ConvexOS Tape System Guide

Document No. 710-003130-000

First Edition
December 1989

CONVEX Computer Corporation
Richardson, Texas USA

ConvexOS Tape System Guide

Order Number: DSW-016

Printed in the United States of America

Revision Information for
ConvexOS Tape System Guide

Edition	Document No.	Description
First	710-003130-000	Released with ConvexOS V8.0, December, 1989

Table of Contents

Appendices

List of Tables

List of Figures

Preface

Purpose

This manual describes how to configure, administer, maintain, and use the ConvexOS tape system. It is a procedural guide—not a reference manual—that explains how to perform the various tape system tasks and describes concepts that underlie these procedures. Reference information (i.e., concise specifications of command syntax and parameter values) can be found in *ConvexOS Programmer's Reference* pages that pertain to the tape system. Reference pages exist for each tape system command that is described in the *ConvexOS Tape System Guide*; cross-references are given to the manual pages whenever appropriate.

Audience

This guide contains information for *system administrators*, who must set up, configure, and maintain the tape system; *users*, who employ tape system resources to read and write tape data; and *operators*, who assist users by servicing tape system resource requests and monitoring the status of the tape system.

Organization

Each chapter of this guide contains information pertinent to one of these three areas of responsibility. These chapters can be distributed separately to those whose responsibilities are clearly confined to one of these roles (e.g. operators or users). Alternatively, the guide can be used as a single unit by readers who must see the "big picture" because their duties cut across all three roles.

In addition to this preface, the following chapters are included in the *ConvexOS Tape System Guide*:

- Chapter 1, "Tape System Overview," provides general information about the functioning of the tape system. This information is particularly important for system administrators who must make decisions about the configuration and operation of the tape system.
- Chapter 2, "Configuring the Tape System," contains information for system administrators about how to use tape system configuration utilities.
- Chapter 3, "Using Tape System Resources" contains information for the tape system user.
- Chapter 4, "Tape System Operations," describes how to use Operator Request Management (*opreq*) to process requests for tape resources.
- Appendix A, "Glossary of Terms," is a glossary of terms used in this publication.
- Appendix B, "Quick Command Reference" contains a summary of tape system commands.
- Appendix C, "Error Message Reference," lists all error messages that are given by the tape system.
- Appendix D, "Reporting Problems," provides instructions for reporting software and documentation problems to the CONVEX Technical Assistance Center (TAC).

Notation Conventions

The following conventions are used in this document:

- Words enclosed in rounded rectangles indicate keyboard keys. For example, RETURN refers to the carriage return key. Words separated by a hyphen and enclosed in rounded rectangles indicate two keys that you must press simultaneously. For example, CTRL-X indicates that you must press the CTRL key while simultaneously pressing the X key.
- The word "enter" in a phrase such as "enter a command" means that you type the command and press the carriage return key. In contrast, the word "type" (for example, "type a line of text") means that you do not press the carriage return key.
- Within command sequences

Boldface	indicates characters that must by typed just as they appear. If the command can be abbreviated, the shortest abbreviation is indicated by upper case letters.
Italics	indicate file names or arguments for which you must substitute actual file names or arguments. Angle brackets (< >) also surround such arguments.
Brackets ([ ])	designate optional entries.
Or Bar (|)	separates alternative choices.
Horizontal ellipsis (...)	shows repetition of the preceding item(s)

Consider the following example:

COMmand *<input_file>* [, . . .] [*<output_file>*]

In this example, **COMmand** may be typed as "com" or "command"; *input_file* indicates a file name that must be supplied by the user; the horizontal ellipsis in brackets indicates that additional input file names, separated by commas, may be supplied; and *output_file* indicates an optional file name.

- References to the *ConvexOS Programmer's Reference* appear in the form *adb*(1), where the name of the man page is followed by its section number enclosed in parentheses.
- `Constant-width font` is used for examples of computer-generated output and FORTRAN and C code.
- The ConvexOS shell prompt is printed as a percent sign (%) unless the user is logged on as *root*; in that case, the prompt is a pound sign (#).

Related Documents

The following documents are provided by CONVEX Computer Corporation to provide additional information about ConvexOS and the tape system:

- *ConvexOS Programmer's Reference* is the standard reference for the ConvexOS operating system.
- *ConvexOS System Manager's Guide* contains the information needed to manage and maintain a CONVEX supercomputer.

The following document is not provided by CONVEX, but contains relevant information:

The ANSI standard for tape labeling (X3.27-1978), is contained in FIPS PUB 79, "Magnetic Tape Labels and File Structure for Information Interchange," available from the Office of Standards Administration, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D.C. 20234.)

Ordering Documentation

To order CONVEX documentation, complete the CONVEX Documentation and Subscription Service Order Form enclosed in the Documentation Catalog included with this manual.

In some situations, the current edition may not be desired. To receive a specific edition of a manual, contact the local CONVEX sales office or call the Technical Assistance Center (TAC).

Technical Assistance

Hardware and software support can be obtained through the CONVEX Technical Assistance Center (TAC):

Within the continental U.S.	1(800)952-0379.
From locations in Alaska, Hawaii, and Canada	1(214)497-4379.
From all other locations	contact the nearest CONVEX office.

Reader's Forum

If you wish to mail your comments to us, please use the form at the end of this manual and list the document page number with your questions and comments. Thank you.

Chapter 1
ConvexOS Tape System Overview

The ConvexOS Tape System consists of operating system software that supports data storage and retrieval on magnetic tape. The tape system is distributed as part of the ConvexOS operating system.

This chapter contains information that is vital to making informed decisions about tape system configuration and operation. This information includes:

- Section 1.1, "Some Words to Know" is an important note about terminology used in this manual.

> **CAUTION** Be sure to read section 1.1, "Some Words to Know," or you may misunderstand important information in this manual.

- A summary of the main features of the ConvexOS tape system is given in section 1.2, "ConvexOS Tape System Features."
- The tape system daemons are explained in section 1.3, "Tape System Daemons."
- Tape system device files are described in section 1.4, "Tape System Device Files."
- An explanation of the tape mounting operation is given in section 1.5, "Understanding the Tape Mounting Operation."

1.1 Some Words to Know

To avoid confusion, the terms "mount" and "device" are used in a more precise sense than usual in this publication:

- In this guide, *mount* refers only to the operation of establishing a logical link to a device file so that the tape system user can access it. By extension, tape drives are also said to be "mounted" when the device that controls them is mounted, and a tape on such a drive is also "mounted". To avoid confusion, "mount" will not be used in this manual to refer to the mere act of placing a tape on a drive, although popular use of this word encompasses this meaning also.
- In this guide, *device* refers only to a special device file, and never to a mechanical device such as a tape drive. A "device" is the same as a "device file" (such files are also called "special files").
- A *node* is a device that has been associated with a tape drive by the tape system configuration utility *tpconfig*. (This utility is described in Chapter 2.)

Refer to Appendix A, "Glossary of Terms" for more information about the terms used in this manual.

1.2 ConvexOS Tape System Features

The following sub-sections describe the main features of the tape system.

1.2.1 ANSI Standard Labels

As of ConvexOS V8.0, the tape system supports reading and writing tapes labeled according to ANSI standard X3.27-1978, as well as unlabeled tapes. Adherence to the ANSI standard permits tapes to be interchanged between CONVEX machines and other systems that support this standard tape format.

Access to labeled tapes is fully "transparent" in the sense that the user need not be concerned with structural details of the labels themselves in order to read or write labeled tapes, nor with the mechanics of creating or verifying labels. All label-related "house keeping" operations are handled by the system. When reading or writing labeled tapes, utilities such as *cat* and *cp* will function as they always have.

The ANSI labeling scheme permits tapes to be identified by Volume Serial Numbers (VSN) that are stored on the tape, and that can be used to identify tapes in mount requests. In addition, users can specify file attributes (access permissions, block size, logical record delimiter, record format, record size, and file name) for reading and writing files on ANSI-labeled tapes.

Tape labeling provides the capability to set access permissions for tapes or files on tapes, and thus provides a measure of security for tape data. The system can be also configured to permit certain users or groups to bypass access restrictions on labeled tapes. Because those given this privilege could alter or destroy sensitive data, system administrators should be cautious in assigning this privilege. (This is done with the *tpconfig* utility.)

Label access restrictions are only enforced if the main tape system daemon—*tpdaemon*—is running.

CAUTION

On VMEbus drives, label verification is done each time the tape is at Beginning of Tape and the user requests to access, (i.e., read or write) the tape. This is done to prevent users from circumventing access restrictions by mounting a tape to which they have access rights (e.g., an unlabeled tape or a labeled tape to which they have read and write privileges), and then substituting a labeled tape to which they do not have access rights.

However, label verification on Multibus drives is performed only when the drive is first mounted. (This is a consequence of constraints imposed by Multibus hardware architecture.) Consequently, the system might not recognize an attempt to wilfully circumvent labeled tape security by switching tapes on mounted Multibus drives.

1.2.2 Operator Request Management

The ConvexOS tape system has options to handle tape mounting requests directly by the tape system or to pass them through an operator. If an operator is available, Operator Request Management (*opreq*) queueing can be turned on. This utility allows tape requests to be queued until they can be serviced. *opreq* provides an interactive windowing interface for the operator.

1.2.3 Multivolume Tape Sets

The ConvexOS tape system supports multivolume ANSI-labeled tape sets. Such sets consist of two or more tapes that are identified as volumes of the same set in the ANSI header label on the tapes. If the Operator Request Management utility *opreq* is active, the system automatically issues requests to the operator to switch tapes whenever necessary.

1.2.4 Optional Control of Drives

If the main tape system daemon—*tpdaemon*—is running, then the tape system will normally control access to all drives. (This daemon should be started when the system is booted and should always be running.) Under these circumstances, drives can be accessed only if the user first issues a *tpmount* request and the request is granted. Once mounted, a drive is reserved for the exclusive use of its owner until it is unmounted. As long as *tpdaemon* is running, the tape device driver will reject any request to directly access a controlled drive (using I/O utilities like *cp* or *cat*).

However, by using the tape system configuration utility *tpconfig*, system administrators may cause some drives to be uncontrolled even when *tpdaemon* is running. Such a drive cannot be accessed through tape system utilities such as *tpmount*, but can be accessed directly by anyone who has access permissions to devices associated with the uncontrolled drive.

1.2.5 Automatic Drive Selection

If the user does not specify a particular drive or device file when mounting a tape, the tape system automatically chooses a drive and device that have the characteristics specified by the user in the mount request (*tpmount*). Default device characteristics for the mount request can be defined via the tape system configuration utility *tpconfig*. These defaults will be used to select a device and drive if the user does not specify them in the mount request.

1.2.6 Automatic Volume Recognition

The tape system also features automatic volume recognition (AVR). When a labeled tape is placed on a drive, the system reads the volume serial number (VSN) of the tape; if the VSN was specified in a *tpmount* request previously submitted by a user, the system will automatically match the tape to the request without further operator intervention. If a matching *tpmount* is not found for a tape, the system will wait for the user to issue such a request, and will then match the tape to the request.

If the tape is unlabeled, AVR will not be used. In this case, the operator can choose a drive to mount for the tape.

1.2.7 ConvexOS Tape System Utilities and Commands

The following utilities and commands are part of the ConvexOS tape system:

- *mt*—Includes a set of tape manipulation commands. *mt* is described in Chapter 3, "Using Tape System Resources"; for reference information, refer to the *mt*(1) page in the *ConvexOS Programmer's Reference.*
- *opreq*—(Operator Request Manager) queues tape mounting requests and provides an interactive, windowing interface for operators to handle tape resource requests. This utility is described in Chapter 4, "Operator Request Management"; for reference information, refer to the *opreq*(1) page in the *ConvexOS Programmer's Reference.*
- *tpattr*—sets attributes of files read or written on ANSI-labeled tapes. *tpattr* is described in Chapter 3, "Using Tape System Resources"; for reference information, refer to the *tpattr*(1) page in the *ConvexOS Programmer's Reference.*
- *tpconfig*—Used to configure the tape system. *tpconfig* can be used interactively to enter multiple configuration commands, in batch mode to process files containing commands, or it can be invoked as a single command from the ConvexOS shell prompt. This utility is described in Chapter 2, "Configuring the Tape System"; for reference information, refer to the *tpconfig*(1) page in the *ConvexOS Programmer's Reference.*
- *tplabel*—Used to write new ANSI labels on a tape. *tplabel* is described in Chapter 3, "Using Tape System Resources"; for reference information, see the *tplabel*(1) page in the *ConvexOS Programmer's Reference.*
- *tpmount*—Used to mount a device and drive. *tpmount* is described in Chapter 3, "Using Tape System Resources"; for reference information, see the *tpmount*(1) page in the *ConvexOS Programmer's Reference.*
- *tpqueue*—Used to obtain information about the tape request queue and utilization of tape drives. *tpmount* is described in Chapter 3, "Using Tape System Resources"; for reference information, refer to the *tpqueue*(1) page in the *ConvexOS Programmer's Reference.*
- *tpunlabel*—Used to remove ANSI labels from a tape. *tpunlabel* is described in Chapter 3, "Using Tape System Resources"; for reference information, refer to the *tpunlabel*(1) page in the *ConvexOS Programmer's Reference.*
- *tpunmount*—Reverses the effect of a *tpmount* command. *tpunmount* is described in Chapter 3, "Using Tape System Resources"; for reference information, refer to the *tpunmount*(1) page in the *ConvexOS Programmer's Reference.*
- *tpwait*—Causes the caller to wait until a background *tpmount* request has been completed. *tpwait* is described in Chapter 3, "Using Tape System Resources"; for reference information, refer to the *tpwait*(1) page in the *ConvexOS Programmer's Reference.*

1.2.8 ConvexOS Tape System Library Functions

A set of C language library functions that correspond to most of the commands described above provides the programmatic interface for the tape system. The library functions are:

- *tpmount()*
- *tpunmount()*
- *tpqueue()*
- *tpattr()*
- *tplabel()*
- *tpunlabel()*
- *tpwait()*
- *tperror()*

These functions are described on the *tape*(3) page of the *ConvexOS Programmer's Reference.*

1.3 Tape System Daemons

The tape system will not function unless certain processes—or *daemons*—are running on the system. These daemons are:

- *tpdaemon*—The main tape system daemon; the pathname of the executable code is */usr/lib/tape/tpdaemon.* If this daemon is not running, the tape system is disabled. Mount requests will not work, drives will not be controlled, and label access restrictions will not be enforced.
- *opreq_daemon*—The daemon that is responsible for *opreq* queueing; the pathname of the executable code is */usr/lib/opreq/opreq_daemon. opreq* queueing allows tape mount requests to "wait in line" until the necessary resources are available. The *opreq* utility can be used to manage tape requests queued by *opreq_daemon.*
- *ansidaemon*—the ANSI labeled-tape daemon; its pathname is */usr/lib/tape/ansidaemon.* (This daemon is automatically started by the tape system whenever it is needed; the user should never directly start an *ansidaemon.*)

Note	Other programs needed by the tape system are located in */usr/convex*; this directory must be in the search path of all tape system users.

1.3.1 *tpdaemon* and *opreq_daemon*

The tape system is not active unless *tpdaemon* is running. This process should be started when the system boots, and should always be running.

> **CAUTION** If *tpdaemon* is not running, devices normally under control of the tape system will not be protected: any user who has permission to access the special device files and who has physical access to the tape drives can use them.

The *opreq_daemon* is responsible for the functioning of the Operator Request Manager (*opreq*). This daemon need not be running if *opreq* will not be used.

tpdaemon and *opreq_daemon* are started by entries in the */etc/rc.std* file, which is executed by ConvexOS when it boots. (The */etc/rc* file is executed first; this file in turn executes */etc/rc.local* and then */etc/rc.std*.) The following lines in the */etc/rc.std* test for the presence of the executable files containing the daemons, and start them if the files are found:

```
if [ -f /usr/lib/opreq/opreq_daemon ]; then
        /usr/lib/opreq/opreq_daemon & echo -n ' opreq_daemon'
fi

if [ -f /usr/lib/tape/tpdaemon ]; then
        /usr/lib/tape/tpdaemon & echo -n ' tpdaemon'
fi
```

If necessary, these daemons can be killed and started by entering appropriate commands. (Use *kill* to kill the daemons, and the name of the file containing the daemon's executable code to start them.) You must, however, be *root* to kill or start the daemons. Refer to the *kill(1)* page of the *ConvexOS Programmer's Reference* for for more information about this command.

If *opreq* is to be used, then *opreq_daemon* must be started before *tpdaemon*. The *portmap* process must be started before *opreq_daemon* and *tpdaemon*. *portmap* is normally started in the file */etc/rc.local*.

You may start *opreq_daemon* with the *-unmount* parameter. The effect of specifying this parameter is to cause tape mount requests that have been cancelled by the user (with *tpunmount*) to be deleted immediately from the *opreq* window. If this parameter is not specified, such canceled requests will continue to be displayed until the operator cancels it by highlighting the request and typing *select-done*. (*opreq* is described in Chapter 4, "Operator Request Management.")

CAUTION

To terminate *tpdaemon* with the *kill(1)* command, use the default signal (SIGTERM). If this signal is not used (e.g., if you enter `kill -9` followed by the PID of the daemon process), the daemon may remain registered with *portmap*, causing future tape system requests to be suspended indefinitely instead of failing.

If *tpdaemon* is killed, all queued tape requests (e.g., *mount, dismount*) are deleted.

You must be *root* to kill or start these daemons.

For reference information about *ansidaemon*, *tpdaemon*, and *opreq_daemon*, see the *ansidaemon*(8), *opreq_daemon*(8) and *tpdaemon*(8) pages in the *ConvexOS Programmer's Reference*.

1.3.2 *ansidaemon*

A "label daemon" is forked by *tpdaemon* whenever a request to mount a labeled tape is processed. This daemon is a filter between the user and the tape device; it handles tape label operations (e.g., recognition, creation, and deletion of labels). Since only ANSI labels are currently supported for the ConvexOS tape system, there is only one kind of label daemon: *ansidaemon*. One *ansidaemon* is started for each mounted tape device through which ANSI-labeled tape I/O will be done; thus, there may be more than one of these processes per drive at any one time. (Each *ansidaemon* exits when its device is dismounted.)

1.4 Tape System Device Files

ConvexOS handles all input and output through special device files (often simply called "devices"). Tape I/O is no exception; it is done through device files that are logically linked to tape drives by the operating system. Thus, after linkages between user, device, and tape drive have been made (as a consequence of the mounting operation), users can employ any ConvexOS commands that are normally used for file I/O (such as *cp* or *cat*) to copy data to or from tape. However, as far as the user is concerned, the destination or source of the data is a file.

Both block and character I/O is supported for the tape system, and is implemented by device files that perform one or the other kind of I/O. For routine data storage and retrieval on tape, character mode should normally be used. For this mode, the kernel will not buffer I/O; everything will be written or read as directed by the user. Using block I/O mode makes a tape function much like a disk (a typical block device). If block mode is used, I/O will be buffered (as disk I/O is).

In addition to device files required for block and character I/O, special device files must be created to support I/O to labeled tapes. Device files are described in the next subsections.

> **Note**
>
> The *ioctl()* system call and, therefore, utilities such as *mt* that use *ioctl()*, can be used only with character devices. Seek operations—implemented by the *lseek()* system call— can only be used with block devices.

1.4.1 Ordinary Tape Devices

Tape system devices are located in the */dev* directory. Their names have the form *mt<n>* or *rmt<n>*, where *<n>* is an integer (e.g., */dev/mt0*, */dev/rmt0*, */dev/rmt4*, etc.). The *mt* devices are used for block I/O; the *rmt* devices are used for character I/O.

Since tape I/O can only be done by directing it through a device, and since each device talks only to one drive, there must be at least one device for each tape drive on the system. In practice, there will usually be more than one. This is because several device files may be defined to differentiate various attributes of the same drive (such as tape density or whether the tape will be rewound when it is closed). In particular, each drive will generally have six *mt<n>* and six *rmt<n>* devices associated with it. (The different device files that are connected to the same drive are sometimes called the "minor devices" or "nodes" of that drive.)

