BRU User's Guide

Copyright © 2003-2013, TOLIS Group, Inc.

Table of Contents

Chapter 1 - Welcome

General Description

BRU is a Backup and Restore Utility with significant enhancements over other more common utilities such as `tar`, `cpio`, `volcopy` and `dump`. BRU is designed for maximum reliability. Dozens of options give it the flexibility to perform almost any kind of backup. It will work with most backup devices, including cartridge, 4mm DAT, 8mm (Exabyte), DLT, and LTO tape drives. It is also ideal for data distribution or software updates. **NOTE**: We refer to "BRU" in this manual for convenience. Please note, that the actual executable file is named "`bru`."

BRU Features

Incremental and Differential Backups

BRU can back up all files created or modified since your most recent backup (or any date you specify) both quickly and easily. Backups can be limited to a given directory or mounted filesystem. For example, BRU can easily back up files in the root filesystem without descending into any of the filesystems mounted on directories in the root filesystem.

Full Backups

Full backups are simply a special case where all files in a given directory hierarchy are backed up. Full backups include all files regardless of their previous backup status. Full backups are the master against which incremental or differential backups are performed.

Multi-volume Archives

Archives created using BRU can span more than a single piece of removable media (such as floppy disks and magnetic tape). When the end of the volume is found, BRU asks you to load the next volume. This volume can be on a different drive if you wish, or you can allow BRU to take the default and use the same drive on which the previous volume was mounted. This multi-volume archive feature allows you to archive files that are larger than the capacity of the media on which the archive is being stored. For example, you might archive a 5 Megabyte database file on several 1.44 Megabyte floppy disks.

It is also possible to tell BRU to use multiple destinations automatically, in a cascade fashion, to create an archive where the multiple volumes are mounted on different physical devices. BRU can select and switch to another output device without operator intervention.

Distribution and Updates

Because it can verify that files written to an archive are complete and intact, BRU is ideal for distributing data or software updates. In addition, BRU can reread a previously recorded archive at any time and verify its internal consistency and data integrity. As of this writing, none of the standard backup utilities available with the UNIX operating system provide anywhere near the level of data integrity checks provided by BRU.

Error Detection and Recovery

Unlike other standard UNIX utilities, BRU is specifically designed to be robust in dealing with errors. Through the use of checksums, redundant information recording, and proprietary error recovery algorithms, BRU is capable of detecting and recovering from a very high percentage of potential data recovery problems.

Customization Features

When BRU runs, it uses an external table of devices (`brutab`). This table can be modified to describe the capabilities and error responses of your own system's archive devices. The first entry in the table is the default archive device. If no device is specified on the command line, this is the device BRU uses. The BRU device table makes it easy for you to customize BRU for a variety of different systems. It also lets you simplify the command line when only the default device is to be used.

When the external table is properly configured, BRU can recover from such common user errors as attempting to use an unformatted floppy disk, inserting a floppy disk upside down, or loading an archive tape out of the expected sequence.

Random Access Capabilities

When the archive device supports random access, BRU uses this feature to greatly decrease the access time to any given file in the archive. The performance differences between BRU and other UNIX utilities can be dramatic when reading archives or searching for a given file in an archive.

Tape drives do not have random access capabilities. However, for tape devices that support quick file access (QFA), BRU 17.0 provides QFA support to speed the process of restoring files. This support is available on selected platforms including Linux and FreeBSD. Check `www.tolisgroup.com/currentbru3.html` for up to date supported platform information.

Special Files

BRU will save and restore all types of filesystems and files with their proper ownership, access attributes, creation dates, and modification dates. BRU can be used to move an entire directory hierarchy from one system to another, with all files, including directories, block special files, character special files, fifos, hard links, and symbolic links reproduced with all attributes intact.

File Comparisons

BRU can compare the contents of an existing archive with the current online files, reporting all those online files that have been modified, have had their attributes changed, or have been removed since the archive was created. For example, using a reference archive of a standard UNIX distribution, BRU can be used to detect which files have been changed or are missing. This is a valuable capability when you encounter system problems and you suspect that the cause is corrupted or missing system files.

File Overwrite Protection

By default, when extracting files, BRU will not overwrite existing online files with older files of the same name. This feature can be overridden when necessary.

File Transport To or From Other Systems

Data storage options may differ on any two given systems. When you are transporting files to or from other systems, BRU will automatically perform byte swapping or word swapping as necessary when the archive is read on the target system. Regular files can easily be moved from one type of system to another. Special files and symbolic links can cause problems on the different target system, however, and transporting such files between different types of systems should probably be avoided.

Increased Speed

BRU has been constantly improved to make it faster. However, on most newer UNIX platforms, the backup speed is limited by the archive device (normally a tape drive) or your network, rather than BRU. This is because the newer hard drives and processors can deliver data much faster than the tape drive can write it to tape. On these systems, BRU can usually be tuned to run at the maximum speed of the tape drive. Of course, the speed will vary, depending on the tape drive, device driver, buffer size, operating system version, CPU speed, I/O subsystem, filesystem performance, kernel parameters, and (in some cases) network performance), among other things. There is no guarantee that it will be faster on your particular system.

Device Name Checking

The current version of BRU checks to see if the specified device name makes sense. This prevents the creation of files in the `/dev` directory. Various other “sanity” checks are performed to prevent users from writing to a device that does not exist.

With many other tools, if you specified a device that did not exist, the tool would create a file with the name you specified. If you were to mistype the destination as `/dev/rmt/on` (instead of `0n`), the backup could be written to a file on the hard disk called `/dev/rmt/on`, rather than the tape drive at `/dev/rmt/0n`, often causing the hard disk to run out of space.

Conventions Used in This Manual

Typographical Conventions

Several typographical conventions are used throughout this manual to make it easier to distinguish between what the system prints on the screen and what you should enter. Words you type, or that appear on the screen as a result of typing a command, appear in a typewriter style font. For example:

```
$ ls /bin/a*
/bin/acctcom
/bin/adb
/bin/ar
/bin/as
/bin/asa
$
```

The “`$`” in the above example is a shell prompt; the remainder of the line is the command you typed. The lines that do not begin with “`$`” are lines that were produced by the typed command.

When they appear in the text, the names of UNIX commands, directories, and files are also set in a typewriter font. The above example uses the UNIX `ls` command and lists several files in the `/bin` directory, including `/bin/adb`.

When special terms are first used, they appear in bold type. For example, “**place-holders**”, in the following paragraph.

Words printed in italics are **place-holders** or **variable names**. They are words that you are to replace with actual values. For example, when the manual uses the string files in the description of a command, you might replace files with /bin/echo /etc/passwd

Command Conventions

Examples of command line entries typically show all options specified separately:

```
bru -c -vvv files
```

However, it is perfectly acceptable to place options that have no required parameters together as part of a single option specifier. In this case, the above example would appear as follows:

```
bru -cvvv files
```

When an option accepts a parameter, for clarity, we show the option and the parameter separated by whitespace:

```
bru -c -o /usr/user1 files
```

In this example, we are creating an archive for a user, and the `-o` option identifies the user. Since BRU can tell what follows the option specifier is the option’s parameter, this command could just as easily have been written as:

```
bru -c -o/usr/user1 files
```

For readability, we avoid using the second form in this manual. But be aware that BRU accepts both forms as legal command options. Many users have come to expect this capability from UNIX. In some places, an option may be indicated as:

```
-o ( name | file | uid )
```

The parentheses, “(“and“)”, mean that a parameter is required. The “`|`” means that any one of the indicated parameters may be supplied, but only one. In this case, either a user name, the name of a file, or a decimal user ID may be supplied.

Definitions and Concepts

archive

Used synonymously with backup. When you back up files you create an archive containing copies of the files. Archive is used both as a verb and a noun.

extract

To restore from an archive.

full backup

A backup of all files on the system. Typically, this includes system files, system programs, and temporary or work files.

incremental backup

An incremental backup copies (backs up) all files that have been modified (changed) since a given date's given backups. A combination of full system backups, usually at weekly intervals, and daily incremental backups is a common backup scheme. Backups are smaller and therefore less time consuming than full backups to run and monitor. They also use less space on whatever media you use.

differential backup

A differential backup includes all files that have been modified or created since the last full backup was performed. Differential backups differ from incremental backups in that files will be backed up multiple times even if they have not been modified since the previous differential backup.

partial backup

A backup of only parts of the system. Typically a partial backup will not back up system files, the files that contain system programs, work files, etc. The files that are to be backed up may be specified by typing file names on the command line, by reading in a file that lists the files you want to back up, or by telling BRU to look at the `bruxpat` file. See Chapter 4, "Archiving Files: The BRU Backup Function," and Chapter 5, "The `bruxpat` File." An incremental backup is a type of partial backup.

restore

Extract files from an archive.

Contacting TOLIS Group

On-line Help

Once you have installed BRU, you may get a quick reference page describing the various BRU command line options by entering the following command:

```
bru -h | more
bru -h | less (BSD and Linux machines)
```

or

```
bru -h | pg (System V machines)
```

This prints to the screen the same information that you'll find in Appendix B, "The BRU Help Command." Since it is sometimes difficult to remember the letters that represent specific parameters, this quick help summary may help.

How To Get Help

We're here to help if you have a problem with BRU that you cannot solve yourself. Before contacting us, please do the following:

- Write down or print out the exact command that you entered when the problem occurred.
- Write down or print out any error messages or warnings. To help us identify the exact error, please include any error numbers.
- Have your BRU serial number, version, and platform information available. This can be found by executing:

```
bru -h | tail -23
```

If you do not have this information, you may be able to find it in the BRU execution log (the file `/var/log/bruexeclog`). This file keeps a record of all BRU commands and errors. It is very helpful if you can send us a copy of the portion that contains the errors by FAX or electronic mail.

Contacting TOLIS Group

Support Phone: (480) 505-1814 (8 a.m. – 5 p.m. MST, Monday - Friday)

FAX Assistance: (480) 505-0492

Internet:

```
support.bru.com
brusales@tolisgroup.com
```

On the web:

```
http://www.tolisgroup.com/
```

Chapter 2 - Installation and Getting Started

Overview

This chapter describes the steps required to install BRU, special system requirements, and tips on getting started using BRU quickly. **NOTE**: Your system may not be covered by the standard installation instructions described here. If this is the case, a separate set of MANUAL INSTALLATION NOTES is provided later in this chapter.

If you are new to UNIX, some of these instructions may seem complicated, but it is difficult to make the installation any easier (it would be easier if UNIX were simpler and all tape drives were the same). If you have any comments or ideas for improvement of this installation process, TOLIS welcomes your input.

Please read through the entire installation instruction section **BEFORE** you install BRU.

If you have difficulty during installation, please contact the TOLIS Group. See "How To Get Help," at the end of Chapter 1, "Introduction." This section outlines the information you will need to provide, where to find it, what our telephone and fax numbers are, and how to reach us using e-mail and the web.

Before You Begin

A few important pieces of information are needed before you begin installation. The necessary items are described below. For convenience (and future reference), record the information in the space provided:

The device names for your tape drive. This varies widely, but is often a name like `/dev/st0, /dev/rmt0, /dev/rStp or /dev/rmt/8`. The BRU install process does it's best to guess the proper device names, however there is a chance that the installer may not guess correctly. If you are unsure of the device name, check with the vendor who sold you the UNIX system or who set up your tape drive.

Enter the rewinding tape drive device name:

____

Enter the non-rewinding tape drive device name:

____

The device name, mount point, and SPECIAL flags for your CDROM drive. Devices are usually `/dev/cdrom, /dev/scd0, /dev/dsk/s0d0t6s0` or such. SPECIAL flags required may include the filesystem type (`-F ISO9660, -t iso9660, -f CD`) and arguments like '`no_root_squash`'. See your OS documentation or contact your OS provider for the proper information. **NOTE**: Many UNIX systems provide an "automounter" that recognizes when you insert a CD and automatically mounts it. If yours does this, please make sure that the mount does not set the "`noexec`" flag. With "`noexec`" set on a mounted CDROM, you will not be able to execute the BRU install programs.

Enter the device name:

____

Enter the mount point:

____

Enter SPECIAL flags for mounting a CDROM

____

If you are installing from a downloaded file or other media, you will need to know the filename or the device name for this alternate media. If you are installing from a tape, the installation device name is the same as the tape drive device name you entered above. If you are installing from a floppy, use the device name that corresponds to your floppy drive. Floppy device names vary, but usually have names like `/dev/fd0`, `/dev/fd096`, `/dev/install`, `/dev/fd1135ds9`, or `/dev/dsk/f0q15dt`.

Enter the installation file or device name:

____

Locate and record the BRU license information. This is provided on the Registration Card included with your product, or from an e-mail if the product was downloaded.

Enter the 'License Data:' from your Registration Card or e-mail:

____

Enter the 'License Key:' from your Registration Card or e-mail:

____

This information will be needed later during the installation procedure.

Using the Standard Installation Process

This process runs directly from the CDROM installation media.

Mount the installation CDROM using

```
   mount -r FLAGS CDROM_DEVICE MOUNT_POINT
```

The `FLAGS`, `CDROM_DEVICE`, and `MOUNT_POINT` are from your notes above. Make the `MOUNT_POINT` your current working directory with `cd MOUNT_POINT`

Run the BRU install program: `./install`

The README file provides any last minute instructions that did not make it into this manual. It is a good idea to review it for any changes in this procedure before continuing. You will be asked for your license information. Enter it from above or your registration card. **NOTE**: If you are installing BRU as a DEMO, you should hit `[ENTER]` at each of these prompts.

The End User License Agreement (EULA) for your product will be displayed. Please review this carefully as it defines legal use of the licensed product and limits the liabilities of TOLIS Group, Inc. You must agree to this license to continue the installation. The installation will proceed automatically from this point.

If you are updating a previous BRU installation, you will be asked if you would like to replace your existing `/etc/brutab` file. Unless you have changed your devices since you last edited this file, answer 'No' and leave the existing `brutab` file in place.

For new installations, you will be asked questions about the device(s) you want BRU to use (you may have multiple). Select the device type from the menu presented and follow the prompts. When prompted, compare the device name presented to the tape device names recorded above. If they match, you may just hit `[ENTER]` to accept the default entry listed. If they do not appear correct, enter the proper names providing the full path to the device (i.e.: `/dev/nst0`, not just `nst0`).

You will be asked for a description for the device. This can be anything string, including spaces, that you would like to use to easily recognize the device in the interface menus. **NOTE**: You may add multiple devices. Once you are finished adding devices, select '`q`' to quit the device configuration menu.

Installing BRU From a 'tarball'

If you received BRU as a compressed tarball (`filename.tar.Z`), either in another product or as a download from our website, these instructions will tell you how to proceed. In order to perform the installation properly, you must be logged in as root or have superuser privileges.

To install BRU, follow these steps

Change to a temporary directory, such as `/tmp` by entering the following command:

```
cd /tmp
```

Use '`uncompress`' to return the file to normal tar format.

```
uncompress filename.tar.Z
```

Use '`tar`' to extract the installation:

```
tar -xvf filename.tar
```

The installer and BRU files have now been copied into the temporary directory.

Finally, execute the install command as outlined above:

```
./install
```

Installation Failure

If errors occurred during installation, first verify that the `tarfile` or CDROM is readable and that the files were actually read.

For a CDROM based install, cd to the CDROM mount point and perform an '`ls -l`' to ensure that the media is readable.

For a tarball install, first, make sure that the file uncompressed properly. If after uncompressing the file, the '`.Z`' extension has been removed, you should be able to execute '`tar -tvf filename.tar`' and have all of the filenames in the BRU install displayed with no errors reported.

Once you know that the distribution is readable, not installing as root could also be causing an installation failure. To install BRU properly, you must be logged in as root or have superuser privileges. The most reported BRU install failure is caused by not running the install as ROOT!

Manually Installing

In the event that the BRU installation reports that BRU has not been ported to your platform, but you know that you have the correct version of BRU, it may simply be a change in the naming convention utilized by your UNIX platform vs. the platform on which the copy of BRU was built.

After extracting the tarball you will find a directory called `brufiles`. Change into this directory with cd and examine the directory contained within. If it appears that the listed platform, ie. `sparc-solaris2.7` should run on your system, `cd` into that platform directory and execute `./bru -h`. If you do not receive a runtime error, then this version of BRU is compatible with your platform. Execute `./installbru` to install from within this directory. If you do receive a runtime error, the copy of BRU you are trying to install is not compatible with your platform. Please contact sales at TOLIS Group for more information.

Once you have received your license information, you may enter it into your existing demo version by executing `/bru/setlicense` and responding to the prompts.

Getting Started With BRU

BRU is a UNIX power-user's tool for performing system backup and recovery. As such, the many variations on the options available from the command line can be quite daunting for a first time user. For most operations, only a small number of options are used.

If you have ever used the UNIX command '`tar`', you're already a good way towards understanding the way BRU operates from the command line. To simplify things, let's examine some of the basic properties of any BRU backup:

- What operation are you invoking? Backup, Restore, Table of Contents, Verify or Estimate?
- What backup device are you using? A tape drive, a disk file, a floppy?
- What file system or selection of files are you backing up or restoring?

Once you know these items, you need to examine the various modes and options that BRU provides. These include:

`-c` - Create (backup) a BRU backup volume

`-x` - Extract (restore) files from a BRU backup volume

`-t` - Table of Contents (file list) of a BRU backup volume

`-i` - Inspect (verify) the contents of a BRU backup volume

`-e` - Estimate the number of volumes a BRU backup will require

`-d` - Difference (compare) the contents of a BRU backup volume with the original files on the file system

`-v` - Verbosity level (up to five `v's` may be specified)

`-f` - What backup device should be accessed

BRU's command line looks like this:

```
bru -mode -[options] [-f device] [path]
```

With these basic modes and options, all basic backup and restore functions can be performed. For example, to backup the contents of the entire system to the first SCSI tape drive under Linux, we would issue the following BRU command:

```
bru -cvf /dev/st0 /
```

This will backup the entire system to the tape drive `/dev/st0`. To examine the contents of a tape made in this manner, we could issue the command:

```
bru -tvf /dev/st0
```

Notice that we don't have to specify a file path for a listing of the tape.

To restore the entire backup to its original path, we would use:

```
bru -xvf /dev/st0
```

This call would automatically restore all files to their original locations.

To verify the contents of a BRU backup volume, there are two options available. The first is TOLIS' recommended option as it requires only the tape drive and BRU to operate. This is the inspect (`-i`) mode.

```
bru -ivf /dev/st0
```

This verification re-reads each buffer block written on the backup volume and recalculates the block's checksum. BRU then compares this calculated checksum with the checksum that was written in the buffer block header. If an incorrect value is detected, BRU will issue a warning that the checksum is bad for the particular file that the error occurred within. However, BRU will restore the available data in the file and any additional data on the backup - unlike other applications which abort at the first sign of a tape error.

The second option is a more widely used method that requires both the backup volume and the original data. This mechanism is called the difference (`-d`) mode.

```
bru -dvf /dev/st0
```

This option reads the data from the tape and performs a bit-by-bit comparison with the original data from the file system. Of course, this mechanism will report problems if files have changed on the file system since the backup was made. This is what makes the inspect mode verification the preferred option for verifying tapes.

Advanced Operations

Now that we understand the basic BRU operations, we can examine some of the more advanced options that can make the difference between a plain backup and a backup that provides enhanced functionality unavailable in other backup utilities, including:

- `-L` - Place a human-readable text label on the backup volume
- `-g` - Read and display ONLY the backup volume information
- `-n` - Select files based on a date and time specification
- `-B` - Run BRU in the background
- `-PA` - Change Absolute paths to relative (strip the leading `/`)
- `-ua` - Unconditionally overwrite ALL files during restore

These options allow you to more selectively control the backup or restore operation performed.

To add the description "Full System Backup" to the backup example from above, we would use:

```
bru -cvVR -L “Full System Backup” -f /dev/st0 /
```

Using `bru -g`, returns the following:

```
# bru -gf /dev/st0
label:           Full System Backup by Tim
created:         Mon Apr 27 08:55:02 1998
artime:          8936925021
archive_id:              3544aa561537
volume:          1
writes:          3
release:         17.0
variant:         0.3
```

```
bufsize:          32768
msize:            0
msize_blks:             0
serial_number:    xxxx-nnnn-n
device:           /dev/st0
user:             root
group:            root
system:           Linux giza 2.0.33 #1 Wed J i686
bru:              Fifth OEM Release
command_line:     bru -cvVR -L “Full System Backup” -f /dev/st0 /
```

As you can see, there’s a lot of information in the volume header. The important items here are the label, creation date, volume, and command line. If you wish to use BRU to perform Incremental or Differential backups, you could use ‘find’ and pipe the results to the BRU command, however, BRU provides a better method for locating files based on their date-time stamps - the ‘`-n`’ option. Using the `-n` option, you can pass a standard date string, like “`01-May-1999`”, or you can create a reference file and pass the name of that file to BRU, like `/etc/LASTFULL`.

Here’s a sample script (change the device for your system) that performs full backups on Saturday morning and differential backups on all other days:

```
#!/bin/sh
#
# This script performs full system backups on Saturday of
# each week and then performs differential (data changed
# since last full backup) backups on all other days.
#
# It would be run by cron at some time in the evening
# when everyone is logged out of the system for best
# coverage.
############################################################

DOW=`date +%w`
if [ $DOW = 6 ]
then
touch /etc/LASTFULL
bru -cvf /dev/st0 -L”Full Backup `date`” /
else
bru -cvf /dev/st0 -L”Differential Backup `date`” -n /etc/LASTFULL /
fi
```

For more information on the uses of BRU, be sure to review the remaining chapters of this manual and visit `http://www.tolisgroup.com`.

Chapter 3 - The* `brutab` *File

BRU works most effectively when it is properly matched with the backup device it will be using. The proper device settings allow BRU to run at maximum speed and increase its ability to detect and recover from errors.

The device parameters are described in a file that is read by BRU when it begins execution. Normally, the BRU device file is named `brutab` and is located in the `/etc` directory. The `/etc/brutab` file may contain entries for several different devices. This file describes: The name and characteristics of the available archive devices (a description can cover more than one such device). When you provide entries in `brutab` describing, for example, the size of the archive tape, then the size parameter need not be provided to BRU on the command line when the program is called.

Values (either symbolic or numeric) that your own system generates for errors of various kinds. This allows BRU to respond properly when certain conditions (like end-of-tape or write protect) are encountered. It also lets BRU generate the correct message when an error is detected.

You may have several versions of `brutab` if you wish. Normally BRU looks for a device parameter file named `/etc/brutab`. However, you can override this default by setting the BRUTAB environment variable to point to the version of `brutab` you want to use.

Here's an example of how to set the BRUTAB environment variable (under the Bourne shell):

```
$ BRUTAB=/tmp/brutab.new
$ export BRUTAB
```

The `brutab` file usually has sample device configurations for many systems on which BRU has already been installed. If your specific device or system is not listed, you can probably find one that is similar. Simply copy the applicable entry and edit it for your device. **NOTE**: The `brutab` file must contain at least two entries: One describes the capabilities of the default archive device, and a second describes the capabilities of the standard input device and its error messages. See Appendix A for example `brutab` file.

The Syntax of `brutab` Entries

Device entries in `brutab` consist of multiple fields separated by white space (spaces or tabs) or a new line character. Entries may span more than one line if you enter your shell's escape character (usually a `\`) immediately before the newline character at the end of the line.

Each `brutab` entry begins with the device name, followed by one or more parameter fields. All tabs and blanks between fields are ignored. Any line that begins with a # character is ignored and will be treated as a comment (except for global `brutab` parameter lines, which are described later).

The device name has the following form:

```
device name | ... | device name
```

For example:

```
/dev/rmt0
```

or

```
/dev/mt0|/dev/mt1|/dev/mt2
```

If the device name field specifies multiple devices, each device must have exactly the same characteristics and capabilities. Note the lack of spaces in the device name field.

There are two types of parameter fields used to define the characteristics of a backup device. Fields are either Alphanumeric or Boolean (true/false). An alphanumeric field consists of the parameter name, followed immediately by an equal sign '=' and the value. There must not be any spaces before or after the equal sign. The value following the equal sign may be a number, a symbolic value, or a string. String values must be enclosed in double quotes.

A Boolean field consists of the parameter name only. If the field is present, the corresponding device capability will be enabled (turned ON). If the Boolean field is absent, the corresponding device capability will be disabled (turned OFF).

Here is an example of various parameters and values:

```
size=640K noautoscan pwerr=EIO \
fmtcmd="format /dev/rfd1135ds9"
```

NOTE: That there is no space between the parameter name and the value. Numeric values may be given in absolute form or appended with a scale factor of:

b or B	Blocks (2048 bytes)
k or K	Kilobytes (1024 bytes)
m or M	Megabytes (1024*1024 bytes)
g or G	Gigabytes (1024*1024*1024 bytes)

The letter T may also be used as a modifier after the scale factors M or G. This is commonly used when specifying tape size.

When T is specified, a multiplier of 1000 will be used (instead of 1024). This is useful when using the tape size given by the tape manufacturer. Tape megabytes are measured differently than memory megabytes. A tape that is specified as 150 tape megabytes will actually hold only 143 memory megabytes.

Listed below are the scale factors when the T (for Tape) modifier is used:

kt or KT	Kilobytes (1000 bytes)
mt or MT	Megabytes (1000*1000 bytes)
gt or GT	Gigabytes (1000*1000*1000 bytes)

This is an example of how to specify the media size for a tape with a manufacturer-specified size of 525 megabytes:

```
size=525MT
```

Certain parameter fields (like `pwerr=EIO`) refer to error codes returned by the UNIX systems. The values for these parameters may be given as digits or in symbolic form (like EIO or ENOSPC).

NOTE FOR PROGRAMMERS: Symbolic error codes are usually defined in the development system header include file `/usr/include/errno.h`.

`brutab` Default Values

If you do not know the value of a parameter for your system, simply leave it out and BRU will use a built-in default value. If you're not sure, try using BRU with the default values first. On most UNIX systems, the default values will work without any problems. If you have trouble with the default values, try using the values from another `brutab` device entry that is similar to yours. If you cannot determine the proper values, you may want to contact us for assistance. See “How To Get Help,” in Chapter 1, “Introduction,” for our hotline and FAX numbers and for our E-Mail and web addresses.

The Default Device

If no device is specified on the command line, BRU will use the first device listed in the `brutab` file. To use a different device, you must either specify the device name with the `-f` option or set the `BRUDEV` environment variable to a device name.

The following command line specifies the device `/dev/rct0`:

```
$ bru -c -f /dev/rct0
```

`If the BRUDEV environment variable is set to a device name, BRU will use that device as the default instead of the first entry in the brutab file. The command shown below sets BRUDEV to /dev/tape and will cause BRU to use /dev/tape as the default device:`

```
$ BRUDEV=/dev/tape
$ export BRUDEV
```

`brutab` Fields

The following sections contain names and descriptions of alphanumeric and boolean `brutab` fields and of global `brutab` parameters. This is information that may be needed during installation. See both Chapter 2, “Installation” and Appendix E, “A Sample `brutab` File.”

Alphanumeric `brutab` Fields

These are the alphanumeric fields currently recognized by BRU. The two most important fields are listed first:

`size=`

Media size in bytes, if known. Zero, if unknown, or if variable-sized media. If a size is given, it must not be larger than the actual media size. For instance, do not specify `size=30G` if you are using a 15 gigabyte tape.

This option should be used with fixed size devices such as tape drives, removable devices (Iomega Zip or Jaz), and floppy disks. With modern tape drives that provide hardware compression, it is not possible to properly define the amount of data that a cartridge will store - in these cases, please use a setting of `size=0`.

If you do specify a size other than `0`, the specified size must not be larger than the actual tape size. If you set it larger, it may cause confusing error messages. This is because BRU is unable to determine the difference between an end-of-tape condition and a real error. To avoid problems, it is better to set the size too small than too large.

