Introduction [1]

This user's guide describes the characteristics and capabilities of the tape subsystem, which is available on the UNICOS and the UNICOS/mk operating systems. The tape subsystem is also called the tape daemon-assisted interface and the Tape Management Facility.

The guide explains the ways in which you may work with the tape subsystem and provides many examples of commonly used commands. It also describes the use of the character-special tape interface.

This publication is organized as follows:

Chapter	Description
1	Introduces the terminology associated with the tape subsystem, discusses tape-related hardware, documents tape interfaces, and describes the tape subsystem's features.
2	Describes the structure of tape formats.
3	Describes the commands that access tapes by using the tape subsystem, as well as tape status and information commands.
4	Describes the use of the tape subsystem from Fortran programs.
5	Describes the use of the tape subsystem from C programs.
6	Describes the use of the character-special tape interface.
Appendix A	Describes system messages.
Appendix B	Describes tape daemon return values.

Appendix C	Lists the user man pages associated with the tape subsystem.

1.1 Terminology

This section describes terminology used throughout this manual and briefly describes the Cray Research systems that run the tape subsystem. It also describes the features of the tape subsystem.

The following terms are associated with the tape subsystem and are used throughout this manual:

Term	Definition
block size	The block size specifies the size (in bytes) of a data block on a tape.
device group	Each tape device belongs to a device group. The device group name is the generic device name in the configuration file. Also referred to as a *resource*.
device name	Each tape device is identified by a device name, which is defined by a device name entry in the tape configuration file.
device type	Each device has a device type, which is specified by a number. The different tape devices available on Cray Research systems.
file identifier	The file identifier is the name of the file recorded in the `HDR1` label of a labeled tape. If specified in lowercase, it is converted to uppercase, per ANSI standard.
job ID	The job ID is the process identification number unique to the shell or batch job currently in use.
label type	The label type may be one of the following: nonlabeled, IBM standard, ANSI standard, or single tape mark format.
path name	Each tape file is defined by a path name. You can specify the path name of the tape file by using the `tpmnt`(1) command. The system creates an entry in the directory specified by the path name. The

	tape device assigned to the tape file may change during volume switching. While a tape device is assigned to a tape file, you may not remove the path name of that tape file; the path name is removed when the tape device is released.
record length	The record length specifies the maximum length of a logical record (in bytes).
volume ID	The volume identifier is a character string that consists of 1 to 6 alphanumeric characters identifying a tape. The volume ID may also be referred to as the *volume serial number (VSN)* or the *internal VSN*.
external VSN	The external VSN is the human readable label applied to the tape's container.
format ID	A format identifier (ID) is the unique identifier for ER90 devices that is recorded on a tape during the volume format. It is a character string that consists of 1 to 6 alphanumeric characters. It is recommended that this label be the same as the volume ID. If you do not specify a format ID, the volume identifier is recorded on the tape as the format ID.

1.2 Hardware

This section describes the hardware of the tape subsystem. It includes a brief discussion of Cray Research systems, tape devices, and loaders.

The tape subsystem runs on Cray Research systems that have either the I/O subsystem Model E (IOS-E or IOS-V) or GigaRing support.

A wide range of tape devices and autoloaders are available on Cray Research systems. For more information on these, see your Cray Research sales representative.

1.3 Tape interfaces

There are two methods for accessing tapes:

- Tape daemon-assisted interface, commonly referred to as the tape subsystem

- Character-special tape interface

The tape daemon-assisted interface, which is called the tape subsystem in this manual, uses a kernel device driver and the tape daemon. It is the standard method of accessing tape devices. This interface supports many functions including tape resource management, device management, volume mounts and dismounts through operator communication or autoloader requests, label processing, volume switching, and error recovery.

Note: Chapters 2, 3, 4, and 5, and appendixes A and B describe the tape daemon-assisted interface. Only Chapter 6 describes the character-special tape interface.

The character-special tape interface to the tape subsystem is similar to the traditional UNIX process of accessing tape devices. It gives you unstructured access to the tape devices so that you can use standard UNIX commands and `ioctl`(2) requests to manage your tapes.

1.4 Tape subsystem features

This section briefly highlights the following features of the tape subsystem:

- Tape subsystem architecture
- Tape label support
- Tape positioning
- Front-end servicing
- User end-of-volume (EOV) processing
- Multifile volume allocation
- Concatenated tape files

This section does not include features of the character-special tape interface. For information on this interface, see Chapter 6, page 121.

1.4.1 Tape subsystem architecture

The basic elements of the tape subsystem are the tape daemon and the tape device driver. The tape daemon is started by the system operator or the system administrator, or it is started automatically as part of the system startup. The tape daemon has super-user privileges. Therefore, it can communicate directly

with the tape device driver to process your requests. You can execute a tape-related command, which builds a request and sends it to the tape daemon, by way of a tape daemon request pipe. Figure 1 shows the architecture of the tape subsystem.