For example, the first tape drive may have the following device files associated with it:

- **mt8** 1600 bpi density, block access, automatic rewind
- **mt12** 1600 bpi density, block access, no automatic rewind
- **mt16** 6250 bpi density, block access, automatic rewind
- **mt20** 6250 bpi density, block access, no automatic rewind
- **rmt8** 1600 bpi density, character access, automatic rewind
- **rmt12** 1600 bpi density, character access, no automatic rewind
- **rmt16** 6250 bpi density, character access, automatic rewind
- **rmt20** 6250 bpi density, character access, no automatic rewind

Devices are usually created when the ConvexOS system is installed. Should it be necessary to create more devices (e.g., because additional drives have been installed), this can be done with the */dev/MAKEDEV* script. The script is invoked with the drive type ("ta") and the unit number of the drive (e.g., "2") as parameter. The script then invokes the ***mknod*** command to create all the needed device files for the drive. For example, to create the devices for drive 2, you would enter the following command:

```
# MAKEDEV ta2
```

Refer to *MAKEDEV*(8) and *mknod*(8) in the *ConvexOS Programmer's Reference* for more information about how to create special device files. In addition, conventions for naming ConvexOS tape devices are explained in the *mtio(4)* pages of the *ConvexOS Programmer's Reference.*

The connection between each device file and the tape drive to which it "talks" is determined by the *major and minor device number* of the file. These device numbers are established when the device is created. The major device number identifies the device driver that performs I/O to the device. The I/O characteristics (density, access mode, rewind status) are determined by the minor device number. The device numbers can be seen by doing a "long list" (*ls -l*) of the */dev* directory. The following is a portion of such a listing:

```
brw-------  1 root       4,  12 Feb 22  1988 /dev/mt12
brw-------  1 root       4,  16 Feb 22  1988 /dev/mt16
brw-------  1 root       4,  20 Feb 22  1988 /dev/mt20
brw-------  1 root       4,   4 Feb 22  1988 /dev/mt4
brw-------  1 root       4,   8 Feb 22  1988 /dev/mt8
```

Major and minor device numbers are the two sets of numbers that appear just before "Feb". For example, the major device number of */dev/mt12* is "4"; the minor device number of this file is "12". Since the major device number is the same for all devices, the same device driver (a magnetic tape driver) is used to handle I/O for all of them. The minor device number has a special meaning for the device driver—this number specifies the tape drive and the attributes of that drive (e.g. tape density, no rewind, etc.).

1.4.2 Labeled-Tape Devices

To permit the ConvexOS tape system to handle labeled tapes, a new set of tape device files must be defined in addition to the ordinary tape device files */dev/mt<n>* and */dev/rmt<n>* described above. (Currently, only ANSI-standard tape labeling is supported.) The labeled tape (*lt*) device files are used in a "filtering" stage of I/O to labeled tapes: output is first directed to the *lt* file and only then to the ordinary device file; input from the tape goes first to the device file, then to the *lt* device file and only then to the user.

lt device files are located in the */dev/lt* directory; by convention, their names have the form *c<n>* or *u<n>*, where *<n>* is an integer, e.g., *c1, u1, c2, u2,* etc.

The *c<n>* and *u<n>* files work together in pairs: each pair of *c<n>* and *u<n>* files can be said to constitute one "labeled-tape device." For example, files *c5* and *u5* make up a labeled-tape device—*lt5*. At least one labeled-tape device must be defined for each tape drive that is on the system and that may be used for labeled-tape I/O. The *u<n>* file is the file that receives input from the tape system user; I/O can be written to, or read from, this file. The *c<n>* file is for internal use by the tape system.

One important difference between labeled-tape device files and ordinary device files is that labeled-tape device files do not correspond to only one particular tape drive. Instead, each labeled-tape file may be linked to any ordinary device file and, therefore, to any tape drive. Such a linkage is made whenever a labeled tape is mounted, and a labeled device file is needed to mediate I/O. These links are temporary; they are dissolved when the device is dismounted.

The ConvexOS installation script runs */dev/MAKEDEV* to create ten labeled-tape devices (i.e., ten pairs of **c**<*n*> and **u**<*n*> files). If they are needed, additional tape devices can be created by running */dev/MAKEDEV* again. The following example shows how *MAKEDEV* can be invoked to create four logical tape devices—*lt0* (made up of *c0* and *u0*), *lt1* (made up of *c1* and *u1*), *lt2* (made up of *c2* and *u2*), and *lt3* (made up of *c3* and *u3*):

```
# cd /dev
# MAKEDEV lt0 lt1 lt2 lt3
# ls -l lt
total 0
crw-------  1 root      20,524288 Aug 16 13:05 c0
crw-------  1 root      20,524289 Aug 16 13:05 c1
crw-------  1 root      20,524290 Aug 16 13:05 c2
crw-------  1 root      20,524291 Aug 16 13:05 c3
crw-------  1 root      20,   0 Aug 16 13:05 u0
crw-------  1 root      20,   1 Aug 16 13:05 u1
crw-------  1 root      20,   2 Aug 16 13:05 u2
crw-------  1 root      20,   3 Aug 16 13:05 u3
#
```

1.5 Understanding the Tape Mounting Operation

Before data can be transferred to or from a magnetic tape on a drive that is under the control of the tape system, the user must establish exclusive control of the drive. Under ConvexOS, control is granted by making certain file linkages and changing the ownership attributes of device files. When control is established over a device—and, therefore, a tape drive—it is said to be *mounted.*

Mounting a tape drive is done with *tpmount.* The use of this command is described in Chapter 3, "Using Tape System Resources"; the information below will be helpful to system administrators who must know the "background" of what happens when a device is mounted. Such knowledge may be required, for example, when making tape system configuration decisions while running *tpconfig* (refer to section 2.1 "Using the Tape System Configuration Utility *tpconfig*").

1.5.1 Mounting for Unlabeled Access

The following is done by the tape system when it mounts a drive for unlabeled tape I/O:

- The tape system *tpdaemon* selects the appropriate device file (i.e., one that has the attributes specified in the *tpmount*), and changes ownership of that file to the ID of the user.
- The *tpdaemon* then creates a file that is, by default, called "TAPE" and located in the user's working directory. This file (also called a "symbolic link") is linked to the device file; any I/O commands directed to the new file will be sent to the device file. (By using the *-s* option of *tpmount*, the user can specify any symbolic link name that is desired.)

For example, here is what an *ls -l* of the working directory of user "dunbar" shows before the *tpmount* command is given:

```
% ls -l
total 376
drwxr-xr-x  2 dunbar          512 Oct  5 14:31 bin
drwxr-xr-x  2 dunbar          512 Oct 16 16:48 gnu
-rw-------  1 dunbar          130 Oct 16 15:54 test
-rw-r--r--  1 dunbar          130 Oct 16 15:38 tfile
-rw-r--r--  1 dunbar          185 Oct 16 15:41 tfile2
-rw-r--r--  1 dunbar          130 Oct 16 15:41 tfile2~
-rw-r--r--  1 dunbar            0 Oct 11 15:58 troff.out
```

A "long list" of the device driver files in directory */dev* might show the following at this point:

```
% ls -l /dev/*mt*
brw-rw-rw-  1 root       4,   0 Jun 14 12:30 /dev/mt0
brw-------  1 root       4,  12 Jun 14 12:30 /dev/mt12
brw-------  1 root       4,  16 Jun 14 12:30 /dev/mt16
brw-------  1 root       4,  20 Jun 14 12:30 /dev/mt20
brw-rw-rw-  1 root       4,   4 Jun 14 12:30 /dev/mt4
brw-------  1 root       4,   8 Jul  8 19:35 /dev/mt8
crw-rw-rw-  1 root       4,   0 Jun 14 12:31 /dev/rmt0
crw-------  1 root       4,  12 Aug  3 13:16 /dev/rmt12
crw-------  1 root       4,  16 Oct  4 14:47 /dev/rmt16
crw-------  1 root       4,  20 Oct 16 17:46 /dev/rmt20
crw-rw-rw-  1 root       4,   4 Jun 14 12:31 /dev/rmt4
crw-------  1 root       4,   8 Oct 14 22:44 /dev/rmt8
```

After "dunbar" has given a *tpmount* command, the list shows a new file, *TAPE*, in the user's working directory:

```
% tpmount
Tape device /dev/rmt20 allocated.

% ls -l
total 25
lrwxrwxrwx  1 dunbar          10 Oct 16 17:46 TAPE -> /dev/rmt20
drwxr-xr-x  2 dunbar         512 Oct  5 14:31 bin
-rw-r--r--  1 dunbar       16407 Oct 11 14:47 config.db
drwxr-xr-x  2 dunbar         512 Oct 16 16:48 gnu
-rw-------  1 dunbar         130 Oct 16 15:54 test
-rw-r--r--  1 dunbar         130 Oct 16 15:38 tfile
-rw-r--r--  1 dunbar         185 Oct 16 15:41 tfile2
-rw-r--r--  1 dunbar         130 Oct 16 15:41 tfile2~
-rw-r--r--  1 dunbar           0 Oct 11 15:58 troff.out
```

The file *TAPE* is linked to a device file, */dev/rmt20*. (The file need not be named *TAPE*; it is possible to specify any name for this "symbolic" file.) The */dev* directory now looks like this:

```
% ls -l /dev/*mt*
brw-rw-rw-  1 root       4,   0 Jun 14 12:30 /dev/mt0
brw-------  1 root       4,  12 Jun 14 12:30 /dev/mt12
brw-------  1 root       4,  16 Jun 14 12:30 /dev/mt16
brw-------  1 root       4,  20 Jun 14 12:30 /dev/mt20
brw-rw-rw-  1 root       4,   4 Jun 14 12:30 /dev/mt4
brw-------  1 root       4,   8 Jul  8 19:35 /dev/mt8
crw-rw-rw-  1 root       4,   0 Jun 14 12:31 /dev/rmt0
crw-------  1 root       4,  12 Aug  3 13:16 /dev/rmt12
crw-------  1 root       4,  16 Oct  4 14:47 /dev/rmt16
crw-------  1 dunbar     4,  20 Oct 16 17:46 /dev/rmt20
crw-rw-rw-  1 root       4,   4 Jun 14 12:31 /dev/rmt4
crw-------  1 root       4,   8 Oct 14 22:44 /dev/rmt8
```

The file ownership of the device file */dev/rmt20* has been changed to "dunbar"; therefore, dunbar can now read and write to this file. Because *TAPE* is linked to the device file, any command that causes I/O (e.g., *cp*) that is directed to *TAPE* will cause data to be read or written on the magnetic tape (the tape on the drive controlled by */dev/rmt20*).

1.5.2 Mounting for Labeled Access

If access to a labeled tape is desired, the mounting operation is somewhat different from that described above. In response to the *tpmount* command, *tpdaemon* creates a file linked to a labeled tape device (i.e., one of the */dev/lt/u<n>* device files). Though the labeled device file will not be shown as linked to any other file by an *ls -l*, I/O is directed through this file to one of the regular character (*/dev/rmt<n>*) device files (again, a device appropriate to the criteria specified in the *tpmount* is chosen).

Note	The devices used for labeled tape access must be character and no-rewind. Block mode or automatic rewind devices cannot be used.

Until the user unmounts the device (using *tpunmount*) no other user is allowed access to the labeled device file or to the regular device file (nor to the tape drive that it "talks" to). In addition, a special labeled tape daemon (*ansidaemon* for ANSI-standard labeled tapes) that is responsible for this I/O stream is forked by *tpdaemon*.

Input destined for the tape is first directed to the linked file in the user's directory, then to the other end of the link (the */dev/lt/u<n>* device file). From there, the ConvexOS kernel sends input to the *ansidaemon*; the *ansidaemon* handles all label-specific operations, then passes the input on to the regular device file. From there, input is sent to the tape drive, just as in the case of unlabeled tapes. Output from the tape is handled in the reverse sequence.

Chapter 2
Configuring the Tape System

This chapter contains information for system administrators on how to configure the tape system. The following topics are discussed:

- Steps that must be taken to prepare the ConvexOS tape system for use and to configure it to local requirements with the tape system configuration utility *tpconfig* are discussed in section 2.1, "Using the Tape System Configuration Utility: *tpconfig*."
- Special configuration issues concerning the Operator Request Management utility (*opreq*) are covered in section 2.3, "Administration of *opreq* Queueing."
- Tape system error message logging is discussed in section 2.4, "Error Message Logging."

2.1 Using the Tape System Configuration Utility: *tpconfig*

The *tpconfig* utility is used to set many of the parameters that govern the functioning of the tape system. *tpconfig* modifies a database of ConvexOS tape system configuration parameters that is contained in the file */usr/lib/tape/config.db*. With the exception of *opreq* queueing status, all changes made via *tpconfig* take effect within a few seconds after you leave *tpconfig*. If the queueing status is changed, the change does not go into effect until there are no active *tpmount* requests. (*tpmount* requests are active until canceled by operator intervention, by the system, or by a corresponding *tpunmount* command.)

Only one *tpconfig* can be used on a system at any one time; the tape system database is locked while *tpconfig* is active. If a second instance of *tpconfig* is started, its access to the configuration database will be read-only.

You must be *root* to change the tape system configuration; if you are not *root*, *tpconfig* will run, but will not make any changes to the tape system configuration file.

CAUTION	Do not attempt to alter the */usr/lib/tape/config.db* file with any text editor; configuration parameters can only be changed with *tpconfig*.

2.1.1 Configurable Parameters

tpconfig has three main uses:

- To set parameters that modify the functioning of the entire tape system. These include:
  - Enable/disable *opreq* queueing—if queueing is enabled, mount/dismount requests will be passed to *opreq*; if it is disabled, such requests will be handled directly by *tpdaemon*.
  - Define label types—currently only the ANSI label type is supported.

  - Set defaults for *tpmount*—the default drive, density, flags, and speed can be set for this command.

- To define characteristics of tape drives. Definable characteristics for each drive include:
  - Drive type—an arbitrary string that identifies the type of logical drive.
  - Control Status—if the logical drive is under tape system control, any device defined as belonging to it can only be accessed after a *tpmount* request has been granted. (Drives will normally be defined as controlled.)
  - Timeout—number of minutes that are allowed to elapse before an idle drive is assumed to be unused; after the drive is timed-out, it is automatically unmounted.
  - Bypass labeling permission—only users that are granted this permission for a particular drive can bypass label verification; such users can access labeled tapes on the drive in unlabeled mode, or mount labeled tapes to which they would not otherwise have access.
  - Allocate drive permission—only users that are granted this permission are allowed to mount any device of this drive in labeled mode (through a *tpmount* request) without also specifying a VSN.
- To define characteristics of nodes (nodes are devices associated with one of the drives defined with *tpconfig*). The following parameters can be set individually for each node associated with a drive:
  - Speed—tape speed of the drive when accessed through this node.
  - Density—tape density of the drive when accessed through this node.
  - Rewind status—whether or not tapes accessed through this device file will automatically be rewound when they are closed.

2.1.2 How to Use *tpconfig*

tpconfig can be used interactively, as a single command, or in "batch" mode. When used interactively, the ConvexOS shell prompt is replaced by the *tpconfig* prompt, and the utility will accept any of the *tpconfig* commands described in this chapter until you exit with CTRL-D. When used as a single command, a *tpconfig* command is entered as a parameter; when the command has been completed, the shell prompt returns. In batch mode, a file containing *tpconfig* commands is fed to the utility. Each method of using *tpconfig* is described in more detail later in this chapter.

2.1.2.1 *tpconfig* Command Format Notation

tpconfig commands are not case sensitive. Either the complete command or a legal abbreviation can be entered. The notation for *tpconfig* commands used in this publication is as follows:

- The shortest unambiguous abbreviation for command keywords and parameters is shown in uppercase, although all letters in keywords can be entered as either upper or lowercase.
- Variables for which an actual value must be substituted when the command is entered are shown in angle brackets and italics (i.e. *<value>*).
- If only one of a set of parameters can be entered, possible choices are separated by "or" bars (|).

- Optional parameters or groups of parameters are enclosed in [square brackets].

For example, the format of the *set control* command is shown as:

SEt Control ON|OFf *<type:unit>*

This means that the command can be entered in any of the following ways:

```
set control off mt:4
set c ON mt:4
set cont OFF mt:4
SET CONTROL ON mt:4
Set Control OFF mt:4
SE C OFF mt:4
se c on mt:4
sE C OF mt:4
```

(This list does not, of course, exhaust all possible permutations.)

When a command is discussed in text, the unabbreviated lowercase form of the command keywords is used, and is shown in italic type.

2.1.2.2 Using *tpconfig* Interactively

To use *tpconfig* interactively, type "tpconfig", followed by a carriage return. The shell prompt will then be replaced by the *tpconfig* prompt: `Tpconfig>`. You may enter any of the *tpconfig* commands explained in section 2.2, "Using *tpconfig* Commands" after this prompt. When the prompt reappears, you may enter additional commands. To leave *tpconfig*, type CTRL-D.

Example 2.1 shows how *tpconfig* can be used interactively. The example also shows how to obtain information about the configuration of the tape system with the *show all* command.

Example 2.1.

First, *tpconfig* is invoked in interactive mode by entering `tpconfig` with no parameters at the ConvexOS shell prompt:

```
# tpconfig
Tpconfig>
```

The Tpconfig> prompt means that the *tpconfig* utility is active and awaiting commands. The *show all* command will display information about the tape system configuration as it is currently defined:

```
Tpconfig> show all
Queueing is: Disabled

Drives:
    mt:0        timeout: 60   Controlled
        Alloc access:
            Users:    All
            Groups:   All
        Bypass access:
            Users:    smith jones
            Groups:   engr
        Nodes:
        /dev/rmt8      speed:       density: 1600  Rewind     Char
        /dev/rmt12     speed:       density: 1600  No rewind  Char
        /dev/rmt16     speed:       density: 6250  Rewind     Char
        /dev/rmt20     speed:       density: 6250  No rewind  Char
        /dev/mt8       speed:       density: 1600  Rewind     Block
        /dev/mt12      speed:       density: 1600  No rewind  Block
        /dev/mt16      speed:       density: 6250  Rewind     Block
        /dev/mt20      speed:       density: 6250  No rewind  Block

Labels:
    ansi daemon: /usr/lib/tape/ansidaemon

Defaults:
    Density: 6250
    Speed: No default
    Drive type: mt
    Mount flags: Character special, No-rewind
Tpconfig>
```

The *show all* command reveals the following information about the tape system:

- **Queueing is:** shows the status of *opreq* queueing for the tape system (in this case, *opreq* queueing is disabled).
- **Drives:** shows the drives that are defined in the *tpconfig* database, along with their characteristics and nodes (associated devices). Drives are defined with the *add drive* command. In the example, one drive—mt:0—has been defined. The following information is shown for this drive:
  - **timeout:** number of minutes that are allowed to elapse before an idle drive is assumed to be unused; after this interval elapses, the drive is unmounted by the system. Timeout can be set with *set timeout.* In the example, drive *mt:0* has a 60-minute timeout.
  - **Controlled** indicates that the control status of *mt:0* is "controlled" (i.e., it is controlled by the tape system). If the drive were uncontrolled, the control status would be shown as "Not controlled." The *set control* command is used to change the control status of drives.
  - **Alloc access:** the users and groups shown below this label are allowed to request mounting a tape without specifying a VSN; users who do not have this permission or who are not members of a group that have this permission must specify a VSN. In the example, all users and groups can

mount tapes without specifying a VSN. The commands for changing drive allocation permissions are described in section 2.2.4, "Set Drive Allocation Permissions."

— **Bypass access:** the users and groups shown below this label are allowed to bypass labeled-tape access restrictions. The commands that are used to change these permissions are described in section 2.2.5, "Set Label Bypass Permissions." In the example, users smith and jones and the engr group have bypass access permission.

— **Nodes:** lists the devices that have been defined to be associated with the drive. Each of these devices is called a "node" of the drive. In the example, drive *mt:0* has eight nodes: *rmt8, rmt12, rmt16, rmt20, mt8, mt12, mt16,* and *mt20.*

The speed, density, rewind status, and mode are shown for each node. In the example, *rmt8* has the following characteristics: no defined speed, a density of 1600 bpi, will automatically rewind on close, and is a character device.

- **Labels:** the daemons for defined label types are shown. Only one label type is currently supported by the tape system: ANSI standard labels.
- **Defaults:** shows the defaults for the tape mounting request (*tpmount*). (This command is described in section 3.1, "*tpmount* and Related Commands.") The default values are used to determine what device will be mounted unless they are overridden by the user when the *tpmount* command is given. The following defaults can be defined:
  - **Density:** default tape density
  - **Speed:** default tape speed
  - **Drive type:** default drive type
  - **Mount flags:** default device mode and rewind status

In addition to *show all*, two other *tpconfig* commands are available to get configuration information: *show drive* shows information about a particular drive (e.g., `show drive mt:0` would show information about the specified drive), and *show defaults* shows only the information under the "Default" heading.

2.1.2.3 Using *tpconfig* as a Single Command

To use it as a single command, enter "tpconfig" at the ConvexOS shell prompt, followed by one of the commands recognized by *tpconfig*. (The commands are described in section 2.2, "Using *tpconfig* Commands and on the *tpconfig*(8) page in the *ConvexOS Programmer's Reference.*) For example:

```
# tpconfig add drive mt:5
```

This command adds a logical drive called "mt:5"; the shell prompt then returns.

2.1.2.4 Using *tpconfig* in Batch Mode

tpconfig may also be used in "batch" mode. To do so, invoke the utility from the shell prompt and direct an input file containing *tpconfig* commands to it:

```
# tpconfig < tpcommands
```

This option is particularly useful in conjunction with the *snapshot* command (described in section 2.2.9.6, "Snapshot."). *snapshot* can be used to obtain an output file of *tpconfig* commands that can then be edited and "fed" to *tpconfig* (via a command like the one above) to produce the desired tape system configuration.

2.2 *tpconfig* Commands

The *tpconfig* commands described in the following sub-sections can be used to change the configuration of the tape system. Use the *show all*, *show drive*, or *show defaults* commands to see the current tape system configuration, and to verify the changes you have made.