Media size can also be specified on the command line with the `-s` option. This command line specification will override any value specified in `brutab`. **IMPORTANT**: The media size must be specified as a whole number, no decimals are allowed. Specifying `size=2.2GT` is **ILLEGAL** and will not work. Instead, you should use `size=2200MT`.

`bufsize=`

Default I/O buffer size for this device. If omitted, the default is 20 kilobytes. If you are using double buffering (`-D`) this `bufsize` value should not be larger than the value of `shmmax`, or the system imposed limit on the size of a shared memory segment, whichever is smaller. The buffer size is the amount of data BRU will transfer from/to the archive device during each read/write operation.

This parameter is usually the one to adjust when trying to maximize BRU's speed. The optimum value varies widely, depending on your system and device. If you are having problems, start by setting `bufsize=16K`. This should work for almost any UNIX system. There is no single "best" value for `bufsize`. On some systems, `bufsize=8K` may produce the best results. On others, setting `bufsize=128K` or larger may be faster. **IMPORTANT NOTE**: Over 75% of all reported BRU problems are caused by buffer sizes that are too large. This is especially true for devices that have a SCSI interface. In many cases, these problems are not apparent when writing to a device (backing up data), but occur when attempting to read (restore data).

DO NOT USE A LARGE BUFFER SIZE SIMPLY BECAUSE IT IS FASTER!

Verify your backup! Use the `-i` or `-d` options to verify the backup, or let AUTOSCAN do it automatically. A fast backup is worthless if it contains errors. Buffer size problems are normally detected by the AUTOSCAN feature, which automatically rewinds and verifies each tape. See Chapter 6, "Archive Inspection and Verification."

The buffer size can also be specified on the command line with the `-b` option. This will override any value specified in your `brutab`.

Listed below are additional alphanumeric `brutab` fields. These are optional and can usually be ignored:

`maxrewindtime=`

The maximum number of seconds that it takes to rewind a tape. Normally, the UNIX system causes the tape to rewind and then returns control to BRU when the rewind is done. However, some tape drives (or device drivers) have a design under which they return control to BRU immediately before the rewind has finished - which causes errors when BRU attempts to read the tape during AUTOSCAN. If your tape drive has this bug, setting `maxrewindtime` will cause BRU to wait before attempting to start an AUTOSCAN. If necessary, a setting of `maxrewindtime=180` seems to work for most tape drives.

`minrewindtime=`

The minimum number of seconds that it takes to rewind a tape. This parameter is only in effect only if `maxrewindtime` is set to a non-zero value. In most cases, `minrewindtime` does not need to be specified and defaults to 25 seconds.

`asbufsize=`

AUTOSCAN buffer size. By default, this is same as the value specified by bufsize. Normally, this is the best value and there is no need to specify `asbufsize`.

`maxbufsize=`

Maximum I/O buffer size for this device. The maximum buffer size this device can handle. If you ask BRU to use a larger buffer size, it will issue a warning that the maximum buffer size has been exceeded. Without this warning, it can be hard to determine the cause of read or write errors that are due to exceeding the capabilities of the device.

`seek=`

Minimum seek resolution. Zero if no seeks allowed. For most seekable devices, the seek resolution corresponds to the physical block size of each data block on the device. All seeks performed on the device will be an integral multiple of this value.

`ederr=`

Error code returned by the device driver when the end-of-device is reached. This allows BRU to detect when it has reached the end of a tape. On many systems, this should be set to `ENOSPC`. For example:

```
ederr=ENOSPC
```

Device drivers vary widely, so the correct value for your system may be different. If you do not know the error code that is returned, do not specify `ederr` or set `ederr=0`.

`prerr=`

Error code returned by the device driver for partial reads. A partial read successfully reads more than zero bytes but less than the requested number of bytes.

`pwerr=`

Error code returned by the device driver for partial writes. A partial write successfully writes more than zero bytes but less than the requested number of bytes.

`zrerr=`

Error code returned by the device driver for zero length reads. A zero length read reads zero bytes.

`zwerr=`

Error code returned by device driver for zero length writes. A zero length write is one that writes zero bytes.

`frerr=`

Error code returned by device driver after an attempt to read from unformatted media. Applies mostly to floppy drives (most tapes are not formatted).

`fwerr=`

Error code returned by the device driver after an attempt to write to unformatted media. This code applies only to devices that must be formatted (like floppies or mini-cartridge tapes). It does not apply to most cartridge tapes, as they do not require formatting.

`wperr=`

Error code returned by the device driver after an attempt to write to media that are write-protected.

`fmtcmd=`

Allows you to specify a format command that BRU will execute if it tries to write to unformatted media (like a floppy disk that has not been formatted). The format command should be enclosed in double quotes. For example, the following command formats a floppy disk under SCO OpenServer:

```
fmtcmd=”format -f /dev/rfd096”
```

If `fmtcmd` is specified, the format field must also be specified or the `fmtcmd` string will not be executed.

`iotimeout=`

Read/write timeout value. This is not a fixed timeout but is the number of seconds added to the maximum time needed to read/write a block of data at 1000 bytes/second. For example, the time needed to write 20,000 bytes would be 20 seconds. If `iotimeout=30` is specified, this value would be added to 20. In this case, BRU will issue a timeout error if the write did not complete within 50 seconds. Normally, `iotimeout` can be left out. It is useful for trouble-shooting purposes if BRU “hangs” mysteriously when writing to a device.

`openretries=`

Number of additional times BRU will try to open a device if the first attempt fails. The default value is 1. Specify `openretries=0` if you want only one attempt at opening a device.

`opentimeout=`

Number of seconds BRU will wait while attempting to open a device. The default value is 3600 seconds. This value must be large enough to include the amount of time it takes to rewind a device, since many devices do not return from an open call until the rewind is complete. Specify `opentimeout=0` to disable the timeout.

The following parameters apply only if double buffering (the `-D` option) is used. If double buffering is not used, they are ignored.

`shmmax=`

Limit on the size of each shared memory segment. BRU will not attempt to allocate a shared memory segment larger than this limit.

`shmseg=`

Limit on the number of shared memory segments. BRU will not attempt to allocate more than this number of shared memory segments. For double buffering, BRU allocates one segment for shared variables, and then at least one additional segment for shared I/O buffers. Therefore, the minimum number of shared memory segments is two.

`shmall=`

Limit on the total amount of shared memory used. BRU will not attempt to allocate more than this total amount of shared memory.

Boolean `brutab` Fields

This section contains a listing of the fields with Boolean capabilities currently recognized by BRU. These Boolean fields are:

`ignoreclose`

Ignore the error that is generated by some devices when they are closed. Some devices with a SCSI interface may need to have this field specified. This will eliminate the `[W003]` warning message that occurs with some tape drives.

`noautoscan`

Disable the AUTOSCAN feature. **DO NOT DISABLE AUTOSCAN UNLESS YOU HAVE A VERY GOOD REASON FOR DOING SO**. The AUTOSCAN feature acts as an early-warning system - detecting minor problems before they become serious. If AUTOSCAN is disabled, your data integrity may be compromised and many types of problems can go undetected.

`noreopen`

Do not close and reopen archive upon media switch. This setting is rarely used but keeping the archive device open usually prevents other processes from accessing the same device.

`rawtape`

Archive device is a raw tape drive. This means that the kernel does not buffer data to and from the device.

`norewind`

Archive device does not automatically rewind to the start of the media after it has finished writing. Note that the size parameter should be zero. **NOTE**: AUTOSCAN will be disabled if `norewind` is specified.

`advance`

Indicates that the device has the capability of advancing the media past read/ write errors. In our experience, very few tape drives (or device drivers) have this capability. Most refuse to advance past hard errors on the media. If this parameter is set, BRU will be able to proceed past bad spots on the media. **NOTE**: It is a serious mistake to define this parameter for devices which are unable to advance the tape drive when an error occurs. If defined incorrectly BRU may not be able to recover from read errors. Most tape drives do not have the ability to advance on error.

qfwrite

For the first write to the first media in this device, request confirmation to proceed. This flag should be used if the device is also used for mounted filesystems, to protect against accidentally overwriting a media that may have been left in the device (floppy drives for example). **NOTE**: Does not work when multiple archive devices are specified.

`shmcopy`

Indicates that the device driver for this device cannot do I/O directly from shared memory. During archive writes, the data will be copied from shared memory to a locally allocated buffer before doing I/O to the archive device. During archive reads, the data will first be read into a locally allocated buffer, and then copied to shared memory. BRU will warn you when it detects conditions that indicate that this parameter should be set, and will automatically switch to copy mode.

NOTE: This parameter is needed only with double buffering (`-D` option). If double buffering is not used, this parameter is not needed.

Global `brutab` Parameters

Several BRU parameters are not specific to a device. These are global parameters and are specified in the `brutab` file in a form that is different from device parameters. Global `brutab` parameters are specified with names that are all **UPPER CASE**. Only one parameter is allowed per line. A line containing a global parameter must begin with the two-character sequence #+ (the pound sign followed immediately by a plus sign).

Normally, the global parameters are specified at the beginning of the `/etc/brutab` file. Global `brutab` parameters can also be set as environment variables. These environment variables will override the settings in the `/etc/brutab` file. To remove any of these environment variables use the unset PARAMETER command i.e. unset BRURAW.

Here is a list of global `brutab` parameters (with default values), showing how they are specified:

```
#+ OVERWRITEPROTECT=YES
#+ RECYCLEDAYS=7
#+ MAXWRITES=200
#+ ZBUFSIZE=500k
```

These parameters are explained in greater detail:

```
SHELL=/bin/sh
```

This variable points to your command shell. This shell is used when BRU spawns a task for running other tasks (such as switching tapes in a loader.)

`OVERWRITEPROTECT=yes`

BRU will check the tape (by reading the first block) before attempting to do a backup. BRU will read the date of the tape before proceeding. BRU will then compare the date in the archive header of the tape to the number of days specified in the RECYCLEDAYS filed. If the tape is newer then the number of days specified by RECYCLEDAYS field BRU will prompt you for a different tape. **NOTE**: This parameter only take effect for devices that have been specified and defined in the `/etc/brutab` file. They have no effect for other devices.

`RECYCLEDAYS=7`

The values specified by RECYCLEDAYS field is read by BRU and compared to the tape. Here's an example: Assume that you are doing today's backup and accidentally try to use the backup tape from yesterday. If `RECYCLEDAYS=7` is set, then BRU will give you a warning and refuse to overwrite yesterday's tape. To continue, you will need to use a blank tape or a BRU tape more than a week old. You should set RECYCLEDAYS to a value that is consistent with your tape rotation schedule.

You can disable date checking by setting `RECYCLEDAYS=0`. However, this will not completely disable OVERWRITEPROTECT. BRU will still do a "tape change" check on multi-volume archives. **NOTE**: This parameter only take effect for devices that have been specified and defined in the `/etc/brutab` file. They have no effect for other devices.

`MAXWRITES=200`

The MAXWRITES option works in conjunction with the OVERWRITEPROTECT option and allows you to limit the number of times a specific piece of media (tape) can be used before it is recommended for removal from tape rotation. If this value is exceeded, BRU will issue a warning and refuse to continue.

Below you will find a few recommendations for various tape types. Although some of these settings might seem low compared to the manufactures specifications we feel that a more conservative value will insure the integrity of your data.

1/4” DC6000 carts	100
4mm DAT	200
8mm Exabyte	50 for non-MP (Metal Particle).
	150 for MP tapes
Mammoth 1 / 2	500
AIT 1/2/3	500
VXA-1/2	500
DLT	1000
SDLT	1000
LTO	1000
1/2” Reel	250

Setting `MAXWRITES=0` will disable checking of the number of writes. **NOTE**: This parameter only take effect for devices that have been specified and defined in the `/etc/brutab` file. They have no effect for other devices. The RECYCLEDAYS and MAXWRITES parameters have no effect unless OVERWRITEPROTECT is specified.

```
BRUTABONLY=no
```

If BRUTABONLY is specified as a global parameter, BRU will permit only the use of devices listed in the `/etc/brutab` file. If the user attempts to use a device that is not listed, BRU will issue a warning message and refuse to continue until the correct device is specified.

```
DEVNAMECHECK=no
```

DEVNAMECHECK causes BRU to perform a “sanity check” on the device name to prevent the creation of a large file in the `/dev` directory. If BRUTABONLY is YES, then this option is inactive.

```
MATCHLEVEL=2
```

MATCHLEVEL determines the level of pathnames matching performed by BRU during a backup or restore. A level of `0` will only match the exact pathname entered (i.e.: `/tmp/123` will not match `./tmp/123` or `tmp/123`, only `/tmp/123` will match). A level of `1` will match relative pathnames that indicate the same path (i.e.: `tmp/123` will match itself and `./tmp/123`, but `/tmp/123` will not match). Level 3 will match any similar pathname (i.e.: `tmp/123` will match `/tmp/123`, `tmp/123` and `./tmp/123`). **NOTE**: This setting can cause some confusion, be aware of what your MATCHLEVEL is set to when doing restores. By default we have set this option to two (2) to allow maximum flexibility when restoring. This is covered in more detail in Chapter 7, “Extracting Files: The BRU Restore Function”

```
MAXFILENAMELEN=255
```

This option allows you to transfer BRU archive files from a UNIX system that supports long file names (greater than 14 characters) to a system with short file names. If you set MAXFILENAMELEN to the maximum length your system will support, it causes files with longer names to be renamed (not truncated). A warning message is issued for each renamed file. The default setting for MAXFILENAMELEN is the maximum file name length your system supports. On certain versions of UNIX (usually older ones), MAXFILENAMELEN defaults to 14. On newer OS's (like Solaris and Linux), it is set to 1023 or more.

`READCHECKLEVEL=1`

On many NFS mounted volumes, it isn't possible for BRU to easily determine if a file is locked. To prevent excessive read errors or a possible hang condition, BRU will pre-read files to check for a lock state. A 0 means no pre-read, a 1 means pre-read only files that appear to be locked and a 2 means to pre-read ALL files (which can slow your backup performance).

`ZBUFSIZE=1000k`

If most of your files are small, (less than a megabyte in size), then a setting of `ZBUFSIZE=500K` (or smaller) would probably work well. If you have many large database files (greater than 10 megabytes), then a setting of `ZBUFSIZE=5M` (or larger) might work better. For example, if BRU compresses a 500K byte file by 50%, then it would need 250K bytes to store the compressed data in its compression buffer. To compress a 2 megabyte file by 50%, 1 megabyte would be needed, etc. NOTE: Setting `ZBUFSIZE` to a large value may cause BRU to fail. This is not a problem with BRU but is due to limits on the amount of memory that can be allocated to a UNIX process. You may be able to increase the maximum available process memory by adjusting your UNIX kernel parameters but this is not recommended unless you are familiar with "kernel-tuning" and know what you're doing.

`BRUHELP=/bru/bruhelp`

This variable points to the location and name of the BRU `readme` or `helpfile`. By default the location of that file is `/bru/bruhelp`.

`BRUMAXWARNINGS=1000`

This variable sets the maximum number of warning that BRU will report before aborting the current operation. By default this is set to 1000.

`BRUMAXERRORS=500`

This is the variable that sets the maximum number of errors that BRU will report before aborting the current operation. By default this is set to 500.

`BRUXPAT=/etc/bruxpat`

This variable sets the location of the `bru` exclusion file. Paths and files listed in this file will be excluded from the backup or restore if the `-X` option is placed in the `bru` command line, refer to Chapter 5, "File Inclusion and Exclusion." By default the file location and name is `/etc/bruxpat`.

`BRURAW=/etc/bruraw`

This variable sets the location of the file containing descriptions of raw data partitions to be backed up or restored by BRU. By default, this files location is `/etc/bruraw`. This file and the commands used with `bru` to activate this file are in Chapter 4, "Archive Creation" later in this manual. This used in conjunction with the `-r` option.

`BRUSMARTREST=/etc/brusmartrest`

This variable sets the name and location of the BRUSMARTREST file. Files listed in this file will be handled as open and restored using a protected method, alleviating the problem of restoring over an open file or shared library. By default this file is located in `/etc/brusmartrest`.

`BRUREMOVELOG=/bru/bruremovelog`

This variable sets the location and name of the results of the SmartRestore. In the event that a restored file's "text busy" flag was set, we rename the original file and then restore the appropriate file. As a result, we create a Bourne shell script that will clean up the old, renamed files. If you are creating scripts to run BRU, it is a good idea to execute this file as the last stage of any restore that is preformed. By default this file is located in `/bru/bruremovelog`

`MOUNTCMD="(null)" /bin/mountcmd`

This parameter allows the user to specify external commands to handle devices. MOUNTCMD is used to specify a command that will be called before BRU attempts to open a device for reading or writing. **NOTE**: If you set this option with a environment variable and want to "unset" it, then use the following command "unset MOUNTCMD" (under the Bourne shell).

`UNMOUNTCMD="(null)" /bin/unmountcmd`

The UNMOUNTCMD should be used to specify a command that will be called after BRU has finished reading/writing. **NOTE**: If you set this option with a environment variable and want to "unset" it, then the following command "unset UNMOUNTCMD" (under the Bourne shell). The MOUNTCMD and UNMOUNTCMD parameters were designed for use with multiple-cartridge tape handlers (stackers or jukeboxes). BRU passes four arguments to the commands specified by MOUNTCMD and UNMOUNTCMD.

Argument	Description
1	device name (i.e. /dev/rct0)
2	volume number (i.e. 2)
3	mode letter (i.e. 'c' or 'x')
4	media size in Kilobytes (i.e. 150000)

In most cases, the commands specified will be shell scripts. Normally, these scripts issue commands that will cause a tape handler to load or unload a tape. See Appendix K, "Using MOUNTCMD and UNMOUNTCMD," for samples of shell scripts that use these commands.

`NOFILESINHDRS=no`

With this option your backups are backwardly compatible with older versions of BRU prior to 14.2.

`DONTLABELALLVOLS=no`

This option toggles BRU's ability to write a Label for each Volume in a multi-volume backup. By default BRU will create a label on each volume during its backup process.

`DIRDATESELECT, ZINBUFFER & ZOUTBUFFER`

NOTE: For Technical Support Use Only. Please leave these settings unchanged.

Determining Parameter Values: `-hh`

BRU may be used with the `-hh` option to print a list of all parameters in the `brutab` file and their values. The `-hh` option may also be used with a device name to print information for that device from the `brutab` file. This allows you to determine how BRU is interpreting parameters. See Appendix B, “The BRU Help Command” for sample command lines and output.

Chapter 4 - Archive Creation: The Backup Function

This chapter describes command line options that are used to create a new archive - BRU's backup function - and contains sample commands for the basic types of backup. The options that can be used with the `bru` command allow you to tell BRU:

- You want to perform a backup
- To estimate how many tapes or disks it will take to create the archive
- What device to use
- How to label the archive
- To set the verbosity level for BRU
- To generate log files
- To wait for confirmation for each action it takes
- Not to cross local filesystem boundaries
- Not to cross boundaries between local and remote filesystems
- To run without user intervention
- To back up a set of files or directories based on a list supplied on the command line
- To back up a set of files by modification date
- To back up a set of files owned by a particular user
- To back up a set of files or directories based on a list supplied in a file or read from `stdin`

NOTE: In this section we talk about writing archives or archiving files. If you are unfamiliar with this way of referring to the backup function, see the "Definitions" section of Chapter 1.

The BRU Command Line

The generalized BRU command line looks like this:

```
bru modes [control options] [selection options] [files]
```

where `bru` is the command and modes tells BRU, for example, whether you are backing up or restoring files. The most commonly used control options specify the device on which an archive is written, the size of the buffer to be used, or the string with which the archive is to be labeled. The selection options tell BRU which files or directories to archive. These might be files owned by a particular user or files created or modified after a particular date.

Options are always start with a hyphen and a letter that indicates the type of option. Some options also require parameters. The following sections describe how each of the most common options is used. See Appendix C, "The BRU Manual Page," for a detailed description of all options.

Command Line Options: Modes

Modes tell BRU what to do, for example, create an archive, list the contents of an archive, or extract files from an archive. There must be at least one mode option on the BRU command line. This chapter is concerned with describing BRU in archive creation mode.

Telling BRU This Is a Backup: -c

The `-c` option creates the archive. It does not require a parameter. All uses of `bru` to create archives will begin with

```
bru -c
```

The command line shown above is the simplest backup command you can give. But it assumes two things: First, since it does not specify a device, it assumes that your `/etc/brutab` file has been set up to specify a default device (see "Telling BRU What Device To Use: `-f`," below; Chapter 3, "Setting Device Parameters: The `brutab` File"; and Appendix E, "A Sample `brutab` File"). Second, it assumes you want to use BRU's default value for files. When writing an archive, the default for BRU is to select all files in the current directory and any subdirectories of the current directory.

Estimating Archive Size: `-e`

Estimating the size of the archive you want to write is similar to creating the archive. If you use the `-e` option instead of `-c`, BRU scans the files you want to back up and tries to tell you how many tapes (or disks if you are backing up to disk) will be needed to create the archive. The estimate is based on the size of the archive media you have either specified on the command line using the `-s` option, or implicitly specified by selecting a device named in the `brutab` file. See "Specifying Media Size: `-s`," below.

The resulting number will be accurate as of the time BRU is run. However, if the file system is active, changes in the files may cause the number of volumes to change. In addition, if the last file to be archived requires one more volume, that information may not be taken into account in the estimate.

Estimate mode cannot be used with data compression since BRU would have to compress each file to find out how much space it would take up in an archive. This would probably take nearly as long as creating the archive. If you must have an estimate that uses file compression, you can create an archive and send it to the null device:

```
$ bru -cvvvv -Z -s 640K -f /dev/null [files]
```

The above command will write a compressed archive to the null device and the verbosity output will indicate how many 640K volumes are needed to save the archive. See "Data Compression: `-Z`," below.

Command Line Options: Control Options

Modes tell BRU what you want to do; control options tell BRU how you want it done. Control options, as the name implies are optional. If you do not provide them, system defaults are used. Either "reasonable" defaults are used or default values are taken from the `/etc/brutab` file, which you will have customized for your system. (Chapter 3, "Setting Device Parameters: The `brutab` File," shows you how to set up a `brutab` file.) Control options that are commonly included in the `/etc/brutab` file are the device on which to create the archive and the size of the archive media.

Telling BRU What Device to Use: `-f device`

If you do not specify a device on the command line, BRU defaults to the first entry in the `/etc/brutab` file. See Chapter 3, "Setting Device Parameters: The `brutab` File."

The example below shows how to specify the device on the command line. It creates an archive from the files in the `/usr/src` directory, using a 60 megabyte tape inserted into the drive mounted as /dev/rmt0. It is important to know that the file that follows the `-f` need not be a device file, although this is the usual case. You may also use files on the system as archiving device files. However, you will experience an error if you set the `BRUTABONLY=YES` global `brutab` parameter in the `brutab` file.

```
cd /usr/src
bru -c -f /dev/rmt0 -s 60MT
```

See “Specifying Media Size: `-s`,” below.

Labeling the Archive: `-L label`

The `-L` option lets you provide a label of up to 63 characters for your archive. There are two ways of providing this label to BRU. The first is to simply specify the label from the command line as shown below.

```
bru -cv -L “Test Label” /usr/bin
bru -c -L “Old Mail 1” -vvvv letter* memo*
```

The second method of passing a label string to BRU, is to create a file and tell BRU to read that file. BRU will read in the first 63 characters of the file as the label for the tape as shown below.

```
bru -c -L “/tmp/label name” /usr/bin
```

The label information appears in the archive header, which can be viewed (or fed to an archive manager program) using BRU’s `-g` option. **NOTE**: That the label string and file name are enclosed in a pair of double quotes. This assures that any shell program you might be running (`sh, csh, ksh, tsh`) will treat the label as a single parameter. If the label is not enclosed in double quotes, BRU will see the individual components of the label string as separate parameters. In the second case, Old would become the label, and each of the words in the rest of the label would be treated as a file name. The following shows how `-g` mode is used to read the label information:

```
bru -g -f oldmail
```

In this example, `oldmail` is a standard UNIX file we are using in place of a device to receive the archived mail files. See Chapter 6, “Archive Inspection and Verification,” and Appendix C, “The BRU Manual Page” for a description of the `-g` option.

Backing up Raw devices: `-r`

This enables BRU to backup or restore raw data partitions. A Raw data partition is sometimes used by large database programs for storage. These partitions can now be backed up with BRU as long as a file called `/etc/bruraw` is created with the definitions of that device included in it. The file should contain entries in the format below :

Raw Device name	Size	Blk Size	Starting Offset
/dev/rfd0135ds9	720k	512	0

NOTE: An entry **MUST** exist in the `bruraw` file or BRU will abort the operation with an error message.

When backing up raw partitions, you must either have the device node in the tree relative to your current location or you must explicitly declare the raw device on the command line as a file name. For example, if you are in `/` and the raw device node in `/dev/rdsk/c0t3d2s1`, you can use this command line to back it up properly:

```
bru -cv -r /dev/rdsk/c0t3d2s1
```

If, however, you are invoking the `bru` command from your home directory, you must include the path to the device node on the command line:

```
bru -cv -r /dev/rdsk/c0t3d2s1 /dev/rdsk/c0t3d2s1
```

NOTE: If any writes occur during the backup, we will issue error messages explaining that some data changed during the backup. We do suggest that if this happens, you should run your backup process again and attempt to get a clean image of that data. BRU starts reading at the front of a file and works it's way through it. We do not lock files so if the information was changed during our backup that information on the tape or archive device will be incorrect.

Changing Default BRU Operations With: `-Q`

This option allows you to change or reset BRU default operations set in the BRUTAB file. By using the `-Q` option with one of it's create mode modifiers (`H` or `L`) BRU will disable that setting and allow you to run with the new settings.

`-QH` - This option will disable BRU from placing small files into tape headers. Using this option allows you to create BRU tapes that are compatible with versions of BRU prior to 14.2. This will override the global BRUTAB setting of `NOFILESINHDRS`.

`-QL` - This option tells BRU to use the literal string as a tape label. This overrides BRU's attempt to look for a file from which to read the tape label. This can be useful if you have a file (in the directory that you are running BRU from) that has the same name as your label string.

Selection Depth for Backup `-U`

BRU, by default will expand directories when given from the command line. With this option BRU can be limited to a certain level of directories that it will expand. By running BRU with the `-U0` (zero) option BRU will only backup the first level of the directory given or the current directory. This can be very useful when you need to backup only the files from a specific directory and none of the files below that directory. For example, the command below will only backup the current directory depth. BRU will backup all files in /home/web and all directory nodes or (names) but it will not pass through and backup files in other directories:

```
bru -c -V -U0 /home/web
```

Below you will see an example of running the option but with the `-U` option set to two (2):

```
bru -c -V -U2 /home/web
```

With this setting BRU backed up all files and nodes in the `/home/web` directory along with `/home/web/test`, `/home/web/test/runone` but BRU would not backup files in the `/home/web/test/runone/demo1`.

Setting the Verbosity Level: `-v` and `-V`

BRU normally does its work silently, returning control to the calling interface (the user terminal, or the spawning process) once it has completed its task. However, if you want to monitor what BRU is doing, a verbosity level selection option is accessible from the command line. Verbosity is specified by the `-v` (lower case) and `-V` (upper case) options. The verbosity options provide five levels of verbosity and a summary mode: `-v`, `-vv`, `-vvv`, `-vvvv`, `-vvvvv`, and `-V`, where a single `v` represents the lowest and five `v's` the highest verbosity level. Using `-V` produces an execution summary without the volume of output generated by the other verbose options. The following example shows how the verbosity option is used with the estimate mode:

```
$ cd /
$ bru -e -vv ./bin/a*
e 2k of 4k [1] ./
e 2k of 6k [1] ./bin
e 72k of 78k [1] ./bin/acctcom
e 68k of 146k [1] ./bin/adb
$
```

The first field in the output contains the mode character, as a reminder of what mode is currently running (in this case the “`e`”, or “`estimate`” mode). The next field gives the amount of space in kilobytes that this particular file will use in the archive. The next field (“of `XXXk`”) gives a running total of the amount of data in the archive, including the current file. The number inside the square brackets is the current volume number. Finally, the name of the file is listed.

