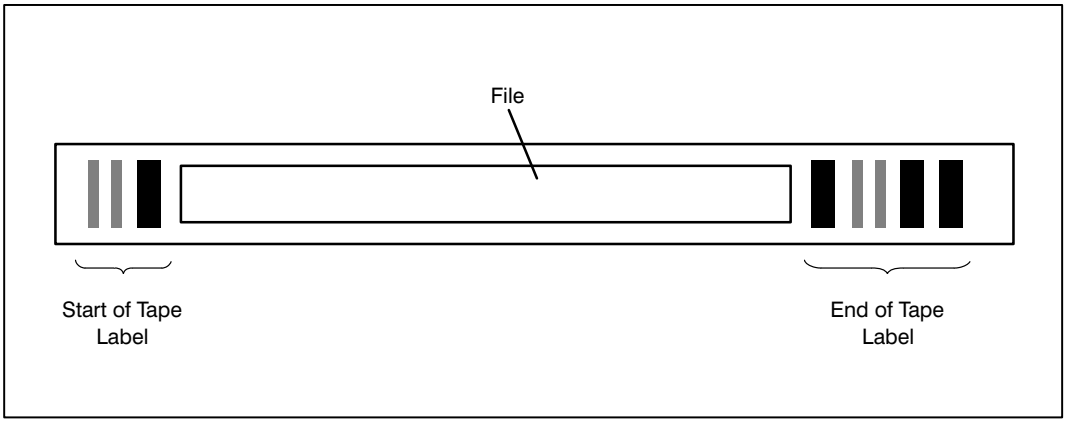

Figure 3. Tape label example.

Tape labels serve several useful purposes:

- They support the IBM concept of extending the file system out to a collection of tapes and they allow the computer to verify the contents of a particular set of tapes.
- They provide an element of protection. Jobs that run on an IBM system must specify the data set name and volume serial numbers of the tapes they want to work with. If the wrong volume is inadvertently mounted on the drive, or if the volumes of the data set are mounted out of order, the operator will be notified and requested to mount the correct volume.
- They describe the contents of the tape and provide the information needed to reconstruct the data set on disk as it has been written to tape.

Copyright © 1994 Thinking Machines Corporation

A label on a tape is essentially a small tape file that appears before any tape files containing data. This file typically contains from one to three 80-character records. If a program such as **dd**(1) is used to read the tape, this file can be extracted and examined by the user. However, the information contained in the label is meant to be read by a program and not a human.

Most tape label formats also include *trailer labels*, which are similar to the labels found on the front of the tape, except that they contain such information as checksums and block counts. It also indicates whether or not the data set contained on this volume extends to other volumes and, if so, identifies those volumes.

Many of the arguments to **itsibmtape** are used to verify the tape labels on the data set volumes. These include the **label**, **dsn**, **vsn**, and **rformat** options. The most important of these options is the **label** option.

The following explains how to process tapes that use the various categories of the IBM Standard Label format, the type most commonly used by far. See the discussion of **itsibmtape** in Chapter 2 for additional details.

label=isl

This format indicates that each volume of the input data set is preceded by an IBM standard label. The data read from this label is compared against the values given in the **dsn** and **vsn** options. If any of these fields do not match, the operator will be alterted and given the option of canceling processing of the mismatched tape, or allowing processing to continue. If no **rformat** option is specified, the data format, block size, and record size in the label are used during processing. When the end of volume on each tape is reached, the number of blocks read from the tape is compared with the block count given in the label. If these counts do not match, the operator is alterted and given the option of aborting or continuing processing. If the tape label indicates that the tape was written in compressed format, and the hardware the tape is loaded on is not capable of reading compressed media, the tape is rejected.

label=osl

This format is used when producing output tapes that already have a valid label written on them. Many data centers initialize the label on new tapes with the volume number the tape has been assigned in the center's library.

Copyright © 1994 Thinking Machines Corporation

This number stays with the tape as long as it is in the library. The first application to write to the tape puts the data set name and other important information in the label as well. When a volume is processed, the data set name and volume number from the label are compared against the data set name and volume number from the command line. If these do not match, the operator is alterted and given the option of aborting or proceeding. If the operator chooses to proceed, the tape will be written, but the *label will not be altered*.

label=osl,wsl

This form of the label option is used when producing output tapes and initializing, rewriting, or overriding the label data on a tape. When a volume is processed, it is first checked for the presence or absence of a label. If no label is found, one is written using the information provided on the command line. If a label *is* found, the operator is alterted that allowing the job to continue will overwrite the label on the tape. If the operator chooses to proceed, the old label data is replaced with the information given on the command line, and the data set is written to the tape.

A.2.2 Tape Files

A tape file is seen as a sequence of records in an IBM environment and a sequence of bytes by a UNIX system.

A tape file is not the same thing as an actual data file. Under UNIX, an entire directory structure may be archived, using the **`tar`** program, into a single tape file and individual UNIX files may then be extracted at a later time. On an IBM system, one data set may span multiple tapes (and therefore multiple tape files) or it may be contained within one tape file.

When a UNIX driver writes a tape file, it organizes the byte stream into a set of tape blocks, all of which, with the possible exception of the last, are of equal size. The last block will be smaller than the rest if the total number of bytes in the file is not an exact multiple of the *block size*. A file's block size is generally chosen to optimize tape drive usage.