2.2.1 Get Help—List *tpconfig* Commands

As shown in Example 2.2, , the *help* command lists the *tpconfig* commands.

Example 2.2.

```
Tpconfig> help
Add Drive <type:unit> [Nocontrol] [Timeout=<N>]
Del Drive <type:unit>
Add Node <path> [Speed=<string>] [Density=<string>] [Rewind] <type:unit>
Del Node <path>
Add Label <label_type> <daemon>
Del Label <label_type>
Add/Del/Set Alloc_drive User_set <user_list...> <type_unit>
Add/Del/Set Alloc_drive Group_set <user_list...> <type_unit>
Add/Del/Set Bypass_labels User_set <user_list...> <type_unit>
Add/Del/Set Bypass_labels Group_set <user_list...> <type_unit>
SEt Control ON|OFf <type:unit>
SEt Default Drive <type>
SEt Default Density <density>
SEt No_default Density
SEt Default Flags [Rewind|Norewind] [Character|Block|Labeled]
SEt Default Speed <speed>
SEt No_default Speed
SEt Timeout <N> <type:unit>
SEt Queueing Enabled|Disabled
SHow All
SHow DEfaults
SHow DRive [<type:unit>]
SHow Labels
SHow Node [<path>]
SNapshot [<file>]
Tpconfig>
```

2.2.2 Turn *opreq* Queueing On or Off

If queueing is turned on, all *tpmount* or *tpunmount* requests will be directed to *opreq*; if it is turned off, the tape system will attempt to handle tape mount or dismount requests directly. (See section 2.3, "Administration of *opreq* Queueing," section 3.1.3, "Completion of *tpmount* Requests," and Chapter 4, "Operator Request Management" for more information about queueing.)

Queueing can be enabled or disabled with the *tpconfig set queueing* command. The format of this command is:

SEt Queueing Enabled|Disabled

> **Note**
>
> A change in the status of *opreq* queueing will not take effect until there are no active *tpmount* requests on the system. (*tpmount* requests are active until they are canceled either by operator intervention, by the system, or by a corresponding *tpunmount* command.) The *tpqueue* command can be used to show active *tpmount* requests.

2.2.3 Define and Delete Drives and Device Nodes

As stated in section 1.4, "Tape System Device Files," one or more device files must have been created for each tape drive installed in the system, and each device file has certain attributes (e.g., tape density, access mode, rewind/norewind). Usually, more than one device file will have been created for each drive, so that varying tape attributes can be invoked by linking to the appropriate file (with *tpmount*).

Before they can be controlled or used (i.e. mounted) by the tape system, devices and drives must be defined via *tpconfig*. When a drive is defined, it can be given certain characteristics (such as control status and time-out duration) that will apply whenever that drive is mounted. The drive definition is also called a "logical drive". When devices are defined, they are associated with logical drives. Such devices are called "nodes" of the drive.

These two steps must be done in order—the logical drive must be created first (with the *add drive* command), followed by the node (defined with *add node*).

The logical drive and node attributes are used by *tpdaemon* to assure tape system security and access to resources. When *tpdaemon* starts, it examines the tape system configuration database file. It then opens one device for every controlled logical drive. As a consequence, the following restrictions apply whenever *tpdaemon* is running:

- No mechanical drive that has been defined as a controlled logical drive and for which nodes have been defined can be accessed by anyone except by using *tpmount*. (I/O cannot be performed directly to a device associated with a controlled drive, since the tape driver will refuse to access a drive if any device that "talks" to it is open, and *tpdaemon* always opens one device for every controlled drive at startup.)
- *tpdaemon* will not permit access via *tpmount* to any device that has not been defined as a node by *tpconfig*.
- A drive defined as uncontrolled can be accessed by performing I/O directly to one of its devices (providing you have the appropriate access permissions to the device file), but cannot be accessed via *tpmount*.
- Similarly, drives that have not been defined via *tpconfig* cannot be accessed with *tpmount*. Direct I/O can be performed to such drives if the user has ConvexOS access privileges to its devices. (Usually, only *root* will have such privileges.)

CAUTION	The logical drives and nodes must parallel the mechanical drives and device files actually present on the system. All drives should be defined. Failure to do this may result in making devices and/or drives inaccessible to tape system users, or in making devices and/or drives vulnerable to unauthorized use.

2.2.3.1 Add Drive

Logical drives are defined with the *add drive* command. The control/no control status, timeout duration, label bypass permissions, and allocation permissions established for the logical drive will apply to all nodes of that drive. The following is the format of the command:

Add Drive <*type:unit*> [**Nocontrol**] [**Timeout=**<*N*>]

Parameters:

<*type:unit*>
This is the name of the logical drive, and consists of a string followed by a colon and an integer. The string is intended to be a mnemonic for the type of drive (e.g. "mt" for "magnetic tape"); the integer is intended to be a unit number for the drive (for example, "mt:4"). However, the system administrator may assign any desired name or number.

Nocontrol
If this optional keyword is included in the command, the logical drive is not under the control of the tape system. Nodes defined to be part of this logical drive cannot be mounted with *tpmount*; they can, however, be directly accessed (e.g., with *cp* or *cat* commands) by any user who has the requisite file permissions.

Timeout=<*N*>
This parameter is used to specify the number of minutes that any drive controlled by a node of the logical drive can be left unused before the tape system automatically unmounts the device file (as though a *tpunmount* had been done by the user). If a drive is timed out, the tape system notifies the user via email. The default timeout is 60 minutes.

Refer to the end of section 2.2.3.2, "Add Node" (immediately below) for an example of creating a drive and nodes.

2.2.3.2 Add Node

In order to use a tape drive, it is not enough to define it with *add drive*. Since access to drives is through device files, such device files must be defined as "nodes" of a drive before that drive can be used. This is done with the *add node* command. Parameters of this command allow specifying several attributes of the node (speed, tape density, and rewind/no-rewind status); these parameters are used as criteria in selecting an appropriate device file when a *tpmount* command is given. (However, all nodes added to a drive share the control/no-control and timeout attributes of the drive as well as the label bypass and allocation permissions.)

CAUTION	If no nodes are defined for a drive, then that drive will be uncontrolled, even though it may have been defined as controlled.

The actual characteristics (including the major and minor device numbers) of each device were fixed when it was created (via MAKEDEV or *mknod*). The *add node* command merely notifies the tape system of the device's characteristics so that *tpmount* can find a device appropriate for the characteristics requested by the user.

The following is the format of the command:

Add Node <*path*> [**Speed**=<*speed*>] [**Density**=<*density*>] [**Rewind**] <*type:unit*>

Parameters:

<*path*>
The absolute pathname of the device file that is being defined as a node (e.g., */dev/rmt8*).

[Speed=<*speed*>]
A string specifying speed of the device.

[Density=<*density*>]
A string specifying density of the device.

[Rewind]
An optional keyword specifying whether the device automatically rewinds tapes when the file is closed. (Closing is done by the *close* system call; this call is made by such standard file transfer utilities as *cp* and *cat* when they have finished writing data. Refer to the *close*(2) page in the *ConvexOS Programmer's Reference* for more information.) If *rewind* is not specified, the device is norewind.

<*type:unit*>
drive with which the node is associated. This must be the type and unit number of a logical tape drive that was previously defined with *add drive*.

Refer to the *mtio*(4) page in the *ConvexOS Programmer's Reference* for information about allowable tape densities. (Node parameters are hardware dependent, so the actual values you use may vary depending on your drive type.)

The tape system examines the device to see whether it is a block or character device when *add node* is used; thus, the device type is not specified with *add node*.

Example 2.3

The following example illustrates the definition of a new drive and nodes:

```
# tpconfig
Tpconfig> add drive mt:1
Tpconfig> add node /dev/rmt9 den=1600 rew mt:1
Tpconfig> add node /dev/rmt13 den=1600 mt:1
Tpconfig> add node /dev/rmt17 den=6250 rew mt:1
Tpconfig> add node /dev/rmt21 den=6250 mt:1
Tpconfig> add node /dev/mt9 den=1600 rew mt:1
Tpconfig> add node /dev/mt13 den=1600 mt:1
Tpconfig> add node /dev/mt17 den=6250 rew mt:1
Tpconfig> add node /dev/mt21 den=6250 mt:1

Tpconfig> show drive mt:1
    mt:1        timeout: 60   Controlled
        Alloc access:
            Users:    None
            Groups:   None
        Bypass access:
            Users:    None
            Groups:   None
        Nodes:
        /dev/rmt9      speed:       density: 1600  Rewind     Char
        /dev/rmt13     speed:       density: 1600  No rewind  Char
        /dev/rmt17     speed:       density: 6250  Rewind     Char
        /dev/rmt21     speed:       density: 6250  No rewind  Char
        /dev/mt9       speed:       density: 1600  Rewind     Block
        /dev/mt13      speed:       density: 1600  No rewind  Block
        /dev/mt17      speed:       density: 6250  Rewind     Block
        /dev/mt21      speed:       density: 6250  No rewind  Block
Tpconfig> ^D
#
```

2.2.3.3 Delete Node

Once they are defined, nodes may be removed from a logical drive with the *delete node* command. The following is the format of the command:

Delete Node *<path>*

Parameter:

<path>
The pathname of the device file.

The following example deletes a node of drive mt:1:

```
Tpconfig> delete node /dev/rmt9
```

2.2.3.4 Delete Drive

Logical drives may be deleted with the *delete drive* command. (If nodes are still defined for the logical drive being deleted, no warning will be given.)

The following is the format of the command:

Delete Drive *<type:unit>*

Parameters:

<type:unit>
The type and unit number of the logical drive being deleted.

The following example deletes drive mt:1:

```
Tpconfig> d d mt:1
```

2.2.3.5 Reset Timeout Parameter of Drives

The timeout parameter is set when the logical drive is defined (with *add drive*). The *set timeout* command can be used to change this parameter once the logical drive has been defined. The following is the format of this command:

SEt Timeout *<N>* *<type:unit>*

Parameters:

<N>
Number of minutes for which the drive can be unused before it is timed out (i.e. unmounted).

<type:unit>
The type and unit number of the logical drive.

The following example sets a timeout of 5 minutes for mt:1:

```
Tpconfig> set timeout 5 mt:1
```

2.2.3.6 Reset Control Status of Drives

The control status of a drive (i.e. whether or not it is controlled by the tape system) is set when the logical drive is defined (with *add drive*). The *set control|nocontrol* command can be used to change this parameter.

This may be useful if it is necessary to quickly switch a drive from controlled to uncontrolled status, or vice versa. (For example, if the system administrator needs exclusive access to a drive for backups, he or she can employ this command to make the drive inaccessible to users. The administrator can then use backup utilities that do not invoke *tpmount*.)

The following is the format of this command:

SEt Control ON|OFf *<type>:<unit>*

Parameters:

ON|OFf
To place the logical drive under tape system control, use *on*; to free the logical drive from tape system control, use *off*.

<type:unit>
The type and unit number of the logical drive.

The following example places mt:1 under tape system control:

```
Tpconfig> set control on mt:1
```

2.2.4 Set Drive Allocation Permissions

With *tpconfig*, the tape system administrator can give certain users or groups permission to mount drives without also specifying a tape VSN. Users who have this permission can use a drive for multiple tapes without entering a *tpmount* each time.

The following *tpconfig* commands can be used for modifying the drive allocation permissions:

- The *add* ... commands add existing users or groups to the list having permission.
- The *set*... commands *replace* the old list of users or groups with the new one.
- The *delete*... commands revoke permission for specified users or groups.

All these commands have two parameters: either *<group_list>* or *<user_list>*, and *<type:unit>*. Because all parameters are similar, they will not be described individually for each command.

<group_list> is a list of either group identifiers (*gid*) or group names. *<user_list>* is a list of either user identifiers (*uid*) or user names. Surround *uid*s and *gid*s with square brackets (for example, [989]); separate items in a list with spaces. If the special asterisk (*) character is used in place of a list, then it denotes all users or groups.

<type:unit> is the name of a logical drive.

If a *uid* or *gid* (i.e., a numeric value) is entered, the corresponding user name or group name (from the *etc/passwd* or */etc/group* file) will be shown when the list of users or groups is displayed in response to any of the *tpconfig show* commands. If a nonexistent *uid* or *gid* is entered, it will be accepted without complaint, and only the entered *uid* or *gid* will be displayed in response to *show*. However, if a user name or group name that is not contained in the */etc/passwd* or */etc/group* files is entered, it will not be added (and an error message will be displayed by *tpconfig*).

Any user can issue a *tpmount* request that includes a specific tape volume serial number (VSN). The tape system will not actually mount the device (i.e., establish the logical links between the appropriate device file and the user) until the requested tape is on the drive or an operator completes the mounting. However, users can only mount the device without specifying a VSN if they have been given permission to do so. Such a request (if successful) has the effect of reserving the tape drive for the exclusive use of the requester until either the requester issues a *tpunmount* command, or the device is unmounted due to a timeout.

> **Note** "VSN" need not necessarily refer to the magnetic VSN encoded on an ANSI labeled tape; it may refer to an identifier recorded on the tape reel, or the storage location of the tape.

The lists for individual users and for groups are separate. For example, it is possible to change or wipe out the group list and leave the individual user list unaltered.

2.2.4.1 Add Groups to Drive Allocation List

To add groups to the list of groups permitted to mount the drive without specifying a VSN, use this command:

Add Allocate_drive Group_set *<group_list>* *<type:unit>*

The following example gives the *doc, engr,* and *sysad* groups permissions to allocate (mount) mt:1 without specifying a VSN:

```
Tpconfig> add a g doc engr sysad mt:1
```

2.2.4.2 Add Users to Drive Allocation List

To add users to the list of users permitted to mount a drive without requesting a specific tape by VSN, use this command:

Add Allocate_drive User_set *<user_list>* *<type:unit>*

The following example gives users *jones, smith,* and *abruzzo* permission, (even if they are not members of the groups added above):

```
Tpconfig> add a u jones smith abruzzo mt:1
```

2.2.4.3 Replace Group Drive Allocation List

To replace the existing list of groups with a new one, use the following command:

SEt Allocate_drive Group_set *<group_list> <type:unit>*

The previous list is deleted and replaced by the new list. *<group_list>* may be a list of group IDs separated by spaces, or it may be an asterisk (*) to signify "all". For example, the following command would allow all groups to mount drive mt:0 without specifying a VSN:

```
Tpconfig> set alloc group * mt:0
```

2.2.4.4 Replace User Drive Allocation List

To replace the existing list of users with a new one, use the following command:

SEt Allocate_drive User_set *<user_list> <type:unit>*

The previous list is deleted and replaced by the new list. *<user_list>* may be a list of user IDs separated by spaces, or it may be an asterisk (*) to signify "all". For example, the following command would allow all all users to mount drive mt:0 without specifying a VSN:

```
Tpconfig> set alloc user * mt:0
```

2.2.4.5 Delete Groups from Drive Allocation List

To delete groups from the list of groups allowed to allocate (mount) drives without specifying a VSN, use this command:

Delete Allocate_drive Group_set *<group_list> <type:unit>*

<group_list> may be a list of group IDs separated by spaces, or it may be an asterisk (*) to signify "all". For example, the following command would allow no groups to mount drive mt:0 without specifying a VSN:

```
Tpconfig> del alloc group * mt:0
```

The following example deletes groups *engr* and *sysadmin* from the list of groups having allocation permission for *mt:1*:

```
Tpconfig> delete a g engr sysadmin mt:1
```

2.2.4.6 Delete Users from Drive Allocation List

To delete users from the list of those allowed to allocate (mount) drives without specifying a VSN, use this command:

Delete Allocate_drive User_set *<user_list> <type:unit> <user_list>* may be a list of group IDs separated by spaces, or it may be an asterisk (*) to signify "all". For example, the following command would allow no users to mount a drive without specifying a VSN:

```
Tpconfig> delete alloc user * mt:0
```

The following example deletes user *smith* from the list of users having allocation permission for *mt:1*:

```
Tpconfig> delete a u smith mt:1
```

2.2.5 Set Label Bypass Permissions

tpconfig can be employed to give users or groups permission to bypass access restrictions on labeled tapes and to disable the record processing and buffering that is normally performed by the label daemon. Those who are granted this permission can use the *-b* option of *tpmount* to override access restrictions on labeled tapes, and they can mount labeled tapes in unlabeled mode. (Of course, all users can mount *unlabeled* tapes in unlabeled mode.)

When a labeled tape is mounted with *-b*, all access restrictions that may be imposed by the label are bypassed. For example, a user who is allowed to use the *-b* option could read or write a labeled tape or files on that tape even though access privileges were restricted by the tape owner. If *-b* is specified in a *tpmount* request, the tape may be mounted either in labeled or unlabeled mode.

As long as the tape is mounted in labeled mode (e.g., by using the *-m label* option in addition to *-b*), normal labeled tape processing operations (e.g. the logical record handling and buffering normally performed by *ansidaemon* and described in section 3.2.3, "Setting File Attributes on Labeled Tapes: *tpattr*,") will still be done, though access restrictions will be ignored.

Users with label bypass permission can also mount labeled tapes in unlabeled mode by specifying *-b* (and, e.g. *-m char*). In this case, the tape will be accessed without the intervention of *ansidaemon*. Everything written on the tape—including the labels—is then accessible as raw data, and none of the record handling or buffering functions of *ansidaemon* will be performed.

Note	Users logged on as *root* can always bypass access restrictions and labeled tape processing by specifying *-b*. (Even *root* must specify the *-b* option of *tpmount* to mount a labeled tape in unlabeled mode.)

CAUTION	Be careful when assigning label bypass permissions, since it permits the user to ignore any access restrictions imposed by the labels on a tape. This could result in the accidental or deliberate destruction or alteration of data.

The label bypass permission commands work like the drive allocation permission commands described in the preceding section. Again, the parameters are very similar and are the same as described in section 2.2.4, "Set Drive Allocation Permissions," so they will not be described individually. The examples in that section also illustrate how the label bypass permission commands work. Once again, the asterisk (*) character in place of a user or group list means "all."

2.2.5.1 Add Groups to Label Bypass List

To add groups to the list of groups permitted to bypass label processing, use this command:

Add Bypass_labels Group_set *<group_list> <type:unit>*

2.2.5.2 Add Users to Label Bypass List

To add users to the list of users permitted to bypass label processing, use this command:

Add Bypass_labels User_set *<user_list> <type:unit>*

2.2.5.3 Replace Group Label Bypass List

This command substitutes the specified list of groups for the existing list. The format is:

SEt Bypass_labels Group_set *<group_list> <type:unit>*

The previous list is deleted and replaced by the new list.

2.2.5.4 Replace User Label Bypass List

To replace the existing list of users permitted to bypass label processing, use this command:

SEt Bypass_labels User_set *<user_list> <type:unit>*

The previous list is deleted and replaced by the new list.

2.2.5.5 Delete Groups from Label Bypass List

To delete groups from the list of those allowed to bypass label processing, use this command:

Delete Bypass_labels Group_set *<group_list> <type:unit>*

2.2.5.6 Delete Users from Label Bypass List

To delete users from the list of those allowed to bypass label processing, use this command:

Delete Bypass_labels User_set *<user_list> <type:unit>*

2.2.6 Defining and Deleting Label Types

At present, the ConvexOS tape system supports only one tape label type: ANSI-standard labels. To provide for future support of multiple label types, the commands *add label* and *delete label* have been provided.

The following are the formats of *add label* and *delete label*:

Add Label *<type> <path>*

Delete Label *<type>*

Parameters:

<type>
A string that denotes the label type being deleted.

<path>
The pathname of the file containing the executable code of the daemon for this label.

2.2.7 Setting Defaults for *tpmount*

The *set default* commands can be used to set defaults for *tpmount*. These defaults will be used whenever *tpmount* is invoked, unless they are explicitly overridden. Refer to section 3.1, "*tpmount* and Related Commands" for information about this command. The *set default* commands are described below.

2.2.7.1 Set Default Drive Type

This command sets the default drive type (not a specific drive) that will be used for any *tpmount* command that does not specify a drive. This is the format of the command:

SEt Default DRive *<type>*

Parameter:

<type>
The type of a drive is the portion of its name before the colon (as defined with *add drive*). For example, logical drives *mt:1* and *mt:2* both have the type *mt*.

The following command will cause any one of the drives that was defined as type "mt" to be used as default for *tpmount* commands:

```
Tpconfig> set default drive mt
```

2.2.7.2 Set Default Density

This command sets the default tape density that will be used for any *tpmount* command that does not specify a density. This is the format of the command:

SEt Default DEnsity *<density>*

Parameter:

<density>
Tape density to be used as default

For example, the following command will cause a density of 1600 bpi to be used as default:

```
Tpconfig> set default density 1600
```

2.2.7.3 Set Default Flags

This command specifies the default rewind/norewind status and the default access mode that will be used for *tpmount* if these values are not specified with *tpmount*.