Generating Log Files Using `-v`

In the previous section, we monitored the output produced by the `-v` option on the screen. The `-v` option also allows you to generate log files that contain this information. If you want to keep a record of how a particular tape archive was created and what files it contains, you can redirect the output of one of the verbose options to a log file. This command line sends the information produced by the `-v` option to the screen:

```
bru -cv file1 file2 file3
```

but this command line sends the log information to `logfile` and any error messages to a different file, `errfile`:

```
bru -cv file1 file2 file3 > logfile 2> errfile
```

If you want both error and log information to go to the same file, you can use a command line similar to this under the Bourne shell:

```
bru -c -v file1 file2 file3 > logfile 2>&1
```

If you want BRU to read the list of files to be archived from a file instead of from the command line, you can use something like:

```
bru -c -vvvv < filelist > logfile
```

where `filelist` is a file containing the list of files, one to a line.

The amount of information you collect in your log files can be varied by using different levels of verbosity.

All error messages and warnings are also recorded in the BRU execution log, `/var/log/bruexeclog`. The `bruexeclog` file is described in Appendix F, “BRU’s Execution Log.”

BRU offers an additional verbosity level that is designed to allow for easier parsing within a script or interface. By setting the verbosity level to 10 `v's`, the output from BRU will appear as:

```
VL:c|e10|1|76|81a4|0|0|3e3199d3|3e10f6ac|3e10f6ac|0|70|803||||/etc/
mtab
```

In this output, the interesting fields are the 3rd and 12th fields separated by the pipe (`|`) symbol. The 3rd field tells you what volume (if it is a multi-volume backup), while the 12th field tells you the logical block address within the volume. In the example above, the `/etc/mtab` file is located on the first volume at the `0x70` (HEX value) block.

Confirmation Option: `-w`

If you are not quite sure what BRU is going to do, you may want to use this option as a test, on a small subset of files. When the `-w` option (wait for confirmation) is specified, for each file, BRU:

- Prints the file name
- Shows the action it is about to take
- Asks for confirmation

If you have a very long list of files, the confirmation option can be tedious. BRU therefore provides a special response that lets you change your mind. If you type a `g` (for "go ahead") instead of `y` or `n`, BRU will stop asking for confirmation. The following example shows how this response is used.

```
$ bru -cvw -f /dev/null ./bin
c ./: please confirm [y/n/g] y
c 2k of 4k [1] ./
c ./bin: please confirm [y/n/g] y
c 2k of 6k [1] ./bin
c ./bin/acctcom: please confirm [y/n/g] y
c 72k of 78k [1] ./bin/acctcom
c ./bin/adb: please confirm [y/n/g] y
c 68k of 146k [1] ./bin/adb
c ./bin/ar: please confirm [y/n/g] g
c 36k of 182k [1] ./bin/ar
c 120k of 302k [1] ./bin/as
c 16k of 318k [1] ./bin/asa
c 4k of 322k [1] ./bin/basename
. . .
```

IMPORTANT NOTE: Be aware that once you have typed g in response to BRU's request for confirmation, you cannot go back to confirmation mode.

Do Not Cross Mount Points: `-m`

There are times when you may want to back up all files in a single directory, but you may also want to limit the selection of files to those on the same mounted filesystem as one of the files in the files parameter. You can do this by including the -m option on the command line when you create the archive. For example, this command line will back up only files in the root directory and exclude any files on mounted file systems:

```
bru -c -v -m /
```

The next example saves all files on both the root and the `/usr filesystems`, but none of the files from other filesystems, such as `/usr2`, is included:

```
bru -c -v -m / /usr
```

If you are not certain which file systems are currently mounted, you can use the `df` command to list this information. The output of `df` varies from one UNIX system to another, but will generally list one filesystem per output line.

Excluding Remote Files: `-R`

Specifying the `-R` option tells BRU to exclude all files on remote systems. It lets you back up files on your local system and not reach out across the network to other systems. The following example backs up only files on the local system:

```
bru -cv -R /
```

For an individual machine that has remote connections and mounted filesystems from remote machines, archiving files from the remote machines could mean generating a very large archive. The `-R` option should be used to limit the backup to your local files.

Running Without User Intervention: `-B`

The `-B` option is used when running BRU in the background. Normally, it is included as part of a shell script that is scheduled to run automatically. For example, the following line might be included in a script that is run by `cron` every night:

```
bru -c -B /usr
```

The `-B` option in the BRU command line tells BRU that it is running in the background and that it can expect no user intervention. It is specifically designed to prevent the situation in which a background backup process needs another tape, and because there is no one present to mount it, the process waits indefinitely for input. If `-B` is used, the process will terminate with an appropriate error code. **NOTE**: The `-B` option will be set automatically if BRU is started as a background process, that is, if the "`&`" shell option is included on the command line. For example:

```
bru -c /usr &
```

This is true when using the Bourne shell; it may not work for other shells.

If an operator is available, you may decide not to use the `-B` option in order to assure that the backup actually takes place. If `-B` is not specified, then BRU will indeed ask for help and the operator may be able to resolve the error condition for a successful run.

The `-I` option provides other options for communicating with BRU in the background. See Chapter 8, "Advanced Uses," "Using BRU with `brutalk`: `-I`," Appendix C, "The BRU Manual Page," and Appendix H, "The `brutalk` Program." The selection of `-B` or `-I` will depend on what the desired recovery is when interaction is necessary.

Specifying Media Size: `-s media_size`

The `-s` option allows you to specify media size in the following format:

```
bru -c -s msize [options] [files]
```

`msize` is usually expressed as kilobytes (K bytes), megabytes (M bytes), gigabytes, or occasionally as bytes. Normally this option is only required with `-c` mode since BRU writes this information onto the archive header block. If specified, `msize` overrides any existing default value, whether built in or read from the archive header.

The following command is an example that writes to a 3.5” floppy disk using a size of 1440KB. It will back up all the files in the `/tmp` directory:

```
bru -cv -s 1440K -f /dev/rfd0135ds18 /tmp
```

Data Compression: `-Z`

Software based data compression is specified using the `-Z` option. In addition to allowing more data to be stored in a smaller space, using compression in software can provide a performance improvement if the backup device used does not offer hardware compression. However, if you are writing to a locally attached backup device that does offer hardware compression, do not use BRU’s software compression. Hardware compression will always be more efficient than software compression and sending compressed data to a compression-enabled backup device can result in the backup becoming larger, rather than smaller.

Command Line Options: File Selection

The files parameter on the command line tells BRU which directories and files to include in an archive. BRU scans its command line, and when it runs out of mode, control, and selection option specifications, interprets the rest of the command line as file names. File names must be specified last. If they are not, the results of the command may be unexpected. It is important to understand that specifying files on the command line is optional. If you do not include the files parameter, BRU will use the default value and include everything in the current directory and in all directories below the current directory.

Using Wildcard Filename Expansion

When you are executing the `bru` command from the UNIX command line, you can specify the files parameter by including wildcards (special characters that the shell expands into matching filenames). BRU recognizes wildcard strings in the same format recognized by the Bourne and Korn shells. For example, if you have the following files in a directory:

```
file1
file2
file3
```

and if you enter this command on the command line (not, for example, as part of an executable shell):

```
bru -cv file*
```

The wildcard character “*” of the above command will be expanded by the UNIX shell and will match the three files. The command is equivalent to

```
bru -cv file1 file2 file3
```

This expansion is done by the UNIX shell, not by BRU.

Wildcard file name expansion can cause problems, especially if the command is entered from a directory in which there are many files or subdirectories. The following command is NOT necessary and is NOT recommended:

```
bru -cv *
```

Using the above syntax can result in your shell attempting to expand the wildcard character and could result in a command line buffer overflow. While BRU will not allow this to become a potential system security exploit, it will most likely cause the BRU command to fail, rather than run. If no files are specified, BRU will automatically back up all files in the current directory. The above command should be replaced with:

```
bru -cv
```

If you have problems when using wildcard names, it may be due to filename expansion limits set by the UNIX shell. To see how the filenames were expanded, check the BRU execution log, `/var/log/bruexeclog`. The "START" entry should contain a list of all the expanded names that were passed to BRU. **NOTE**: Wildcard characters are used in a slightly different manner when restoring from an archive. Wildcard characters used to specify file names with `-x` (file extraction) must be enclosed in double quotes. See Chapter 7, "Extracting Files: The BRU Restore Function," "Using Wildcard Filename Expansion."

Using Pathnames

Files may be specified using either relative pathnames or absolute pathnames. The default used by BRU is to save each file by specifying the path to the file as relative to the current directory. This takes the form:

```
./subdir/filename
```

Naming a file using its relative pathname allows you to transport it from one filesystem to another because the path is relative to the current directory. For example, files that came from `./usr/tests` on one machine may be restored to `./system/qa` on another machine without the need to tell BRU anything about either the source or the target directory (since everything is relative to the current directory).

See Chapter 7, "Extracting Files: The BRU Restore Function," "Where To Put the Files: Pathnames," for a note on specifying file names. If files are specified, the names as stored will no longer be specified as relative to the current directory unless you explicitly include "`./`" in your file specifier(s). If you want your files to be portable from one machine (or filesystem) to another, you may want to specify files using a relative pathname rather than an absolute pathname. If you have backed up your files using absolute pathnames, BRU has an option (`-PA`) that allows you to convert an absolute pathname to a relative pathname when restoring the files. See the section on "Converting Pathnames from Absolute to Relative Form: `-PA`, " in Chapter 7, "Extracting Files: The BRU Restore Function.

NOTE: The string that describes the path to a file must not exceed the MAXPATH value for the shell within which BRU is being executed. For normal Bourne Shell, `/bin/sh`, this is usually 1023 characters.

Selecting Files from `stdin: -`

As an alternative to specifying the files to be saved on the command line, you can provide BRU with a file that it can read, or you can pipe the output of a command to BRU and tell BRU to interpret this output as the file list. In both cases, you must use the special filename "`-`". When you use a hyphen on the BRU command line in place of the files option, BRU reads the standard input stream for a list of files to process rather than collecting them from the command line. This usage takes the form:

```
bru -c [options] -
```

The special filename "`-`" is typically used in conjunction with a UNIX pipe or with the standard input stream redirected to a file containing a list of filenames. The two commands

```
cat /tmp/filelist | bru -c -PEPf -
```

and

```
bru -c -PEPf - < /tmp/filelist
```

are equivalent. Both create an archive of the files you have listed in `filelist`. Note the use of the `-PEPf` option, which causes BRU to expand any directories in the list. If the `-PEPf` is not specified, BRU will NOT expand directories, but will simply archive a single entry for each directory (just the directory name).

The following command backs up all files owned by a specific user, in this case user `jim`. The UNIX find command translates the symbolic user name `jim` to the correct user id. In this case, the `-PEPf` option is not needed, because the find command expands the directory and passes all the pathnames to BRU:

```
cd /
find . -user jim | bru -c -
```

Here is an alternate way to let BRU do the same thing. Since BRU is not reading a list from `stdin`, it will automatically expand the directories and will back up all the files owned by `jim` in the current and any lower directories:

```
bru -c -o jim /
```

Reading from `stdin` and Writing to `stdout`

A slightly different form of the BRU command is used when reading a list of files from `stdin` and writing the archive to `stdout`. Here's an example:

```
bru -c -f - - < filelist > bru.out
```

where `filelist` contains the list of files to be saved, creates an archive with the specified files and writes it to the file `bru.out`. The first "`-`" is taken as the parameter for the `-f` option, specifying that the output is to be written to the standard output stream. The second "`-`" is taken as the special filename in place of the files arguments, and directs BRU to read the list of files to archive, from the standard input stream.

Selecting Files by Date: `-n date`

Backups you schedule on a regular basis - daily, weekly, monthly, or at whatever interval is convenient for your application - can be full backups or incremental backups. The `-n` option is used to perform incremental backups (the '`n`' stands for newer than). It tells BRU to save only files that were created or modified after the date specified as a parameter. By default, it will cause BRU to only save files that were modified (or the `inode` status changed) after the date specified as a parameter. The default mode check the modification time and the status (`inode`) change time. If either time is greater than the specified date, the file will be selected for backup. To change the default comparison method (i.e. compare with modification time only), use the date type modifier example

```
bru -c0n 7-DEC-94,m
```

If you have a large number of files to back up, you can save time by performing incremental backups, for example, on a daily basis, and then running a weekly full backup. This way, only the files that changed since your last full backup will be archived during the week. Depending on your supply of backup tapes, you can then consider reusing an old backup tape from two or three weeks ago for a given weekly backup. Once a month you may want to do a full backup of all files and save these archives virtually indefinitely.

The frequency with which you perform backups, and how long you preserve them, depends on how far back you think users may need to retrieve earlier versions of files. This decision will depend on your system and your needs.

Date Formats for Use with `-n`

The date may be specified in one of four ways:

Format 1:

The variables in the first format have the following meanings:

DD

The day of the month.

MMM

The first three characters of the name of the month

YY

The last two digits of the current year

HH

The hour in 24 hour format

MM

The minute

SS

Seconds

The first format takes the following form:

```
-n DD-MMM-YY[,HH:MM:SS][,amc]
```

For example,

```
-n 07-dec-92,01:00
```

schedules a backup for 1:00 a.m. on December 7th, 1992. The three-letter month parameter is case insensitive. It could just as well have been written `DEC` or `Dec`.

The seconds may be omitted. As indicated by the square brackets, the entire time specification is optional. If present, it is separated from the date specification by a single comma (`,`). If no time is specified, the default value, `00:00:00`, is used.

There are three optional date types:

`a` - Access time. Access time can be set by the touch command. The list command `ls -lu` shows the access time.

`m` - Modification time. Modification time is shown by the `ls -l` command. The modification time can also be set by the touch command.

`c` - `Inode` change time, sometimes mistakenly referred to as “`create`” time but also known as status change time. Modifying a file, changing permissions, or moving a file will change this time. The time is shown by `ls -lc`. The status change time is set by the system. This time cannot be set by any user programs.

Format 2:

The following is a second format. All components are specified numerically.

```
-n MM/DD/YY[,HH:MM:SS]
```

Format 3:

A third format may also be used to specify the date following the `-n`:

```
-n MMDDHHMM[YY]
```

In this case, the month is specified numerically (`MM`) and must include a leading zero for one-digit months (for example, January must be entered as `01`, not as `1`). Note that there are two `MM` fields. The first is the month, the second is the year. If the year is not specified, BRU defaults to the current year.

Here is an example that schedules a backup for February 17th, 1993 at 1:30 a.m.:

```
-n 0217013093
```

Format 4:

The fourth option tells BRU to look for a specific file and use its modification date as the argument to `-n`.

```
-n pathname
```

You can use this option to create a script that will automatically create an archive for the appropriate time interval. Thus the file name you specify may be nothing more than a dummy file whose sole purpose is to provide a pathname whose modification time is controllable. **NOTE**: The `-n` option can also be used when retrieving files from an archive, that is, when using the `-x` mode. In this case, it tells BRU to extract only files that are newer than the date specified. See Chapter 7, “Extracting Files,” “Selecting Files by Date: `-n`,” for sample command lines.

Resetting Access Time: `-a`

The option `-a` causes BRU to reset the access time of the files processed. This is important when using the `atime` (access time) of a file for date processing. Since the action of reading a file for backup accesses it, telling BRU to reset the `atime` will change the file’s `atime` to the last real access, rather than the access just performed by BRU.

Inverse Date Specification

In rare cases you may need to back up files older than a given date. You can do this by placing an exclamation point (`!`) immediately before the date. The exclamation point reverses the sense of the search. For example,

```
bru -c -n !10-Jan-92
```

will back up all files older than January 10th, 1992. Be aware that you may have to use your shell’s escape character to use the `!` on the command line.

Selecting Files by User: `-o`

You may want to back up files owned by a particular user. The `-o` option lets you tell BRU the identity of the user whose files you want to archive. BRU accepts three different forms of user identification:

- User name
- Numeric user ID
- File owner

Selecting Files by User Name: `-o login_name`

The user name is an ASCII string that identifies the user. It must correspond to a user name in a password file (normally in `/etc/passwd`). The first entry on each line in the `/etc/passwd` file is the user name. Note that this is the user's login name, not the full name of the user. A typical entry is:

```
    jeff:yrrZaI78sqbro:8:2:Jeff Dough:...
```

This option has the form:

```
   -o login_name
```

In the above entry, jeff is the user name you would specify. It is also the information that appears as the file owner when an ls -l command is issued. The following is an example of selecting the files to be archived by user name:

```
   bru -c -o jeff files
```

This command will create an archive containing only files owned by user `jeff`.

Selecting Files by File Owner: `-o pathname`

If you use a pathname to a file as the argument to the `-o` option, BRU assumes that the owner of that file is the user whose files are to be archived. This option has the form:

```
   -o pathname
```

An example is:

```
   bru -c -o /usr/dough files
```

BRU selects all files in files that are owned by the owner of the file `/usr/dough`.

Selecting Files by User ID: `-o decimal_value`

You can also specify the owner by his or her decimal identifier. The decimal identifier is the `uid` value in the `/etc/passwd` file. This option has the form:

```
   -o decimal_value
```

An example is:

```
   bru -x -o14720
```

The numeric user ID is used less often than the other two forms of the -o option, but is available for experienced UNIX users who want to use it. In some circumstances, it is possible for files to have user id numbers that do not match any user listed in the `/etc/passwd` file.

Forcing Overwrite of Archives (OVERWRITEPROTECT): `-O`

If you have the OVERWRITEPROTECT and RECYCLEDAYS set in the BRUTAB file, archives will not be overwritten unless their creation age is older than the setting for RECYCLEDAYS. To override this from the command line, use the `-O` (capital oh) option.

The Shell Scripts `fullbru` and `incbru`

Two shell scripts, `fullbru` and `incbru`, have been included with BRU to help you perform backups. These are normally installed as `/bin/fullbru` and `/bin/incbru`. The `fullbru` script will perform a full backup. It will back up all files on your system. `incbru` will perform an incremental backup; it will back up only the files that have changed since the last time `fullbru` was run. The `incbru` script does most of the work. It was written to be as simple as possible and may not work correctly on your system. Feel free to modify it to suit your needs. See Appendix G, "The Shell Scripts `fullbru` and `incbru`" for copies of the scripts and examples of their use.

Interrupting BRU

BRU normally catches both interrupt (`SIGINT`) and quit (`SIGQUIT`) signals. The keys that generate these signals are system dependent and can be discovered with the UNIX `stty` command (`stty -a` for example). When BRU receives an interrupt (`SIGINT`) signal during archive creation or extraction, it completes its work on the current file before cleaning up and exiting. By contrast, when BRU receives a `SIGQUIT` signal during archive creation or extraction, it terminates the program immediately. If an archive is being created, it will be truncated at the point at which `SIGQUIT` is received. If a file is being extracted from an archive, the file will not be restored properly. An interrupt always leaves an archive in a consistent state for future reads. It is therefore the recommended method for stopping BRU.

The `bruxpat` File and the `-X` Option

In addition to the standard BRU options for selecting files-by date, user, or owner - BRU lets you include or exclude files by name. This feature is enabled with the `-X` (uppercase X) option. The `-X` option can be used when you are creating an archive (`-c` mode) or extracting files from an archive (`-x` mode). Additionally, the `bruxpat` file is used when using BRU's software compression to exclude specified file types from the compression pass.

When you include the `-X` option on the BRU command line, BRU will apply include/exclude patterns specified in the `/etc/bruxpat` file (the default pattern file). The filenames encountered by BRU as it does a backup (or restore) will be compared to the patterns. If a filename matches any of the exclude patterns, it will not be backed up (or restored). Normally patterns will be read from the `/etc/bruxpat` file. BRU will use patterns from a different file if the environment variable BRUXPAT is set to the name of the new file. For example:

```
BRUXPAT=/etc/newbruxpat
export BRUXPAT
```

The `bruxpat` file is also used to disable compression on certain files (`-Z` option). If many of your files are already compressed, you can enter a line in the `bruxpat` file that tells BRU not to compress files whose names indicate that they are already compressed. You may, for example, want to disable compression for files with filenames that end in "`.Z`" or "`.z`" or MP3 files (which are already highly compressed).

Specifying Include/Exclude Patterns

Each line in the `bruxpat` file consists of a control field and a pattern. The pattern is separated from the control field by a space or tab (whitespace). The include/exclude patterns can be specified as shell wildcard expressions or as regular expressions (as used by commands like `grep` and `sed`). Lines beginning with the '`#`' character are treated as comments.

The control field consists of two characters. The first character specifies the type of action, the second specifies the type of pattern. The control field characters are described below:

The Control Field: Action Type Characters

`i`	Include this pathname if the pattern matches. The pathname is accepted and no further patterns are applied.
`x`	Exclude this pathname if the pattern matches. The pathname is rejected and no further patterns are applied.
`z`	Disable compression for pathnames that match this pattern. Applies only if compression is specified (the `-Z` option).

The Control Field: Pattern Type Characters

`s`	The pattern is a shell style wildcard pattern except that '/' characters are not treated as special characters.

`r`	The pattern is a regular expression.
`l`	The pattern is a literal string.

The Pattern Field

The pattern may be one of the three types specified above under "Pattern Type Characters." The way in which shell wildcard patterns and regular expressions are formed can be found in the standard UNIX documentation. **NOTE**: Be aware of any trailing whitespace on ANY non-comment line. This can cause unexpected problems with BRU.

A Sample `bruxpat` File

Here is a typical `bruxpat` file with patterns used to include and exclude files:

```
#
# Sample bruxpat
# Not all of these are necessary and are
# provided for example ONLY!
# Include the C runtime startup file but
# exclude all other object
# files.
is */crt0.o
xs *.o
# Exclude all core files.
xs */core
xs core
# Exclude files and subdirectories in
# the temporary directories. Handle
# both relative and absolute
# form pathnames.
xs ./usr/tmp/*
xs /usr/tmp/*
xs ./tmp/*
xs /tmp/*
# Don't waste time trying to compress
# files that are already compressed
zs *.[Zz]
zs *.gz
zs *.bz2
zs *.mp3
zs *.gif
zs *.jpg
```

The patterns act as filters that are applied to pathnames currently known by BRU. A pathname currently known by BRU is any pathname included by default (the current directory and everything below the current directory) or specified on the BRU command line (either explicitly or read in from files). The `-X` option will not cause BRU to select or include any files that are not part of the current directory or that are not among the directories or files specified as part of the BRU command line.

Testing Include/Exclude Patterns

Filter patterns may produce unexpected results, especially if several patterns are specified. It is wise to do a test before you attempt a backup. The following command will test which files will be extracted from an archive, and will write the pathnames selected to `xtest.out` using the extract pattern(s) specified in `/etc/bruxpat` (or the file named in the environment variable BRUXPAT). Test your parameters without doing an actual backup by using the following BRU command:

```
bru -evvvv -X > ctest.out
```

Running BRU in -e mode will not perform an actual backup; instead it will print all the selected pathnames, in this case sending them to the file named `ctest.out`. If this file contains the pathnames you expect, then it is safe to do a backup with the

`-c option substituted for -e:`

```
bru -cvvvv -X
```

You can test patterns used for file extraction in a similar manner by using the `-t` option. The following command will test which files will be extracted from an archive, and will write the table of contents to `xtest.out` using the extract pattern(s) specified in `/etc/bruxpat` (or the file named in the environment variable BRUXPAT):

```
bru -tvvvv -X > xtest.out
```

Again, if `xtest.out` contains the pathnames you expect, it is safe to restore files, substituting -x for -t:

```
bru -xvvvv -X
```

NOTE: To use the combination of the –T along with the –X options, refer to Chapter 7, "Extracting Files."

Chapter 6 - Archive Inspection and Verification

This chapter shows the options BRU provides for inspecting an archive or verifying its contents. These options allow:

- Inspect an archive
- List its contents
- Compare the archive to the current contents of a filesystem
- Create and list the contents of the archive list file

See Appendix J, "Archive Inspection and Verification Examples," for examples of the commands used and the output generated.

Archive Inspection

BRU provides three types of archive verification. The first (`-i`, inspect mode) reads the entire archive to ensure that the data and the internal storage structures are intact. The second (`-d`, differences mode) reads each file in the archive and compares it with the corresponding file on the disk. The third, default, method is called AUTOSCAN and is performed automatically. BRU also provides a way to list the contents of an archive without performing any verification (`-t`, table of contents mode).

BRU's `-g` option allows you to read the archive information block. This block contains the archive label and other information about the archive.

Inspecting an Archive: `-i`

When you use the `-i` (inspect) option on the BRU command line, BRU reads the archive tape and verifies the checksums of the data written. If data or internal storage structures are inconsistent, BRU will issue warning messages. You can stack `-i` with `-c` to manually inspect the backup that is run:

```
bru -c -i
```

This will back up all files in the current directory, then rewind and do a checksum verification of the archive. This is the equivalent of BRU's AUTOSCAN mode, and is not required unless you have disabled the AUTOSCAN feature with the `-A` command line option or defined `noautoscan` in the `/etc/brutab` for the specific device.

You can tell BRU to inspect an archive at any time. Unlike the `-d` option described later in this chapter, the files used to create the archive do not need to be on line.

Listing the Table of Contents: `-t`

The -t option lists the table of contents of the archive. If the verbose option (`-v`) is included on the command line, BRU provides a more detailed table of contents. The `-v` option, when used with `-t`, provides the equivalent of an `ls -l` performed on your UNIX system. The `-vv` option, when used with `-t`, provides information about hard or symbolic links, as well as file sizes and so on.

When run without any of the verbose options, the table of contents is simply the list of files in the archive:

```
$ bru -t -f bru.out
/
/bin
/bin/cat
/dev
/dev/rfloppy0
/dev/bru.q
/usr
/usr/lib
/usr/lib/news
$
```

Adding `-v` to the command line produces a listing of the table of contents that is similar to that produced by `ls -l`:

```
$ bru -tvf bru.out
drwxr-xr-x 12 root root 0 Jun 11 10:33 /
drwxr-xr-x 2 bin bin 0 Jan 29 20:49 /bin
-rwxr-xr-x 1 bin bin 9525 Nov 20 1987 /bin/cat
drwxr-xr-x 5 bin sys 0 Sep 20 16:25 /dev
prw-rw-rw- 1 root root 0 Sep 2 00:40 /dev/bru.q
drwxr-xr-x 25 root bin 0 Aug 29 15:10 /usr
drwxr-xr-x 27 bin bin 0 Sep 18 23:09 /usr/lib
lrwxrwxrwx 1 news news 0 Feb 4 00:34 /usr/lib/news
$
```

Dumping the Archive Information Block: `-g`

The `-g` mode prints formatted information from the archive header. This information is stored in the first block (2K) of the archive. The archive header contains information about when and where the archive was created, who created it, etc. It also contains an optional 63-character label (specified by the user who created the archive).

The following example shows how to use the `-g` mode to list the archive header information:

```
$ bru           -g
label:          My Backup
created:        Mon May 13 16:01:59 1995
artime:         7951357191
archive_id:            2f64cee7590a
volume:         1
release:        17.0
variant:        1
bufsize:        32768
msize:          1998848000
msize_blks:            976000
serial_number:  xxx-xxxxx
device:         /dev/nst2
user:           root
group:          bin
system:         odt20 odt20 3.2 2 i386
bru:            SCO UNIX
```

```
command_line:     bru -cvG /home/html/htdocs
```

The archive information block fields are described below:

`label`:

Label that was specified with the `-L` option when archive was created. Blank if no label was given.

`created`:

Date when BRU started to create the archive.

`artime`:

Integer representing the archive creation time. The value of `artime` is the number of seconds since January 1, 1970 (UNIX time).

`archive_id`:

A hexadecimal number that can be used as unique archive identifier. This number is derived from the time (first 8 digits) and the process ID (last 4 digits).

`volume`:

Which volume of the archive this tape belongs to.