A tape file is terminated by a *tape mark* (also called a *file mark*), which is written by the drive following the file. A tape mark is an end-of-file indicator that is visible to the applications programmer as an end-of-file condition on

Version 7.3, April 1994
Copyright © 1994 Thinking Machines Corporation

the tape device. Figure 4 illustrates a number of tape file layouts with tape marks in place. Note that the last file on the tape is followed by two tape marks, one for the file and the other indicating that it is the last file.

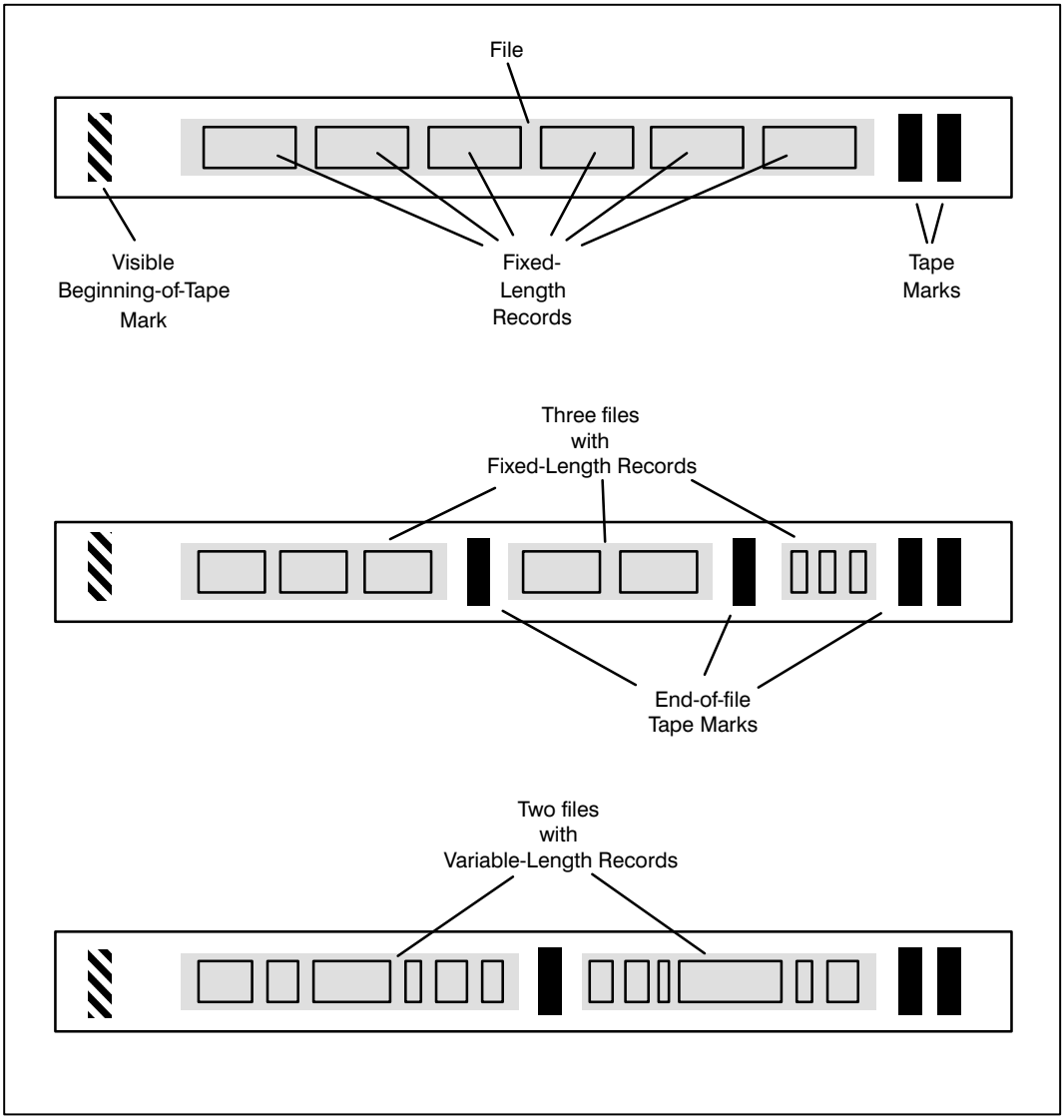

Figure 4. Tape mark examples.

All tape drives include special hardware to fast-forward past a tape mark, leaving the tape ready for reading the next file on the tape. A drive can also

Copyright © 1994 Thinking Machines Corporation

position itself on a specific block on the tape by performing a FSR (forward space record) or BSR (backward space record) operation. In addition, most drive units support a high-speed access method called *fixed-block mode*, which allows collections of similarly sized records to be extracted in a single operation.

The only indicators on a tape that must be obeyed are the BOT (beginning of tape) and EOT (end of tape) markers. The BOT marker notifies the tape drive that it has reached the load point when the tape is first mounted. It also tells the tape drive when to stop rewinding the tape. The EOT marker notifies the application that the physical end of the tape has been reached during tape advancement (reading or writing).

NOTE: For a more complete and detailed discussion of magnetic tape labels and file structure administration, see the IBM publication, *MVS/Extended Architecture Magnetic Tape Lables and File Structure Administration*, Order Number GC26-4145

Copyright © 1994 Thinking Machines Corporation