If *rewind* is specified, the tape is, by default, automatically rewound when the file is closed; if *norewind* is specified as a default flag, the tape is not rewound. The rewind status default can be overridden by specifying *-r* or *+r* with *tpmount*. (Closing is done by the *close* system call; this call is made by such standard file transfer utilities as *cp* and *cat* when they have finished writing data. Refer to the *close*(2) page in the *ConvexOS Programmer's Reference* for more information.)

The access mode default can be set to character, block, or labeled access. The default is used unless overridden with the *-m* flag of *tpmount*.

The format of the command is:

SEt Default Flags [Rewind|Norewind] [Character|Block|Labeled]

Parameters:

Rewind|Norewind
If this parameter is specified, either "rewind" or "norewind" is made the default.

[Character|Block|Labeled]
Whichever one of these is specified becomes the default access mode for *tpmount*.

For example, the following command sets automatic rewind and labeled access as defaults:

```
Tpconfig> se de f rewind labeled
```

2.2.7.4 Set Default Speed

This command specifies the default tape speed. The speed should be given as the speed of one (or more) of the nodes that has been defined with an *add node* command. The default speed can be overridden with the *-i* option of *tpmount*. This is the format of the command:

SEt Default Speed *<speed>*

Parameter:

<speed>
The tape speed, as specified for one or more nodes with the *add node* command.

The following example specifies a default speed of 50 ips. A device defined (via *add node*) as having a speed of 50 ips will be used for *tpmount* commands that do not specify a speed.

```
Tpconfig> set default speed 50
```

2.2.8 Eliminating Default Density and Speed

If there is no default speed or density, and if no speed or density is specified with *tpmount*, speed or density are not criteria the tape system will use in selecting a device. Under these circumstances, a device of *any* speed or density may be selected by the tape system in response to *tpmount*. To permit this, it may be necessary to undo the previous selection of a speed or density default. The following two commands can be used for this.

2.2.8.1 Set Default Speed to None

To eliminate any previously set speed default, use this command:

SEt No_default Speed

2.2.8.2 Set Default Density to None

To eliminate any previously set default density, use this command:

SEt No_default Density

2.2.9 Getting Tape System Configuration Information

The *show* commands can be used to obtain information about the tape system. In addition, the *snapshot* command produces an output file of *tpconfig* commands that reflects the current configuration. This output file can then be edited (with any text editor) and "fed" to *tpconfig* to produce an altered configuration, or it can be used in unedited form to restore the configuration to a previous state.

2.2.9.1 Show All

Displays all defined entities (logical drives and nodes along with their associated characteristics, labels, defaults, queueing status of the tape system, etc.) The format is:

Show All

An example illustrating *show all* can be found in section 2.1.2.2, "Using *tpconfig* Interactively."

2.2.9.2 Show Defaults

Shows the currently defined defaults and queueing status for the tape system. The format is:

Show DEfaults

2.2.9.3 Show Drive

Shows the characteristics of the specified logical drive. The format is:

Show DRive [<*type:unit*>]

Parameter:

[<*type:unit*>]
Identifier of the logical drive to be shown; if none is specified, information for all defined logical drives will be shown.

2.2.9.4 Show Labels

Shows all currently defined tape label types. The format is:

Show Labels

2.2.9.5 Show Node

Shows information about the specified node. The format is:

Show Node [*<path>*]

Parameter:

<path>
Pathname of the device file of the node; if none is specified, information about all defined nodes will be shown.

2.2.9.6 Snapshot

In addition to the *show* commands for displaying information about the current tape system configuration, the *snapshot* command can be used to produce an output file of *tpconfig* commands. If this file is subsequently directed into *tpconfig*, these commands will produce the tape system configuration that was in effect when the *snapshot* command was given. This file can be used to save the current configuration so that it can be restored at a later time. In addition, the output file can be edited (using a standard text editor) to produce the desired configuration. (The *tpconfig* database itself cannot be directly edited.) The format of the command is:

Snapshot [*<file>*]

If no output file is given, it will be sent to *stdout*.

To use the output file of *snapshot* (either unchanged or edited) as input to *tpconfig*, enter the following at the shell prompt:

```
% tpconfig < <file>
```

2.2.10 Initial Defaults

The ConvexOS tape system is shipped with the following default configuration:

- Queuing is disabled
- No users or groups are allowed to bypass label processing
- All users can give *tpmount* command without specifying a VSN
- Default density: 6250 (for *tpmount*)
- No default speed (for *tpmount*)
- One controlled drive (mt:0) is defined, along with associated nodes.
- No-rewind character special device is default for *tpmount*.

2.3 Administration of *opreq* Queueing

The ConvexOS tape system can be configured to send requests to mount tape devices to an interactive Operator Request Manager (*opreq*). Instructions for *opreq* are given in Chapter 4, "Operator Request Management."

If *opreq* queueing is not enabled, *tpmount* requests are processed immediately by *tpdaemon*. If the requested device is available, the request is fulfilled; if the device is not available, it fails. (There is, however, an option (*-q*) of the *tpmount* command that causes *tpdaemon* itself to queue requests. If this option is used, any command that cannot be carried out immediately because a tape resource is lacking will not fail, but will be suspended until the resource is available. Refer to the *tpmount*(1) page of the *ConvexOS Programmer's Reference* and Chapter 3, "Using Tape System Resources," for more information about *-q*.)

If *opreq* queueing is enabled, requests to mount devices and to install specific tapes (*tpmount*) are passed on to *opreq*. Logical links entailed in mounting a device are then only made when the operator notifies *tpdaemon* (via *opreq*) that the tape is on the drive and should be mounted, or when the Automatic Volume Recognition (AVR) capability recognizes that the tape specified in the *tpmount* request has been placed on a drive.

2.3.1 Enabling *opreq* Queueing

To activate *opreq* queueing, the system administrator must enable this option with the *tpconfig* utility. This procedure is described fully in section 2.2.2, "Turning *opreq* Queueing On or Off.").

2.3.2 Operator Privileges

To start an instance of *opreq*, the operator invokes the */usr/convex/opreq* process at the ConvexOS prompt:

```
# /usr/convex/opreq
```

Some of the operations that can be performed through *opreq* (e.g., mounting and unmounting tape devices) require that *opreq* be running as *root*. If it is not desirable to grant *root* privileges to the operator, then the *op* command can be used to grant limited superuser privileges to the operator that will apply only to *opreq*. (Refer to the *op*(8) page in the *ConvexOS Programmer's Reference* for more information about *op*.)

2.3.3 *opreq* Terminal Characteristics

opreq configures its display to fit the characteristics of a wide variety of terminals and operator preferences. When *opreq* starts up, it does the following things to determine the characteristics of the operator's terminal:

- It finds out the terminal type by examining the TERM environment variable.
- Next, *opreq* must determine the characteristics of this terminal type as configured for the system. To do this, it examines the TERMCAP environment variable. The following possibilities exist:

— If the variable is empty, *opreq* uses the characteristics specified for the terminal type (obtained from TERM) in the */etc/termcap* file.

— If the TERMCAP variable contains the pathname of a file, *opreq* uses the terminal characteristics specified in that file.

— If the TERMCAP variable contains a list of terminal characteristics (instead of a file name), *opreq* uses the information from the variable that pertains to the terminal type.

To change the display width of the *opreq* window, change the number of columns indicated in */etc/termcap* file for the operator's terminal type, and change the number of terminal columns via *stty*. (Refer to the *stty*(1) page in the *ConvexOS Programmer's Reference* for more information about this command). The change will not take effect until *opreq* is restarted.

opreq uses standard VT100 graphic characters whenever possible, as well as the standard VT100 keyboard layout (including the cursor keys).

2.3.4 Changing *opreq* Window Defaults

The system administrator may change several of the *opreq* window default characteristics (including the labels in the title bar) by editing a file called */usr/lib/opreq/.opreqrc*. This file contains *opreq* window configuration and manipulation commands (e.g.,*window-open*, *configure-status*, *configure-type*, and *configure-title*). These commands are all described in section 4.2.3.5, "Configure the *opreq* Window," and 4.2.3.7, "Opening, Closing, and Moving Between Windows"; they are also listed in Appendix B, "Command Quick Reference." For example, the file may contain the following command:

```
configure-title uid status type vsn ring drive time-in comment
```

This command causes the listed labels to be displayed in the title bar of the *opreq* window when it opens. You may remove any of these titles if you don't want them displayed, or add others that you want shown.

Changing */usr/lib/opreq/.opreqrc* affects the defaults for all *opreq* users. Instead of doing this, users may customize their windows by copying this file to their current working directory or to their home directories and then editing it. When *opreq* starts, it will look first in the current directory and then in the user's home directory for an *.opreqrc* file; only if no such file is found will it use the file in */usr/lib/opreq/.opreqrc*.

Note	The *opreq* utility is generally run as *root*. If this is so, *opreq* will consider the home directory of the user to be the root directory (i.e., "/"). Thus, if a local *.opreqrc* file is to be used, *opreq* should be started from the directory in which that file is located.

2.3.5 Resetting *opreq*

If *opreq* will not start because of an error condition (e.g. as a result of the abnormal exit of a tape system process, or because of improper installation of some portions of the system), it

may be necessary to reset the *opreq* tape queueing system. To reset *opreq*, kill *opreq_daemon, and* use *tpconfig* to turn off queueing and quit *tpconfig*. Then remove all "share" files. (These are files located in */usr/lib/opreq* that have names beginning with "share..."—e.g., *share1*, *share2*, etc.). After this has been done, turn queueing on again, and restart *opreq*.

2.4 Error Message Logging

The tape system uses the ConvexOS system logging facility to record error and warning messages. By default, these messages are stored in a file called */usr/adm/log/tapelog*. The */etc/syslog.conf* file is used to configure system logging; the following line in that file causes tape system messages to be logged to */usr/adm/log/tapelog*:

```
tape.debug                              /usr/adm/log/tapelog
```

You may edit this line to change the log file or message level. (Refer to the *syslog*(3) and *syslogd*(8) pages in the *ConvexOS Programmer's Reference* for more information about ConvexOS system logging).

If your tape system is behaving abnormally, check the log file for error messages.

Chapter 3
Using Tape System Resources

The ConvexOS tape system provides three key capabilities for tape system users:

- **Mounting tape drives**—Commands that provide this capability are discussed in section 3.1, "*tpmount* and Related Commands":
  - *tpmount* (mount a tape drive)
  - *tpunmount* (unmount a drive)
  - *tpwait* (suspend execution of the shell until *tpmount* is completed)
  - *tpqueue* (examine tape queue).
- **Label-handling**—This includes writing ANSI-standard labels on tapes, removing them, and setting file attributes on labeled tapes. Commands that provide this capability are discussed in section 3.2, "Labeling Tapes and Setting File Attributes":
  - *tplabel* (label an unlabeled tape)
  - *tpunlabel* (remove a tape label and delete tape data)
  - *tpattr* (set file attributes on labeled tapes).
- **Tape manipulation**—Includes moving tape forward or backward on a record-by-record or file-by-file basis, rewinding the tape, etc. Commands that provide this capability are available under the *mt* utility, and are discussed in section 3.3, "Tape Manipulation Commands."

3.1 *tpmount* and Related Commands

Before data can be transferred to or from a tape, the user must establish communication with, and exclusive control of, the drive on which the tape is installed. Under the ConvexOS tape system, communication and control are granted in response to a *tpmount* command. (Events that occur when the system grants exclusive access to a tape drive are described in section 1.5, "Understanding the Tape Mounting Operation.") Thus, *tpmount* must be invoked before tape data can be read or written. When the ConvexOS tape system has carried out this command, a tape drive is reserved exclusively for the user who invoked *tpmount*.

> **Note**
>
> If *opreq* queueing is disabled, and you want to access a labeled tape, put the tape on the drive before entering *tpmount*; otherwise, *tpmount* may fail. (An error message will inform you that the tape drive is "not on-line or not ready.")

The following table shows the parameters that may be specified along with *tpmount*; all parameters are optional.

Table 3-1: *tpmount* Parameters

Parameter	Meaning
-a <*device_file*>	Link to the specified device file (one of the files in */dev*) to fulfill the mount request. This forces use of a particular tape drive (the one to which that device "talks"); unless the user has been granted permission to do otherwise, a Volume Serial Number (VSN) must be specified (with the *-s* option) if *-a* is given. (Permission to specify a device without requesting a particular tape can be granted to users or groups by the system administrator through *tpconfig*.) The *-a* option overrides any other drive selection parameters specified on the command line or in the selection criteria defaults set with *tpconfig*. (Refer to section 2.1, "Using the Tape System Configuration Utility: *tpconfig*" for information about *tpconfig*.) **NOTE** Only no-rewind devices can be used for labeled tapes.
-B	Place the *tpmount* request in the background. (Refer to section 3.1.3.3, "Executing *tpmount* in Background or Foreground" and section 3.1.3.4, "*tpwait*" for information about running this request in the background and about the related *tpwait* command.)
-b	Bypass label access restrictions and processing. This option permits mounting labeled tapes in labeled mode while bypassing access restrictions imposed by the tape labels. In addition, labeled tapes can also by mounted in unlabeled mode when the *-b* option is used. Unless this option is specified, a labeled tape cannot be mounted in block or character mode. **CAUTION** Exercise care in using the *-b* option; it will cause any access restrictions imposed by tape labels to be ignored. This could result in accidental or deliberate bypass of security restrictions and consequent alteration or destruction of data. The privilege of using the *-b* option can be restricted to certain users and groups with *tpconfig*. However, *root* can always use *-b*.
+b	Turn off label bypassing previously set with the *-b* option on the same command line; used with compound *tpmount* commands requesting simultaneous access to multiple tapes. (Refer to Section 3.1.2.2, "Examples of Compound *tpmount* Commands" for more information.)
-c <*comment*>	Comment; if *opreq* queueing is enabled, any comment entered after this flag will be sent along with the request to the *opreq* window.

Parameter	Meaning
-d *<density>*	Density; desired tape density.
-f *<n>*	Skip forward *<n>* files on the tape when it is first accessed; valid only for labeled tape.
-i *<speed>*	Speed; desired tape speed.
-l *<label_type>*	Label type; only ANSI labels are now supported and only "ansi" can be specified (this option need never be given).
-m *<mode>*	The access mode; may be *block*, *char*, or *label*. A default access mode can be specified with *tpconfig*; the *-m* option need be specified only if a mode other than the default is desired. **NOTE** Only no-rewind devices can be used for labeled tapes.
-q	Queue the request; if *tpmount* cannot be completed immediately, the request will be queued (i.e., will wait) until the needed resource is available. This sort of queueing is not the same as the queueing implemented with *opreq* (e.g., the *-q* flag does not require the presence of an operator); *-q* has no effect if *opreq* queueing is turned on.
+q	Turn off queueing set with the *-q* option previously on the same command line; used with compound *tpmount* commands requesting simultaneous access to multiple devices. (Refer to example 3.1.2.2, "Examples of Compound *tpmount* Commands" for more information.)
-R	Read-only access; (informs the operator that the write ring should be removed from the tape).
+R	Reverses *-R* option if it was used previously on the same command line; used with compound *tpmount* commands requesting simultaneous access to multiple devices.
-r	Automatic rewind on close; cannot be used with labeled tape.
+r	Specifies no rewind of tape; reverses *-r* option if it was used previously on the same command line.

Parameter	Meaning
-s <*name*> [VSN [,VSN]...]	Name of the symbolic link that will be created by *tpmount* and that will point to the tape device, as well as VSN (Volume Serial Number) of desired tape, or multiple VSNs. If VSNs are given, a symbolic link name must also be given. If more than one VSN is specified, these are taken to be the volumes of a single ANSI tape set. (Multiple VSNs must be separated by commas.) A VSN must be specified unless the user has been granted permission to omit it by the system administrator. (This is done with *tpconfig*. Refer to section 2.2.4, "Set Drive Allocation Permissions.") **NOTE** The VSN can be any printable character string; it need not be numeric. All alphabetic characters in VSNs will be recorded on the tape as upper case, and all VSNs specified in a *tpmount* will be converted to upper case. Magnetic VSNs (i.e. VSNs recorded on ANSI-labeled tapes) can be up to six characters long. VSNs longer than six characters can be specified in *tpmount* requests, but only the first six characters will actually be compared against the magnetic VSN. (Any additional characters specified as part of the VSN will be displayed in the *opreq* window, and may have special meaning for the operator. Examples would include location of the tape, or an identifier written on a paper label affixed to the tape.)
-t <*drive_type*>	Drive type; one of the drive types defined by the system administrator with *tpconfig*. (The type is the string before the colon of the drive name, e.g., if the drive name is mt:1, the drive type is "mt".)

3.1.1 How Devices are Selected

In response to *tpmount*, the tape system selects a suitable device based on the parameters specified in the *tpmount* and on any defaults that are in force. The following criteria are used for selecting the device:

- If a device name is specified in *tpmount* (via the *-a*) parameter, that device is used if it is available. If it is not available, the request fails, unless the request is being queued (either by *opreq* or the *-q* option of *tpmount*).
- If no device name is given, the tape system selects a device based on the criteria of speed, density, rewind/no rewind status, and access mode. Any or all of these four criteria can be specified on the *tpmount* command line (using the *-i, -d, -r,* or *-m* options). The default values defined in *tpconfig* are used for any of these parameters not specified on the command line. The tape system selects any available device that has characteristics (as defined with *tpconfig*) that fit the criteria. If no devices fitting the criteria exist, or if all such devices are already in use (i.e. mounted), the request fails, unless the request is being queued (either by *opreq* or the *-q* option of *tpmount*).
- If a logical tape drive type (*-t* option) is specified, the choice of drives is limited to one of the devices defined in *tpconfig* as belonging to that logical drive type. If none of the available devices in that group fit the other selection criteria, then the *tpmount* request fails, unless the request is being queued (either by *opreq* or the *-q* option of *tpmount*).

Generally, if parameters are specified that do not fit any device known to the tape system, *tpmount* fails. An exception to this is a *tpmount* command that specifies a device file name (with

-a). If the specified device exists, then *-a* overrides any other device characteristics given in the command, and the device specified with *-a* is used.

If *opreq* is enabled, the operator may choose a drive for the *tpmount*. The system will choose a device based on the criteria specified with *tpmount* (or on the defaults defined for *tpmount* if no criteria were specified).

> **Note**
>
> If *opreq* queueing is disabled and there is more than one tape drive on the system, you must give an unambiguous drive specification with *tpmount*—the tape system must be informed of which drive you want to mount. This can be done through specifying characteristics that apply only to one drive, or by specifying a device name with *-a*. If the drive specification is ambiguous, *tpmount* will fail.

3.1.2 Entering *tpmount* Commands

The *tpmount* command must be entered from a ConvexOS shell; it may have no parameters (thus invoking defaults), or any of the parameters described above. There are two basic kinds of *tpmount* commands: *simple* and *compound*. A simple *tpmount* requests that only one tape device be mounted; this is the most common use of *tpmount*. A compound *tpmount* requests that two or more devices be mounted at the same time.

In the examples that follow, *opreq* queueing is assumed to be disabled. The *tpconfig show all* command can be used to show the configuration of the tape system. (Refer to Chapter 2, "Configuring the Tape System" for more information about *tpconfig*.)

In the examples below, the following defaults are assumed to be in force:

```
% tpconfig show all
Database access is read-only

Drives:
    mt:0        timeout: 60   Controlled
        Alloc access:
            Users:    All
            Groups:   All
        Bypass access:
            Users:    None
            Groups:   None
        Nodes:
        /dev/rmt8       speed:       density: 1600  Rewind     Char
        /dev/rmt12      speed:       density: 1600  No rewind  Char
        /dev/rmt16      speed:       density: 6250  Rewind     Char
        /dev/rmt20      speed:       density: 6250  No rewind  Char
        /dev/mt8        speed:       density: 1600  Rewind     Block
        /dev/mt12       speed:       density: 1600  No rewind  Block
        /dev/mt20       speed:       density: 6250  No rewind  Block

Labels:
    ansi daemon: /usr/lib/tape/ansidaemon

Queueing is: Disabled

Defaults:
    Density: 6250
    Speed: No default
    Drive type: mt
    Mount flags: Character special, No-rewind
```

Here, *tpconfig* shows that the tape system is configured as follows:

- All users may mount drives without specifying a tape VSN ("Alloc access")
- No users may bypass tape label processing (*root* is an exception: users with *root* privileges can always bypass label processing)
- One logical drive—*mt:0*—has been defined, along with associated devices
- Only ANSI labels are recognized
- *opreq* queueing is disabled
- Default density is 6250 bpi
- No default speed has been defined
- Default drive type is "mt"
- The default mount mode is "character", with no automatic rewind

(For the sake of simplicity, some of the devices normally defined for this drive are not shown.)

3.1.2.1 Examples of Simple *tpmount* Commands

The following examples illustrate some uses of simple *tpmount* commands.

Example 3.1.

```
% tpmount
Tape device /dev/rmt20 allocated
%
```

In example 3.1, *tpmount* is entered without parameters. Accordingly, the default density (6250), drive type (mt), access mode (character), and rewind status (No-rewind) are used as criteria. (Because no default speed is given in *tpmount*, and because "No default" is the default speed defined for *tpconfig*, tape speed is not used as a criterion in selecting the device.) One device of the single defined drive (mt:0) fits these criteria: */dev/rmt20*. Consequently, the tape system selected this drive.