`release`:

The release of the BRU software used to create the archive.

`variant`:

Information about the particular variant of BRU. A revision control string.

`bufsize`:

Buffer size (in bytes) used to create this archive.

`msize`:

Media size (in bytes) in effect when archive was created.

`msize_blks`:

Same as above, except media size is specified as the number of 2K blocks.

`serial_number`:

The serial number of the BRU license doing

`device`:

The name of device used to create this archive.

`user`:

The user who created this archive.

`group`:

The primary group of the user who created this archive.

`system`:

Information about the system on which this archive was created.

`bru`:

Information about the version of BRU used to create this archive.

`command_line`:

The command used to create the archive.

Archive Verification

Reporting File Differences: `-d`

If the `-d` (differences) option is included on the BRU command line, BRU compares the archive (tape) file with the corresponding file on the disk. If any differences are found, they will be reported and the information will be written to `stdout`. The amount of information reported is determined by the difference level (the number of `-d` options) and the verbosity level (number of `-v` options).

NOTE: The `-d` option will read the archive first, and then compare it with the corresponding file on disk. If there is no corresponding file on the disk, BRU issues a warning message. If the disk file differs from the archived file, BRU reports the difference.

If there are disk files that are not on the archive tape, BRU is not aware of the files and no messages will be generated. Differences will be reported only for files that exist on the archive tape.

BRU counts the number of files that are different and lists the count in the execution summary. If there are no differences, the count will be zero. Also, if any differences are found, the exit code returned by BRU will be set to 1.

You can use the `-d` option when you create an archive:

```
bru -c -d
```

Or you can use it alone, to check which files have changed since the archive was created:

```
bru -d
```

Using the `-d` option causes BRU to report whether a file has changed either its size or its contents. BRU compares the file and the archive as byte-streams, that is, it treats both files as a stream of characters and reports any difference between the two.

Like `-v`, `-d` allows you to specify levels of difference checking. If you call BRU with the -dd option, in addition to comparing the files and the archive as byte streams, BRU reports the following:

- Differences in the file modification date (`mtime`)
- Changes in the access mode (`chmod` has been used on the file)
- Changes in the number of links for non-directory files
- Differences in the contents of symbolic links
- Differences in the owner identification (`chown` has been used on the file
- Differences in group ID (`chgrp` has been used on the file)

When you specify the option as `-ddd`, BRU also reports

- Differences in host device
- Differences in major/minor device for special files
- Time of last access for regular files

If the option is specified as `-dddd`, BRU reports all other differences except for the time of the last status change, major/minor device numbers for non-special files, and size differences for directory files (there may be empty directories).

The `-dddd` mode is generally meaningful only during a verification pass of a full backup on a filesystem that no users are currently accessing (that is, no files are being created or modified).

Difference Mode Examples

The following example illustrates how to check for files that have changed their size or contents. In this case, we want to list only the names of the files that are different:

```
$ bru -d
“./file1”: size contents
```

The next example is similar, except that single level verbosity is specified. In this case, all the files will be listed and each difference will be reported on a separate line:

```
$ bru -d -v
./file1
“./file1”: file size different
“./file1”: file contents different
./file2
./file3
```

Now we’ll increase the difference checking level to detect any changes in the file modification date or in file permissions. No verbosity is specified, so only the differences will be reported:

```
$ bru -dd
“./file1”: size mtime contents
```

Any combination of `-d` and `-v` options can be specified. You can also use the `-V` (capital V) option to generate an execution summary that shows the number of archive files that are different. Here’s an example:

```
$ bru -d -V
**** bru: execution summary ****
Started: Tue Mar 14 08:50:19 1995
Completed: Tue Mar 14 08:50:53 1995
Archive id: 2f65ba6b0193
Messages: 0 warnings, 0 errors
Archive I/O: 0 blocks (0Kb) written
Archive I/O: 110 blocks (220Kb) read
Files written 0 files (0 regular, 0 other)
Files read 66 files (66 regular, 0 other)
Files in headers: 60
Write errors: 0 soft, 0 hard
Read errors: 0 soft, 0 hard
Checksum errors: 0
Difference count: 1
```

Using BRU To Find System Problems

Many UNIX system problems are caused by files that have changed (or are missing). BRU’s differences mode can often help you in trouble-shooting these types of problems-if you do a little advance preparation. First, use BRU to create a “reference” backup of your entire system (or just the basic UNIX system programs in the root filesystem). This should be done when your system is properly configured and working properly. Save this reference archive for the future (make sure that the tape is write-protected and properly marked with a label).

If you experience system problems later on, you can use the reference archive with the `-d` mode to detect any changes in your system. Quite often, this will help you in tracking down difficult problems. The reference archive tape might also be useful in case you need to recover from a system crash.

AUTOSCAN Verification

A large percentage of backup problems are caused by tapes (or other media) that are unreadable. If your tape drive goes bad, it can write tapes that contain errors. Or your tape may contain bad spots. Tape drives are notoriously poor at detecting errors while writing. Usually no errors will be reported and problems will not be detected unless the tape is actually read.

Unfortunately, this is often too late. You may be trying to restore the only copy of a file from a backup tape that was never verified. Unless your archive tapes are verified immediately after you create the archive, any tape drive (or bad tape cartridge) problems may go undetected. Without verification, your backup tape may be worthless.

The AUTOSCAN feature automatically detects problems of this type. AUTOSCAN is enabled by default. With AUTOSCAN enabled, BRU will automatically rewind your tape and “scan” it by reading all the data that was just written. It will read each block of data and verify the checksums. If there are no problems, BRU will continue with the backup and ask for the next volume (if any). If errors are detected, BRU will issue warning messages.

The AUTOSCAN feature is normally enabled when the `-c` (create mode) option is used.

When AUTOSCAN Is Disabled

AUTOSCAN is disabled for devices that have the `norewind` field specified in the `brutab` file (since BRU cannot rewind the device, it won’t be able to read it). It is also disabled when writing to devices with unknown parameters (devices that are not listed in the `brutab` file).

AUTOSCAN is automatically disabled if the `-i` or `-d` options are specified – that is, if BRU is invoked in inspect mode or in differences mode. (See Appendix C, “The BRU Manual Page,” or Appendix D, “Table of BRU Modes and Options,” for a description of all of BRU’s modes.) Since these options also read the tape and detect any errors in the data, AUTOSCAN would be redundant. You can disable AUTOSCAN by specifying the `noautoscan` field as part of the `brutab` entry for a device. This is not recommended, however. For most tape drives, this scan of your archive tape will take only a few extra minutes (it is typically much faster than writing) and it can help you avoid many future problems.

Chapter 7 - Extracting Files: The Restore Function

This chapter describes the command line options used to get files out of an archive—BRU's restore mode. The `-x`, or extract, option allows you to do a full restore or a partial restore. You may want to do a full restore, for example, when a hard disk has crashed and you have replaced it with an empty formatted disk. You may want to do a partial restore to install a software update, or more commonly, when a user has deleted a file accidentally.

This chapter follows the same outline as Chapter 4, "Archiving Files: BRU's Backup Function." Since many of the command line options are used with both backup (`-c`) and restore (`-x`) modes, they will be summarized in this chapter—but the examples will be exclusively of their use with the extract option. You may want to look at the corresponding sections in Chapter 4, or at the BRU manual page in Appendix C, for a more complete description of these options. We've organized the manual this way for two reasons: First, separating the backup and restore functions is more useful for reference. Although backup and restore command lines may include some of the same options, their use is different. Second, by the time you are restoring files, you will be familiar with many of the command line options common to both backup and restore functions. You will not need to hear about them again in detail.

The options that can be used with the `bru` command allow you to tell BRU:

- That you want to restore files
- What device to use
- The verbosity level
- To generate log files
- To wait for confirmation of each action it takes
- To change the ownership of extracted files
- To specify files using wildcard name expansion
- To extract files regardless of dates
- To extract only files that do not already exist on the system
- To extract files from an archive using a list
- To extract files from an archive based on modification date
- To extract files from an archive based on ownership
- Where to put the extracted files

NOTE: In this section we talk about extracting files from archives. If you are unfamiliar with this way of referring to the restore function, see the "Definitions" section of Chapter 1.

The BRU Command Line

This is the BRU command line format:

```
bru modes [options] [files]
```

`bru` is the command itself and modes tells BRU, for example, whether you are backing up or restoring files. The most commonly used options specify the device and the level of verbosity. The files tell BRU which files or directories to extract. The sections described in this chapter show how each of the most common options is used.

Command Line Options: Modes

The mode tells BRU the action to perform. For example, the `-x` mode extracts files from an archive, the `-t` mode lists the contents of an archive, and the `-c` mode creates an archive. There must be at least one mode option on the BRU command line. This chapter describes BRU in extract (restore) mode.

Telling BRU To Restore: `-x`

STOP! Before you extract files from an archive, write-protect your tape (or other media). This will avoid data loss that can result from entering the wrong command. It's easy to make a mistake, like entering `bru -c` instead of `bru -x`.

The `-x` mode tells BRU that you want to extract files from an archive. It does not take a parameter. All uses of `bru` to extract files must begin with

```
bru -x
```

The command line shown above is the simplest extract command you can execute. It assumes two things: First, since it does not specify a device, it assumes that your `/etc/brutab` file has been set up to specify a default device. (See "Telling BRU What Device To Use: `-f`," below; Chapter 3, "Setting Device Parameters: The `brutab` File "; and Appendix E, "A Sample `brutab` File.") Second, it assumes you want to use BRU's default value for files. When reading an archive, the default for BRU is to select all files in the archive. **NOTE**: By default, BRU will **NOT** overwrite files unless the file on the archive is newer than the file on disk (refer to the `-u` option).

Command Line Options: Control Options

Telling BRU What Device to Use: `-f device`

The -f option is used to specify the archive device. If the `-f` option is not used, BRU will use the first device listed in the `brutab` file. The example below will create an archive from the files in the `/usr/src` directory, using a tape inserted into the drive named `/dev/rmt0`. The `-f` option requires a parameter.

```
cd /usr/src
bru -c -f /dev/rmt0
```

Setting the Verbosity Level: `-v`

The `-v` option tells BRU to print information about the procedure it is performing. The `-v` option provides five levels of verbosity and a summary mode: `-v`, `-vv`, `-vvv`, `-vvvv`, `-vvvvv`, and `-V` (capital V), where a single `v` represents the lowest and five `v's` the highest verbosity level. `-V` (capital V) produces an execution summary. The following is an example of verbosity output:

```
$ cd /
$ bru -xvvvv -f /dev/rct0
x   2k of   4k [1] ./
x   2k of   6k [1] ./bin
x  72k of  78k [1] ./bin/acctcom
x  68k of 146k [1] ./bin/adb
x  36k of 182k [1] ./bin/ar
x 120k of 302k [1] ./bin/as
  . . .
$
```

The first field in the output contains the mode character, as a reminder of what mode is currently running (in this case the "`x`," or "`extract`" mode). The next field gives the amount of space in kilobytes this particular file will use in the archive. The next field ("of `XXXk`") gives a running total of the amount of data in the archive, including the current file. Note that the first value is 4k. This represents the 2K header block for directory "`/`" and 2K for the archive header block. The number inside the square brackets is the current volume number. Finally, the name of the current file is listed.

Generating Log Files: `-v`

If you want to keep a record of how a particular tape archive was extracted, you can redirect the output of any of the verbosity options described above to a log file. For example:

```
bru -xvf /dev/rmt0 > extract.log
```

This will write a list of the extracted files to a file named `extract.log`.

Label Option `-L`

The `-L` option during restores is used as a comparison. BRU will read the label string given and attempt to match it to the label of the tape or archive device. If the label does not match on the first tape, BRU will abort the operation. If the label does not match on subsequent tapes, a warning is issued, but the extraction or restore will continue.

There are two ways of providing this label to BRU. The first is to simply specify the label from the command line as shown below:

```
bru -xv -L “Test Label” /usr/bin
```

The second method of providing the label string to BRU, is to create a file and tell BRU to read that file. In this fashion BRU will read in the first 63 characters of the file as the label for the tape as shown below:

```
bru -x -L “/tmp/label name” /usr/bin
```

The label information appears in the archive header, which can be viewed (or fed to an archive manager program) using BRU's `-g` option. **NOTE**: That the label string and file name are enclosed in a pair of double quotes. This assures that any shell program you might be running (`sh, csh, ksh, tsh`) will treat the label as a single parameter. If the label is not enclosed in double quotes, BRU will see the individual components of the label string as separate parameters. In the second case, `/tmp/label` would become the label, and each of the words in the rest of the label - in this case 'name' - would be treated as a file name.

Command line override option: `-Q`

The `-Q` options along with the `-x` "Restore" option change default BRU operations as described below. Below we have laid the options out in the command line format `-QL` for instance.

`-QL`

Use a literal string as a tape label. This will override BRU's attempt to look for a file from which to read the tape label. This can be useful if you have a file that is the same name as the label you wish to apply to the restore command.

`-QR`

Disable SmartRestore. This option turns off BRU's handling of open or shared files. It is not recommended that you override this setting

`-QS`

Translate Symbolic Links. This option is used in conjunction with the `-T` option will force the translation of symbolic links as explained in the `-T` option found above.

`-QV`

Ignore “Incorrect Volume” warnings. When restoring from a multi-tape set or beginning a restore from other then the first tape is a set, BRU will normally issue a warning message. Using this option BRU will not issue the warning message and continue the restore process.

Depth Selection on Restore Option: `-U`

This option allows you to set the depth at which BRU will work during its restore process. Files more then the -`U(#)` levels will not be processed. For example, if the current directory is `/home/webmaster`, `-U0` (zero) will only restore the files and directory nodes in the `/home/webmaster` directory. While the directory node (name) will be restored the files in side each directory will not be restored.

With a level of `-U2` (two), BRU will restore `/home/webmaster/test`, `/home/webmaster/test/runone`, `/home/webmaster/test/runtwo`, but not restore `/home/webmaster/test/runone/demo1`. In this configuration BRU can be used to easily restore a subdirectory from a total system backup.

Confirmation Option: `-w`

When the `-w` option is specified, BRU will wait before attempting to extract a file. For each file, it will do the following:

Print the file name, show the action it is about to take, ask for confirmation. BRU also provides a special response that lets you change your mind. If you type a `g` (for “go ahead”) instead of `y` or `n`, BRU will stop asking for confirmation. The following example shows how this response is used:

```
$ bru -xvw -f /dev/rct0
x ./: please confirm [y/n/g] y
x   2k of   4k [1] ./
x ./bin: please confirm [y/n/g] y
x   2k of   6k [1] ./bin
x ./bin/acctcom: please confirm [y/n/g] y
x  72k of  78k [1] ./bin/acctcom
x ./bin/adb: please confirm [y/n/g] y
x  68k of 146k [1] ./bin/adb
x ./bin/ar: please confirm [y/n/g] g
x  36k of 182k [1] ./bin/ar
x 120k of 302k [1] ./bin/as
x  16k of 318k [1] ./bin/asa
x   4k of 322k [1] ./bin/basename
. . .
```

Changing Ownership of Extracted Files: `-C`

BRU stores the user and group ownership with the archived files and normally restores files with the original user and group ownership. In some cases it may be necessary to change the ownership of restored files to that of the user and group running BRU. The `-C` option is not an option you will normally use. It is typically used for extracting files from tapes that were written on a different system (with different user and group IDs). The ownership of such extracted files can be changed by using the `-C` option (be sure you use a capital C). For a full description, see Chapter 8, “Advanced Uses,” “Ownership of Extracted Files.”

Command Line Options: File Selection

There are several ways to specify which files you want to extract. For example, you can:

- Use wildcard file name expansion
- Flag files to be extracted unconditionally-that is, without regard to date
- Tell BRU not to overwrite any existing files
- Extract files read from the standard input
- Extract files by date
- Extract files by user
- Use include/exclude patterns
- Tell BRU where to put extracted files: Relative and Absolute pathnames

Be careful to specify file names last. If your file specifications are not in the correct position on the command line, the results of the command may be unexpected. If files are not specified, BRU defaults to all files on the archive tape. That is, the command

```
bru -x
```

is equivalent to

```
bru -x “*”
```

Using Wildcard File Name Expansion

BRU recognizes wildcard strings in the same format as recognized by the Bourne or Korn shells. However, when such strings are passed to BRU in extract mode, they must be quoted to prevent their expansion by the shell. For example, the following command will extract all files in any subdirectory one level below the current directory that start with any character between “`a`” and “`h`” and end with “`.c`”; it will also extract all files in any subdirectory three levels down:

```
bru -x “./*/[a-h]*.c” “./*/*/*”
```

The command in the next example will extract all files in the `./etc` subdirectory that do not end in “`.old`” (the “`!`” operator is a BRU enhancement of wildcard expansion that does not exist in the shell):

```
bru -x “./etc/!*.old”
```

Use the “`!`” operator with extreme caution. In particular, since each pattern is applied independently to determine a match, be wary of including more than one pattern with a “`!`” operator on the command line. For example, the pattern “`!file1 !file2`” will match all files, including files “`file1`” and “`file2`”. Since “`!file1`” matches pattern “`file2`” and “`!file2`” matches pattern “`file1`”, you have defeated the exclusion.

The “`!`” is mostly useful for simple exclusion tasks such as excluding a single file. For other file exclusions, it is usually easier to use the `-X` option. See Chapter 5, “File Inclusion and Exclusion.” **NOTE**: Wildcard characters used to specify file names in `-x` mode must be enclosed in double quotes. Wildcard characters used with `-c` are not quoted.

Using Include/Exclude Patterns

The `-X` option can be used to select files when restoring. Refer to Chapter 5, “File Inclusion and Exclusion,” for details.

Unconditional File Type Extraction: `-u flag`

Normally BRU will not supersede existing files while extracting files from an archive - that is, BRU will not overwrite an existing file with an older archived file of the same name. Specifying the `-u` flag option causes files of the type specified by the flags listed below to be unconditionally selected for extraction. Such files will overwrite any existing files of the same name regardless of modification times. If the verbosity level is two or higher (`-vv` through `-vvvv`), BRU will print a message for each file that is not superseded. The confirmation option (`-w`) may also be used with the `-u` flag option.

The available flags are:

`a`	Select all files
`b`	Select block special files
`c`	Select character special files
`d`	Select directories
`l`	Select symbolic links
`p`	Select fifos (named pipes)
`r`	Select regular files
`f`	Select regular files (the same as `r`)

Multiple flags select all files that match any of the given types. For example, `-ubcd` will select all block, character, and directory files. The `-ua` option will select all files and is equivalent to specifying each of the other individual flags.

Existing directories are never overwritten. If you select directories, only their attributes may be modified. The `-u` option merely allows directory attributes to be set back to some previously existing state.

If you select symbolic links, only the contents of the link will be modified. It is impossible under some versions of UNIX to change the access time, the modification time, or the file mode of a symbolic link.

The following example will extract all regular files in an archive, regardless of their modification dates. Any existing files with the same names will be overwritten. If any of the extracted files has been changed since the archive was created, all such updates will be lost.

```
bru -x -ur
```

The next example specifies that all of the files under directory `./usr/doe` are to be extracted from the archive on the default device, and are to unconditionally overwrite any existing files with the same names:

```
bru -x -ur ./usr/doe
```

The `-E` File Extraction Option

When extracting files from an archive, BRU will normally replace an existing file only if the corresponding archive file has a more recent date. For example, if the existing file has a modification date of January 1, 1990, it would be overwritten by an archive file (of the same name) with a date of February 1, 1990. Most of the time, this is the desired behavior.

In some cases, you may not want to overwrite an existing file, even if the archive file is newer. This is often true when doing software updates. For example, a software update may contain configuration files with dates that are newer than existing configuration files. If these carefully created configuration files are overwritten, the software may not work properly and the configuration files will need to be re-created.

The `-E` option lets you avoid these problems. If `-E` is specified on the extract command line, BRU will not replace any existing file, even if the archived file is newer. It will only extract files that do not currently exist. The following is a typical use of the `-E` option:

```
bru -xv -E
```

Of course, if you really do want to extract everything from an archive, use the `-ua` option. This will perform an unconditional extraction. BRU will overwrite all files with the same names as the archived files-regardless of the date. See "Unconditional File Type Extraction: `-u`," above. **NOTE**: The `-E` and `-ua` options should not be used together.

Extracting Filenames Read from `stdin: -`

The hyphen (`-`) is a special filename. When it is specified on the BRU command line in place of the files option, it tells BRU to read the standard input stream (`stdin`) for a list of files to process rather than collecting them from the command line. The rules governing the use of the hyphen as a special filename are the same for both reading and writing archives. This usage takes the form:

```
bru -x [options] -
```

This option is typically used in conjunction with a UNIX pipe or with the standard input stream redirected to a file containing a list of filenames. The commands

```
cat /tmp/filelist | bru -x -
and
bru -x - </tmp/filelist
```

are equivalent and extract the specific list of files you have named. **NOTE**: `filelist` must contain all the files, that you wish to restore (not just the directory names). `filelist` must also be in the same order. You may wish to list the files with '`bru -tv >filelist`' then edit `filelist` for the files you want to restore. **NOTE**: BRU will read your list of files BUT WILL NOT DO WILDCARD EXPANSION of that list of files. BRU expects the list of files to be completely expanded when it receives the list. Therefore WILDCARDS CAN'T BE USED in your list of files for restoring data. **NOTE**: If you use a hyphen (`-`) in place of a list of files on the BRU command line, you are not allowed to use the "`-f -`" option to read an archive from the standard input on the same command line. These two usages conflict. The following command is illegal:

```
bru -x -f - - <filelist
```

It is not possible for BRU to read both the archive and the list of files from the standard input.

Extracting Files by Date: `-n date`

The way dates are specified is the same for both backup and restore functions. The command line differs only in the use of

-c for backup and `-x` for restore. In the case of a backup, you are asking BRU to archive files created or modified after the date you specify on the command line (see Chapter 4, “Archiving Files: The BRU Backup Function,” “Selecting Files by Date: `-n`”). In the case of a restore, you are asking BRU to extract files from an existing archive based on the dates on which the files were last modified.

For the details of date specification, see Chapter 4.

Inverse Date Specification

If you need to extract files older than a given date, you can do this by placing an exclamation point (`!`) immediately before the date specification on the command line. The exclamation point reverses the sense of the match. For example,

```
bru -x -n !14-Mar-92
```

will extract all files created or modified before March 14, 1992.

Extracting Files by User: `-o`

File extraction can be limited to files owned by a particular user in the same way files are selected by user when you are creating an archive. The `-o` option lets you tell BRU the identity of the user whose files you want to extract from an archive. BRU accepts the following three forms of user identification:

- `User name`
- `Numeric user ID`
- `File owner`

Although these three ways of using the `-o` option were discussed in Chapter 4, “Archive Creation: The BRU Backup Function,” examples of their use in restoring files may be useful here. If you have questions, see Chapter 4, which goes into greater detail.

Extracting Files by User Name: `-o username`

The user name must correspond to a user in one of the password files. It identifies the owner of the archived files you want to extract. This will extract all files owned by the user bob:

```
bru -xv -o bob
```

Extracting Files by User ID: `-o uid`

You can also specify the owner by the decimal identifier derived from the `uid` value in the `/etc/passwd` file. The decimal value is used less often than the other forms of the -o option. This command line will extract all files owned by the user with user ID 112:

```
bru -xv -o 112
```

In some circumstances, it is possible for files to have user ID numbers that do not match any user listed in the `/etc/passwd` file.

Extracting Files by File Owner: `-o pathname`

If you use the pathname to a file as the argument to the `-o` option, BRU assumes that the owner of that file is the user whose files are to be restored. BRU selects all the files that are owned by the owner of pathname. For example, if owner.file is owned by bob, BRU selects all files owned by user bob:

```
bru -xv -o owner.file
```

Where To Put the Files: Pathnames

The placement of extracted files is largely determined by the way in which files and directories were specified when the archive was created. Archived files may have been stored using either relative pathnames (pathnames that begin with a "`./`" or an alphanumeric character) or absolute pathnames (pathnames that begin with a "`/`" character). An absolute pathname always begins at the root of the system.

If the archive was created by specifying an absolute pathname, for example, `/usr/me/myfile`, then the files will be restored in that exact location.

If the file specification at the time of archive creation, did not begin with the "`/`" character, for example:

```
./currentstuff/myfile
```

or

```
currentstuff/myfile
```

then the files will be restored in the current directory when BRU is run to extract them. In the above case, if `/tmp` is the current directory, then the file will be restored as

```
/tmp/currentstuff/myfile
```

If you specify a relative path when you back files up, then the BRU archive tape can be used to restore a filesystem to a different device than the device used to create it. An entire directory can be moved from one machine to another without first determining that there is enough space on a particular filesystem to install it. You simply have to choose a filesystem where there is enough space and issue the command to extract the directory from a location in that filesystem. The relative pathnames of the files assure that the installation occurs in the "current directory."

There are no particular advantages to specifying absolute pathnames when creating a BRU archive, and in general, absolute pathnames should be avoided.

NOTE: When trying to match filenames,

```
abc/xyz
```

is **NOT** equivalent to

```
./abc/xyz
```

The filename must be specified in exactly the same way as the name of the file stored in the archive. You can use

```
bru -tv
```

to see how the named is stored. See Appendix C, "The BRU Manual Page," for a description of the `-t` option.

Translate on Restore Option: `-T file`

Translate on restore allows the user to translate, rename or relocate files based on the content of a translation file. The file should be created in a two-column format. The first column should list the files you want to have translated, the second column should list the new directory or name you want the files translated to. This translation applies to directories, names and extensions. By default, symbolic links will not be translated. See the `-Q` option mentioned earlier in this manual. To help in the understanding of this option we will work with this example. To restore all of the files from /home/bill and place them in `/home/paul`, you would create a file that had the following text. For this example we will call this file trans.

Contents of the file trans

```
/home/bill /home/paul
```

Command line:

```
bru -xvvv -T/dir/trans /home/bill
```

No further interaction is then needed. During the restore BRU will translate all files in the `/home/bill` directory to the new `/home/paul` directory. You can also have `bru` handle multiple translation. Simply by adding more lines to the same file you could have `bru` translate all of the files from many directories to different directories. See example below:

```
/home/bill /home/paul
/usr/lib         /usr2/lib
/home            /u2
```

The translate file can contain as many translation lines as necessary, but each line must consist of a pair of entries. **NOTE**: When using BRU with the Translate on Restore option and the `-PA` option, BRU will translate those files inside of the translate file and convert the remainder of the files with the `-PA` option as described below.

Converting Pathnames from Absolute to Relative Form: `-PA`

If you specified an absolute pathname for the files you backed up (an absolute pathname is one that starts with a slash character "`/`"), then BRU will attempt to restore them in the root directory with exactly the same pathname. This is fine, unless you want to move the files to a different directory.

If you want to restore files with absolute form pathnames in a different directory, you must use the `-PA` option. The `-PA` option will translate the leading slash character from an absolute pathname to "`./`" and permit extraction into the current directory.

Here's an example of how the `-PA` option is used. First we back up a file with an absolute format pathname:

```
bru -cvf /dev/tape /etc/termcap
```

Now we restore the file to a different location:

```
cd /u/mydir
bru -xv -PA -f /dev/tape
```

This creates the file `/u/mydir/etc/termcap`. If the `-PA` option had not been used, the file would have been restored as `/etc/termcap`.

The `-PA` option is needed only if you created an archive with absolute pathnames. It has no effect if the archive already contains pathnames in relative format (BRU's default mode). **NOTE**: To extract individual files using the `-PA` option, you must specify the filename without a leading "`/`" character. For example, if a file was archived as `/usr/bin/man`, the normal command to extract a single file,

```
   bru -x /usr/bin/man
```

will restore the file to that exact location. The command you must use to restore the file to the current directory at the time of extraction is:

```
   bru -x -PA usr/bin/man
```

Note the difference in the files argument.