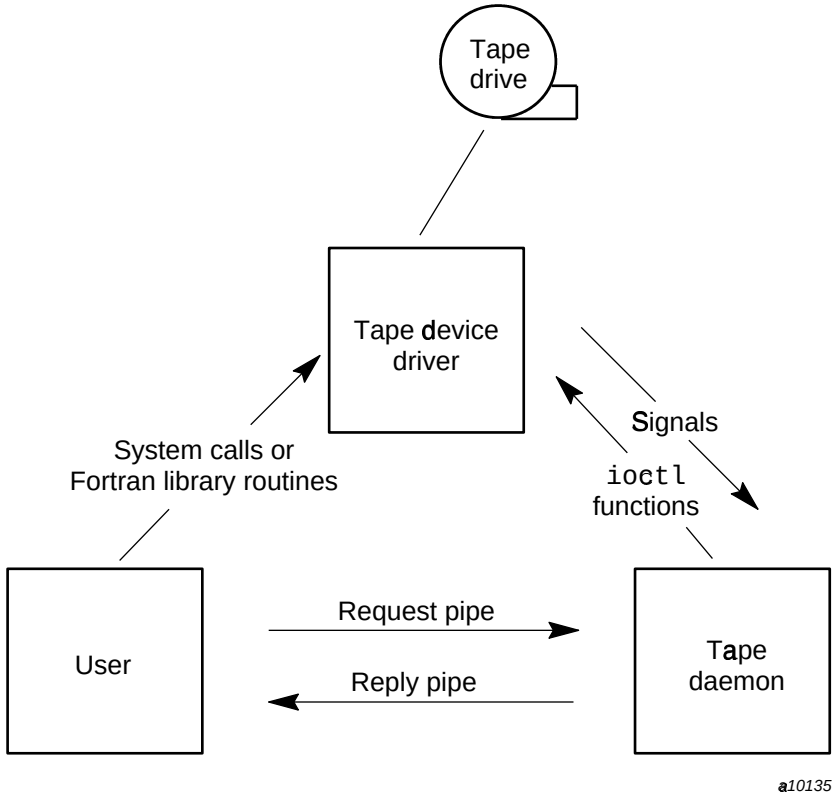

Figure 1. Tape subsystem architecture

The tape device driver signals the tape daemon if any of the following conditions occur:

- You issue an `open`(2) request to the tape path name.
- You issue a `close`(2) request to the tape path name.
- You issue the first I/O request to the tape path name.

- An I/O error occurs.
- A tape mark is read.
- An end-of-tape is detected during a `write`(2) operation.
- An end-of-file is detected, requiring tape mark processing.

If any of these conditions occur, your job is suspended until the tape daemon finishes processing. At this point, the tape daemon requests that the tape device driver either issue you an error message or allow you to continue. Figure 2 illustrates this process.

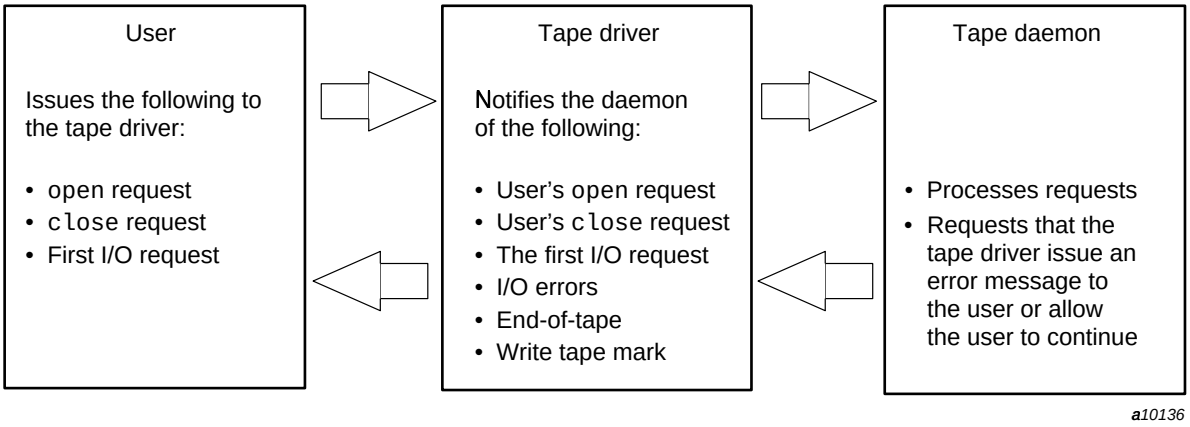

Figure 2. Communication between the user, tape driver, and tape daemon

1.4.2 Tape label support

The tape subsystem supports ANSI standard labels, IBM standard labels, single tape mark format tapes, or no labels. *Single tape mark format tapes* do not have labels and are terminated by a single tape mark at the end-of-volume, whereas a normal *nonlabeled tape* is terminated by two tape marks at the end-of-volume. Also, bypass-label processing is available to privileged users with proper user database (UDB) permission. *Bypass-label processing* lets privileged users read or write tape labels as regular files.