Example 3.2

```
% tpmount -d 1600 -m block
Tape device /dev/mt12 allocated
%
```

In example 3.2, a density of 1600 bpi and block access mode is specified. Combined with the default rewind status (no rewind), this leaves device */dev/mt12* as the only possibility.

3.1.2.2 Examples of Compound *tpmount* Commands

Any of the parameters—except for *-B*—may be repeated in the same *tpmount* command. This means that more than one device and more than one tape VSN may be requested in a single command. This way of constructing *tpmount* commands may be useful if simultaneous access to two or more drives is required. For example, suppose that drives A and B are both needed at once. If devices controlling A and B are specified in a separate *tpmount*, then one drive, e..g., A, may be mounted while B is unavailable. Because A cannot be used without B, a system resource is needlessly made unavailable. In addition, if another user has requested another pair of drives, for example A and C, a deadlock may result. This can be avoided by specifying both drives on the same *tpmount* command line; if this is done, the tape system will not mount one drive unless the other is available at the same time.

CAUTION

If two or more users issue compound *tpmount* requests, and if *opreq* is not enabled, a deadlock may occur. For example, if there are two drives on the system—A and B, and if two users simultaneously issue compound *tpmount* commands that request the mounting of both drives, the tape system could allocate drive A to one user and B to the other. Thus, neither compound *tpmount* request could be fulfilled, and a deadlock will have resulted. *opreq* automatically prevents such deadlocks.

Many of the options have both "-" and "+" versions; the "-" version turns the option "on," while the "+" version turns it "off". This feature is useful in commands that request a multiple mount. Because any option specified in such multiple requests affects all of the remainder of the command line, it may be necessary to reverse the options. For example, if more than one mounting operation is requested in the same command, label bypass may be specified for one but not the other. The following example illustrates such a case.

In example 3.3, we will assume that an additional logical drive (mt:1) and its nodes have been defined:

```
Tpconfig> show drive mt:1
    mt:1        timeout: 60   Controlled
        Alloc access:
            Users:    All
            Groups:   All
        Bypass access:
            Users:    None
            Groups:   None
        Nodes:
        /dev/rmt9       speed:       density: 1600  Rewind     Char
        /dev/rmt13      speed:       density: 1600  No rewind  Char
        /dev/rmt17      speed:       density: 6250  Rewind     Char
        /dev/rmt21      speed:       density: 6250  No rewind  Char
        /dev/mt9        speed:       density: 1600  Rewind     Block
```

Example 3.3.

```
% tpmount -B -b -a /dev/rmt8 -s ONE VSN123 +b -m label -a /dev/rmt21 -s TWO VSN456t
```

In example 3.3, two requests are made to mount tapes. Label processing is bypassed for the first tape (with *-b*); because the default mount mode is character, the tape will be mounted in character (unlabeled) mode even if it is labeled. Label bypass is turned off for the second tape (with *+b*); if it is labeled, it will be mounted in labeled mode (because *-m label* was specified), and all access permissions recorded in the label and the file attributes will be obeyed.

Note	When requesting multiple devices in the same *tpmount*, be sure to specify a different symbolic link (e.g., ONE and TWO in the example) for each one. Also, since each drive can only be mounted once, the tape system cannot simultaneously mount devices that are associated with the same drive (e.g. */dev/rmt8* and */dev/rmt12* could not both be mounted at once).

3.1.3 Completion of *tpmount* Requests

In regard to the way in which *tpmount* requests are processed, there are two fundamentally different ways in which the tape system may be configured—*opreq* queueing may be enabled or disabled. The way in which the completion of *tpmount* requests is handled differs depending on the status of *opreq* queueing. This status is determined by *tpconfig* (described in section 2.2.2, "Turn *opreq* Queueing On or Off").

3.1.3.1 *opreq* Queueing Disabled

If *opreq* queueing is disabled, the tape system attempts to fulfill any *tpmount* request immediately. If the requested drive is available and the command is otherwise in order, the tape system mounts the drive and notifies the user. If the requested drive is busy or the request cannot be carried out for some other reason, then the tape system cancels the request and notifies the user of failure.

An exception to this is a *tpmount* request with the *-q* flag; if this flag is used, the tape system will queue the request if the drive is busy, and will complete the request when the drive is free. Note that queueing with the *-q* flag is not the same as *opreq* queueing; using the flag does not require *opreq* to be running, nor does it require operator intervention. If *opreq* is enabled, the *-q* flag has no effect.

3.1.3.2 *opreq* Queueing Enabled

If *opreq* queueing is enabled, *tpmount* does not directly cause the tape system to mount the requested drive. Instead, the system notifies the tape operator via the *opreq* utility that a mount request has been made. The tape system mounts the drive either when the operator notifies it to do so (e.g., when the correct tape is on the drive), or when the Automatic Volume Recognition (AVR) capability detects the presence of the tape specified in the *tpmount* request. (Of course, the operator may choose not to mount the tape, and cancel the mount request instead.)

3.1.3.3 Executing *tpmount* in Background or Foreground

In general, the shell from which a *tpmount* request was made will be suspended until the mount operation is complete or has failed. By waiting in a suspended state, the shell is sure to receive any notifications or error messages that pertain to the *tpmount*. Another advantage of waiting is that it may be desirable for the shell to do nothing further until the *tpmount* request has been completed (for example, a script that includes a *tpmount* may be intended to halt until the tape is

available).

It is possible, however, to run *tpmount* in the background, and so free the shell for further commands while the mounting operation is being processed. One method is to type the keyboard interrupt character (usually CTRL-C) after *tpmount*:

Example 3.4.

```
% tpmount
^C
Mount request placed in the background.
Use tpunmount to cancel tape request.
%
```

Typing CTRL-C causes *tpmount* to be run in the background, and the shell prompt returns.

Another method is to use the *-B* flag with *tpmount*; if this flag is used, the command will be run automatically in the background.

CAUTION

If you run *tpmount* in the background, you will not see error messages that pertain to the request (unless you invoke *tpwait*). For example, if you request a drive that is not available and *tpmount* fails, no notification will be given.

3.1.3.4 *tpwait*

If *tpmount* is running in the background, it may be necessary to bring it to the foreground. This can be done with the *tpwait* command. *tpwait* can be used without parameters if only one *tpmount* is active (a *tpmount* is "active" until it has been removed with *tpunmount*, regardless of whether or not the mounting operation requested in the *tpmount* has been performed). If more than one *tpmount* is active, then the *tpmount* that is to be brought to the foreground must be specified with the *-s <symbolic_name>* option of *tpwait*. (The *-s* option specifies the name of the logical link that is associated with the *tpmount*.)

In example 3.5, a device mount is requested; the *tpmount* is then placed in the background, and a second device is mounted. Finally, the first request is returned to the foreground to await its completion. (The example assumes that *opreq* queueing is turned on; *tpmount* requests are not necessarily completed immediately.)

Example 3.5.

```
% tpmount -s MYLINK1
^C
Mount request placed in the background.
Use tpunmount to cancel tape request.
% tpmount -s MYLINK2
^C
Mount request placed in the background.
Use tpunmount to cancel tape request.
% tpwait
tpwait: ambiguous request
% tpwait -s MYLINK1
```

As shown in the example, merely entering *tpwait* when more than one *tpmount* command has been queued results in an error message ("ambiguous request"); in this case, the symbolic link name must be specified to identify the request that is to be brought to the foreground. When *tpwait* has been entered successfully, the shell is suspended until the mount has been done; a message to that effect will then be displayed, and the shell prompt will return:

```
% tpwait -s MYLINK1
Tape mounted/Drive allocated.  Device: /dev/rmt21
%
```

3.1.4 Unmounting Devices with *tpunmount*

After the desired tape I/O work has been done, the user must unmount the tape device with the *tpunmount* command. This command reverses the file linkages that were created with *tpmount*, and frees the drive for other users. *tpunmount* may also be used to cancel *tpmount* requests that have not yet been completed (i.e., the device has not yet been mounted). Unless *-k* is specified, *tpunmount* will take the drive offline.

Table 3-2: *tpunmount* **Parameters**

Parameter	Meaning
-k	If this flag is given, the *tpunmount* will not take the drive offline. The tape will, however, be rewound. This option is ignored if *opreq* queueing is enabled (in this case, the tape is always taken off-line).
-s <*symbolic_name*>	If only one drive is currently mounted for the user, then *tpunmount* can be entered without parameters—the tape system automatically dismounts the mounted drive. If more than one drive is mounted by the user issuing the *tpunmount*, the name of the link to the device file must be specified with the *-s* option. If no mounted drive matches the symbolic name, an error message ("no such request") is given, and no drive is dismounted.

The following are examples of *tpunmount*:

Example 3.6.

```
% tpmount
Tape device /dev/rmt20 allocated.
% tpmount -s TAPE2
Tape device /dev/rmt22 allocated.
% ls -l
total 30
lrwxrwxrwx  1 dunbar           10 Oct 23 14:17 TAPE -> /dev/rmt20
lrwxrwxrwx  1 dunbar           10 Oct 23 14:17 TAPE2 -> /dev/rmt22
drwxr-xr-x  2 dunbar          512 Oct 23 14:16 bin
-rw-r--r--  1 dunbar        12343 Oct 12 14:41 config.db
drwxr-xr-x  2 dunbar         1024 Aug 30 04:43 gnu
-rw-rw-r--  1 dunbar         2865 Sep  5 15:54 hosts
-rw-r--r--  1 dunbar          145 Oct  4 13:59 show.def
drwxr-xr-x  2 dunbar          512 Oct 23 14:05 test
% tpunmount -s TAPE
% tpunmount
% ls -l
total 26
drwxr-xr-x  2 dunbar          512 Oct 23 14:16 bin
-rw-r--r--  1 dunbar        12343 Oct 12 14:41 config.db
drwxr-xr-x  2 dunbar         1024 Aug 30 04:43 gnu
-rw-rw-r--  1 dunbar         2865 Sep  5 15:54 hosts
-rw-r--r--  1 dunbar          145 Oct  4 13:59 show.def
drwxr-xr-x  2 dunbar          512 Oct 23 14:05 test
```

In example 3.6, two tape drives are mounted. By listing the current working directory (with *ls -l*), the user can see that two links have been made for the drive (TAPE and TAPE2).

By specifying the name of one of the logical links (TAPE) with *tpunmount*, the user can unmount the drive designated by this link. After this has been done, the second drive can be unmounted by invoking *tpunmount* without parameters.

3.2 Labeling Tapes and Setting File Attributes

The ConvexOS tape system supports ANSI-standard tape labels. These labels contain information about the *volume* (i.e. the entire tape or multivolume tape set) and about each file on the tape. The *tplabel* command is used to turn an unlabeled or blank tape into a labeled one; *tpunlabel* can be used to remove the label. Because *tpunlabel* writes blanks over the label at the beginning of the tape, all data on the tape becomes inaccessible if this command is used. If it is mounted after *tpunlabel* is invoked, the tape will then be treated as unlabeled by the tape system. Attributes for individual files on labeled tapes are stored in file labels, and can be set with *tpattr*.

Before a labeled tape can be read or written, or a new label placed on an unlabeled tape, the drive must have been mounted in labeled mode. This is done by entering a *tpmount* command with the *-m label* option. (The option need not be specified if labeled access mode was specified as the default when the system administrator configured the tape system with *typconfig*.) If an unlabeled tape is recognized on a drive mounted in labeled mode, a warning is displayed (refer to example 3.7, below).

3.2.1 Creating Labels: *tplabel*

Labels are created with the *tplabel* command. Labels can be created only on tapes that are not already labeled (to erase labels, use *tpunlabel*, described below). *tplabel* has two parameters, described in Table 3-3.

Table 3-3: *tplabel* Parameters

Parameter	Meaning
-a	If this option is specified, the tape being labeled will have restricted access. The owner (identified by the login name of the label's creator) can always read and write the tape; but if *-a* is given, other users will not be allowed to do so. CAUTION Permission to bypass label processing with the *-b* option can be granted by the system administrator with *tpconfig*; any user who can bypass label processing can bypass the access restrictions created by *-a*. (*root* is always able to do this.)
-s <symbolic_name>	If the user has more than one mounted drive, then the symbolic name linked to the drive on which the tape is to be labeled must be specified with the *-s* option of *tplabel*.

The VSN that is assigned to the tape (and stored in the ANSI volume label) will be the VSN that was specified in *tpmount*; if no VSN was specified, then the tape will have a VSN composed of 6 blanks. (Six characters is the maximum length of a VSN.) The VSN and access restriction can only be changed by deleting the label and then relabeling the tape.

In example 3.7, an unlabeled tape is mounted in labeled access mode. *tplabel* is then used to label it.

Example 3.7.

```
% tpmount -m label -s MYTAPE2 RDD978
tpmount: Warning: tape is not labeled; use tplabel to label tape
% tplabel -a -s MYTAPE2
%
```

After the tape is labeled, it is left mounted and further I/O may be performed on it.

3.2.2 Removing Labels: *tpunlabel*

Labels can be removed with the *tpunlabel* command. To remove a label, first mount the drive in labeled mode with *tpmount*. Only the owner of a tape or a user logged on as *root* can unlabel it.

tpunlabel has one parameter: *-s <symbolic_name>*; as in *tplabel*, this parameter is used to specify the symbolic link to the drive on which the tape is to be unlabeled.

> **CAUTION** Since *tpunlabel* writes blanks at the beginning of the tape, all data on the tape becomes inaccessible if this command is used. In effect, all data on the tape is deleted.

When a tape is unlabeled, it can be given a new label with *tplabel*. To use the tape for unlabeled tape I/O, unmount the drive, then mount it in an unlabeled mode (either block or character). This is done in example 3.8.

Example 3.8.

```
% tpmount -m label -s TAPE2 RDD978
Tape RDD978 mounted.  Device: /dev/lt/u1
% tpunlabel -s TAPE2
% tpunmount -s TAPE2
% tpmount -m block -s TAPE2 RDD978
Tape RDD978 mounted.  Device: /dev/mt21
```

When the drive is first mounted, the ANSI label device */dev/lt/u1* is shown; when the drive is mounted in unlabeled mode, the device */dev/mt21* is shown. The tape may be left on the same drive between these two mountings; it is just the logical links that have changed between the first and second mountings. (Labeled access requires that an ANSI label device be interposed between the user and the tape device; with unlabeled access, the label device is not used.)

Also note that in the second mounting, a VSN number is specified, even though the tape is now unlabeled, and the number is not actually part of the magnetic data on the tape. Though not required for unlabeled tapes, VSNs can be used in *tpmount* requests for such tapes to give information to the operator. For example, the VSN might be used to inform the operator of the identity of the tape that is to be placed on the drive; in this case, it may correspond to the number written on a paper label or to the rack location of the tape.

3.2.3 Setting File Attributes on Labeled Tapes: *tpattr*

The ANSI tape volume label contains information such as the VSN and access restrictions that pertain to the tape as a whole. In addition, individual files on the tape can be given certain attributes with the *tpattr* command. When this command is given, it sets the attributes for all files subsequently written to the tape until either the drive is dismounted or the attributes are reset with another *tpattr*. *tpattr* is used primarily to specify tape file attributes before they are written. However, if something other than block newline is specified as the record delimiter when files are written, then *tpattr* must also be called before the files are read to specify the correct delimiter. (Refer to section 3.2.3.3, "Record Delimiters.")

Attributes that can be given to files by specifying them as parameters of *tpattr* are listed in Table 3-4 (all parameters are optional):

Table 3-4: *tpattr* Parameters

Parameter	Meaning
-a r|w|rw|""	Sets file accessibility for users other than the tape owner; unlike the access restriction that may be applied to the tape as a whole, this applies only to the file. If *-a r* is specified, other users have only read privileges; if *-a w* is specified, other users have only write privileges; *-a rw* gives other users both read and write privileges. If a pair of quotes ("") follows *-a*, other users have no access privileges to the file. The default access permission is read-only. (No special privileges are normally given to users logged on as *root*. However, root could use the *-b* option of *tpmount* to bypass labeled tape access permissions.)
-b *<blocksize>*	The physical block size for the file. The default blocksize is 2048 bytes. (Refer to the description of tape blocks in section 3.2.3.1, "Blocks" for more information).
-d newline|syscall|blocknl	Logical record delimiter; may be a newline character (*newline*), a system call (*syscall*), or blocked newline (*blocknl*—same as newline, but the tape system automatically reads the number of records required to fill the buffer of the *read* system call). Refer to section 3.2.3.3, "Record Delimiters" for more information about record delimiters. The default record delimiter is block newline.
-f F|D|U	The record format; may be fixed (*-f F*), decimal-based variable (*-f D*), or unformatted (*U*). The default record format is fixed. (Refer to section 3.2.3.2, "Logical Records" for more information about record format.)
-i *<file_identifier>*	An identifier for the file; can be no longer than 17 characters. Alphabetic characters will be stored and read in upper case (even if they are specified as lower case). Therefore, file identifiers on labeled tapes cannot be differentiated by case. The default file identifier is NONAME.
-r *<record_size>*	Logical record size in bytes. The default record size is 128 bytes.
-s *<symbolic_name>*	Name of the symbolic link that points to the drive on which the destination tape is mounted.

Four parameters described in Table 3-4 govern the way data is written to the tape file:

- Block size
- Logical record delimiter
- Record format
- Logical record size

These parameters are set when the tape file is written. When a tape file is being read, the values established when the file was written are automatically used, except for the logical record delimiter (Refer to section 3.2.3.3, "Record Delimiters").

3.2.3.1 Blocks

For ANSI-labeled tapes, the ConvexOS tape system supports block-buffering ("blocking") of I/O data; this means that I/O is buffered on a block-by-block basis by *ansidaemon*. A *block* is a physical characteristic of the arrangement of data on a tape; it is the same as a *physical record*; such blocks or records are sections of contiguous data on the tape that are separated by *interrecord gaps*.

Because labeled-tape I/O is buffered, it is asynchronous. When data is written to tape by a user, each *write* system call goes to *ansidaemon*, and is held until an amount of data equal to the block size is accumulated; at that point, *ansidaemon* sends it to the tape. Similarly, an entire block of data is retrieved from tape by *ansidaemon* when it receives a *read* system call, even though a smaller amount may have been requested in the *read*.

Block size can be adjusted with the *-b* parameter of *tpattr*.

CAUTION	Because I/O to ANSI-labeled tapes is asynchronous, failure of a write may not be detected immediately. The failure may occur when *ansidaemon* actually writes a block of data to tape, not when the user makes a write request.

Though the ANSI standard specifies a maximum block size of 2 kbytes (the *tpattr* default), the ConvexOS tape system allows blocks up to 64 kbytes. If the tape is to be transported to a non-CONVEX machine, be sure to verify that the target system accepts larger blocks before assigning a block size greater than 2 kbytes.

A larger block size means that less system resources are spent on tape I/O, because it is more efficient to write large amounts of data at long intervals than to write small amounts frequently. However, a large block size also means that data is written less frequently to tape. This will increase the likelyhood of data loss if there is any abnormal termination (e.g, a program or system crash).

For efficient utilization of storage media, block size—especially for fixed-size records—should be set to a multiple of the record size whenever possible. In general, a block size set equal to the record size will minimize I/O buffering.

3.2.3.2 Logical Records

Blocks (or physical records) are *physical* units of I/O, and the user need never deal with them—blocks exist solely for the convenience of the tape system. In contrast, *logical records* are part of

the intrinsic structure of the data; a logical record is a unit of data that is visible or significant to users of the data. Common examples of a logical record are a line of text terminated by a newline character, or a record from a database table.

Three kinds of logical record formats supported for tape files by the CONVEX tape system can be specified by the *-f* parameter of *tpattr*:

- **Fixed-length** (*-f F*) records all have the size specified with the *-r* parameter of *tpattr*. When the tape system encounters an end-of-record delimiter while writing data to tape (refer to section 3.2.3.3, "Record Delimiters" for information about delimiters), and if the number of bytes of data is less than the record size (as specified with the *-r* parameter), the record is padded with space characters to the specified length. Writing a record longer than the specified size results in an error message returned by the *write* system call (ETPTRUNC); the record is not written. When a fixed-length record is read from a tape file, all trailing spaces are deleted. (This means that any spaces at the end of a record will be lost, even if they were originally present.)
- **Variable-length** (*-f D*) records may be of any length up to 9,999 bytes. The length of the record is always indicated by an index at the beginning of the record. The index is four bytes long, and specifies the length of the record in bytes, including the length of the index. The index uses ASCII representation of base 10 numbers. (Because the length index must be accommodated, the maximum amount of data that can be contained by a variable-length record is actually 9,995 bytes).

  When the tape system encounters an end-of-record delimiter while writing variable-length records, it prepends the record length to the record. When variable-length records are read, this length index is used to locate the end of the record. Variable-length records will not be padded, nor are trailing spaces stripped when read.