Chapter 8 - Advanced Uses

The most important uses of BRU-creating archives, inspecting or verifying archives, and extracting files from archives have been covered in earlier chapters. In this chapter, we describe some of the more advanced uses of BRU. Among these advanced uses are:

- Using BRU's internal Encryption
- Using a different I/O buffer size
- Using an explicit media size
- Reading archives from `stdin` or writing archives to `stdout`
- Using multiple archive devices
- Controlling ownership of extracted files
- Ways to increase BRU's speed
- Saving archive space by using data compression
- Setting up BRU to run with `brutalk`
- Setting the interaction options
- Running BRU from `cron`
- Controlling BRU's I/O streams
- Tuning BRU for shared memory
- Using BRU to save and restore sparse files
- Using BRU with a bootable floppy
- Using BRU with remote backup devices
- Using BRU to do "live" system backups

Using BRU's Internal Encryption: `--key or –key=`

Telling BRU the Buffer Size: `-b bufsize`

You can tell BRU to use a specific buffer size for reading and writing data. Normally BRU uses a default buffer size if no buffer size for the output device is specified in the `brutab` file. However, you can change the value BRU uses by specifying the `-b` option on the command line. This option specifies the size of the data block that is written or read with each access to the archive device. When writing, BRU stores the buffer size information in the archive header. When reading the media, BRU will use this buffer size (unless a different value is specified by the `-b` option).

When you use the `-b` option, you tell BRU to use a buffer that is `bufsize` large. As with other numeric parameters, you can specify the size in bytes, kilobytes, or megabytes. A typical value is 32K, used as the default value. When you provide a parameter to specify the buffer size, that parameter must be an even multiple of 2K (that is, a multiple of 2048 bytes). A value of 2K is the absolute minimum size that BRU will accept. The maximum value is determined by your hardware and system software. If you specify a value that is not a multiple of 2K, BRU will round up to the next direct multiple of 2K. Here is an example that specifies a buffer size of 64K:

```
bru -b 64K [options] [files]
```

NOTE: Some tape devices have fixed buffer sizes. If you have such a device, you must set the BRU `bufsize` to a multiple of the device's buffer size or your write attempts will fail with a write error. If you experience errors, try adjusting the buffer size and writing small test backups until the write succeeds. This value, or some multiple of this value, will be the best setting for the device.

Setting the Archive Media Size: `-s msize`

You can use the `-s` option to tell BRU the capacity of the media you are using. With most modern operating systems and tape devices, this setting is not necessary. However, you may wish to set the media size for devices like removable disks or disk files. **NOTE**: When you use this option, it overrides any value you may have specified in the `brutab` file with the `size=` parameter. If you have two or three different sizes of tape that you use for your backups, but only one tape drive, you can use the `-s` option each time you run BRU to specify the size of the tape you'll be using. However, this is tedious, and specifying a size larger than the actual size is a serious error.

A better solution is to create "aliases" for the tape drive, and change your `brutab` file appropriately. Simply duplicate the basic `brutab` entry for the device, but change the device name and size= parameter. Here's an example:

```
/dev/tinytape size=10M [other parameters]
/dev/medium   size=20M [other parameters]
/dev/bigtape  size=30M [other parameters]
```

Now create links for each new device name:

```
$ cd /dev
$ ln st0 tinytape
$ ln st0 medium
$ ln st0 bigtape
```

The paths `/dev/st0`, `/dev/tinytape`, `/dev/medium`, and `/dev/bigtape` all point to the same device, but BRU considers them to be separate devices with the characteristics given in the `brutab` file.

When the `-s` option is specified, it overrides any other default value, even that read from the tape header during a read or scan of the archive. The value that you give with the -s option should be an integral multiple of the BRU buffer size. However, BRU does not indicate any error when the value is not a multiple. Instead, BRU calculates the effective media size by silently rounding down to the nearest multiple of the buffer size.

One very handy use of the `-s` option is to create an archive in a set of normal files, for transfer via modem or email to another system (using mail, ftp, or another file transfer utility) where the receiving system limits the size of incoming attachments. By splitting the archive into several smaller pieces, large archives can be transmitted in small chunks, reducing retransmission overhead in case of errors. For example:

```
bru -cv -s600K -f bru1 -f bru2 -f bru3 [files]
```

will cause BRU to write its archive into the files called `bru1`, `bru2`, and `bru3`. Each file would have a maximum size of 600K bytes. These files could then be sent to another system (remote) with the UNIX `scp` (secure copy) command:

```
$ scp bru1 user@remote:/usr/spool/uploads
$ scp bru2 user@remote:/usr/spool/uploads
$ scp bru3 user@remote:/usr/spool/uploads
```

and unpacked on that system with the commands:

```
$ cd /usr/spool/uploads
$ bru -xv -f bru1 -f bru2 -f bru3
```

Reading `stdin` or Writing `stdout` (standard I/O streams)

If the pathname supplied as the parameter for the `-f` option is a hyphen (`-`), then BRU uses the standard input (`stdin`) for archive reading or standard output (`stdout`) for archive writing, as appropriate. When extracting files from an archive, (`-x` option), the "`-f -`" causes the input archive to be read from `stdin`. The filenames to be extracted must be in the same order as they appear in the archive (or files will be skipped). If the filenames are not in the same order, you must use the `-Pf` option. This will cause BRU to sort the names (by building an internal filename table) before it does the extraction.

An example would be extracting a `gzip'd` BRU file in a single command line:

```
gunzip -c gzipfile.bru.gz | bru -xvf -
```

This allows `gunzip` to expand the compressed file and pass the results via `stdout/stdin` through to BRU for extraction.

To perform the reverse operation, you can instruct BRU to send its output to `gzip` for compression as follows:

```
bru -cvf - /files | gzip > brufile.bru.gz
```

The resulting file will be a BRU archive that has been compressed with `gzip`. This can also be accomplished using BRU's external compressor option, described later in this chapter.

Using Multiple Files/Devices: `-f device`

You may use the `-f` option multiple times on the same command line to tell BRU to use multiple files or multiple paths (which could be devices, for example) to store the archives. If multiple `-f` options are given, each path is added to a list of paths to cycle through each time a volume change is required. When the end of the list is reached, BRU automatically cycles back to the first path and waits for confirmation to continue the cycle again. Any input other than a carriage return will cause BRU to use the newly entered path and to abort the cycling for the remainder of the current run. This feature is known as device cycling.

The following example will do a full system backup (starting at root). It will first write data to `/dev/tape1`. When it is full, it will automatically continue on `/dev/tape2`. **NOTE**: When using this option all devices need to be the same size as the first device. BRU takes the size of the first device and uses that size for all other device when device cycling starts. If you specify a size of zero (0), BRU will then continue to write to each device until it receives an end of tape message. Then, BRU will switch to the new drive or device:

```
bru -cv -f /dev/tape1 -f /dev/tape2
```

To do a multiple device restore, the command is similar:

```
bru -xv -f /dev/tape1 -f /dev/tape2
```

BRU commands run from `cron` often use the device cycling feature. This allows BRU to get as much work done as possible before interaction is necessary.

Device cycling can be used with the global `brutab` parameters `MOUNTCMD` and `UNMOUNTCMD` to allow writing to jukeboxes or tape libraries. If these parameters are used, then BRU will continually cycle through all the devices until it has completed reading or writing. It will not wait between devices or when it returns to the first device-any waits should be done by the shell scripts specified by `MOUNTCMD` and `UNMOUNTCMD` see Appendix K, “Using `MOUNTCMD` and `UNMOUNTCMD`”.

Ownership of Extracted Files: `-C`

When extracting files, BRU restores the owner and the group to that which is stored in the archive for each file. By specifying the `-C` option, you instruct BRU to change the owner and group ID to that of the person running BRU, including that of root. If the system administrator root is performing a filesystem restore, this option should not be used, since root would end up owning all of the files.

The `-C` option is useful when you are importing files and directories from other systems because it allows the person running BRU to assign all of the files to himself. It is best to check the archive by running BRU with the `-tv` options to see who owns the files in the archive and thus to determine whether or not the `-C` option is appropriate to use.

BRU enforces standard UNIX access security. It makes every attempt to prevent normal users from archiving or extracting files to which they might not normally have access. If a user does not have read access to a file, that user will not be able to back it up, restore it from an archive, or change its ownership with the -C option. **SECURITY NOTE**: Anyone with physical access to the archive media and a knowledge of the structure of the file records can recover the archive’s contents by writing their own file extraction program. If data security is important, you should protect the archive media from unauthorized physical access.

Increasing BRU’s Speed

BRU has a number of options that can, at times, increase the operating speed of the program. These options include using a larger I/O buffer and using double buffering.

Before attempting to tune BRU for optimum performance, you should know the maximum speed of your archive device. On many newer systems, BRU is limited by the speed of the tape drive. If your tape drive is already operating at top speed, it is impossible to adjust BRU’s parameters to make it run faster.

Listed below are typical maximum speeds for various types of tape drives:

Drive Type	**Speed**
DDS4	4 MB/sec
8 mm	500 KB/sec
SLR100	2.5MB/sec
VXA-1	3MB/sec
DLT8000	6MB/sec
LTO	2MB/sec
Mammoth 2	12MB/sec

The above speeds are the native throughput for the drives listed. Hardware data compression will increase these values depending upon how well the data compresses. The numbers are based on information gathered from the tape drive manufacturers. This list is meant only as a guide, the actual maximum speed of your tape drive may be different. To obtain the rated maximum speed for your drive, refer to the manufacturer's specifications.

During the I/O operations, BRU calculates the tape read/write speed and records the value in its execution log, `/var/log/bruexeclog`. Here is a typical log entry (this would be on a single line in the log):

```
20010903 04:19:10|4331|root|[L182] wrote 256670 blocks on volume [1],
0:48:59, 174 Kb/sec
```

In this case, BRU was writing to a 4mm DDS1 drive. It took 2,939 seconds (0:48:59) to write 513340 Kbytes (256670 x 2 Kb/block), so the tape speed is 174 Kb/sec. This is very close to the maximum DDS1 speed of 180 Kb/sec, so it is likely BRU is "tuned" for optimum performance. Any further adjustments would probably be futile.

TUNING HINT #1: Make sure you read or write enough data to get an accurate tape speed measurement. Some types of tape drives (like DATs) take several seconds to load the tape before they actually begin reading or writing. This extra time will skew the speed measurement (the apparent speed will be slower) on small amounts of data. To avoid this problem, run your tape speed tests for two minutes or longer. The longer the test, the more accurate the speed measurement will be.

TUNING HINT #2: Do not use BRU's compression (`-Z` option) when tuning for optimum speed. Data compression can be used after you have determined the optimum parameters. If your tape drive supports data compression (hardware compression), it is usually faster than BRU's compression. If this is the case, BRU's compression option should not be used.

Specifying a Larger (or Smaller) Buffer Size

BRU operation can sometimes be performed more quickly if a larger buffer is specified, particularly with streaming tape drives. However, on some systems, a smaller buffer size will result in greater speed. The default buffer size, if the `-b` option is not specified, is read from the `brutab` file. Normally, this is set to `bufsize=32K`. If you use a larger buffer, two things happen:

1: Your performance might improve due to larger blocks being read and written in a single action to the archive device.

2: If you are using a tape unit as the archive device, using a larger buffer may allow you to store more data on the tape since the proportion of tape that stores data goes up as the number of inter-record gaps decreases.

A smaller buffer size (like 2K or 4K) is often faster when using tape devices that have small buffers or on systems with tape controllers or device drivers that do their own buffering.

There are several things that can affect the size of the buffer and therefore the tape block size that is written to the archive:

Available memory size

If you don't have enough free memory to allow this size buffer to be used, BRU will issue an error message and quit.

Device limitations

If your archive device driver has a specified limit on the block size that it can support and you have specified a buffer that is larger than that size, then you may experience problems. Even though the amount of memory will be allocated for BRU to use, reads or writes to the archive device will fail, usually with the error message “no such device or address”.

Improperly specified size

If you specify a size that is not an even multiple of 2048 bytes (also called 2K), the buffer size will be silently rounded up to the next multiple of 2K.

Archive header information

If you are extracting files, the buffer size read from the archive header takes precedence over the default buffer size or a buffer size read from the `brutab` file for the given archive device. Any buffer size option given on the command line takes precedence over any other source of the buffer size.

Double Buffering: `-D`

If your system uses System V shared memory (your system administrator can tell you if it does), then depending on the constraints of the archive device driver, using double buffering (the `-D` option) can sometimes show a dramatic increase in the I/O rate of the archive device. It is not uncommon to see the speed increase by a factor of two. Unfortunately, it is also not uncommon to see the speed decrease slightly also. This feature is very hardware and software dependent, and the only way to discover if you should use it for a particular device is to try it and see if it helps.

Turning Off Error Checking: `-F`

DO NOT USE THIS OPTION! If that didn’t scare you off ... running BRU with the `-F` option disables calculation and checking of error-sensing values. You’ll generally only use this option when you are absolutely sure that the archive media is essentially perfect; that there is no chance for an error to occur in the recording process.

If you record an archive with the `-F` option, you must read it back with the same option. **NOTE**: That some of BRU’s automatic features, such as AUTOSCAN and byte swapping (mentioned in the Introduction), are not functional if error checking is disabled.

The option is provided primarily to facilitate the output of one BRU (perhaps that creates an archive) to be piped into the input of a second BRU, that extracts data from an archive. In this case, BRU becomes a fancy copy program. For example, to copy all of the files from `/usr/u1` to `/usr/u2`, the following command could be given:

```
$ (cd /usr/u1; bru -cFf -) | (cd /usr/u2; bru -xFf -)
```

Note that in this case, the “`-`” at the end of each of the two parts of the command follows the use of the -f option. In this case, the “`-`” does **NOT** mean that BRU should look for a file list to be provided by reading the standard input as explained for the “`-`” option in an earlier chapter. Instead, the hyphen is the file path (see the `-f` option explanation). It may look a little confusing, but BRU knows the difference. If you want a file list to be provided from `stdin`, just make sure the `-f` option does not appear on the command line with a conflicting “`-`” usage. This option can also be used to transfer files to a different system. As long as BRU is installed on both machines. In the example below we are using the `rsh` command. This command may be different on different UNIX platforms.

```
   $ bru -cFf - | rsh hostname “(cd /tmp; bru -xFf -)”
```

Using Data Compression: `-Z`

If you specify the `-Z` option, BRU will use compression to make the final size of the archive smaller. The compression algorithm supports multiple levels of compression - the higher the level, the more aggressively BRU will work to compress your data. Also, the higher the number, the harder your CPU will be tasked to perform the compression. The default is to use level 3 compression, but you may adjust this for your needs by specifying the `-N` level option, where level is the desired compression level (1 to 6) Example.

```
   $ bru -c -Z -N 6 ./
```

NOTE: The `-N` option has no effect unless the `-Z` or `-S` options are specified. Using this compression technique can result in space savings of 0% to 90% or more depending on the kinds of files being stored. Typically, most files will compress about 30 to 50 percent, however sparse files containing lots of redundant data or zeros, such as typical large database files, may compress as much as 90% or more. If `USIZE` is the uncompressed size of a file (as shown by `ls -l`), and `CSIZE` is the compressed size of a file, BRU defines the compression ratio in percent as:

```
   Ratio = 100 * ( 1 - (CSIZE / USIZE) )
```

One experiment, using a “typical” mix of files consisting of some text files, some executable binary files, and some miscellaneous data files, gave the following results:

Typical Compression Ratios:

Level	Archive Size	Ratio
None	7434K	0%
1	4239K	69%
3	4178K	70%
6	4055K	72%

Compression is not the default mode because not all versions of BRU know how to deal with compressed files (early versions of BRU did not support the `-Z` option).

The disadvantage to the use of compression is that it takes time and extra memory to compress or decompress the files. The overall impact on the archiving time depends on the spare CPU capacity of your system. If BRU is spending most of its time waiting for I/O to complete, then compression may have no noticeable effect on your archiving time. On the other hand, if your system is very busy, or has a slower CPU, then compression can easily double or triple your backup time.

One last hint, the `-S` option can be used to turn on automatic compression for files larger than a certain threshold size (see the discussion of the `-S` option later in this chapter). In this case, only those files larger than the specified threshold will be compressed. This is generally much faster than turning on compression for all files, while still providing significant space savings. When using the `-S` option, it is not necessary to specify the `-Z` option. **NOTE**: If your tape drive provides hardware data compression, use it instead of BRU's compression option. Hardware data compression is normally much faster and more aggressive than BRU's compression. Since it is generally not possible to gain anything by compressing data that is already compressed, **DO NOT** use both types of compression. If you do, your tape speed will decrease and tape usage will increase.

Using An Alternate Compressor With BRU

BRU 16.0 and later provide a mechanism which allows you to utilize an external compression engine instead of the built in LZO engine. This allows BRU to be used with more modern compression engines as they become available.

To use an external compressor, BRU looks at the global variable "`BRUZEXTERN`" to determine the program that should be used. You can set this as either an environment variable:

```
BRUZEXTERN="/usr/bin/bzip2" ; export BRUZEXTERN
```

or as a global variable in the `/etc/brutab` file (refer to "Chapter 3 - The `BRUTAB` File" for more information on global variables):

```
#+BRUZEXTERN="/usr/bin/bzip2"
```

The only requirement is that the compression engine accepts `stdin` as the compression source and send the resulting compressed data to `stdout`. Also, you must be able to call the engine for decompression using a `-d` option to the original program (i.e.: `bzip2 -d`).

Once a tape is created using an external compressor, you do not need to set the `BRUZEXTERN` for BRU to be able to extract the compressed archive. The setting for `BRUZEXTERN` is added to the archive header and future uses of BRU to extract files from this archive will know which program to use for decompression automatically.

Using BRU with `brutalk: -I args`

Using the `-I` option, BRU can be set to run interactively with a program such as `brutalk` (see Appendix I, "The `brutalk` Program"). If there is no terminal present, you can direct BRU to write its queries to a fifo (first-in-first-out buffer file) and read responses from a different fifo. The interactive program, in turn, watches the output from BRU and directs its activities accordingly, responding through BRU's input response fifo, just as though the responses were made from an interactive terminal.

When BRU was first installed on your system. The installation created the two fifos listed below. The commands listed below, can be used to create the fifos if for some reason they where not created during install, or you need to create additional fifos.

```
$ mknod /dev/bru.q p
$ mknod /dev/bru.r p
```

To create different fifos i.e. for running multiple BRU jobs.

```
$ mknod /dev/myfifo.q p
$ mknod /dev/myfifo.r p
```

In order to use the `brutalk` feature you will need to add the option listed below to your `bru` command line.

```
-Ib
```

If you created a different set of fifos, (i.e.: if you are running multiple BRU jobs) then you would add the following argument to your BRU command

```
-Iq,/dev/myfifo.q -Ir,/dev/myfifo.r
```

to the desired BRU command line which ultimately gets executed. The first time BRU needs to communicate with an operator, it will open the two fifos, write a query to the `bru.q fifo`, and wait for a response from the `bru.r fifo`. The `brutalk` program, which is a simple program provided with BRU, can be used to read the query and send a reply. Here is the command to start `brutalk` if you are running the standard fifos as explained in the previous example:

```
brutalk
```

If however you are writing to a different set of fifos simply run the example below

```
brutalk [-t ttyname] /dev/bru.r
```

or

```
brutalk [-t ttyname] /dev/bru.q /dev/bru.r
```

The second form is used if you want `brutalk` to timeout after a few seconds if there are no queries ready to be read. The `brutalk` program will continue to read queries and send replies until either BRU exits, or a CTRL-D (EOF) is typed at the terminal. The optional `ttyname` parameter can be used to force `brutalk` to interact with the user via some terminal steam other than `/dev/tty`. See Chapter 4: “Archiving Files: The BRU Backup Function,” “Running without User Intervention: `-B`.” If the `-B` option is used, BRU is forced to run in the background and will terminate if a problem occurs.

The selection of either method depends on what the desired recovery is when interaction is necessary. If you prefer to have the backup terminate and not rerun, then use the `-B` option. However, if you want to be able to recover from an unexpected situation (for example, if the tape is simply off line), and to rerun or continue the backup, then use the `-I` option, and use `brutalk` to resolve the problem when it is convenient. `brutalk` can also be used by someone dialing into the system to check on BRU’s progress, since it can be run by anyone from any terminal. NOTE: Only the permissions on the read/write fifos enforce security.

Running BRU from `cron`

Using the interactive option allows you to run BRU from `cron` (`cron` runs programs at specified times and frequencies). If no interaction with the user is required, running from `cron` is no different than running directly from a terminal. However, when interaction is necessary there are basically two options; terminate, or find some way to communicate with the operator (or another program masquerading as the operator). The `-B` option provides for simple termination while the `-I` option provides for communication with an operator. BRU recognizes the following parameters for the `-I` option:

```
b          default fifos
q,fifoname1      write queries to a fifo
r,fifoname2      read responses from a fifo
```

```
l,filename        write log information to a file
```

You can use the device cycling feature to specify that more than one archive device contains tapes that are ready for a nightly backup. For example, an operator might load a set of tapes onto several drives, and schedule `cron` to begin a daily backup at midnight, when all of the programmers and office staff have gone home. BRU will do as much work as it can without interaction, and then wait patiently for an operator to provide additional information in the morning. As an example, assume that the `crontab` file contains the following `crontab` entry:

```
30 02 * * * sh -c /etc/dailybru /usr
```

and the file `/etc/dailybru` contains:

```
# Backup the specified files.
# Use fifos to communicate if necessary.
FILES=$*
IOPS='-Ir,/dev/bru.r -Iq,/dev/bru.q'
DEVS='-f /dev/rmt0 -f /dev/rmt1'
bru -c $IOPS $DEVS $FILES 2>/etc/dailybru.log
```

where `/dev/rmt0` and `/dev/rmt1` are tape units that the operator loads before leaving.

Every day at 2:30 a.m., `cron` will run the `/etc/dailybru` script, causing BRU to begin backing up all files and subdirectories under `/usr`. BRU will continue to run until it finishes or until it hits the end of the tape in `/dev/rmt1`, at which point it will wait for the operator to communicate with it via the fifos `/dev/bru.r` and `/dev/bru.q`. The file `/etc/dailybru.log` will contain any warnings or error messages issued by BRU.

This use of device cycling allows BRU to use multiple devices without the operator's presence being required. When BRU runs out of space on the first volume, instead of the operator (who has gone home) having to physically mount another volume, BRU automatically switches to a different volume, physically mounted elsewhere, to continue the archive creation operation.

Setting Up a `cron` Entry

Many UNIX system administrators like to schedule their backups to run automatically, often in the middle of the night. This can be accomplished by using the `cron` facility. `cron` executes commands specified in a `cron` table that is created by the user. This table can be listed or modified with the `crontab` command.

Here is an example of how to list the current contents of the `cron` table. The commands shown are typical entries for the root user:

```
# crontab -l
17 5 * * 0 /etc/cleanup > /dev/null
0 2 * * 0,4 /usr/lib/cron/logchecker
0 3 * * * /usr/lib/cleantmp > /dev/null
20 1 * * * /usr/bin/calendar -
```

Each line in the `crontab` consists of six fields. The first five determine the time. The sixth field is the command that will be executed. The six fields are interpreted as follows:

Field	**Description**
1	minutes (0 to 59)
2	hour (0 to 23)
3	day of month (1 to 31)
4	month (1 to 12)
5	day of week (0 to 6, 0=Sunday)
6	the command to execute

Some sample `crontab` entries are described below.

Run BRU every day at 2:30 a.m.:

```
30 2 * * * /bin/bru -c /
```

Run BRU at 10:00 p.m., Monday through Friday:

```
0 22 * * 1-5 /bin/bru -c /
```

Run the `fullbru` command at 3:00 a.m. every Saturday:

```
0 3 * * 6 /bin/fullbru
```

Run the `incbru` command at 4:00 a.m., Tuesday through Friday:

```
0 4 * * 2-5 /bin/incbru
```

Updating the `crontab` involves three steps:

1: Create a file that contains a copy of the current `cron` table.

2: Edit the file (or simply append a line to add a new entry).

3: Update the `cron` table with the edited file.

Here's a simple example that will add a new entry to the `cron` table:

```
# crontab -l > cron.lst
# echo "30 2 * * 2-6 /bin/bru -c" >> cron.lst
# crontab cron.lst
```

To verify that the entry has been added, list the contents of the `cron` table again:

```
# crontab -l
17 5 * * 0 /etc/cleanup > /dev/null
0 2 * * 0,4 /usr/lib/cron/logchecker
0 3 * * * /usr/lib/cleantmp > /dev/null
20 1 * * * /usr/bin/calendar -
30 2 * * 2-6 /bin/bru -c
```

Note that the new entry for BRU has been added. For more information on creating `crontab` entries, refer to your system documentation on `cron` and `crontab`.

BRU's I/O Streams

BRU has several I/O streams that it uses for providing information to the operator and for accepting input when such input is required. The names of the streams used internally are:

`stderr`

This is the stream to which all error and warning messages are written, when such messages are **NOT** part of a query to the user to which a response is required. For example, failure to open a particular file for archiving because of permissions would generate an error message to the `stderr` stream.

`logfp`

This is the stream to which all information generated as a result of the `-v` option is written. Also, output as a result of the `-h` option is written to `logfp`. If the user specifies a `-Il` option to BRU, then `logfp` is opened using that file name. Otherwise, `logfp` is connected to `stdout`, unless the archive is being written to `stdout`, in which case `logfp` is switched to `stderr`.

`ttyout`

This is the stream to which all messages are written that are part of a query to the user for input, to which a reply is required before execution continues. Execution is blocked until a reply is received (see `ttyin`). When BRU is run in the foreground, `ttyout` is connected to the user's terminal by explicitly opening the file specified by the `-Iq` option to BRU. If there is no `-Iq` option specified, then `/dev/tty` is tried. If this open fails, then `ttyout` is connected to `stderr`.

`ttyin`

This is the stream from which replies are read in response to queries posted to the `ttyout` stream. Execution is blocked until the reply is read from the `ttyin` stream. When BRU is run in the foreground, `ttyin` is connected to the user's terminal by explicitly opening the file specified by the `-Ir` option to BRU. If there is no `-Ir` option specified, then `/dev/tty` is tried. If this open fails, then `ttyin` is connected to `stderr`.

Tuning Shared Memory Parameters

When System V style shared memory is available, BRU can use this facility to provide double buffering. The basic idea behind double buffering is to provide overlapping (simultaneous) I/O to both the disks and the archive device, typically a tape unit. Thus for creating archives, files will be read off disk and placed into archive buffers in memory, while previously created buffers are simultaneously being written to the tape. This is accomplished by BRU splitting into two cooperating processes, a parent process one which does the archive device I/O and a child process which does the disk I/O. If your hardware is not capable of supporting simultaneous I/O to both the disk and tape devices, then double buffering is of no advantage. Unfortunately, this hardware design flaw seems to afflict a large number of micro/mini computer systems, particularly those with all the peripherals attached via a single SCSI bus.

Your maximum backup speed may be limited by your tape drive. Before attempting to increase BRU's speed by using double- buffering, check on rated speed of your tape drive. For information on tape drive speeds, refer to previous section in this chapter, "Increasing BRU's Speed."

Shared Memory Parameters

To discuss tuning of the shared memory system as used by BRU, we need to define the following parameters:

`shmmax`

This is the maximum size of any single shared memory segment as used by BRU. The absolute upper limit on this value is set by the kernel. Additionally, an artificial limit can be imposed on a per device basis by defining a value in the `brutab` entry for the device `shmmax=64K` for example). If a `brutab` value is not specified, a default value is picked by BRU - typically 64K bytes.

`shmseg`

This is the maximum number of shared memory segments which are used by BRU. Note that BRU uses one segment for shared variables and the rest for shared archive buffers. Thus the absolute minimum value for BRU is two. The absolute upper limit on this value is set by the kernel. Additionally, an artificial limit can be imposed on a per device basis by defining a value in the `brutab` entry for the device `shmseg=16` for example). If a `brutab` value is not specified, a default of 5 is picked by BRU.