1.4.3 Tape positioning

Tape positioning lets you move to the beginning of a tape block. Tape movement may be forward or backward; however, tape positioning directives cannot be used to circumvent normal tape label processing or label checking unless you have tape-manager permission or bypass label permission and use an absolute track address positioning request (TR_PABS). You can position the tape file relative to a tape mark, tape block, or volume; or you can position the tape file to an absolute track address.

1.4.4 Front-end servicing

While processing tape labels, the tape subsystem requests permission and volume serial numbers from a specific front-end system. Front-end servicing is optional.

1.4.5 User end-of-volume processing

User end-of-volume (EOV) processing lets you gain control at the end of a tape volume. For EOV processing or positioning to a tape block, it is necessary to know that the file being processed is a tape file.

You may request to be notified when end-of-volume is reached. In addition, you can request special user EOV processing, which includes the reading, writing, and positioning of the volume before and after a volume switch. After special processing has completed, you must request that the tape subsystem resume normal processing.

1.4.6 Multifile volume allocation

Multifile volume allocation lets you process a multifile volume tape without the need for the system to unload and load tapes between files.

1.4.7 Concatenated tape files

The concatenated tape file feature lets you read multiple tape files as though they were one tape file. An end-of-volume status is returned for all concatenated files read, until the last file and its end-of-file is encountered.

1.5 Tape performance

This section briefly describes how to improve the performance of the tape subsystem. The tape transfer rate and system CPU time impact tape performance. The tape transfer rate is largely determined by the tape block size, but can be affected by other parameters such as buffer size and system buffer size.

System CPU time is largely a function of the number of system calls used for tape I/O; the primary determinant of system CPU time is the library buffer size.

You can improve tape performance by adjusting system buffering and kernel interfaces.

1.5.1 System buffering

System buffering is used to buffer user data when necessary. Buffering of tape data in the system buffer is optional and can be disabled with the `-U` option on the `tpmnt`(1) command. The defaults set by the system administrator are an important factor in the tape subsystem performance. For Fortran I/O, this means that data is transferred directly between the library buffer and the tape device. For C programmers using system calls directly for tape I/O, this option also imposes certain restrictions. See Section 5.2.

There are two advantages to using the `-U` option of the `tpmnt`(1) command:

- Because the system buffer is not used, the limit on the maximum tape block size imposed by the value of the system parameter `TAPE_MAX_PER_DEV` is removed. With the `-U` option, tape blocks are limited in size only by the size of the user's buffer, the device limit, and for blocked I/O, the `-b` option of the `tpmnt`(1) command.
- In some cases, performance is enhanced because the tape data is only copied twice, rather than three times on each transfer (once between the user's data area and the library, next between the library buffer and the system buffer, and then between the system buffer and the tape device).

In practice, the `-U` option is advantageous only with very large buffers or large tape blocks. For this discussion, large buffers are those over four times the maximum block size (MBS) of the tape, where the MBS is at least 128 Kbytes. In other cases, tape processing with the `-U` option is more expensive in system CPU time and slightly slower in terms of the tape data rate.

System buffering is an important factor in tape subsystem performance. The default is system buffering of all tape data. The `tpmnt -U` command disables

system buffering for blocked I/O. If the MBS (as specified by the `tpmnt -b` command) is less than half the size of `TAPE_MAX_PER_DEV`, the kernel driver divides this buffer into two equal parts and performs asynchronous read-ahead and write-behind operations (double buffering). If the MBS is larger than half the size of `TAPE_MAX_PER_DEV`, the system buffer is used as a single buffer and tape performance will suffer. Tape blocks larger than `TAPE_MAX_PER_DEV` will result in an error. For byte stream I/O, the ER90 driver divides the buffer into two equal parts and performs asynchronous read ahead and write behind operations (double buffering). (For information about the ER90 format, see Section 2.3, page 23.) The `TAPE_MAX_PER_DEV` is configurable by the system administrator. See the *Tape Subsystem Administration*, Cray Research publication SG–2307, for details.

1.5.2 Kernel interface

The kernel interface to tape I/O is through the standard `read`(2) and `write`(2) system calls. It is called transparent I/O.

Flexible file I/O (FFIO) library routines provide another way to perform tape I/O. Fortran I/O is based on FFIO, and the C library contains FFIO routines. See Chapter 4, page 41, for further information. Also see the *Application Programmer's I/O Guide*, Cray Research publication SG–2168, for more information on FFIO.

1.6 Tape multilevel security

When discussing the UNICOS tape subsystem as used on a UNICOS multilevel security (MLS) system in this manual, it is assumed that you have read and understood the following information:

- *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111
- *General UNICOS System Administration*, Cray Research publication SG–2301 MLS chapter of publication SG-2301

Warning: If your site is running a Trusted UNICOS system, you must thoroughly understand the information and follow all procedures discussed in the documents listed previously and noted throughout the rest of this manual to properly use the UNICOS tape subsystem on a Trusted UNICOS system.

UNICOS systems with the UNICOS MLS feature require a user's security label to be equal to the tape security label to write to that tape. The user's security label must dominate the tape security label to read to that tape.

All files on a tape must be at the same security label.

All tape activity can be audited. The tape security administrator can tune the tape subsystem to control what can be audited.