  The *-r* parameter of *tpattr* specifies the actual maximum length of the records of a specific tape file, including the length index. Thus, the maximum amount of data that can be contained in a variable-length record of a tape file is four bytes less than the record length specified with the *-r* parameter. (The maximum length specified by *-r* for variable-length records may in no case exceed the maximum size of 9,999 bytes.)
- **Unformatted** (*-f U*) records have no record structure. Such a record type might be used, for example, in binary files containing executable code. If the record type is unformatted, the tape system will write a block of data to tape whenever a *write* system call is initiated by the user. Using the unformatted record type is similar to using the system call record delimiter (refer to section 3.2.3.3, "Record Delimiters"); however, the blocksize will vary, depending on the amount of data sent with each system call, and no record size or record delimiters will be written. Since there is no ANSI standard for this record type, care should be taken to avoid compatibility problems if the tape will be read on a non-CONVEX system.

CAUTION	If the logical record being read from tape is larger than the size of the *read* system call buffer, the record will be truncated to fit the buffer. The rest of the record will be lost. Therefore, care must be taken to keep the logical record size from exceeding the size of the buffer provided by the tool that will read the data. (E.g., for *cp*, this is 64 kbytes.)

3.2.3.3 Record Delimiters

A *record delimiter* signifies the end of a logical record to the tape system; the system then writes the data contained in the logical record into a tape block. (The system will write as many logical records into a block as will fit in their entirety.) Three kinds of record delimiters can be specified with the *-d* parameter of *tpattr* for fixed and variable length record types (the record delimiter for unformatted records is always a system call):

- **newline** (ASCII code: decimal 10). If this delimiter is used, newline characters in the input are not written to tape. Each time the tape system encounters a newline, it assumes that it has reached the end of a logical record. If the record type is "variable," the tape system prepends the number of bytes in the record. If the record type is "fixed," the tape system pads the record with spaces if it contains fewer bytes than the specified record size. In either case, the newline character is not be written to tape, but is added again when the record is read.
- **blocked newline** (the default). Using this delimiter has the same effect as specifying the *newline* delimiter when writing data to tape, but it substantially enhances the efficiency of *read* operations.

  This is because when *newline* is used as the delimiter, each *read* system call returns only one logical record from *ansidaemon*, no matter how large the buffer used for the call. For example, if *cp* is used to copy a file containing newline-delimited records of a text file from tape, *cp* issues a series of *read* system calls to get the data. The first *read* causes *ansidaemon* to retrieve a block of data from the tape. However, it hands off only one logical record to the *read* call. Thus, though *cp* uses 64 kbyte buffers for its *read* and *write* calls, each buffer will contain only one line of text. This is inefficient, because a large number of *read* calls will have to be issued by *cp* to get the whole file. (*write* calls are not subject to this limitation.)

  Using the *blocked newline* delimiter mode avoids this problem because the buffer of the *read* is filled to capacity. The newline characters are restored just as they would be, had *newline* mode been specified.

> **CAUTION** When using *blocked newline* delimiter mode, data will be read from tape up to the capacity of the *read* buffer; this means that the tape will not necessarily advance only one logical record at a time, as it would with *newline* mode. Thus, care must be exercised to verify the tape position when using any of the *mt* tape movement commands after reading data.

- **system call**. If this delimiter is specified, the tape system assumes that each *write* system call terminates a record (such system calls are issued by any ConvexOS command—such as *cp*—that writes data). Newline characters are not removed. If the record format is "fixed," the record is padded to requisite size with spaces. If the record format is "variable," the record size is prepended to the record, but it is not padded. If fast "write-through" is desired (i.e., if it is important to write data as soon as possible after the user causes a *write* system call), the system call record delimiter should be chosen and the block size should be set equal to the record size.

  The number of bytes written by system calls may vary. However, the *write* system calls issued by *cp* each write 64 kbytes; thus, any *cp* command writes one or more 64 kbyte records if the system call record delimiter is being used. This means that the block size

and record size should be set to 64 kbytes. (Because variable-length records cannot contain more than 9,999 bytes, fixed-length records must be used in this case.)

> **CAUTION** The same record delimiter must be used when reading a tape file as was used when writing the file. Thus, if the default record delimiter (block newline) was not used, *tpattr* must be specified with the correct delimiter before the file is read. (The delimiter specified in this *tpattr* stays in effect until the tape is unmounted or another *tpattr* is issued.) Failure to do this will result in extra or missing newline characters in the output.

Example 3.9.

The following example shows several uses of *tpattr* to set tape file attributes.

```
% tpmount -m label -s LBLTP 123456
Tape 123456 mounted.  Device: /dev/lt/u0
% tpattr -i /etc/passwd -r 150
% cp /etc/passwd LBLTP
% tpattr -i acctdata -f U -a""
% cp /usr/adm/acct LBLTP
% tpattr -i helloworld.c -f D -r 80
% cat helloworld.c > LBLTP
```

The following things are done in this example:

- A labeled tape is mounted. Its logical link is a file called LBLTP, and its VSN is 123456.
- A *tpattr* is used to label the first file that will be written as */ETC/PASSWD* (*-i* is used to do this); the record type is the default (*fixed*), and the record size is declared to be 150 bytes (with *-r*). (All file identifiers are turned to upper case when written, since the ANSI standard requires this.)
- A *cp* command is used to copy the file called */etc/passwd* to the logical link *LBLTP*.
- The next *tpattr* sets the file identifier as *ACCTDATA*; in addition, the record size is *undefined* (*-f U*), and users other than the owner are given no access privileges to the tape file (by specifying empty double quotes after *-a*).
- The file */usr/adm/acct* is copied to the logical link LBLTP.
- *tpattr* sets the next tape file name to *HELLOWORLD.C*. The *variable* record format is specified (*-f D*), and the record length is 80 bytes. (By default, the newline character is the record delimiter.)
- The file *helloworld.c* is written to the logical link *LBLTP* with the *cat* command.

3.3 Tape Manipulation Commands

Because the tape system mounting operation creates a file in the user's working directory that is logically linked to the tape drive, reading and writing data on tape is much like reading or writing data to a normal ConvexOS file. Once the tape has been mounted, the same ConvexOS commands used to perform I/O with normal files are used for tape I/O. When sending data to the tape, the logical link is used as the target of the command; when getting data from tape, the logical link is the source from which data is obtained.

For example, the following commands all write data to the tape (the examples assume that a tape is mounted and linked to *TAPE* in the current working directory and that *myfile* is a normal file in the current working directory).

```
% cp myfile TAPE
% cat myfile > TAPE
% grep "string" myfile > TAPE
% echo "hello, world" > TAPE
```

All of the following commands all obtain data *from* the tape:

```
% cp TAPE myfile
% cat TAPE > myfile
% grep "string" TAPE
```

Despite the similarities, there are some differences between disk file I/O and tape; these differences result from the physical characteristics of tape: it is a sequential access medium. In the examples above, *TAPE* is not like a disk directory; it cannot, for instance, be searched for files with the *ls* command. A series of files copied to *TAPE* is written to the tape in sequence; after each file has been copied, the tape is at the end-of-file marker after that file. At that time, doing a *cat* of *TAPE* will not return any data, because the tape head is at the end of the last file on the tape. To *cat* any of the previous files, it is necessary to rewind the tape to the beginning of that file. In addition, each command that reads data from the tape (*cat*, *grep*, *cp*, etc.) operates on only one file. For example, if you want to *cat* all files on the tape (once you have rewound the tape to its beginning), you must issue a *cat* command for each file.

A set of special ConvexOS tape manipulation commands—the *mt* commands—provide the capability to move the tape forward or backward (e.g., rewind the tape, move backward/forward a specified number of files or records, etc.). Table 3-5 lists the effects of the *mt* commands on labeled and unlabeld tapes. Refer to the *mt*(2) page in the *ConvexOS Programmer's Reference* page for additional information.

> **NOTE** The *mt* commands work only for tapes mounted in character or labeled modes. They do not work for block mode.

The effect of some *mt* commands differs depending on whether the tape being manipulated is an ANSI-labeled tape or not. The following table shows the *mt* commands, along with their effects on both labeled and unlabeled tapes.

Table 3-5: Effect of *mt* Commands

mt Command	Unlabeled (Character) Mode	Labeled Mode
eof <*n*>, weof <*n*>	write <*n*> end of file markers	write <*n*> empty ANSI files
fsf <*n*>	forward space <*n*> files	forward space <*n*> logical files
fsr <*n*>	forward space <*n*> records	forward space <*n*> logical records
bsf <*n*>	back space <*n*> files	back space <*n*> logical files
bsr <*n*>	back space <*n*> records	not allowed
rewind	rewind tape	rewind tapeset (leave at first ANSI file)
offline, rewoffl	take tape offline	not allowed - use *tpunmount*
gap <*n*>	write <*n*> interrecord gaps	ignored
status	return tape status	return (physical) tape status

The *bsf* and *fsf* commands work the same for labeled and unlabeled tape. Though the labels and tape marks that are found between files are different for a labeled tape and an unlabeled tape, these two tape movement functions have the same result in both cases. *bsf* moves the tape backward until the specified number of end-of-file markers have passed the tape head. The tape is then stopped just before that last end-of-file. *fsf* makes the tape move forward until the specified number of end-of-file markers have passed by the tape head. The tape is stopped just after the last marker.

Thus, if you are currently positioned in file 3, and you want to read from the beginning of file 2, you should issue an `mt bsf 2` command, and then `fsf 1`.

The following figure (3-1) depicts such a situation in general terms.

Figure 3–1: Tape Movement for *bsf* and *fsf* commands

The following symbols are used in this figure:

* is an end-of-file mark
^ marks the current position of the tape head

a. The tape is positioned at the beginning of file 3:

```
-------------------------------------------------------------------
<file 1 data>  *  <file 2 data> *  <file 3 data>  *
-------------------------------------------------------------------
                                  ^ (ready for file3)
```

b. An `mt bsf 2` command will make the tape driver scan backwards until the head has crossed two end-of-file marks. The tape will be positioned at the end of file 1:

```
-------------------------------------------------------------------
<file 1 data>  *  <file 2 data> *  <file 3 data>  *
-------------------------------------------------------------------
          ^ <------
```

c. Now, an `mt fsf 1` will position the tape at the beginning of file 2 (the tape driver scans forward past one tapemark):

```
-------------------------------------------------------------------
<file 1 data>  *  <file 2 data> *  <file 3 data>  *
-------------------------------------------------------------------
        ------>  ^
```

CAUTION	Any attempt to read past the end of tape marker on an ANSI-labeled tape will fail; an error message ("No such file or directory") will be given. On unlabeled tapes, reading will be attempted (and may return spurious data).

Example 3.10

Example 3.10 is a continuation of Example 3.9. In that example, the file *helloworld.c* was the third file written on the tape. In Example 3.10, the tape is rewound, positioned at the *helloworld.c* file, and the file is read with *cat*.

```
% mt rew
% mt fsf 2
% cat LBLTP
/* helloworld.c */
#include <stdio.h>

main()
{
    printf ("hello, world!\n");
}
```

Chapter 4
Operator Request Management

The ConvexOS Operator Request Management Utility (*opreq*) provides an interactive, window-based interface for computer operators. At present, only the ConvexOS tape system works through *opreq*; other subsystems may be given an *opreq* interface in the future.

opreq allows computer operators to service requests for tape system resources and to monitor the status of the tape system. If *opreq* queueing has been enabled by the system administrator (the procedure for doing this is described in Chapter 2), messages that give information about the status of the tape system and about tape system resource requests are displayed in the *opreq* window.

Requests for mounting tape devices—i.e., *tpmount* commands—function differently depending on whether *opreq* queueing is enabled. If it is enabled, *tpmount* requests are queued. This means that if there is more than one request for access to a given drive, the requests will "wait in line" until they can be processed. If *opreq* is not enabled, such requests are handled directly by the tape system, and either succeed or fail immediately. (An exception to this is the *-q* option of *tpmount*, which is described in Table 3-1 and section 3.1.3.1, "*opreq* Queueing Disabled.")

The *tpconfig show all* or *show defaults* commands (described in section 2.1.2.2, "Using *tpconfig* Interactively") can be used to find out whether *opreq* is enabled. This chapter presumes that *opreq* is enabled on your system.

When you invoke *opreq* at a terminal, an *opreq* window is opened. This window shows messages pertaining to the tape system. One important kind of message notifies you of queued *tpmount* requests. Other messages that can be shown in the window give additional information, such as the VSN of tapes that have been recognized by the tape system, requests to change tapes (in multi-volume tape sets), *tpunmount* requests, etc.

In addition to giving information that helps the operator place the required tapes on appropriate drives (or remove them), the *opreq* window also provides facilities to assist the operator in initiating device mounting and dismounting operations, and to change the status of requests (e.g., to cancel them). The type of information shown about each message is configurable; refer to section 4.4.1, "Selecting Message Fields for Display" for information about how to do this.

If there is more than one operator, multiple instances of *opreq* may be active on the system at any one time. If this is so, operators can assign tape requests to themselves by using the *select work* command to prevent conflicts.

4.1 Starting *opreq*

To start *opreq*, enter the following at the ConvexOS shell prompt:

```
% opreq
```

> **Note** — */usr/convex* must be in your path, or this command will not work.

An *opreq* window appears on your screen, and looks something like this:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE-------------TIME IN------- 1+
| joe  Ready   Mount-Tape   1234      No    mt:0 @ mach1      4:27pm Aug 23   |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
+ -----------------------------------------------------------------------------+
Command:
```

An *opreq* window has three main parts:

- **Title Bar**—The title bar is the series of labels that appears across the top of each window (e.g., "UID," "STATUS." etc. in the example above). These labels refer to the message information that is shown beneath them. You can choose the labels that appear in the title bar; refer to section 4.4.1, "Selecting Message Fields for Display" for information about how to change the labels.
- **Messages**—One or more messages about the status of the tape system may appear in the window under the title bar. In the example above, there is one message. As indicated under the TYPE label, it is a request to mount a tape device. When a user enters a *tpmount* command, such a message appears in the *opreq* window. The message indicates the originator of the request—in this case, a user with the ID "joe" (shown under the "UID" label), the status ("Ready" means that the device has not yet been mounted), the Volume Serial Number (VSN) of the requested tape, a drive on which the tape can be placed ("mt:0"), and other information. These units of message information that appear under the title bar label are called *fields*.
- **Command Line**—This is the line beginning with the `Command:` prompt at the bottom of the screen. You may enter commands to *opreq*; begin typing, and the command line displays the text. The `Command:` prompt may change; for example, if you have typed "configure-type", *opreq* changes the prompt to `Configure:`. There will only be one command line, even if multiple *opreq* windows are being displayed on the screen. Refer to section 4.2.3, "*opreq* Commands" for more information about commands.

More than one *opreq* window may be opened on the same terminal at once. The window number shown at the right end of the title bar (in the example, it is number "1") will be incremented for each additional window. Although multiple windows can be open, only one window can be *active* at any one time at a single terminal. The active window is always indicated by a highlighted, or reverse video, border. (The appearance of reverse video depends on characteristics of the display. If characters are normally shown as white-on-black, reverse video will be black-on-white; the opposite is true if characters are normally shown as black-on-white.) The command line always applies to the active window (the window with the highlighted border).

Refer to the description of the *window...* commands in section 4.2.3.7, "Opening, Closing, and Moving Between Windows" for information about how to open additional windows.

4.2 Using the *opreq* Window

To use *opreq*, you must understand three basic operations that can be performed in the window:

- Highlighting
- Selecting menu items
- Entering commands

4.2.1 Highlighting

Before you can act on any item displayed in an *opreq* window, the item must be *highlighted* (i.e., shown in reverse video). Typically, one item of each kind (e.g., messages or menu items) will be highlighted at one time. This area of highlighting is called the *selection cursor.*

The selection cursor can be moved. To highlight different items, move the selection cursor up or down by pressing the "up" or "down" cursor keys, or by typing `up` or `down` on the command line.

Various actions can be performed on highlighted items. For example, when a message is highlighted, it can be acted on by typing a command. Thus, to cancel a *tpmount* request, you would highlight the message that corresponds to it and type *select-cancel.*

4.2.2 Switching or Selecting Menu Items

The *opreq* window can be made to display a variety of menus. Some menus present a selection of options that can be switched on or off, while others allow selection of a particular item from a group.

For example, to select a drive to be mounted in response to a *tpmount*, move the selection cursor in the drive menu until the desired drive is highlighted, then press RETURN.

To turn the UID option on the *configure-title* menu on or off, move the selection cursor until it is over the UID entry in the title bar label menu, and then press RETURN. In general, the state of *opreq* menu items that can be toggled is indicated by an "X" next to the item. If there is an "X", the item is turned on; pressing RETURN will switch it off. The reverse is true if the item is already turned off, i.e., there is no "X"; pressing RETURN will switch it on.

4.2.3 *opreq* Commands

opreq recognizes a number of commands; to see a list of them, type a question mark (?). (Refer to section 4.2.3, "*opreq* Commands for a description of these commands).

4.2.3.1 Entering Commands

As you type *opreq* commands, they appear next to the `Command:` prompt at the bottom of the window. If more than one *opreq* window is being displayed on your screen, the commands will always apply to the active window.

It is not necessary to type the complete command; *opreq* completes words in commands as soon as it recognizes an unambiguous portion of the word. For example, *opreq* recognizes both *configure type* and *configure title* commands. If you type "c", the word "configure-" appears after the `Command:` prompt, because this is the only word beginning with "c" known to *opreq* that can occur at the start of a command. If you then type "t", nothing happens, other than that the letter is added to those on the command line. At this point, the remainder of the command is still ambiguous, and *opreq* does not know how to complete it. (If you are not sure of which command you want to enter, you can type a question mark (?) to show all possible command completions.) As soon as you type "y", the complete command (*configure-type*) appears on the command line. (You should now stop typing; if you attempt to type the rest of the word, your terminal will emit a "beep".)

To carry out a command that appears on the command line, press (RETURN) . If you change your mind about executing a command that you have typed before you press (RETURN), backspace over the command to delete it. To cancel an interactive command that has begun executing (e.g., by displaying a menu), enter *cancel.*

4.2.3.2 Display Information

The *display...* commands cause information to be displayed in the *opreq* window. There are two such commands:

- *display-messages*—causes the message display in the window to be refreshed. Messages that have a "done" or "canceled" status will not be cleared from the window until the display is refreshed. (If the window has been configured to display cancelled messages, they will not be removed even if the display is refreshed.)
- *display-avail-drives*—If this command is given, a list of available drives appears superimposed on the window.

4.2.3.3 Move Selection Cursor

You can move the selection cursor up, down, left or right by using the "up" or "down" cursor keys. In addition, you can move the cursor by typing these movement commands:

- *move-up*— Move selection cursor up.
- *move-down*— Move the selection cursor down.

4.2.3.4 Change Status of Selection

These commands change the status of the selected message. To use them, first move the selection cursor (the highlighted area) over the desired message, then enter the command. There are three *select...* commands:

Note	*opreq* must be running with *root* privileges to use the *select...* commands. Contact your system administrator if you lack these privileges.

- *select-cancel*—cancels the highlighted message. (If canceled messages are not being displayed, the message will disappear when the *display-message* command is next invoked, or a new window is opened.)
- *select-done*—changes the status of the message to "done" (i.e., the indicated action—such as a tape mount—has been performed.) If the message is a mount request, a list of available drives is displayed; an "X" appears next to the drives that are available. The operator must then select the drive on which the tape has been placed, and press RETURN. If a VSN is specified in the *tpmount*, the operator cannot mount a drive that has a labeled tape with a different VSN on it.
- *select-work*—assigns the message to the operator who enters the command, and prevents other operators from working on it.

4.2.3.5 Configure the *opreq* Window

The *configure...* commands determine what information is displayed for each message and allow you to restrict the kinds of messages that will be displayed. (The use of these commands is more fully described in section 4.4, "Configuring the *opreq* Window.") There are three such commands:

- *configure-title*—changes the labels that appear in the window title bar (and, consequently, the information displayed for each message).
- *configure-type*—determines which types of messages will be displayed.
- *configure-status*—chooses messages to be displayed on the basis of status.

4.2.3.6 Display Message Field

This group of commands can be used to display a specified field for a selected message. (Since there may not be enough room in the window to display all information for each message, these commands can be used to show information that is not routinely displayed.)

The message field commands have the format *field-<field_name>*, where *<field_name>* is the name of any message field supported by *opreq*. For example, *field-MID* would cause the message identifier of the selected message to be displayed. The possible field names are described in section 4.4.1.1, "Description of Message Fields."

4.2.3.7 Opening, Closing, and Moving Between Windows

The *window...* commands can be used to open new *opreq* windows, close windows, or move between windows. There are three such commands:

- *window-open*—opens a new *opreq* window. After you enter this command, a small box appears on the screen. This box is used to mark the position of the upper left corner of the new window. Move the box to the desired location by using the cursor keys or *move-...* commands, then press RETURN . Next, position the bottom right corner of the new window in the same way. (To cancel the command, enter *cancel.*)
- *window-close*—closes the active window.

- *window-goto*—can be used to move between multiple *opreq* windows. There are several ways to use this command:
  - — If there are only two windows, the other window is brought into the foreground automatically when *window-goto* is entered.
  - — Typing the number of the window after the window-goto command moves the specified window to the foreground. (The number of each window is in the upper right corner.)
  - — The cursor keys can be used to move a special cursor that appears when this command is typed, and more than two windows are open on the screen. When the cursor is positioned over the desired window, press the RETURN key.

4.3 Using *opreq* for Tape Operations

If *opreq* queueing is enabled and a user enters a *tpmount* command, a message appears in the *opreq* window. This message indicates to the operator that exclusive access to a tape drive has been requested by a user.

4.3.1 Examples of Tape Mount Requests

Suppose that user "jane" enters a *tpmount* command. At the shell prompt, she enters:

```
% tpmount
```

At about the same time, user "ann" also enters a *tpmount* from her terminal:

```
% tpmount -m label -s TAPE1 AX3343
```

Finally, user "joe" enters this command:

```
% tpmount -m label -s TAPE1 AX1745
```

As a consequence of these commands, the following messages appear in the *opreq* window:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE-------------TIME IN------- 1+
|jane   Ready   Mount-Tape              Yes  mt:1 @ dragonne   3:20pm Nov  7   |
|ann    Ready   Mount-Tape   AX3343     Yes  mt:1 @ dragonne   3:22pm Nov  7   |
|joe    Ready   Mount-Tape   AX1745     Yes  mt:1 @ dragonne   3:23pm Nov  7   |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
+ ------------------------------------------------------------------------------+
Command:
```

These messages reflect the *tpmount* commands issued by the users. The status of all messages is "Ready"; this means that no device has yet been mounted in response to the request, but that the operator may initiate mounting. The entry under "DRIVE" is only a suggestion; it indicates the first available drive (and the name of the system on which it resides) found by the tape system. The operator is not necessarily limited to using this drive; a menu of all available drives will be shown in response to a *select-done*. (Refer to section 4.4.1.1, "Description of Message Fields" for a description of all message fields that may be shown.)