`shmall`

This is the maximum amount of shared memory that will be used by BRU. The absolute upper limit on this value is set by the system kernel, and is a system wide limit. The actual limit per user is roughly the product of `shmseg` and `shmmax` or the kernel limit whichever is smaller. Additionally, an artificial limit can be imposed on a per device basis by defining a value in the `brutab` entry for the device `shmall=256K` for example). If a `brutab` value is not specified, a default value of 320k is used.

Greedy Mode

Note that by setting all three of these previous parameters in your `brutab` file to arbitrarily huge values, you can cause BRU to operate in what has been termed greedy mode. That is, BRU will allocate the largest shared memory buffer that it can, and then allocate and attach as many of them as it can. Since this can be terribly wasteful of memory and seriously impact system performance for other users, the defaults are much more sensible. Typical greedy mode parameters would be `shmseg=1000, shmmax=1M, shmall=1000M`. The defaults are usually more like `shmseg=5, shmmax=64K, shmall=320K`.

Determining Kernel Parameters

In theory, the best way to determine your absolute maximum limits as enforced by the kernel is to look up this information in your kernel `config` directory or consult your system manuals. In practice, most users are not supplied with, or have access to, the kernel `config` files. Seldom does the documentation mention the specific values that were used to configure the kernel shipped with the system.

The BRU distribution may include (for those systems that support System V style shared memory) a simple program (`shmtest`) which probes the shared memory system to try to determine the values dynamically. A similar facility is incorporated into BRU.

To run this program, locate the executable program `shmtest` and simply run it:

```
$ shmtest
shmseg = 99
shmmax = 262144 (256K)
shmall = 2093056 (2044K)
```

The number of segments of various sizes that can be actually allocated and simultaneously attached are:

```
2 segments of size 262144
```

```
4 segments of size 131072
8 segments of size 65536
16 segments of size 32768
16 segments of size 16384
16 segments of size 8192
16 segments of size 4096
16 segments of size 2048
16 segments of size 1024
16 segments of size 512
16 segments of size 256
16 segments of size 128
16 segments of size 64
16 segments of size 32
16 segments of size 16
```

Tuning Shared Memory for a Given Device

The optimum use of shared memory generally depends on the nature of the archive device, the hardware configuration, possibly the version of the operating system, and other assorted variables. In short, you must test your device in a systematic manner, to try to determine the optimum parameters to put in the `brutab` entry for that device.

Testing consists of setting constraints on BRU's shared memory usage via the `brutab` entries for `shmseg`, `shmmax`, and `shmall`. Since BRU will naturally try to use as much shared memory as it can get without violating these constraints. Reducing `shmmax` will cause BRU to use smaller buffers, but more of them, providing `shmseg` is set to some large value.

By systematically changing `shmmax, shmseg`, and `shmall`, BRU can be forced to pick specific values, and thus it's performance can be measured at each set of values. This is sort of like Boyle's Law applied to BRU. (Boyle's Law says that for an ideal gas, the pressure times the volume is a constant, or commonly paraphrased as "a gas will expand to fill it's container" - kind of like your users' data on your filesystems, eh??). The most natural performance measure is the archive throughput, in kilobytes per second. For a given value of `shmmax`, measure and plot the throughput versus the number of shared memory segments set by `shmseg`. Each plot of constant `shmmax` will give a performance curve which shows the optimum value for the number of shared segments (the smallest number of segments that gives the highest throughput).

Repeat this several times for various values of `shmmax`. Since BRU will only use segments that are multiples of the I/O buffer size, use test values that reflect this relationship. For example, the default I/O buffer size for BRU is 32K. Thus the natural test points for constant `shmmax` would be 32K, 48K, 64K, 96K, 128K, etc.

NOTE: Running BRU with verbosity level four (`-vvvv`) will cause it to print a message showing the values for the size and number of the shared memory segments that it selected. This verifies the constraints set by changing the `shmmax` and `shmseg` parameters.

This testing may seem time consuming, but depending upon all the hardware and software variables, there can be dramatic differences in throughput, so the testing is worthwhile in many cases.

Handling Sparse Files: `-S threshold`

With the UNIX filesystem, it is possible to create files which take up much less space in the filesystem than their size would indicate. These files are generally known as sparse files, and commonly occur in database or scientific applications. A sparse file can be loosely defined as one in which large areas of null bytes are created by seeking to a particular file offset before writing any actual data. The following example shows creation of a sparse file using the UNIX `dd` command to create an empty file, seeking to an offset of 1M in the file, and then writing a string at that offset:

```
$ df
/   /dev/dsk/c0d0s0   20330 blocks  9670 i-nodes
/x  /dev/dsk/c5d0s0   13242 blocks  7168 i-nodes
$ echo “End Of File” | dd of=sparsefile bs=1k seek=1k
0+1 blocks in
0+1 blocks out
$ ls -l sparsefile
-rw-rw-r— 1 fnf sys 1048588 Oct 5 09:50 sparsefile
$ od -c sparsefile
0000000 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
*
4000000 E n d   O f   F i l e \n
4000014
$ df
/   /dev/dsk/c0d0s0   20330 blocks  9670 i-nodes
/x  /dev/dsk/c5d0s0   13236 blocks  7167 i-nodes
```

Note that the free space in the /x filesystem, where the sparse file was created, was 13242 blocks before creating the 1M sparse file, and 13236 blocks afterwards. Thus the sparse file actually uses only 6 disk blocks (each 512 bytes), or 3K bytes of actual disk space.

Now look what happens when we copy the sparse file to another file:

```
$ df
/   /dev/dsk/c0d0s0   20330 blocks  9670 i-nodes
/x  /dev/dsk/c5d0s0   13230 blocks  7167 i-nodes
$ cp sparsefile bigfile
$ df
/   /dev/dsk/c0d0s0   20330 blocks  9670 i-nodes
/x  /dev/dsk/c5d0s0   11170 blocks  7166 i-nodes
$ ls -l sparsefile bigfile
-rw-rw-r— 1 fnf sys 1048588 Oct 5 09:50 sparsefile
-rw-rw-r— 1 fnf sys 1048588 Oct 5 10:04 bigfile
$ od -c bigfile
0000000 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
*
4000000 E n d   O f   F i l e \n
4000014
```

The thing to note here is that when we copied the file, the process of copying the file did not preserve the sparseness of the file. The copy of the file named bigfile now actually takes up about 1M of disk space (2060 blocks). Note that BRU can be used to copy the file while preserving the sparseness, as given in an example earlier. From the viewpoint of a user process reading or writing the file, both files look identical, as is shown by the dump of the file using the UNIX od command.

However, the original sparse file only uses 6 disk blocks and the copy uses 2060, quite a difference! BRU has a special option, the -S option, which turns on features that reduce the amount of archive space used by sparse files, and that preserve the sparseness of the file when it is extracted from the archive.

Saving Sparse Files

During creation of the archive, the -S threshold option causes all files larger than the specified threshold size (the argument for the -S option) to be automatically compressed, saving considerable space in the archive. When the -S option is specified, the -Z option is also enabled. Sparse files typically compress by 90% or more. For example, let's show how the sparse file we created previously compresses more than 99.5% (BRU reports 100% due to rounding):

```
$ bru -cvv -S 100K -f bru.out sparsefile
c (100%)  6k of  8k [1] sparsefile
$ ls -l bru.out
-rw-rw-r— 1 fnf project  10240 Oct 5 10:43 bru.out
```

Restoring Sparse Files

During extraction, blocks of null bytes are created in the filesystem using seeks, rather than actually writing the blocks of null bytes out. This mimics the original technique used to create the original copy of the sparse file, but it does impose a computation overhead to test each block before seeking or writing the block.

Sparse files are **NOT** created automatically when restoring. The -S threshold option must be specified. Only files that were previously compressed (with the -Z or -S options) can be restored as sparse files. When restoring, only files with sizes larger than the threshold value will be created as sparse files. All others will be restored as regular files. Here's an example of how we delete and then restore the sparse file. We'll also use df to check the disk space before and after we run BRU:

```
$ df
/   /dev/dsk/c0d0s0  20330 blocks  9670 i-nodes
/x  /dev/dsk/c5d0s0  13202 blocks  7166 i-nodes
$ rm sparsefile
$ bru -xvv -S 100K -f bru.out
x (100%)  6k of  8k [1] sparsefile
$ df
/   /dev/dsk/c0d0s0  20330 blocks  9670 i-nodes
/x  /dev/dsk/c5d0s0  13202 blocks  7166 i-nodes
```

In this example we removed the original copy of the sparse file, extracted the copy from the archive, and ended up with exactly as much free disk space as before. If you have a filesystem that is very full, and very fragmented, it is useful to use BRU to do a full backup of the filesystem, delete all the files (or remake the filesystem), and then restore all the files from the backup using the -S option. This will this remove most of the fragmentation, resulting in faster access to your files. It may also free up some of the disk space that was being consumed by blocks of null bytes.

WARNING: Frequently there are files which could be sparse files but simply were not created that way. In most cases, application programs cannot tell the difference between a sparse file and a regular file. However, some UNIX system files cannot be turned into sparse files. Be sure you know enough about UNIX and the minimum set of files necessary to run before you try this on a critical filesystem like the root filesystem! **DO NOT** restore the system kernel (usually named `/unix`) as a sparse file. If the kernel file is restored as a sparse file, it will be impossible to boot your system.

Using a Bootable Floppy or Tape

Since BRU requires the operating system to be up and running (it is not a standalone program), you may have been wondering how you would restore your hard drive if the entire disk was somehow wiped out by a catastrophic failure. For most systems, the answer is a bootable floppy or bootable tape. These will contain a select subset of your full system files, just the files necessary to boot and run the system without having to access any files from your hard disk. The procedure involved in creating a bootable floppy or tape will vary, depending on your specific version of UNIX. **NOTE**: This section contains general information only-the steps involved may be different for your system. You will have to perform some of the steps before you need to restore your drive, so don't wait until catastrophe strikes. Make preparations now.

Creating a Bootable Floppy

If you are using BRU with Linux on x86 systems, or Caldera/SCO Openserver, we provide a ready-to-use solution called CRU - Crash Recovery Utility. CRU takes the following steps into account when creating your boot/root floppies and recovering your system. For more information on CRU, please refer to the CRU README file in the CRU directory or visit our website at

`http://www.tolisgroup.com/CRU_Info.html`.

If you are using another OS, you will need to consult your system documentation to find out exactly how to create a bootable recovery floppy. Once you have a bootable floppy, you can boot your system and run a minimum set of utilities to reconstruct your hard drive (or drives). The exact procedure for creating a bootable floppy is different for each system, so we can't give you explicit instructions here.

Once you have created a bootable floppy, you will need to copy the BRU executable to the `/bin` directory of the floppy, and copy the `brutab` file to the `/etc` directory of the floppy. To do this, you would generally run the mount command to mount the bootable floppy as an additional temporary mounted filesystem. Assuming that you have a directory called `/mnt`, and that your mountable floppy device name is `/dev/fd0`, the sequence of commands would be something like this:

```
$ mount /dev/fd0 /mnt
$ df
/      (/dev/root ):  12345 blocks  3564 i-nodes
/mnt   (/dev/fd0 ):     534 blocks    65 i-nodes
$ cp /bin/bru /mnt/bin/bru
$ cp /etc/brutab /mnt/etc/brutab
$
```

Depending on how your bootable floppy was created, you may have to copy other critical executables, like format, `fsck`, `ls`, `rm`, `mkdir`, etc., to your bootable floppy. Take time now to make a complete list of all your files on the floppy, and check the discussion of the restore process to make sure you have everything you will need. If you have a spare drive, you might even consider making a backup of your active drive, replacing it with the spare drive, and attempting a restore to ensure that you have all the necessary files available and have the technique mastered.

Formatting and Partitioning Your Hard Drive

In the following discussion, it is assumed that you have had a catastrophic failure of some sort which has rendered your hard drive unbootable and unusable, and you have booted off your bootable floppy or tape. You might even have replaced your old drive with a brand new drive which has never been used before. **DO NOT** attempt any of the examples with your working system drive or you will likely destroy all of the information it contains.

TOLIS Group assumes no responsibility for any damage you cause while attempting these steps on a live system.

In preparing your new disk for use on your system, you may or may not be required to perform a low level format of the physical drive. Most newer IDE hard drives do not require low level formatting, and - in fact - may be damaged if you attempt to low level format them. If you are using SCSI disks, you can perform low level formatting via your SCSI HBA BIOS routines on most PC platforms or through special system commands on workstation class systems. This formatting may or may not include testing and marking bad blocks (blocks which are unusable). As with making a bootable floppy, the exact procedure for formatting a new hard drive is system dependent, so we can't give you an example which will work for all cases. You will need to consult your system documentation, or the documentation which came with your new hard drive.

Once your drive is formatted, generally it must be partitioned to allocate space for various filesystems on it. It is customary to partition large drives into several filesystems, to better organize your disk storage. As with formatting, the process required to partition a hard drive is system specific, so you will again need to consult your system documentation. **NOTE**: That you do not have to partition the drive exactly like the original. As long as there is sufficient room in the partition in which you are attempting to restore your files, BRU does not care about partition sizes. Thus you can use this procedure to repartition and restore your main drive if you discover that your original partitioning does not match your current needs. **IMPORTANT NOTE**: We strongly recommend having at least two complete, verified backups of any data on your hard disk before you deliberately destroy the data by reformatting and repartitioning.

Mounting and Restoring Each Partition

Assuming that your backups have been organized on a filesystem (partition) basis, with various full and incremental backups of each partition (filesystem), this section describes how you would restore each partition. Note that it assumes that you have made all of your backups with relative pathnames (filenames relative to some current directory) rather than absolute pathnames (filenames beginning with a "`/`" character). If you have used absolute pathnames in your backup, you may use BRU's `-PA` option to convert the absolute paths "`/`" to relative "`./`" during the restore operation. More on the `-PA` option later in this chapter. We'll use the root filesystem of your hard disk as an example. Restoring your other filesystems is similar.

The first step is to create an empty filesystem in the root partition of your hard drive, using the `mkfs` or format program on your bootable floppy. Some UNIX systems may use a different command to create filesystems (like the `divvy` command under SCO OpenServer).

After creating a filesystem, you can then mount it on an empty directory on your floppy (we assume your root partition device name is /dev/dsk/0s0 and the raw device name is /dev/rdsk/0s0):

```
$ mkfs /dev/rdsk/0s0 100000
$ mount /dev/dsk/0s0 /mnt
```

At this point, you can now restore the root partition on your hard drive to the state of the last full backup:

```
$ cd /mnt
$ bru -xvf /dev/tape -PA
x   2k of   4k [1] ./bin
x 112k of 116k [1] ./bin/adb
.
.
x 240k of 2420k [1] ./unix
```

Note that because all of the files were stored with pathnames relative to the current directory ("."), when we changed our current directory to /mnt, our current directory became the same directory which was the "/" directory when the backup was made while the system was running off of the hard drive. Thus the directory that appears as /mnt/bin while running off of the floppy, will become just /bin when we reboot off of the restored hard drive.

After restoring the full backup, we will next restore each incremental backup to recover files which changed since the full backup. If you make your incremental backup for each day include all files changed since the last full backup (rather than just files changed during that day) which is normally referred to as a differential backup, then you will only need to restore the latest differential. Otherwise, you will need to restore each incremental made since the full backup. Together, the full backup plus all of the incremental or differential backups should be sufficient to recover to the point where the last backup was made:

```
$ bru -xvf /dev/tape
x   4k of    8k [1] ./etc/passwd
x  85k of   93k [1] ./bin/NEWls
.
x 240k of 2420k [1] ./NEWunix
$ bru -xvf /dev/tape
.
```

NOTE: That if you run out of space while restoring the incrementals, it means that you have restored some files which were deleted between backups. BRU does not make note of deleted files (nor should it; file configuration management should be accomplished by a separate program). To solve this problem, you will need to remove these unnecessary files by hand before proceeding with restoration of the current incremental backup.

Using Remote Devices

BRU incorporates a special feature that is similar to that used in the remote dump rdump and remote restore rrestore programs. Instead of specifying the name of a local device to use as the archive, you can give the name of a device on another system connected to the local system via some high speed network such as ethernet.

In the example below the local system, the one running BRU is named `myhost`. The remote system, the system with the desired archive device attached is called `remotehost`, and the remote device is known as /dev/rmt0. To backup the files from the local system "`myhost`" and store them on the remote system "`remotehost`", `/dev/rmt0` device, you would run the command below on the local host "`myhost`":

```
bru -c -vv -f remotehost:/dev/rmt0
```

To restore files from the same remote system you would run this command:

```
bru -x -vv -f remotehost:/dev/rmt0
```

You should also create an entry for the remote device in `myhost's brutab` file. As long as you have the appropriate programs and permissions to be able to run a remote shell on `remotehost`, and you have the remote device server program `/etc/rmt` on `remotehost`, then BRU should be able to read and write the remote device `/dev/rmt0`. Note that some versions of the `/etc/rmt` program have a maximum I/O buffer size of 10K, which is smaller than the default I/O buffer size used by BRU. If you get an error using remote devices, try setting your I/O buffer size to 4K using the `-b` option. Using remote devices can sometimes be very slow. Once you have established that you can successfully read and write a remote device, you might try the double buffering option to increase the speed of remote reads and writes. **NOTE**: The `/etc/rmt` program is not furnished with certain versions of UNIX. If you cannot find a copy (source code can be found at many Internet sites), please contact TOLIS Group. A complete source code listing (and full explanation) can also be found in the following book for C programmers: UNIX Network Programming, W. Richard Stevens, Prentice-Hall, Englewood Cliffs, NJ, 1990, ISBN 0-13-949876-1.

Crash Recovery and Absolute Pathnames

The capability to restore files in a different directory is crucial when you are trying to restore system files after a crash. Crash recovery normally involves booting the system with a temporary root directory. This directory may reside on a floppy disk and probably does not have enough room for any new files. It may even be read only. If you try to extract an archive with absolute pathnames, BRU will attempt to create them in the temporary root directory. This is NOT what you want. What you want is to create the files in the root directory of your original hard disk. To do this, you must mount the hard disk's root filesystem on a directory (like `/mnt`). After the hard disk root filesystem has been mounted, change to the mount directory, and extract the archive while converting the absolute pathnames to relative form. Here is an example of the commands you might use to do this:

```
# mount /dev/rdsk/0s0 /mnt
# cd /mnt
# bru -x -PA -f /dev/tape
```

NOTE: These are only sample commands. The exact commands and device names may be different for your system.

Restoring Shared Library Files - SmartRestore

Many newer versions of UNIX (like System V, Release 4) use shared library files. These files contain common C-language routines that are called by most UNIX programs. Nearly every UNIX program uses the `libc shared` library, so this file is always active. Shared library files are normally stored in one of the library directories like `/lib`, `/shlib` or `/usr/lib`. Shared library filenames are often distinguished by a special suffix like "`_s`" or "`.so`" or "`.so.1`" (for example, `/shlib/libc_s` or `/usr/lib/libc.so.1`).

WARNING: Serious problems can occur if you try to restore (overwrite) an active shared library file-usually your system will crash! BRU does its best to be intelligent about the way it handles restoring these types of files through the SmartRestore functionality. If a file is in use (text file busy), BRU will attempt to relocate the existing file before restoring the copy in your archive. The text busy flag and memory pointers will follow the move to the file's new location, so your system will continue to operate. However, the old version of the in-use file will still be on the system when you next reboot.

When restoring using the SmartRestore process, BRU will examine the contents of the `/etc/brusmartrest` file for specific files to handle this way even if the file's text busy flag is not set. Example entries for the `/etc/brusmartrest` file are:

```
s  *.so    # Be careful with shared libraries
s  *.so.*
s  */bru   # don't overwrite the bru program
```

When BRU restores a file that matches one of these specifications, the original file is moved to `/usr/tmp/` and named `DELETE.XXXXXX` where the X's are numbers. A shell script is created called `/bru/bruremovelog` that serves as both a log file of the files restored using this method and a simple script to delete the moved original files once the system is rebooted. Another way to avoid problems when restoring files, is to exclude any active shared library files using the `-X` option with appropriate entries in the `/etc/bruxpat` file. Here are two examples of exclusion patterns that can be placed in your `bruxpat` file:

For System V, Release 4:

```
xs *.so.*
xs *.so
xs */bru
```

For SCO UNIX:

```
xs /shlib/*_s
xs */bru
```

These patterns may not be correct for your system or backup method. Please refer to your system documentation for details on the location of any shared libraries on your system.

Appending to Existing Archives

Tape drives allow you to write several archives, or filesets on a single tape. To do this, you must prevent the tape from performing a rewind after it has completed writing (the normal mode of operation). This can be accomplished by using the `norewind` device node. When the `norewind` device is used, BRU will begin to write data at the current tape position and will stop the tape, without rewinding, when the backup has completed. Be aware that when the `norewind` device is used, it is impossible to rewind the tape and BRU's automatic verification features will be disabled.

If you use the `norewind` device, you must position the tape at the proper file mark before attempting to read or write. Tape utilities that do this are normally furnished with your UNIX system (or can be furnished by your tape drive vendor). The names of these tape utilities vary widely, but common names are: tape, `tapecntl` and `mt`. Refer to your system documentation for specific details. **NOTE**: You should not attempt to store multiple archives on a single tape unless you have very good reasons for doing so. Do not do it simply to save on tape costs.

Many newer tape drives can handle several Gigabytes and it may seem appealing to have the capacity to store a week's worth of backups on a single tape. Do this, and you're "putting all your eggs in one basket." You could lose all your backup data if your tape is damaged. The cost of a few more tapes will seem cheap compared to the trouble you'll have trying to restore (or re-create) your data.

Live System Backups

When backing up files on a UNIX system, it is always best if the system is quiet and no file changes are occurring. That's why many system administrators choose to run their backups late at night.

On some busy UNIX systems, especially those with active database programs, file changes occur 24-hours a day. If files are changing, BRU will attempt to archive them, but errors may occur. If a file changes while BRU is reading it, a warning message will normally be issued. Other errors may occur if BRU is attempting to compress a file that is changing. In either case, the data that was archived may not be correct. In fact, it may contain serious errors.

BRU does not perform any file-locking, so there is no way for BRU to stop a file from changing during a backup. In most cases, file-locking would be worthless anyway, especially for database files. This is because BRU has no way of determining if a file is part of a changing database or of knowing which groups of database files should be locked. Theoretically, BRU could simply lock each file individually, but this would probably cause the database program to crash. Most database programs have complex locking schemes and do not expect anyone else to be locking their files.

Fortunately, some of the database vendors have recognized the problems in trying to do hot backups and furnish special commands that allow you to temporarily "freeze" the database files while a backup is done. Any changes made to the database during the backup will be written to a special temporary area. Once the backup is done, all the changes will be applied to the database files. The following example illustrates how to put an Oracle database into backup mode, do the backup with BRU, and then return the database to normal.

First, issue SQL commands to "freeze" the Oracle database:

```
connect internal
alter tablespace name begin backup
exit
```

Now run BRU and back up the database files:

```
bru -cv /database
After the backup is done, "unfreeze" the database:
connect internal
alter tablespace name end backup
exit
```

The above database commands are shown only as an example. They may not work correctly for your particular database. For additional information, refer to your database documentation or contact your database vendor.

Chapter 9 Graphical User Interface

BRU offers an easy to use graphical interface known as "`xbru`." XBRU allows even the most novice user to properly manage BRU backup and restore operations with no knowledge of the underlying tape commands required to perform the operations.

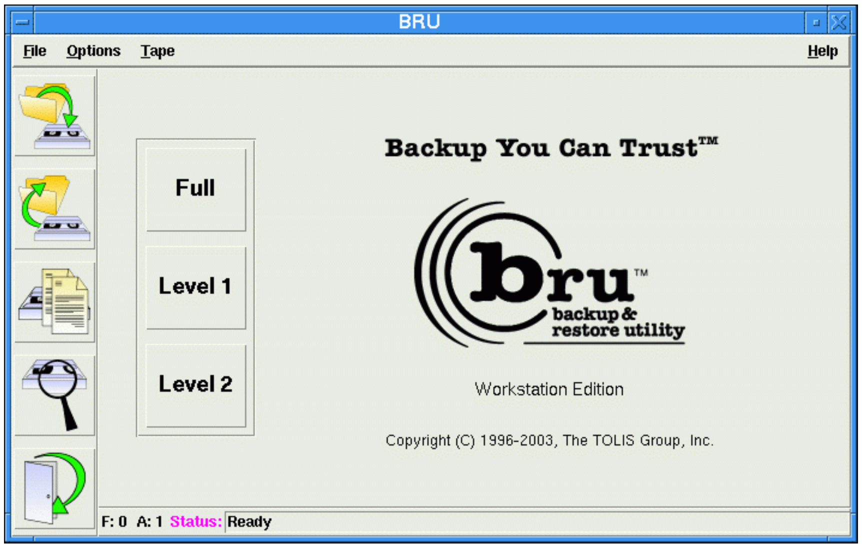


Figure 1 - XBRU Main Interface Window

Interface Options

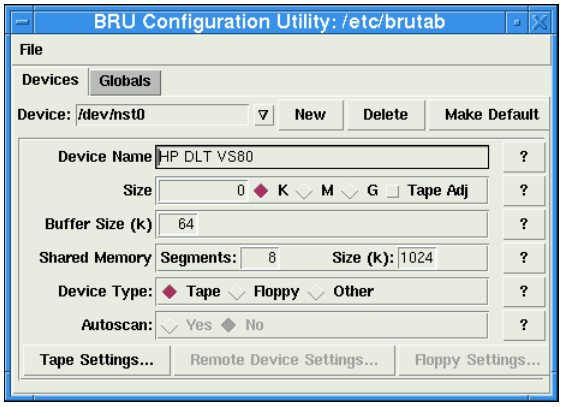
When using the graphical interface, you may perform any of the following operations:

- Create backups interactively.
- Create, save, and load backup definitions
- Schedule backup definitions to run in the future
- Restore files from archives
- List contents of archives
- Verify contents of archives
- View, and clear, the BRU Execution Log
- Perform selected tape I/O operations.
- Setup E-mail notification when specified events occur

XBRU Configuration

During BRU installation, basic parameters were selected for the tape devices on your system. From the perspective of normal BRU operations, all of the necessary information is included in this basic `/etc/brutab` file. However, to make the best use of the graphical interface, it may be necessary to modify these default settings.

From the File menu, select the “Configure BRU” option (you must be running `xbru` as root for this option to appear).



Device: /dev/nst0

Device Name: HP DLT VS80

Size: 0 [✓ K] M G ☐ Tape Adj

Buffer Size (k): 64

Shared Memory: Segments: 8 Size (k): 1024

Device Type: [✓ Tape] Floppy Other

Autoscan: Yes [✓ No]

Figure 2 - Main Configuration Dialog

XBRU uses the non-rewinding version of the various tape devices on your system. This is required to allow XBRU to manage the tape motion. A non-rewinding device is typically named the same as a rewinding device, with the exception of having an ‘`n`’ included in its node name. If a rewinding device is listed as “`/dev/st0`”, the non-rewinding device would be “`/dev/nst0`”. Users that are not certain of device identification should consult their operating system documentation.

Tape Configuration Dialog

It is very important that the device name given in the Device Configuration dialog and the properties set on this page match. For further illustration, several sample `brutabs` can be found at: `http://www.tolisgroup.com/brutabs3.html`.