Note

The operator must know how to interpret user requests. For example, Jane did not specify a VSN; does this mean that she wants a blank tape, or an unlabeled tape? Or does she want a specific labeled tape, and expects the operator to know which one? Ann specified a VSN; does she want a (previously labeled) tape that has this VSN, or does she want the operator to put a blank tape on the drive so that she can label it with this VSN?

Procedures must be established so that the operator has the information necessary to interpret user requests. For example, labeled tapes should have paper labels corresponding to their magnetic VSN.

For this example, we will assume the following:

- Jane wants a blank tape, and intends to use it in unlabeled mode.
- Ann wants a blank tape, and will label it.
- Joe wants a previously labeled tape with the specified VSN.

To service these requests, the operator might first highlight Ann's request and type the *opreq* command *select-done*. The drive selection menu appears in the upper right corner of the *opreq* window to permit the operator to choose the tape drive that will be used for Ann:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE-------------TIME IN------- 1+
|jane   Ready   Mount-Tape               Yes  mt:1 @ dragonn+ VSN = AX3343      +
|ann    Ready   Mount-Tape   AX3343      Yes  mt:1 @ dragonn|X mt:1 @ dragonne  |
|joe    Ready   Mount-Tape   AX1745      Yes  mt:1 @ dragonn|X mt:0 @ dragonne  |
|                                                          |X mt:2 @ dragonne  |
|                                                          +-------------------+
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
+ -----------------------------------------------------------------------------+
Command:
```

This menu shows all the drives installed on the system; the available drives are marked with an "X" on the left. (In the example, all three drives are available.)

The operator then highlights the desired drive (mt:1, for example), and presses RETURN . The drive selection menu disappears, and the status of Ann's request changes from "Ready" to "Mounted". The system is now waiting for a tape to be placed on mt:1. When the operator does this, the tape system will recognize the presence of the tape and notify *opreq*; *opreq* will then change the status of the message from "Mounted" to "Done". This means that the tape system has mounted the drive—it has established the logical linkages required to allow the user to read and write tape data. When mounting has been done, Ann is notified by a message that appears on her terminal that her *tpmount* command has been completed. When the operator later removes the tape from the drive, he or she should place a paper label on the tape with the VSN (AX3343) to visually identify it.

Although the operator selects the drive, the operator need not be concerned with internal details of the mounting process. Logical linkages between device files are made automatically by the tape system on the basis of the tape characteristics (e.g., block or character access, tape speed, density, etc.) specified in the *tpmount*. (Refer to section 1.5, "Understanding the Tape Mounting Operation" for a description of the mounting process.)

In contrast to Ann, Jane only wants to mount a drive; she does not specify a particular tape. Depending on the operating procedures of your site, this could mean different things. For example, Jane may want to place her own tape on the drive once the operator has mounted the drive, or she may be relying on the operator to know what tape she wants, or she may want the operator to put a blank tape on the drive.

In any case, the operator can cause the drive to be mounted regardless of whether the tape is already on it by highlighting Jane's request and typing *select-done*. The drive selection window will open again; this time the operator will choose a drive other than mt:1, since that one is being used by Ann. (If there is only one drive on the system, Jane will—of course—have to wait until Ann is done.) Because the *tpmount* did not specify a tape VSN, the status of the message will be immediately changed to "Done," even if no tape is on the drive.

Assuming that the operator chooses drive "mt:2," the screen will look like this after the drive selection has been made:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE-------------TIME IN------- 1+
|jane   Done    Mount-Tape              Yes  mt:2 @ dragonne   3:20pm Nov  7   |
|ann    Done    Mount-Tape   AX3343     Yes  mt:1 @ dragonne   3:22pm Nov  7   |
|joe    Ready   Mount-Tape   AX1745     Yes  mt:1 @ dragonne   3:23pm Nov  7   |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
+ -------------------------------------------------------------------------------+
Command:
```

4.3.2 Mounting Without Operator Intervention: AVR

In the example above, the operator typed the *select-done* command to fulfill Jane and Ann's requests. The operator may proceed in a like fashion with Joe's request. Alternatively, the operator could rely on the Automatic Volume Recognition (AVR) capability of the tape system to complete the request.

To do this, the operator would simply place the requested tape (the one with the VSN AX1745) on the drive without first selecting Joe's request and entering *select-done*. The tape system will then automatically match the request to the VSN of the tape that it recognizes on the drive, mount the drive, and change the status of the message in the *opreq* window to "Done."

When the tape system detects the presence of a tape on a drive, a message of type "AVR" is shown in the *opreq* window (provided, of course, that the window has been configured to display this message type). Since the tape is labeled, the VSN will be shown.

The VSN of a labeled tape detected by AVR can be used to match the tape to a *tpmount* request. The system can either match the detected VSN to a queued *tpmount* (as in the case of Joe, above), or it can match a newly entered *tpmount* that specifies a VSN to the VSN of a tape that was previously recognized on a drive. When the tape system finds such a match, the drive is automatically mounted, and the corresponding *opreq* message is given a status of "Done."

AVR cannot be used to automatically mount drives with unlabeled tapes; such drives are mounted only when the operator performs a *select-done* on the mount request.

4.3.3 Cancelling Mount Requests

The operator can cancel tape mounting requests that do not already have a status of "Done," "Mounted," or "Cancelled" at any time by highlighting the corresponding message, then typing *select-cancel*. This may be done, for example, because the operator cannot, for some reason, service the request. When a message is cancelled, the STATUS entry for the message is changed to "Cancel".

If a user enters a *tpunmount* request for a *tpmount* that has not yet been done, the status of the message is likewise changed to "Cancel." If the *tpmount* has been done, *tpunmount* generates a separate message having a type "Unmount Tape."

4.4 Configuring the *opreq* Window

The *opreq* window can be "customized" to show only messages having a certain type or status, and to show only selected types of information for each message.

4.4.1 Selecting Message Fields for Display

Each message that is displayed in the *opreq* window has certain types of information associated with it. For example, every message has a user ID and a status. Each of these types of information is called a message *field*. You may not desire to have all fields displayed in the window (e.g., if the window is being shown on an 80-column display there may not be room, or you may not wish to show fields that are never used at your installation). To change the displayed message fields (and the corresponding labels in the title bar), enter the following command:

```
Command: configure-title
```

A menu appears superimposed over the left side of the window:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE-------------TIME IN------- 1+
|+---------------+ount Tape            --    mt:0 @ mach1      4:27pm Aug 23   |
||    MID        |                                                              |
||  X UID        |                                                              |
||  X STATUS     |                                                              |
||  X TYPE       |                                                              |
||    ASSIGNED   |                                                              |
||  X VSN        |                                                              |
||  X RING       |                                                              |
||  X DRIVE      |                                                              |
||    LINK       |                                                              |
||  X TIME IN    |                                                              |
||    TIME OUT   |                                                              |
||  X COMMENT    |                                                              |
|+---------------+                                                              |
+ -------------------------------------------------------------------------------+
Configure:
```

This menu shows the possible labels that can be displayed in the title bar of the window (these are also the names of all the message fields). The "X" next to the menu item indicates that the item will appear in the menu bar. The menu item that is now selected (i.e., that can currently be changed) is highlighted. Switch the selected item off or on by pressing the carriage return key; change which item is selected by pressing the "up" or "down" cursor keys.

To save the current state of the title labels and exit the menu, enter:

```
Configure: done
```

To cancel all changes you made since you entered *configure-title*, enter:

```
Configure: cancel
```

The *configure-title* command is effective only for the current *opreq* session; when *opreq* is restarted, the default labels will again be shown. (Refer to section 2.3.4, "Changing *opreq* Window Defaults," for information about making permanent changes to the window defaults.)

4.4.1.1 Description of Message Fields

Labels that can be made to appear in the title bar are listed below, along with an explanation of the kind of information contained in the corresponding message field.

MID— Message ID. Each message in the system is automatically assigned a unique identifier. The MID is provided for compatibility with future features, and need not ordinarily be displayed.

UID—User ID; shows the ID of the user who triggered the message (e.g., by issuing a *tpmount* command).

STATUS—Indicates the current status of the message. The status can be changed with the *select* command (refer to section 4.3, "Using *opreq* for Tape Operations"). Messages to be displayed in the window can be filtered on the basis of their status (refer to the description of *configure-status* in section 4.4.2.2, "Selecting Messages According to Status"). The following entries may appear under "STATUS":

- **New**—Just entered the system (messages are usually changed to a different status almost immediately).
- **Working**—The message has been assigned to an operator (with the *select work* command, or is being worked on. (See the *select* command under *opreq Commands*). The name of the operator to whom the message has been assigned will appear under "ASSIGN", if this label is being displayed in the title bar.
- **Done**—The action indicated by the message has been completed. Messages having a status of "Done" will continue to be displayed until the operator issues a *display-messages* command (refer to section 4.2.3.2, "Display Information").
- **Cancel**—The message has been canceled either by the user or by the operator. Messages that have been cancelled will continue to be displayed until the operator issues a *display-messags* command (refer to section 4.2.3.2, "Display Information").
- **Waiting**—Completion of the required action is awaiting availability of a resource. (For example, a tape mount request might be waiting for a drive to become available.)
- **Ready**—The message is ready to be serviced.

- **Mounted**—This status is shown for a *tpmount* that specified a tape VSN if the operator has requested that the status of the message be changed to "Done," but AVR has not yet verified that the requested tape is on the specified drive.
- **Remote**—Not implemented; included for compatibility with future features.

TYPE— The type of message; the following types are supported:

- **Mount Tape**—The message is a request to access a tape. If a user issues a *tpmount* command when queueing is enabled, a message of this type will be displayed. (Refer to the description of *tpmount* in section 3.1, "*tpmount* and Related Commands.")
- **Unmount Tape**—The message is a request to unmount a tape. If a user issues a *tpunmount* command, or if the system unmounts a tape (e.g., due to a timeout), a message of this type is created.
- **Auto Vol Rec**—The message has been generated by the tape subsystem to indicate that a tape has been recognized on a device by Automatic Volume Recognition (AVR). If the tape has a VSN (i.e., is a labeled tape), the VSN will be shown.
- **Replace Tape**—The message has been generated by the tape subsystem to notify the operator that a tape must be replaced (e.g., because it is part of a multivolume tape set).
- **Info**—Not implemented; included for compatibility with future features.
- **Cancel**—Not implemented; included for compatibility with future features.
- **Problem**—Not implemented; included for compatibility with future features.

Messages can be displayed selectively by type. Refer to the description of *configure-type* in section 4.4.2.1, "Selecting Messages According to Type" for more information.

ASSIGN— Operator to which the message is assigned. If the message has been assigned to a specific operator, the operator's name will appear under this label. (A message is assigned when the *select-work* command is invoked on it.)

TIME IN— The time at which the message entered the system. Messages are displayed in the order in which they were received.

TIME OUT— The time at which the message leaves the system. This time will never be displayed (the field has been included for compatibility with future features); the label should, therefore, not be shown in the title bar.

COMMENT— Optional comment for message. Any comment that the user specified along with *tpmount* will be displayed under this label.

LINK— MID of linked message. If the message is linked to another message, the MID of that message will be displayed under this label. (One mount message might, for example, be linked to another if a compound *tpmount* request specified more than one drive.)

VSN— Volume Serial Number. The VSN is given only in messages that have the following types:

- **Mount Tape**—If a VSN appears in a message of this type, that VSN was specifically requested in the *tpmount* command. If no VSN appears, the *tpmount* was only a request for access to a tape drive, not for a specific tape. In response, typically, the operator will mount the drive and permit the requester to access the drive.
- **Auto Vol Rec**—A message of this type indicates automatic volume recognition—a tape has been placed on a drive, and has been recognized by the tape subsystem. If the subsystem finds a VSN, it will be shown; if none appears, the tape has no VSN.

- **Replace Tape**—Because this message must specify a particular tape that is to be put on the drive, the message must include a VSN.

RING— This field notifies the operator whether the tape should have a "write" ring.

DRIVE— list of drives. For "mount" messages, this is the first drive found by the system on which the tape can be placed; for "Auto Vol Rec" and "Replace Tape" messages, it is the specific drive to which the message applies.

4.4.2 Selecting Message Display Criteria

The operator may set criteria that determine which messages will be displayed in the *opreq* window. Selection can be made on the basis of message *type* and message *status*.

4.4.2.1 Selecting Messages According to Type

The operator may configure the *opreq* window so that only messages of certain types appear. (Refer to section 4.4.1.1, "Description of Message Fields" for information about possible message types.) To configure the display for message type, enter:

```
Command: configure-type
```

A menu of all possible message types is then superimposed on the window:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE-------------TIME IN------- 1+
|+-----------------+nt Tape            --   mt:0 @ mach1      4:27pm Aug 23    |
||  X Mount Tape   |                                                           |
||  X Unmount Tape |                                                           |
||    Auto Vol Rec |                                                           |
||  X Replace Tape |                                                           |
||    Problem      |                                                           |
||    Info         |                                                           |
||    Cancel       |                                                           |
|+-----------------+                                                           |
+ -------------------------------------------------------------------------------+
Configure:
```

Only messages of the types that are marked by an "X" will be displayed in the *opreq* window. To select a message type for display, use the cursor keys to move the selection cursor (highlighted area) to the desired type, and then press the carriage return. To deselect a type that is marked with an "X", do the same thing.

4.4.2.2 Selecting Messages According to Status

The operator may configure the *opreq* window so that only messages having a certain status appear in the screen. (Refer to section 4.4.1.1, "Description of Message Fields" for information about possible message status.) To configure the display for message status, enter:

```
Command: configure-status
```

A list of all possible status values will then appear superimposed on the window:

```
+ UID--STATUS--TYPE---------VSN-------RING--DRIVE------------TIME IN------- 1+
|+------------+Unmount Tape            --   mt:0 @ mach1      4:27pm Aug 23    |
||    New     |                                                                 |
||    Working |                                                                 |
||    Done    |                                                                 |
||    Cancel  |                                                                 |
||    Deleted |                                                                 |
||    Waiting |                                                                 |
||  X Ready   |                                                                 |
||    Mounted |                                                                 |
|+------------+                                                                 |
+ -------------------------------------------------------------------------------+
Configure:
```

Only those messages that have a status that is marked with an "X" in this list will be shown in the *opreq* window. Select message status for display just like message type (above).

4.4.3 Changing *opreq* Window Defaults

Using the *opreq* window configuration commands affects the configuration of the window only for the current session; if *opreq* is killed and then restarted, the defaults will be restored. You may change several of the *opreq* window default characteristics (including the labels in the title bar) by copying a file called */usr/lib/opreq/.opreqrc* to your home directory, or to the directory that you are in when you invoke *opreq*. By editing this file, you can "customize" your *opreq* window.

> **CAUTION** Edit only your copy of */usr/lib/opreq/.opreqrc*. Changing the original file will change defaults for all *opreq* users; only the system administrator should edit this file.

The */usr/lib/opreq/.opreqrc* file contains *window-open*, *configure-status*, *configure-type*, and *configure-title* commands. (These commands are all described in section 4.2.3, "*opreq* Commands.") For example, the file may contain the following command:

```
configure-title uid status type vsn ring drive time-in comment
```

This command causes the listed labels to be displayed in the title bar of the *opreq* window when it opens. You may remove any of these titles if you don't want them displayed, or add others that you want shown.

When *opreq* starts, it first looks in your current directory, then your home directory for a *.opreqrc* file; if no such file is found, it will use the file in */usr/lib/opreq/.opreqrc*.

Note	*opreq* is generally run as root; if this is so, *opreq* will consider the home directory of the user to be the root directory (i.e., "/").

APPENDIX A
Glossary of Terms

The following terms are used in this publication. Note that some of these terms (e.g., "device," and "mount" are used in a restricted way in this publication to reduce ambiguity.

AVR
is an acronym for "automatic volume recognition". This is the capability of the tape system to recognize a labeled tape when it is placed on a drive.

access
(when used as a verb) means to read or write data through a device or drive, or on a tape.

allocate
is a synonym for *mount* (see below); it refers to the operation of establishing a logical link to a device file (and, by extension, to a tape drive) and changing its ownership so that the user can access it.

device
refers to a *logical device*—that is, a device file, e.g., */dev/mt4* not to a mechanical device, such as a tape drive.

drive
refers to a mechanical tape drive.

input
data read from tape.

label
is labeling information recorded on the tape; this is also sometimes referred to as a "magnetic label." References to paper labels on the tape reel will be made explicit, i.e., "paper label".

logical drive
refers to an entity defined with *tpconfig* that should correspond to a mechanical drive; a logical drive is composed of a group of *nodes* (see below).

mount
refers to the operation of establishing a logical link to a device file and changing its ownership so that the user can access it. By extension, tape drives are also said to be "mounted" when a device that controls them is mounted, and a tape on such a drive is likewise "mounted". To avoid confusion, the word "mount" is not used in this manual to refer to the mere act of placing a tape on a drive, although popular use of this word encompasses this meaning also.

node
is a device that has been defined (via *tpconfig*) to be associated with a logical drive.

output
data written to tape.

VSN
is an acronym for "volume serial number" and is an identifier for a tape. A VSN may be "paper"--i.e. an identifier recorded on the paper label on a tape for the convenience of the operator, or it may be "magnetic". Magnetic VSNs are part of the information recorded in the label of ANSI-labeled tapes.

APPENDIX B
Command Quick Reference

This appendix contains a summary of tape system commands intended for quick reference. For detailed information about these commands, consult the appropriate sections in this manual or the *ConvexOS Programmer's Reference.*

tpconfig Commands

The following commands are recognized by *tpconfig*:

Table B-1: *tpconfig* Command Summary

Add Allocate_drive Group_set *<group_list> <type:unit>*	Add groups to drive allocation list
Add Allocate_drive User_set *<user_list> <type:unit>*	Add users to drive allocation list
Add Bypass_labels Group_set *<group_list> <type:unit>*	Add groups to label bypass list
Add Bypass_labels User_set *<user_list> <type:unit>*	Add users to label bypass list
Add Drive *<type:unit>* [Nocontrol] [Timeout=*<N>*]	Define new drive
Add Label *<type> <path>*	Define new label type
Delete Allocate_drive Group_set *<group_list> <type:unit>*	Delete groups from drive allocation list
Delete Allocate_drive User_set *<user_list> <type:unit>*	Delete users from drive allocation list
Delete Bypass_labels Group_set *<group_list> <type:unit>*	Delete groups from label bypass list
Delete Bypass_labels User_set *<user_list> <type:unit>*	Delete users from label bypass list
Delete Drive *<type:unit>*	Remove drive
Delete Label *<type>*	Delete label type
Delete Node *<path>*	Remove node
SEt Allocate_drive Group_set *<group_list> <type:unit>*	Replace group drive allocation list
SEt Allocate_drive User_set *<user_list> <type:unit>*	Replace user drive allocation list
SEt Bypass_labels Group_set *<group_list> <type:unit>*	Replace group label bypass list
SEt Bypass_labels User_set *<user_list> <type:unit>*	Replace user label bypass list
SEt Control ON|OFf *<type>:<unit>*	Toggle drive control status
SEt Default DEnsity *<density>*	Set default density for *tpmount*
SEt Default DRive *<type>*	Set default drive for *tpmount*
SEt Default Flags [Rewind|Norewind] [Character|Block|Labeled]	Set default flags for *tpmount*
SEt No_default Density	Eliminate default density for *tpmount*
SEt No_default Speed	Eliminate default speed for *tpmount*
SEt Queueing Enabled|Disabled	Toggle *opreq* on or off
SEt Timeout *<N> <type:unit>*	Change timeout for drive
SEt Default Speed *<speed>*	Set default speed for *tpmount*
Show All	Display all configuration values
Show DEfaults	Show default values for *tpmount*
Show DRive [*<type:unit>*]	Show information about specified drive
Show Labels	Show defined label types
Show Node [*<path>*]	Show information about specified node
Snapshot [*<file>*]	Store present configuration in file

opreq Commands

The following commands are recognized by *opreq*:

Table B-2: *opreq* Command Summary

configure-status	Choose messages to be displayed by status
configure-title	Change labels in title bar
configure-type	Determine what types of messages are displayed
display-avail-drives	show list of available drives
display-<*message_field*>	Show specified message field
display-messages	Refresh message display
move-down	Move selection cursor down
move-up	Move selection cursor up
select-cancel	cancel highlighted message
select-done	Change status of message to "done"
select-work	Assign message to operator
window-close	Close *opreq* widow
window-goto	move between *opreq* windows
window-open	Open new *opreq* window

mt Commands

The following is a list of the *mt* commands:

Table B-3: *mt* Commands

mt Command	Unlabeled Mode	Labeled Mode
eof <*n*>, weof <*n*>	write <*n*> end of file markers	write <*n*> empty ANSI files
fsf <*n*>	forward space <*n*> files	forward space <*n*> logical files
fsr <*n*>	forward space <*n*> records	forward space <*n*> logical records
bsf <*n*>	back space <*n*> files	back space <*n*> logical files
bsr <*n*>	back space <*n*> records	not allowed
rewind	rewind tape	rewind tapeset (leave at first ANSI file)
offline, rewoffl	take tape offline	not allowed - use *tpunmount*
gap <*n*>	write <*n*> interrecord gaps	ignored
status	return tape status	return (physical) tape status

APPENDIX C
Error Message Reference

This appendix contains a description of error messages that can be returned by the tape system. Since it is impossible to predict all possible combinations of circumstances that could cause error messages, this list cannot be exhaustive. Only those error messages that are likely to be given and that serve as helpful diagnostic indicators for users are listed in this appendix. If you encounter error messages that are not described, this indicates a serious internal error. In that case, please contact the CONVEX Technical Assistance Center (TAC).

These error codes are either shown on-screen or are sent to the error log file (usually */usr/adm/log/tapelog*), or both. The error codes for the programmatic tape system interface that correspond to these messages and that are returned by the tape system library functions are not documented in this appendix. To see these codes, examine the */usr/include/tape.h* file.

Messages recorded in the */usr/adm/log/tapelog* file will usually be labeled to indicate their severity. The following labels may be used:

EMERG	system is unuseable
ALERT	action must be taken immediately
CRIT	critical conditions
ERR	error conditions
WARNING	warning conditions
NOTICE	normal but significant condition
INFO	informational; no action required
DEBUG	for debugging only; should not normally be seen

General Tape System Error Messages

The following error messages may be sent to your terminal or to the log file by the tape system. When you see them on your screen, these messages will have the format

`<command>:<error message>`

where *command* is the tape system command that caused the error message. For example, if a user who is not *root* attempts to unmount a tape mounted by *root*, the following message will be shown:

```
% tpunmount: no authorization
```

This indicates that the *tpunmount* command failed, and that the reason for this failure was a lack of authorization to unmount the requested device.

Messages are listed in alphabetical order.

`ambiguous request`

Meaning: The user has more than one active tape request, and has issued a command that did not satisfactorily specify the request to which it pertains. For example, if the user has mounted several devices and then issues a *tpunmount* without parameters, the system will not know which device to unmount. In this case, reissue the command and use *-s* to give the symbolic link path.

`can't create symbolic link`

Meaning: The symbolic link to the tape device could not be created. The reason for the failure is appended. For example:

```
tpmount: can't create symbolic link
/mnt/guest/TAPE: File exists
```

In this case, a symbolic link called "TAPE" (the default link name) already exists; use *-s* with *tpmount* to specify a different link name.

`can't get current working directory`

Meaning: The *getwd(3)* call failed. This will happen if the current working directory doesn't exist.

`device is opened by another process`

Meaning: The user attempted to unmount a device (using *tpunmount*), but a process still has the device open. That process must close the device before it can be unmounted.

`empty request`

Meaning: The tpmount request didn't ask for any tapes. (This message can only be given in response to a *tpmount*() library call.)

`internal tape system error`

Meaning: Report the problem to CONVEX Technical Assistance Center (TAC); include a copy of the error log file (usually */usr/adm/log/tapelog*) and the command that generated the error.

`invalid drive type`

Meaning: The drive type specified with *tpmount* (or the default drive type, if the default is used) does not match any defined drive type.

`invalid label type`

Meaning: The label type specified with *tpmount* is invalid, or the label daemon (defined with *tpconfig*) doesn't exist.

`invalid path to symbolic link`

Meaning: The path to the user's symbolic link has been removed.

`magnetic vsn doesn't match paper vsn`

Meaning: The VSN in the VOL1 header label on the tape doesn't match the VSN that was requested with *tpmount*.

`mount not yet complete`

Meaning: The *tpmount* has not been completed yet. Enter *tpwait* to wait for mount.

`mount request failed - tape is labeled`

Meaning: The user tried to mount a labeled tape in block or char mode (and didn't use the label bypass flag).

`no authorization for request`

Meaning: An unauthorized action has been attempted. (For example, a user who does not have the requisite privileges attempted to mount a drive without specifying a VSN, or a user attempted to dismount a device mounted by another user.)

`no drives meet the required speed, density, and type`

Meaning: Several of the *tpmount* options restrict the list of devices that can meet the users requirements. This error is returned if none of the devices match all of the requirements given by the user. These options include:

-d density
-i speed
-m mode
-b bypass label mode
-a device
-r automatic rewind on close
-t drive_type

`no entry in password file`

Meaning: The user doesn't have an entry in the */etc/passwd* file.

`no matching drives are available at this time`

Meaning: There is at least one device that meets the requirements specified in *tpmount*, but all of them are busy now and queueing is disabled.

`no such request`

Meaning: The specified tape request doesn't exist. This message would be seen if a user enters a *tpunmount* command, and specifies a symbolic link (with *-s*) that has no corresponding *tpmount* request. Enter "tpqueue -l" to get a list of active *tpmount* requests.

`queueing is disabled - please specify device with tpmount -a option`

Meaning: A tape was specified with the tpmount command (by VSN); queueing is disabled and the other tpmount parameters match more than one drive. The user must specify which drive the tape is on by giving more tpmount arguments (usually *-a*).

`server didn't respond`

Meaning: *tpdaemon* didn't respond to your request. This message is always followed by an additional RPC (remote procedure call) error message that explains why the call failed.

`symbolic link used by another request`

Meaning: The symbolic link specified in a *tpmount* command (or the default— */TAPE*) is the same as is already being used by another tape request. The symbolic link path must be unique for each request.

`tape cannot be labeled`

Meaning: The tape cannot be labeled. The most likely reason is that there is no tape on the drive.

`tape cannot be unlabeled`

Meaning: The tape cannot be unlabeled. The most likely reason is that there is no tape on the drive.

`tape device must be no-rewind for labeled tapes`

Meaning: The user attempted to mount a tape using label mode and specified an automatic rewind device with the tpmount -a option. Only no-rewind devices can be used in labeled mode.

`tape drive is not on-line or not ready`

Meaning: The user attempted to mount a tape in label mode, or to label/unlabel a tape, and the drive is either not on-line or not ready.

`tape is already labeled`

Meaning: An attempt was made to label a tape that is already labeled; use *tpunlabel* to unlabel the tape first.

`tape is already unlabeled`

Meaning: An attempt was made to unlabel an unlabeled tape.

`tape mount canceled by operator`

Meaning: Queueing is enabled and the operator cancelled the *tpmount* request via *opreq*.

`tape must be mounted as labeled to use tplabel/tpunlabel`

Meaning: The user attempted to *tplabel* or *tpunlabel* a tape that was not mounted in label mode.

`unable to connect to server`

Meaning: It was not possible to establish a connection with *tpdaemon*. This message is always followed by an additional RPC (remote procedure call) error message that explains why the call failed. One common reason for this message is that a tape system command was issued when *tpdaemon* was not running. *tpdaemon* must be running to use the tape utilities.

`Warning: tape is not labeled; use tplabel to label tape`

Meaning: This warning is shown when a unlabeled tape is mounted in label mode. It does not indicate an error condition, but is meant to remind you to use *tplabel* to label the tape.

tpconfig Error Messages

The following error messages may appear on your screen while you are running *tpconfig*.

```
Ambiguous command.
Ambiguous keyword match
```

Meaning: Not enough characters were given to distinguish the command or keyword from another command or keyword. Enter a more complete form of the command.

```
Cannot create new database.
```

Meaning: The database */usr/lib/tape/config.db* does not exist, and you do not have permission to create a new database. Make sure that you are logged on as *root*.

```
Cannot open database: /usr/lib/tape/config.db
```

Meaning: The indicated database could not be opened. Check the access permissions of the database with *ls*(1).

```
Database access is read-only.
```

Meaning: You do not have permission to make changes to the tape system configuration database. (Only *root* has permission.) You may, however, read information from the database. Only the *show* commands work when the database is read-only.

```
Database does not exist. Creating new database.
```

Meaning: Because */usr/lib/tape/config.db* did not exist, a new database is being created.

```
Database is readonly.
```

Meaning: The user issued a command to change the configuration but only has read-only access. (Only *root* has permission to change the database.)

```
Error reading defaults
Error reading drives
Error reading labels
Error reading node
```

Meaning: The configuration database is corrupt. Check for other error messages. The problem can be fixed by removing the old configuration file and creating a new one.

```
Error writing defaults
Error writing drives
Error writing labels
Error writing nodes
```

Meaning: A write error occurred while writing out a new configuration. These messages are always accompanied by a system error message.

```
Interrupt.
```

Meaning: The keyboard interrupt character (usually <CONTROL-C>) was typed. If you want to exit *tpconfig*, type the EOF character (usually <CONTROL-D>).

```
Missing quote.
```

Meaning: Somewhere on the command line there is a mismatched quote character.

```
Node must be either block special or character special.
```

Meaning: Only character special files and block special files can be given as a node in the *add node* command.

```
readdrives: malloc failed
readlabels: malloc failed
readnodes: malloc failed
```

Meaning: There is not enough memory available for *tpconfig*.

```
readnodes: stat failed on /dev/rmt?
```

Meaning: One of the nodes defined in the database no longer exists. You should delete that node with *delete node*.

```
The database is being modified by another person.
Access is read-only.
```

Meaning: The configuration database is locked, probably because another user is running *tpconfig*. Only one user can run *tpconfig* at any one time.

```
Unable to write out configuration.
```

Meaning: *tpconfig* could not write the configuration back to */usr/lib/tape/config.db*. Verify that you still have write permission and that the disk is not full.

```
Warning: Invalid daemon path.
```

Meaning: The daemon specified with *add label* does not exist.

opreq Error Messages

The following error messages may be displayed on your screen or sent to the log file to indicate error conditions while you are using *opreq*.

`At limit`

Meaning: An attempt was made to move past the edge of the screen.

`Invalid command`

Meaning: A command was issued that *opreq* does not recognize.

`Invalid drive`

Meaning: The selected drive cannot be mounted.

`Invalid window`

Meaning: An attempt was made to select a window that is not available.

`No current message`

Meaning: A message operation has been attempted, but no message is currently selected. Move the selection cursor over the desired message.

`No current window`

Meaning: An operation was attempted that requires a window to be active, but no window is active.

`No other window`

Meaning: An attempt was made to move to another *opreq* window; no other window exists.

`Not implemented`

Meaning: You have attempted to use an unimplemented command or feature.

`Not ready`

Meaning: A *select-done* command was attempted on a message that is not in "Ready" status.

`Restricted command`

Meaning: You do not have permission to carry out the command. (To use its full capabilities, *opreq* must be running as *root*.)

`Tape server: opreply() failed`

Meaning: Internal error: an attempt to send a message via *opreply*() has failed.

`Tape server: timed out`

Meaning: The tape drive has not been used for longer than the defined time-out period; it has been unmounted.

APPENDIX D
Reporting Problems

This appendix introduces the CONVEX Technical Assistance Center (TAC) and *contact* utility. The *contact* utility is an online system for reporting problems to the TAC. To learn *contact* by using it, enter **contact** at the system prompt and then answer the questions as they appear on the screen. To find out more about using *contact*, read through this appendix. It describes prerequisites and tips for using *contact* and the step-by-step process *contact* takes you through.

D.1 Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation question, contact the TAC. This group stands ready to solve such problems.

D.2 The *contact* Utility

The TAC recommends using the *contact* utility to report a hardware, software, or documentation problem. The *contact* utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke *contact*, it prompts you for information about the problem. When you finish your report, *contact* electronically mails it to the TAC. You are notified within 48 hours that the TAC has received your report.

D.3 Prerequisites

To use *contact* requires

- a UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- full path name of the program or utility in question
- version number of the program or utility in question

D.3.1 UUCP Connection

Before using *contact*, check with your system administrator to be sure there is a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX-based system to another. The *uucp* (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

D.3.2 Finding the Program Path Name

To determine the full path name of the program or utility in question, use the *which* command. The next screen illustrates using the *which* command to find the full path name of the loader (*ld*) utility.

```
>which ld
/bin/ld
>
```

In this example, the full path name of the loader is */bin/ld*.

For more information on the *which* command, refer to the *which*(1) man page. You can also use the *info* online information system. Enter **info which** at the system prompt.

If you use the C shell (*csh*), you can also use the *whence* command to find the program path name. The *whence* command works like *which*, but faster.

D.3.3 Finding the Program Version Number

To determine the version number of the program or utility in question, use the *vers* command. The next screen illustrates using the *vers* command to find the version number of the loader (*ld*) utility. Enter **vers**, then the path name of the program or utility.

```
>vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader version number is 7.0.

For more information on the *vers* command, refer to the *vers*(1) man page. You can also use the *info* online information system. To do so, enter **info vers** at the system prompt.

D.4 Tips on Using the *contact* Utility

The *contact* utility is interactive and easy to use. This section lists tips to help use it efficiently. In particular, this section tells how to

- use a *.contact* file
- abort a contact session
- resubmit an aborted report
- suspend a contact session
- move from one prompt to another
- use tilde-escape sequences in the *contact* utility

D.4.1 Using a *.contact* File

When you invoke *contact*, it prompts for information regarding the problem. The first prompt is for your name, title, phone number, and company name. You can, however, create a *.contact* file to skip this first prompt. Follow these steps:

1. Create a *.contact* file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke *contact*, it automatically includes the *.contact* file as input for the first prompt and proceeds to the next prompt.

D.4.2 Aborting the Report

To abort a contact report, either enter the interrupt key (usually CTRL-C) or choose the abort option when prompted by the *contact* utility. Using CTRL-C to abort does not save the contents of the report. Using the abort option saves the contents of the report in a file named *dead.report* in your home directory.

D.4.3 Submitting the *dead.report* File

After you abort a contact session, the *contact* utility saves the report in a file named *dead.report* in your home directory. Using the *contact* command with the *-r* option automatically merges the contents of the *dead.report* file into the new contact session. Enter

contact -r

and *contact* finds the *dead.report* file in your home directory and merges it into the contact report. You can then edit the report. When you end the editing session, *contact* returns to the final prompt, which asks you to review, edit, submit, or abort the report.

D.4.4 Suspending a Report

Sometimes it is necessary to stop in the middle of a contact report and return to the shell (for instance, to suspend the contact session to find the program path name or version number). To suspend the contact session, press CTRL-Z. To return to the contact session, press **fg**. Using CTRL-Z and the *fg* (foreground) command lets you toggle back and forth between the *contact* utility and the shell. You cannot, however, use CTRL-Z and *fg* to toggle back and forth if you are using the Bourne shell (*sh*).

D.4.5 Ending a Response

The *contact* utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press RETURN. Other prompts require more than a one-line response; to move to the next prompt, press CTRL-D.

D.4.6 Tilde-Escape Sequences

The *contact* utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. The following tilde sequences are recognized by *contact*:

~e	start the text editor (defined in the EDITOR environment variable)
~h	display a list of available tilde-escape sequences
~p	print the contact report to the terminal screen
~r *filename*	read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence works only for prompts that allow more than a one-line response.
~~	insert a single tilde as the first character in the line

D.5 Using the *contact* Utility

The *contact* utility prompts for the following information:

- your name, title, phone number, and corporate name
- name and version of the product
- one-line summary of the problem
- detailed description of the problem
- priority of the problem
- instructions on how to reproduce the problem
- comments about the problem
- comments about the documentation relating to the problem
- files to include in the contact report

The following is a step-by-step discussion of these prompts.

Step 1a To invoke the *contact* utility, enter **contact** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software or documentation question. The next screen illustrates the *contact* command and the system response.

```
>contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

Step 1b If there is a *.contact* file in your home directory, *contact* skips the first prompt. The next screen illustrates the *contact* command and the system response when a *.contact* file is in your home directory.

```
>contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

Step 2 The *contact* utility prompts for the version number of the product. If you do not know the version number, use CTRL-Z to suspend the session. Use the *which* (or *whence* if you use *csh*) and *vers* commands to find the version number of the product. Use the *fg* command to return to the session and enter the version number in the form X.X or X.X.X.X.

Step 3 The *contact* utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Please make this summary as descriptive as possible in one line.

Step 4 The *contact* utility prompts for a detailed description of the problem. Please make this description as complete as possible. Include source code and a stack backtrace when possible. (Refer to the *adb*(1) or *csd*(1) man page for information on obtaining a stack backtrace.) The more information you provide, the quicker the TAC can isolate and solve the problem.

Step 5 The *contact* utility prompts for the priority of the problem. The next screen illustrates this prompt and priority levels from which to choose; you must enter a priority number.

```
Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.
>
```

Step 6 The *contact* utility prompts for an explanation of how to reproduce the problem. Please include the command syntax and options you used and anything else you did to make the program run.

Step 7 The *contact* utility prompts for any other pertinent comments. Please include all relevant information.

Step 8 The *contact* utility prompts for suggestions regarding documentation supporting the product. Indicate whether the documentation could be revised to address the problem.

Step 9 The *contact* utility asks for names of files necessary to reproduce the problem. The next screen illustrates the *contact* prompt and sample user response.

```
Are there any files that should be included in this report (yes | no)?
>yes
Please enter the names of the files, one to a line (^D to terminate)
>test.f
>~/subroutines/sub.f
>
```

Note

Tilde-escape sequences are not recognized in responses to this prompt. In *contact*, a tilde in this section means your home directory. This convention is based on use of the tilde for expanding file names in *csh*.

If files specified are small text files, they are automatically included in the contact report. If the files are too large to be included in this report, *contact* gives further instructions on how to submit these files.

To specify a directory, combine directory files into a single file using the *tar* command (refer to the *tar*(1) man page for further information) or enter each file name in the directory on a single line in the contact report.

Step 10 The *contact* utility prompts you to review, edit, submit, or abort the contact report. The next screen illustrates this prompt.

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

Review — review the text of the contact report. You are then prompted again to select an option.

Edit — edit the text of the contact report. If you choose to edit the report, *contact* puts you in your default text editor.

Submit — sends the report to the CONVEX TAC. You are notified within 48 hours that the TAC has received the report. This option exits the *contact* utility and returns you to the shell.

Abort — saves the text of the report in a file named *dead.report* in your home directory. This option exits the *contact* utility and returns you to the shell.

Index

ConvexOS Tape System Guide
Document No. 710-003130-000, First Edition

Reader's Forum

Please use this form to submit comments or questions concerning the clarity and usefulness of this manual. Constructive critical comments are most welcome and help us continue in our efforts to generate quality customer documentation. Please list the page number for questions or comments.

_____

_____

_____

_____

_____

_____

_____

_____

From:

Name_____ Title_____

Company_____ Date_____

Address and Phone No._____

FOR ADDITIONAL INFORMATION OR DOCUMENTATION:

Location	Phone Number
From all locations in continental U.S.	1(800)952-0379
From locations in Alaska, Hawaii, & Canada	1(214)497-4379
From all other locations	Contact nearest CONVEX office

Direct mail orders to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

(Fold Here First)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

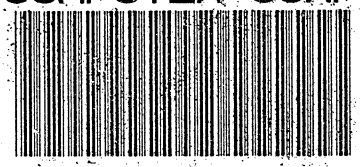
CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851

(Fold Here Second)

(Tape or Staple)

DSW-016

CONVEX COMPUTER CORPORATION



CONVEXOS TAPE SYSTEM GUIDE
710-003130-000

PRINTED IN U.S.A.