The relevant changes would be the change in the device name (“`/dev/st0`” to “`/dev/nst0`”), the change in size (from 4GB to unspecified, or blank), the change of the “`rewind`” flag to “`norewind`”, and the addition of the “`noautoscan`” flag. The “`devname`” entry is only to provide a name for the device that can be read by the user and will not affect XBRU’s ability to utilize it. It is highly recommended that the “`devname`” value be set. There are additional entries that MAY need to be entered into the `brutab` device entry for non-rewinding devices by clicking on the “`Tape Settings`” button. Variables and Examples are:

`rewindcmd`: Command to rewind tape drive. Example:

```
rewindcmd=”mt -f /dev/nst0 rewind”
```

`erasecmd`: Command to erase tape. Example:

```
erasecmd=”mt -f /dev/nst0 erase”
```

`rfmcmd`: Command to wind tape to the next file mark/archive. Example:

```
rfmcmd=”mt -f /dev/nst0 fsf”
```

`bfmcmd`: Command to rewind tape to previous file mark. Example:

```
bfmcmd=”mt -f /dev/nst0 bsf”
```

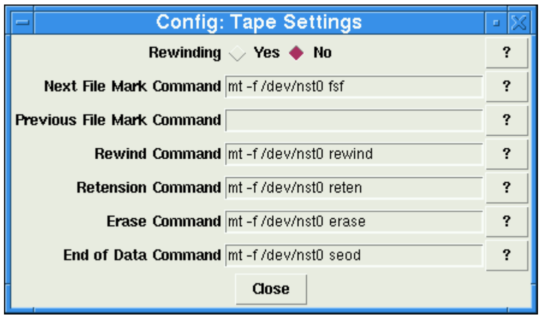
On Linux:
`bfmcmd="mt -f /dev/nst0 bsfm 2"`

eodcmd: Command to wind tape to end of recorded data. Example:

```
eodcmd="mt -f /dev/nst0 seod"
```

`retencmd`: Command to retension tape. Example:

```
retencmd="mt -f /dev/nst0 reten"
```



Rewinding Yes [✓ No]

Next File Mark Command: mt -f /dev/nst0 fsf

Previous File Mark Command:

Rewind Command: mt -f /dev/nst0 rewind

Retension Command: mt -f /dev/nst0 reten

Erase Command: mt -f /dev/nst0 erase

End of Data Command: mt -f /dev/nst0 seod

Figure 3 - Tape Settings

Consult the system documentation for information on how these commands should be implemented (hint: try '`man mt`' or '`man tape`'). If the command sequences are not set, XBRU will make assumptions based on the operating system. NOTE: The command to rewind a tape to the previous file mark does not work with some devices. In those cases, `bfmcmd` should be removed. Subsequently, XBRU will use a slower, but more reliable, method of rewinding and spacing forward to the appropriate file set.

Linux Device Notes

For details on Linux specific tape device configuration please visit: `http://www.linuxtapecert.org`.

Hot-Button Backups

XBRU provides a GUI with three shortcut buttons for creating backups: "`Full`", "`Level 1`", and "`Level 2`". These operate using the active settings for backup device, compression, include/exclude patterns, etc. (see "`Backup Options`" later in this chapter). These will override the "`Skip Mount Points`" setting so any mount points encountered will be backed up. However, XBRU will ask for confirmation if backing up of remote volumes is enabled. See Figure 5 for further information.

Starting XBRU

To launch XBRU, type "`xbru`" at the command prompt.

Backup Operations

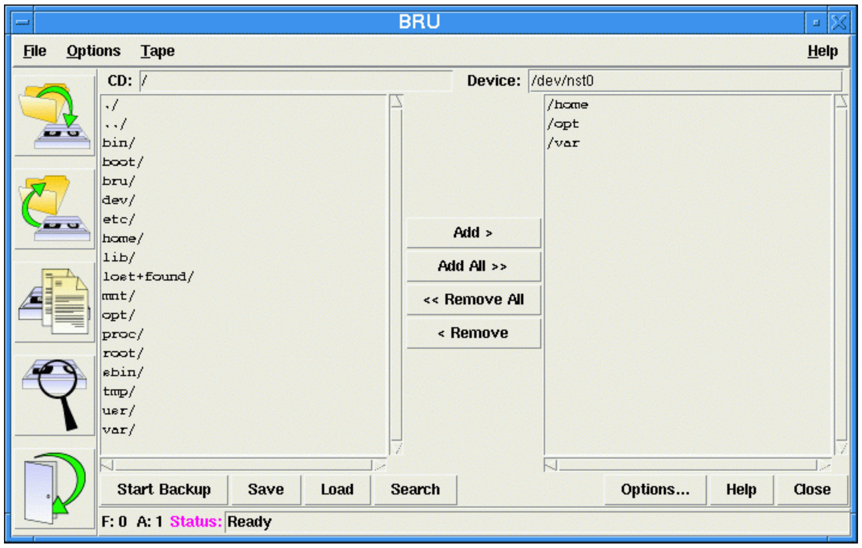


Figure 4 - Backup Interface

Select the "`Backup`" button or choose "`Backup`" from the "`File`" menu to bring up the file selection interface. The box on the left will display the name and contents of the current directory. To change directories, edit the "`CD`" entry or double-click on the directory entries in the listing. The box on the right will display the current files/directories selected for backup.

File Selection

Add files or directories to the backup list by highlighting them in the current directory listing and selecting "`Add`". All entries in the current directory can be added by selecting "`Add All`". Alternatively, files can be added by simply double-clicking on them. Clicking the right mouse button is equivalent to selecting "`Add`". A directory cannot be added if a file or directory within that directory is already in the backup list, as this would result in backing up duplicate files. Any subdirectories/files must be removed before the parent directory can be added. To remove, use the mouse to select entries from the list and select "`Remove`". "`Remove All`" will clear the selected list.

Both listings support multiple selections. Selecting an item, then <Shift>-selecting another item, will select all items between the two. <Ctrl>-click will select individual items. Selecting "`Add`"/"`Remove`" will affect all of the selected items.

Search

XBRU also provides a search function, accessible by clicking on the "`Search`" button. This button will display a search window where a shell-style pattern can be entered and/or recursive searching (optional) can be initiated. As in shell protocol, the "*" character will match 0 or more characters, "?" matches any single character, and `[a-c]` or `[a,c]` will match either a range of characters between '`a`' and '`c`', or only '`a`' or '`c`', respectively.

Recursive searches will match the pattern against all files and subdirectories found. Otherwise, the search pattern is only matched against files and directories in the current directory. The pattern entered should match the entire path desired. Examples are as follows:

`file`	Matches "`file`" in the current directory
`*file`	Matches any paths ending with "`file`", such as "`dir/file`", `dir/badfile`, or "`somefile`"
`dir/*`	Matches any paths beginning with "`dir/`"
`*.gif`	Matches any paths ending with "`.gif`"

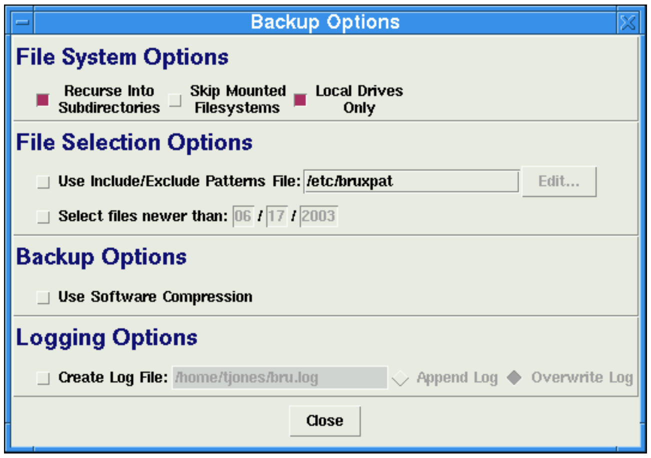
Backup Definitions

Backup Definitions allow the user to define a specific set of backup options, schedule selected backup processes to occur at a later time, or quickly retrieve files/directories. The backup interface displays option buttons to "`Load`" and "`Save`" backup definitions.

Selecting "`Save`" will display a prompt for a definition name. Note that there are spaces for multiple users to allow different users to have definitions of the same name. Clicking "`Save`" will store all relevant information, which includes the current device selection, backup options such as compression and include/ exclude pattern settings, and selected files and directories. Once a definition has been saved, a prompt will appear that allows the user to schedule the definition to run at some specific time in the future. To change any information in a definition, the changes must be completed, the definition must be saved under the same name and "`overwrite`" must be selected at the ensuing prompt. Definitions are stored in files located in a user directory within XBRU's installation directory. Consequently, the user saving the definition MUST have write access to XBRU's install directory.

XBRU will create the subdirectory for the user if it does not exist. Selecting "Load" will present a dialog showing all current definitions. After selecting the desired definition, XBRU will set the device and backup options to the previously saved parameters.

Backup Options



[✓ Recurse Into Subdirectories] Skip Mounted Filesystems [✓ Local Drives Only]

Use Include/Exclude Patterns File: /etc/bruxpat

Select files newer than: 06 / 17 / 2003

Use Software Compression

Create Log File: /home/tjones/bru.log Append Log [✓ Overwrite Log]

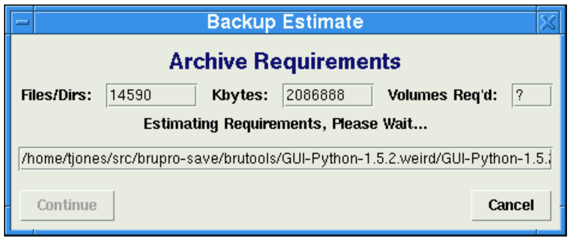
Figure 5 - Backup Options Dialog

The "Backup Options" interface can be accessed by either selecting the "Options" button from the Backup File Selection window, or by selecting "Backup" from the "Options" menu. "Backup Options" allows a user to select such items as incremental backups, mount point handling, compression settings, logging options, and more. Refer to the previous chapters for more information on how different options will affect backups.

To save all settings for future use, select "Save Options" from the "Options" menu.

Running the Backup

After selecting at least one file or directory for backup, click the "Start Backup" button to begin the backup process. This will result in the appearance of several dialog windows designed to handle such things as overwriting/appending to tapes and labeling backups. Another window that will appear is the "Estimate" dialog.



Files/Dirs: 14590 Kbytes: 2086888 Volumes Req'd: ?

Estimating Requirements, Please Wait...

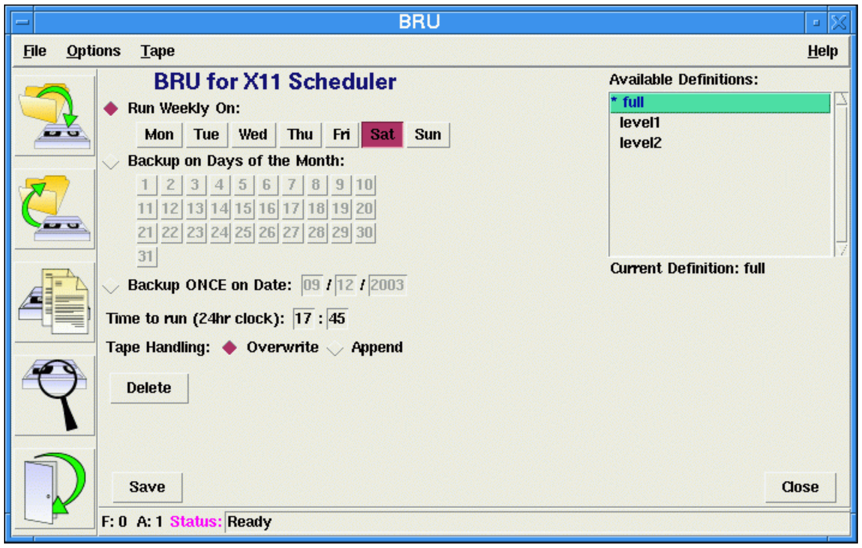
/home/tjones/src/brupro-save/brutools/GUI-Python-1.5.2.weird/GUI-Python-1.5.

Figure 6 - Estimate Dialog

The "Estimate" dialog will display how many files will be backed up, how large the backup will be in kilobytes, and approximately how many volumes the backup will consist of, provided the tape size has been set. The volume estimate does not take compression into account if it is activated. At this point, the backup may be completed without any further interaction. Once the estimate is complete, select the "Continue" button to perform the actual backup. If you don't click a button within 30 seconds, XBRU will automatically begin the backup.

The next window to appear will be the "Backup Progress" dialog, which displays a progress meter and various statistics about the backup. This allows the user to visually monitor the backup activity. Once the backup is complete, XBRU will attempt to rewind the tape, if applicable, and verify the backup. The meters will be reset to show the verification progress. When finished, the "Cancel" button in the "Backup Progress" window will change to say "Done" and XBRU's status line will change to say "Ready".

Scheduling Backup Definitions



Run Weekly On: Sat

Backup on Days of the Month:

Backup ONCE on Date: 09 / 12 / 2003

Time to run (24hr clock): 17 : 45

Tape Handling: [✓ Overwrite] Append

Available Definitions:
* full
level1
level2

Current Definition: full

F: 0 A: 1 Status: Ready

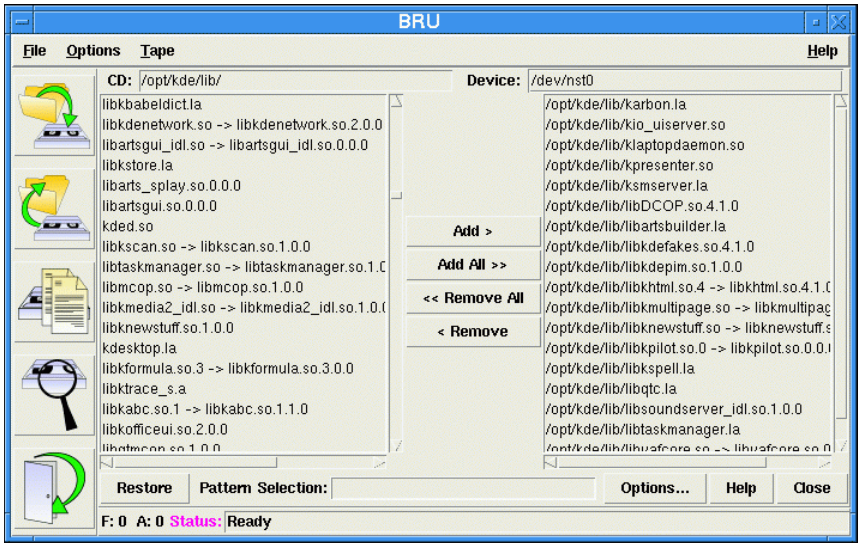
Figure 7 - Scheduler Dialog

XBRU provides a scheduler for setting up backup definitions to run at later times. The scheduler dialog is accessed through the dialog window that appears after saving a definition, or by selecting "`Scheduler`" from the "`File`" menu. When activated, the scheduler will display a calendar with a "`Definition`" window located in the top right hand corner of the window. This displays a list of all previously saved definitions available to the user. Currently scheduled definitions will appear with an asterisk.

To schedule a definition, double-click on the definition name. Select the day(s) of the week that the definition should run, keeping in mind that it is possible to select more than one day of the week. Multiple backups can also be scheduled. An example might be scheduling a definition labeled as "`Daily`" to perform a backup (Backup Options, Section 3.5) every Monday through Thursday at 6:30pm, then scheduling a backup labeled as "`Weekly`" to run on Fridays at 6:30pm. A user can also select whether the backup will overwrite or append to the tape in the drive when the definition runs (only applicable to tape devices). For more information on BRU's scheduler and its use of cron, see the application notes at `http://www.tolisgroup.com/applnotes.html`. **NOTE**: "`Save`" **MUST** be selected in order for any changes to take effect. Selecting a different definition or closing the window will cause any unsaved changes to be discarded. To unschedule or completely delete any definition(s), highlight the desired definition and select "`delete`".

Restoring Files

Retrieving Archive Contents



CD: /opt/kde/lib/

Device: /dev/nst0

```
libkbabeldict.la
libkdenetwork.so -> libkdenetwork.so.2.0.0
libartsgui_idl.so -> libartsgui_idl.so.0.0.0
libkstore.la
libarts_splay.so.0.0.0
libartsgui.so.0.0.0
kded.so
libkscan.so -> libkscan.so.1.0.0
libtaskmanager.so -> libtaskmanager.so.1.(
libmcop.so -> libmcop.so.1.0.0
libkmedia2_idl.so -> libkmedia2_idl.so.1.0.(
libknewstuff.so.1.0.0
kdesktop.la
libkformula.so.3 -> libkformula.so.3.0.0
libktrace_s.a
libkabc.so.1 -> libkabc.so.1.1.0
libkofficeui.so.2.0.0
libqtmcop.so.1.0.0
```

```
/opt/kde/lib/karbon.la
/opt/kde/lib/kio_uiserver.so
/opt/kde/lib/klaptopdaemon.so
/opt/kde/lib/kpresenter.so
/opt/kde/lib/ksmserver.la
/opt/kde/lib/libDCOP.so.4.1.0
/opt/kde/lib/libartsbuilder.la
/opt/kde/lib/libkdefakes.so.4.1.0
/opt/kde/lib/libkdepim.so.1.0.0
/opt/kde/lib/libkhtml.so.4 -> libkhtml.so.4.1.(
/opt/kde/lib/libkmultipage.so -> libkmultipag
/opt/kde/lib/libknewstuff.so -> libknewstuff.s
/opt/kde/lib/libkpilot.so.0 -> libkpilot.so.0.0.
/opt/kde/lib/libkspell.la
/opt/kde/lib/libqtc.la
/opt/kde/lib/libsoundserver_idl.so.1.0.0
/opt/kde/lib/libtaskmanager.la
/opt/kde/lib/libwafcore.so -> libwafcore.so.0
```

Pattern Selection:

F: 0 A: 0 Status: Ready

Figure 8 - Restore Dialog

Before any files can be selected for restore, XBRU must retrieve the contents of the archive.

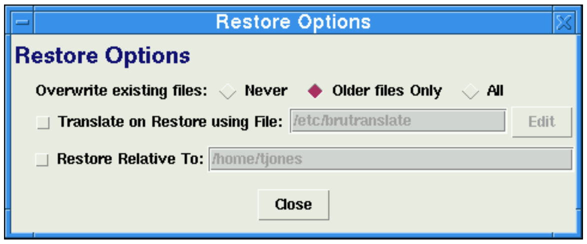
Selecting Files

Once XBRU retrieves the contents of an archive, a listing similar to the one used for selecting files for backup will be displayed. The box on the right will display the contents of the current "`directory`" (shown as stored on tape). Change directories by double-clicking on the selected directory. Double-click on files to add them to the restore list. Multiple files can be selected by holding down the <Shift> or <Ctrl> key while selecting files. Selecting "`Add`" will add all selected files.

Restoring

Once all files/directories have been selected to be restored, click on the "`Restore`" button. A window will appear displaying each file as it is restored.

Restore Options



Restore Options

Overwrite existing files: Never [✓ Older files Only] All

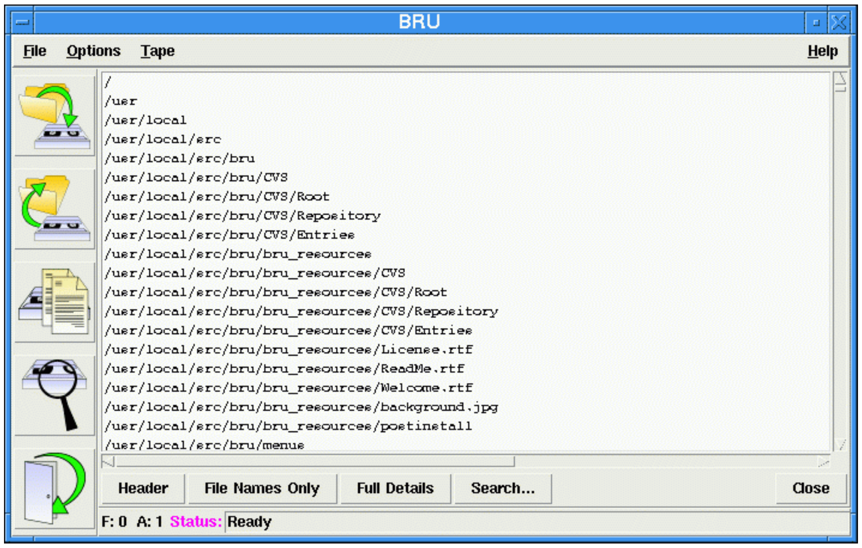
Translate on Restore using File: /etc/brutranslate

Restore Relative To: /home/tjones

Figure 9 - Restore Options Dialog

Restore options can be accessed by clicking the "`Options`" button on the "`Restore`" interface or by selecting "`Restore`" from the "`Options`" menu. The "`Restore Options`" interface can be used to control features such as selecting which files are overwritten during a restore operation, restoring files to different names (Translate on Restore), and converting absolute pathnames to relative pathnames. See the "BRU Technical Manual" for further details regarding these options. "`Restore`" settings can be saved for future use by selecting "`Save Options`" from the "`Options`" menu.

Listing Archives



```
/
/usr
/usr/local
/usr/local/src
/usr/local/src/bru
/usr/local/src/bru/CVS
/usr/local/src/bru/CVS/Root
/usr/local/src/bru/CVS/Repository
/usr/local/src/bru/CVS/Entries
/usr/local/src/bru/bru_resources
/usr/local/src/bru/bru_resources/CVS
/usr/local/src/bru/bru_resources/CVS/Root
/usr/local/src/bru/bru_resources/CVS/Repository
/usr/local/src/bru/bru_resources/CVS/Entries
/usr/local/src/bru/bru_resources/License.rtf
/usr/local/src/bru/bru_resources/ReadMe.rtf
/usr/local/src/bru/bru_resources/Welcome.rtf
/usr/local/src/bru/bru_resources/background.jpg
/usr/local/src/bru/bru_resources/postinstall
/usr/local/src/bru/menus
```

F: 0 A: 1 Status: Ready

Figure 10 - Listing Dialog

List Options

XBRU also has options for displaying the contents of archives. Selecting the "`List`" button from the main display will display a listing interface. Three options are available: BRU's "`Header`", a "`File Names Only`" listing, and a "`Full Details`" listing.

Header

This option displays BRU's archive header record, which lists items such as the archive label, creation date, BRU version and serial number used to create the archive, buffer size, system/machine the archive was created on, and a variety of other pertinent information. This is useful for determining how old the archive on the tape is, where it originated from, etc.

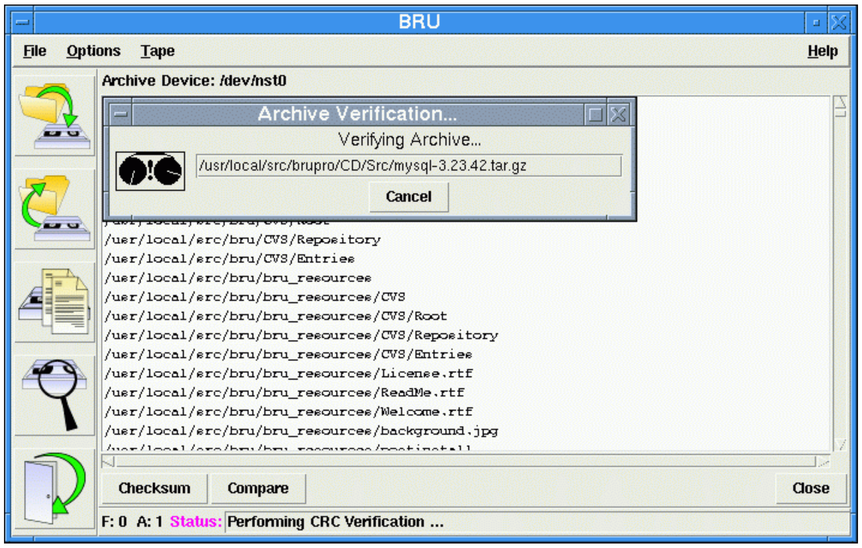
File Names Only

This option displays the only the filenames and paths of the files in the current archive.

Full Details

This option scans the entire archive, extracting such information as file names, permissions, owners, sizes, modification times, etc. A confirmation prompt will also appear for this process, as BRU must scan the entire archive to retrieve this information.

Verifying Archives



Archive Device: /dev/nst0

Verifying Archive...

/usr/local/src/brupro/CD/Src/mysql-3.23.42.tar.gz

```
/usr/local/src/bru/CVS/Repository
/usr/local/src/bru/CVS/Entries
/usr/local/src/bru/bru_resources
/usr/local/src/bru/bru_resources/CVS
/usr/local/src/bru/bru_resources/CVS/Root
/usr/local/src/bru/bru_resources/CVS/Repository
/usr/local/src/bru/bru_resources/CVS/Entries
/usr/local/src/bru/bru_resources/License.rtf
/usr/local/src/bru/bru_resources/ReadMe.rtf
/usr/local/src/bru/bru_resources/Welcome.rtf
/usr/local/src/bru/bru_resources/background.jpg
```

F: 0 A: 1 Status: Performing CRC Verification ...

Figure 11 - Verification Dialog

Verification Options

Selecting the "`Verification`" button from the main display will display XBRU's "`Archive Verification`" window. BRU employs two methods for verifying archives.

Checksum Verification

As archives are created, BRU automatically calculates a checksum for each block of data written to the archive and stores it in the header for that block. Checksum Verification (also known as "`Anytime Verify`") reads each block of data, recalculates the checksum for that block of data, and compares it to the checksum that was written into the block header. XBRU will list each file as it is verified, along with any errors found. At the end of the operation, the total number of checksum errors found will be displayed. If no errors are found, then the integrity of the backup has been preserved and a user should have no trouble restoring it. One advantage to the Checksum Verification is that it can be done days, weeks, even months later with no loss of accuracy. Checksum Verification also allows for the process to be performed on a system other than the one the archive was created on.

Differences Verification

Differences Verification performs the same kind of data-integrity check used by many other backup products. Frequently referred to as "bit-level verification", this mode allows BRU to compare the files in the archive to the files on the hard drive. Any discrepancies, such as changed modification times, content differences, or files in the archive that are not on the disk, will be noted. After BRU has completed the differences listing, it will insert a line stating "`***END`

`OF DIFFERENCES ***.`" If this is the only line that appears, the files in the archive are identical to the files on the hard drive.

Advanced Uses

Multiple Archives on One Tape

XBRU supports writing multiple archives to a single tape. This option is only available for tapes and does not function with disks. This is particularly useful in environments where restoring a single file is a frequent and time-consuming process, as BRU has to read an entire archive during the restore operation. This can result in the amount of time taken to restore a single file being comparable to the amount of time it took to create the backup in the first place. This time factor can be reduced by splitting the backup into sections. For example, a directory such as "`/home`" is typically a good candidate for being a separate backup. If a separate archive is created for "`/home`", BRU only needs to scan that smaller archive during the restoration of any files in it.

Creating the Archives

Using XBRU to create multiple archives on a tape is a simple process. When creating a backup, XBRU will display a prompt requesting a selection to "`Append`" or "`Overwrite`" the tape. Selecting "`Append`" will create multiple archives on the tape.

Working with Multiple Archives

Once multiple archives have been created on a single tape, specific archives can be selected with the "`Next Archive`" and "`Prev Archive`" options from the "`Tape`" menu. These options are used to navigate to and from different archives on the tape. When an archive is accessed, listings, verifications, and restores will become operational. When finished with the archive, the tape should be repositioned to the beginning of the archive. Use the "`Rewind`" button to reset the tape to the first archive.

Other Tape Functions

XBRU also supports a number of other useful tape operations. These options are available from the "`Tape`" menu.

Erase

The extent of this command's functionality is based on the type of tape drive used. For some drives, it will write out fresh block information, allowing the tape to be used with multiple archives. For others, it will write an "`End of Recorded Data`" (EOD) mark to the beginning of the tape, effectively erasing it. Regardless of any of its proprietary functions, "`Erase`" will almost certainly **DESTROY** any existing data on the tape.

Retension

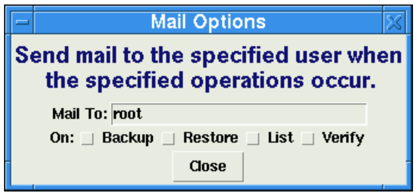
Retensioning a tape will forward it all the way to the end of the tape and then rewind it back to the beginning. This is useful for tapes that have been used many times over and may be experiencing write or read errors.

This option is also recommended before writing to any Travan drive (TR-3).

Rewind

"`Rewind`" returns a tape to its beginning. This can be useful if a user has been working with multiple archives and wants to move quickly to the beginning of the tape.

E-mail Notification



Send mail to the specified user when the specified operations occur.

Mail To: root

On: Backup Restore List Verify

Figure 12 - E-Mail Settings Dialog

XBRU can be configured to forward e-mail to a specified user when scheduled events (i.e. Backups, Restores, Listings) occur. This feature can be activated by running XBRU as root and selecting the "`Mail-To:`" option from the "`Options`" menu. A dialog will appear where a user's name can be specified and events can be selected to trigger the e-mail notification. Use "`Save Options`" from the "`Options`" menu to store the settings as "`System-Wide defaults`". **NOTE**: The e-mail notification settings can only be accessed as root.

Appendix A - Files That Are Installed

The BRU files that should be installed on your system are listed below. For simplicity, we often refer to a tape drive rather than listing all the possible media. Where we have done this, it should be understood that all media are included.

`/bin/bru`

The bru backup and restore program.

`/etc/brutab`

The parameter file that customizes BRU for your system. You will need to further customize this file by entering the parameters for your tape drive.

`/etc/bruxpat`

A file in which you may specify patterns that will match files you want to include in or exclude from your backups and restores. The `bruxpat` file permits BRU to archive and extract files selectively. See Appendix J, “The `bruxpat` File.”

`/bin/incbru`

A shell script that uses BRU to perform an incremental backup. The `incbru` command will speed daily backups by causing BRU to back up only files that have been changed since the last time `fullbru` was run. You may want to customize the `incbru` script to handle your specific backup needs. See Appendix H, “The Shell Scripts `incbru` and `fullbru`.”

`/bin/fullbru`

A shell script that will perform a full system backup (level 0) as part of an incremental backup procedure. The `fullbru` script works with `incbru`, described above.

`/bin/showxlog`

A shell script that displays the last few lines of the BRU execution log file, `/var/log/bruexeclog`.

`/bin/brutalk`

A program used to communicate with BRU when BRU is run in background mode. See Appendix I, “The `brutalk` Program.”

`/bru/README`

A file that contains specific information for your type of system. `/bru/README` also contains information that may have been too late to be included in this manual.

`/bru/bruhelp`

The contents of this file `BRUHELP`. This file is a complete list of all of the BRU options and flags.

`/etc/brusmartrest`

This file contains information regarding to restoring open or shared files.

`xbru/`

The `xbru` directory, is installed within `/usr/local/xbru` by default but you may select otherwise during installation. This directory contains all of the files associated with BRU's graphical interface. Symbolic links are created between `/usr/local/bin` (default - you may choose otherwise) and the actual executables in the `xbru` directory.

The file names shown above are where the BRU files are installed on most standard UNIX systems. On some versions of UNIX, the locations may vary slightly. Additional files and utility programs may also be installed in the `/bru` directory. These are optional and are not needed to make BRU work properly.

The `/bru` directory contains a spare copy of the `bru` executable as `/bru/bru`. This is not the version that is normally executed. `/bru/bru` is stored as a safety precaution, in case the executable copy, stored as /bin/bru, is destroyed. **NOTE**: The main copy of BRU is installed in the `/bin` directory because /bin is normally part of the root filesystem. Since the root filesystem is always mounted, this ensures that BRU will always be available to restore files. If you install BRU in a directory that is part of another (non-root) filesystem, you may have problems if your system fails: If the filesystem on which BRU is installed is damaged and `unmountable`, you will be unable to use BRU to restore files after such a system failure.

For this reason, please do not move `/bin/bru`, or the `/bru` directory from your `/` (root) filesystem. In the event that your system fails, having them there can mean the difference between recovery and closing your doors.

Appendix B - The BRU Help Command

This section describes BRU's two help options, `-h` and `-hh`. The first prints an abbreviated manual page for quick reference. The second lists `brutab` parameters and their current values.

Using `bru` with the `-h` Option

This is the mini-manual page that is printed when BRU is invoked with the `-h` (help) option. It is intended to be a concise reminder of the various options to BRU and their meanings. This sample page may not be identical to the version printed by your particular copy of BRU. Generally, this is a duplication of the BRU `manpage`. Please refer to "Appendix C: The BRU Manual Page."

Using `bru` with the `-hh` Option

BRU may also be invoked with the `-hh` option. This version of the command looks up all the parameters for the default device and prints their current values. It is a good way to verify parameter values after they have been changed. The `-hh` option has the following variant: It may be followed by `-f` and a device name. For example:

```
# bru -hh -f /dev/rmt0
```

In this case, the command will print the current parameter values for the named device. This command is used mostly to verify that BRU is reading the proper values from the `brutab` file. Invoking `bru` with `-hh` produces a listing similar to the following:

```
**** environment variables / global brutab parameters ****
BRUTABONLY=NO
DEVNAMECHECK=NO
MATCHLEVEL=2
MAXFILENAMELEN=512
MAXWRITES=1000
OVERWRITEPROTECT=YES
READCHECKLEVEL=1
RECYCLEDAYS=1
ZBUFSIZE=1048320
ZINBUFSIZE=4096
DIRDATESELECT=NO
ZOUTBUFSIZE=6144
MOUNTCMD="(null)"
UNMOUNTCMD="(null)"
BRUHELP="/bru/bruhelp"
BRUMAXWARNINGS=1000
BRUMAXERRORS=500
BRUXPAT="/etc/bruxpat"
BRURAW="/etc/bruraw"
BRUSMARTREST="/etc/brusmartrest"
BRUREMOVELOG="/usr/adm/bruremovelog"
BRUTMPDIR="/usr/tmp"
```

```
 **** brutab parameters for device: /dev/rct2 ****

devname=”EXABYTE Tape”
tape: ON
rawtape: ON
reblocks: OFF
ignoreclose: OFF
noautoscan: OFF
rawfloppy: OFF
shmcopy: ON
format: OFF
eject: OFF
qfwrite: OFF
reopen: ON
noreopen: OFF
norewind: OFF
advance: OFF
openrw: OFF
largefile: OFF
resetcmd=”(null)”
fmtcmd=”format -f /dev/rct2”
rfmcmd=”(null)”
wfmcmd=”(null)”
retencmd=”(null)”
rewindcmd=”(null)”
positioncmd=”(null)”
locatecmd=”(null)”
ejectcmd=”(null)”
openretries=1
iotimeout=0
iowait=0
opentimeout=3600
maxrewindtime=0
minrewindtime=25
size=1952000K
bufsize=20480
asbufsize=262144
maxbufsize=0
minbufsize=0
blocksize=0
endnulls=0
shmseg=7
shmmax=204800
shmall=0
seek=0
prerr=0
pwerr=0
zrerr=0
zwerr=0
frerr=0
```

```
fwerr=0
wperr=5
ederr=0
rmtsh="no-rsh"
rmtsvr="/etc/rmt"

**** system info ****
compile date: Wed Feb  1 14:32:00 1995
compiled on: odt20 odt20 3.2 2 i386
```

Using `bru` with the `-hhh` Option

BRU, when run with this option will produce a list of the error code definitions recognized by BRU.

```
**** error code definitions ****

EPERM=1
ENOENT=2
ESRCH=3
EINTR=4
EIO=5
ENXIO=6
E2BIG=7
ENOEXEC=8
EBADF=9
ECHILD=10
EAGAIN=11
ENOMEM=12
EACCES=13
EFAULT=14
ENOTBLK=15
EBUSY=16
EEXIST=17
EXDEV=18
ENODEV=19
ENOTDIR=20
EISDIR=21
EINVAL=22
ENFILE=23
EMFILE=24
ENOTTY=25
ETXTBSY=26
EFBIG=27
ENOSPC=28
ESPIPE=29
EROFS=30
EMLINK=31
EPIPE=32
EDOM=33
ERANGE=34
```

```
EWOULDBLOCK=90
EINPROGRESS=91
EALREADY=92
ENOTSOCK=93
EDESTADDRREQ=94
EMSGSIZE=95
EPROTOTYPE=96
ENOPROTOOPT=118
EPROTONOSUPPORT=97
ESOCKTNOSUPPORT=98
EOPNOTSUPP=99
EPFNOSUPPORT=100
EAFNOSUPPORT=101
EADDRINUSE=102
EADDRNOTAVAIL=103
ENETDOWN=104
ENETUNREACH=105
ENETRESET=106
ECONNABORTED=107
ECONNRESET=108
ENOBUFS=63
EISCONN=110
ENOTCONN=111
ESHUTDOWN=112
ETIMEDOUT=114
ECONNREFUSED=115
ELOOP=150
ENAMETOOLONG=78
ENOTEMPTY=145
ENOMSG=35
EIDRM=36
ECHRNG=37
EL2NSYNC=38
EL3HLT=39
EL3RST=40
ELNRNG=41
EUNATCH=42
ENOCSI=43
EL2HLT=44
EDEADLOCK=56
```

Appendix C The BRU Manual Page

The full BRU manual page follows. Appendix B is an abbreviated form of this manual page. It contains a help list of bru options and short definitions. It also shows you how to list brutab parameter values and device information. Appendix D contains a table of options cross referenced to the bru modes with which they may be used.

NAME

bru - backup and restore utility

SYNOPSIS

bru modes [control options] [selection options] [files]

DESCRIPTION

Bru is a UNIX filesystem backup utility with significant enhancements over other more common utilities such as tar, cpio, volcopy, and dd. Some of bru's capabilities include:

o Full or incremental backup with quick and easy restoration of files.

o Multiple physical volumes per archive.

o Data integrity assurance via checksum computation on every archive block.

o Ability to save and restore directories, symbolic links, block special files, and character special files.

o Comparison of archives with current directory hierarchy.

o Ability to recover files from corrupted archives or damaged media with minimal data loss.

o No inherent maximum archive buffer size.

o Improved performance through random access archive I/O when available.

o Automatic byte or half word swapping as necessary when reading archives produced on other machines.

o Recognition of filename generation patterns in the same form as the shell for files read from an archive.

o Intelligent handling of large, sparse files.

When files are specified on the command line then the actions to be performed are limited to those files. If a named file is a directory then it and all its descendants are used. If no files are specified then the default for writing archives is all files in and below the current directory. The default for reading archives is selection of all files in the archive.

If "-" is given instead of files then the standard input is read to obtain the file list. This is useful in conjunction with the find command to provide finer control over files selected for backup. Obviously this mode is only valid when bru is not also reading its archive from the standard input.

DEFAULTS

Various default parameters, such as archive device name and size, archive buffer size, controlling terminal name, etc., are system dependent. These defaults, along with version, variant, and other miscellaneous internal information, may be discovered via the -h mode.

MODES

One or more of the following modes must be specified. The order of execution, from highest priority to lowest, is ecitxdgh.

-c Create a new archive. Forces a new archive to be created regardless of whether one currently exists. Writing starts at the first block.

-d Differences between archived files and current files are detected and reported. May be specified more than once, as -dd, -ddd, or -dddd, to control level of difference checking.

When specified as -d, bru reports when it discovers that a regular file's size (st_size) or contents (when compared as byte streams) has changed since the archive was made.

When specified as -dd, bru reports additional differences in modification date (st_mtime) access mode (st_mode), number of links (st_nlink) for non-directory files, differences in the contents of symbolic links, owner id (st_uid), and group id (st_gid).

When specified as -ddd, bru reports additional differences in host device (st_dev), major/minor device (st_rdev) for special files, and time of last access (st_atime) for regular files.

When specified as -dddd, bru reports all differences except time of last status change (st_ctime is not capable of being reset), major/minor device numbers for non-special files (meaningless), and size differences for directory files (may have empty entries). The -dddd mode is generally meaningful only during a verification pass with full backups of quiescent filesystems.

-e Estimate media requirements for archive creation with same arguments. Prints estimated number of volumes, number of files to be archived, total number of archive blocks, and total size of archive in kilobytes. If the media size is unknown or unspecified, it is assumed to be infinite. Estimates are for uncompressed data only; the -Z option will be ignored.

-g Dump archive info block in a form more easily parsed by programs implementing a complete filesystem management package. Performs no other archive actions.

-gg List archive list of files created during the bru -cvG mode. This has been deprecated and is no longer used in BRU 17.0.

-h Print help summary of options. Also prints some internal information such as version number and default values for archive pathname, media size, archive buffer size, etc.

-i Inspect archive for internal consistency and data integrity. When -vv option is also given, prints information from archive header block.

-t List table of contents of archive. When used with the -v option, will give a verbose table of contents in the same format as the "ls -l" command. When used with the -vv option, will also indicate what files are linked to other files and where symbolic links point.

-x Extract named files from archive. If an archived file is extracted (see -u option) then the access mode, device id (special files only), owner uid, group uid, access time, and modification time are also restored. If the -C flag is given (see below), then the owner uid and group uid will be changed to that of the current user.

Nonexistent directories are recreated from archived directories if possible. Otherwise they are created with appropriate defaults for the current user. Extracted or created directories are initially empty.

CONTROL OPTIONS

Many of the control options are similar in function to their tar or cpio equivalents.

Sizes are specified in bytes. The scale factors G, M, k, or b, can be used to indicate multiplication by 1024*1024*1024 (one Gigabyte), 1024*1024 (one Megabyte), 1024, or 2048 (the size of a bru tape block) respectively. Thus “10k”, “5b”, and “10240” all specify the same number of bytes.

-a Reset the access times of disk files that have been read while performing other actions.

-b bsize

Use bsize as the archive input/output buffer size. The minimum is the size of an archive block (2k or 2048 bytes) and the maximum is determined by available memory and I/O device limitations. If bsize is not an even multiple of 2048 bytes, it will be rounded up. Normally this option is only required with -c mode since bru writes this information in the archive header block. If specified, bsize overrides any existing default value (generally 20k), whether built in or read from the

archive header.

-B Useful in shell scripts where bru is run in the background with no operator present. Under these conditions, bru simply terminates with appropriate error messages and status, rather than attempting interaction with the terminal.

-C Change the owner (chown) and group of each extracted file to the owner uid and group gid of the current user. Normally, bru will restore the owner and group to those recorded in the archive. This flag causes bru to follow the system default, with extracted files having the same owner and group as the user running bru, including root. (Under 4.2 BSD, the default group is that of the directory in which the file is created.)

The -C option is useful with archives imported from other systems. In general, it should not be used by the operator or system administrator when restoring saved file. Use the -tv option to see the owner and group of files stored in the archive.

-D Causes bru to use double buffering to the archive device on systems that have System V-style shared memory. Depending on hardware constraints, double buffering may dramatically increase the archive device I/O rate but may adversely affect the error recovery algorithms.

-f path

Use path as the archive file instead of the default. If the path is “-” then bru uses the standard input for archive reading or standard output for archive writing, as appropriate.

If multiple -f options are given, each path is added to a list of files to cycle through each time a volume change is required. When the end of the list is reached, bru automatically cycles back to the first path and waits for confirmation to continue the cycle again. Any input other than a carriage return will cause bru to use the newly entered path and to abort the cycling for the remainder of the current run.

-F Checksum computations and comparisons are disabled. This mode is useful when the output of one bru is piped to the input of another bru, or when the data integrity of the archive transmission medium is essentially perfect. Archives recorded in this mode must also be read in this mode. Be aware that some of the automatic features of bru, such as automatic byte swapping and AUTOSCAN, are not functional in fast mode.

-Iqrlb

When BRU is running in the background, there is no way to interact with it. The -I qrlb options allow a users to setup a set of read and write fifos that a second program BRUTALK and work with to responed with BRU. Below we have listed the “qrlb” options that are currently valid with the -I option.

```
q,fifo write queries to fifo
    r,fifo read responses from fifo
    l,file write log info to file
    b Use the default fifos created durning install.
           /dev/bru.q   (Default send fifo)
           /dev/bru.r   (Default reply fifo)
```

The q and r options are useful for interacting with bru when it has been run without a controlling terminal. See the discussion under RUNNING FROM CRON.

-L str (c)

In create mode, label tape with given string (63 char max).

-L file

In create mode, the first 63 characters in file are used as the label for the archive members. In extract mode the first 63 characters within file are used for the comparison as described below.

-L str (x)

In extract mode, only restore if the label of the archive EXACTLY matches the given string. If the string does not match on the initial tape, the operation aborts. If the label does not match on subsequent tapes, a warning is issued, but the extraction continues.

-r rawdev

This enables BRU to backup or restore raw data partitions. A BRURAW file must exist and contain entries that define the raw data to be accessed. See the discussion of the BRURAW global variable below for information on the BRURAW file and location. The file contains entries in this format:

```
    Raw Device Name      Size    Blk Size  Starting Offset
    ________-            __      ____      ________-

    a sample entry would be:

    /dev/rfd0135ds9      720k    512             0
```

An entry MUST exist in the BRURAW file or bru will abort the operation with an error message.

When backing up raw partitions, you must either have the device node in the tree relative to your current location or you must explicitly declare the raw device on the command line as a file name. For example, if you are in / and the raw device node is /dev/rdsk/c0t3d2s1, you can use this command line to back it up properly:

```
bru -cvV -r /dev/rdsk/c0t3d2s1
```

If, however, you are invoking the bru command from your home directory, you must include the path to the device node on the command line:

```
bru -cvV -r /dev/rdsk/c0t3d2s1 /dev/rdsk/c0t3d2s1
```

Additionally, the filesystem being backed up MUST be locked or unmounted to prevent any writes from occurring during the backup. If this is not done, the data could be corrupted during the process of backing it up.

-T file

Translate on restore - rename or relocate files based on the contents of a translation file. The file can be any file name. It is an ASCII text file which contains two columns - all files that contain the text in column 1 will have that text translated to the text in column 2.

* It is important to note that there can be NO empty fields in this file. You MUST provide both columns for each line in the translation file.

This translation applies to directories, names and extensions. By default, symbolic links will not be translated. See the -Q option below. Translate works with all BRU modes except create (-c).

To restore all of the files in /home/bob to /home/ted, you should have the following information in your file.:

```
/home/bob    /home/ted
```

No further interaction is required by the user. Also multiple translations can occur within a single restore. We could perform both of the above translations and others by placing the columns into a single translation file as follows:

```
/home/bob    /home/ted
/usr/lib     /usr2/lib
/home  /u2
...etc
```

The translation file can contain as many translation lines as necessary, but each line must consist of a balanced pair of entries.

-Q HLRSV

Changes default BRU operation as described below:

H This turns off placing small files into tape headers. Using this option allows you to create BRU tapes that are compatible with versions of BRU prior to 14.2

L Use a literal string as a tape label. This overrides BRU's attempt to look for a file from which to read the tape label. This can be useful if you have a file that is the same name as the label you wish to apply.

R Disable SmartRestore. This turns off BRU's handling of open or shared files. It is not recommended that you override this setting.

S Translate Symbolic Links. Used in conjunction with the -T option (see above) will force the translation of symbolic links.

V Ignore "Incorrect Volume" warnings. When restoring from a multi-tape set or beginning a restore from other than the first tape in a set, use of this option will prevent the normal warning about having the incorrect

volume and continue with the restore.

-l Ignore unresolved links. Normally bru reports problems with unresolved links (both regular and symbolic links). This option suppresses all such complaints.

-m Do not cross mounted file system boundaries during expansion of explicitly named directories. This option applies only to directories named in files. It limits selection of directory descendants to those located on the same filesystem as the explicitly named directory. This option currently applies only to the -c and -e modes.

-N level

Use level as the compression level if -Z is also specified. The default is to use level 3. The allowable values are 1 - 9, with 1 being lowest. The higher the level, the more agreesively BRU will work to compress the file. This will be at the cost of processor power - higher compression uses more CPU power.

-O Overwrite the archive regardless of the settings for OVERWRITEPROTECT and RECYCLEDAYS.

-p Pass over files in archive by reading rather than seeking. Normally bru will use random access capabilities if available. This option forces reads instead of seeks. Use only on disk files or disk media.

-P flags

Pathname options that provide explicit control of expansion of directories, automatic archiving of parent directories, etc. Possible characters for flags are:

e Disable automatic expansion of explicitly named directories.

E Enable automatic expansion of explicitly named directories

f Disable filter mode. Builds internal tree of pathnames before doing anything with the pathnames from the input list.

F Enable filter mode. Each pathname from the input list is treated individually, and processing is done with that pathname before the next pathname is examined.

p Disable automatic archiving of parent directories of explicitly named files or directories.

P Enable automatic archiving of parent directories of explicitly named files or directories.

See the discussion under DIRECTORIES.

-R Remote files are to be excluded from the archive. If the system does not support remote filesystems, this option is ignored.

-s msize

Use msize as the media size. The effective media size will be computed from msize since it must be an integral multiple of the input/output buffer size (see the -b option). Normally this option is only required with the -c mode since bru writes this information in the archive header block. If specified, msize overrides any existing default value, whether built in or read from the archive header.

-S size

Enable options to deal more intelligently with sparse files (files with lots of null bytes). When used in conjunction with the -c mode, turns on automatic file compression for files that are larger than the specified size. When used in conjunction with the -x mode, seeks will be used to create blocks of null bytes in the output file, rather than actually writing null bytes. See the discussion under SPARSE FILES.

-v Enable verbose mode. May be specified more than once, as -vv, -vvv, or -vvvv, to get even more verbosity. The -vvvv form of this option includes an execution summary (see -V).

-V Print execution summary only. The -vvvv option includes an execution summary as part of its output.

-X Apply exclusion patterns specified in the file /etc/bruxpat (or in the file specified by the BRUXPAT environment variable).

-w Wait for confirmation. bru will print the file name and the action to be taken and will wait for confirmation. Any response beginning with ‘y’ or ‘Y’ will cause the action to complete. Any response beginning with ‘g’ or ‘G’ will cause the action to complete and will reset the -w option so that no further confirmations will be requested. Any other response will abort the action.

-Z Use LZW file compression. This is not the default because not all versions of bru know how to deal with compressed files. When the -v option is also selected, the compression ratio for each file is printed as a percentage. When this flag is used in conjunction with the -t option on an archive that contains compressed files, the actual archive file sizes and names are printed, rather than the original values before archiving.

A limited amount of backward compatibility with non-compressed versions of bru is provided. Archives read by older versions will appear to contain files that were precompressed prior to archiving. The public domain compress utility can be used to decompress such files after dearchiving. See the -N option.

FILE SELECTION OPTIONS

The file selection options control which files are selected for processing. Note that some options are valid only with specific modes.

-n date

Select only files newer than date. The date is given in one of the forms:

Format	Example
DD-MMM-YY[,HH:MM:SS],[amc]	12-Mar-84,12:45:00.ac
MM/DD/YY[,HH:MM:SS],[amc]	3/12/84,ac
MMDDHHMM[YY],[amc]	0312124584,ac
pathname	/etc/lastfullbackup

The time of day is optional in the first two forms. If present, it is separated from the date by a comma.

By default, the modification and create times are used for comparison. Other times can be used by specifying letters after the date (a=access time, m=modification time, c=create time). For example:

-n 14-Apr-84,15:24:00,ac

If the date is preceded by an exclamation point (!), then files older than the specified date will be selected.

If date is the pathname of a file, then the modification date of that file will be used. This is useful in automated backups when a dummy file is “touched” to save the date of the last backup. NOTE: do not use the [amc] modifier with tis option.

-o user

Select only files owned by user. User may be specified in one of three ways:

As an ascii string corresponding to a user name in the password file.

As the pathname of a file, in which case the owner of that file is used.

As a numeric (decimal) value. This value will be the user ID as found in the passwd file.

-u flags

When used in conjunction with -x mode, causes files of the type specified by flags to be unconditionally selected regardless of modification times. Normally bru will not overwrite (supersede) an existing file with an older archive file of the same name. Files which are not superseded will give warnings if verbose mode level 2 (-vv) or higher is enabled. Possible characters for flags are:

a	select all files (same as specifying all flags)
b	select block special files
c	select character special files
d	select directories
l	select symbolic links
p	select fifos (named pipes)
f	select regular files (same as r)
r	select regular files (same as f)

-U #Selection depth for backup or restore of files in relation to the current directory depth. Files more than (#) levels Under this level will not be processed.

For example, if the current directory is /home/ted, -U0 (zero) will only backup the files and directory nodes in the home/ted directory. While the directory nodes will be backed up, any files in directories off of this level will not be backed up. With a level of 2, BRU will backup files in /home/ted/test, /home/ted/test/runone, /home/ted/test/runtwo, but not in /home/ted/test/runone/demo1.

This same feature works for restoring files and directory trees as well.

Selection of directories implies that only their attributes may be modified. Existing directories are never overwritten, this option merely allows their attributes to be set back to some previously existing state.

Selection of symbolic links implies that only the contents of the link will be modified. It is currently impossible under 4.2 BSD to change access time, modification time, or the file mode of a symbolic link.

EXAMPLES

Create (-c) a new archive of all files under “/usr/src”, writing archive to file (-f) “/dev/rmt0” using multiple tapes with a maximum size (-s) of 30 megabytes per tape.

```
bru -c -f /dev/rmt0 -s 30M /usr/src
```

Create (-c) a new archive on the default device in the first pass, archiving all files in and below the current directory which have been created or modified since 3 P.M. on 14-Jan-92 (-n). Then do a second pass to verify that there are no differences (-d) between the archive and current files. Each file is listed (-v) as it is processed.

```
bru -cvd -n 14-Jan-92,15:00:00
```

Archive all files owned (-o) by user “user1” using the default archive device.

```
bru -c -o user1 /
```

Copy a directory hierarchy from “/usr/u1” to “/usr/u2”.

```
(cd /usr/u1; bru -cf - ) | (cd /usr/u2; bru -xf -)
```

Extract (-x) the regular file “/usr/guest/myfile” unconditionally (-ur) from an archive on file (-f) “/dev/rf0”. Since the device size was recorded in the header block, it need not be specified. Note that option arguments do not need to be separated from their corresponding option flag by whitespace.

```
bru -x -ur -f/dev/rf0 ./usr/guest/myfile
```

Extract (-x) all C source files in “/usr/src/cmd” that have names beginning with characters ‘a’ through ‘m’. Wait (-w) for confirmation before extracting each file.

```
bru -xw ‘/usr/src/cmd/[a-m]*.c’
```

Inspect (-i) a previously created archive on the default device, dumping the contents of the header block for inspection (-vvv) and verifying the internal consistency and data integrity of the archive.

```
     bru -ivvv
```

Back up the entire root filesystem without crossing mounted (-m) filesystem boundaries. The archive will be written to file (-f) “/dev/rmt0” using an I/O buffer size (-b) of 10k bytes. A record of all files processed will be written to file “brulogfile” for future reference.

```
     cd /
     bru -cvm -f /dev/rmt0 -b 10k >brulogfile
```

DIAGNOSTICS

Most diagnostics are reasonably informative. The most common have to do with meaningless combinations of options, incompatible options, hitting memory or device limits, unresolved file links, trying to archive or restore something to which access is normally denied, or problems with media errors and/or archive corruption.

DEVICE TABLE

bru contains an internal table of known devices and their characteristics. bru first looks for an environment variable BRUTAB, which contains the name of the dynamically loaded file if it begins with a ‘/’ character, or contains device descriptions if the first character is not ‘/’. If there is no BRUTAB environment variable, the file /etc/brutab is loaded. If neither of the preceding is found, an internal default

description is loaded.

SIGNAL HANDLING

bru normally catches both interrupt (SIGINT) and quit (SIGQUIT) signals. When an interrupt is caught during archive creation or extraction, bru completes its work on the current file before cleaning up and exiting. This is the normal way of aborting bru. When a quit signal is caught, an immediate exit is taken.

Note that during file extraction, a quit signal may leave the last file only partially extracted. Similarly, a quit signal during archive creation may leave the archive truncated.

When either interrupt or quit is caught at any time other than during archive creation or extraction, an immediate exit is taken.

ERROR RECOVERY

When properly configured for a given software/hardware environment, bru can recover from most common errors. For example, attempts to use unformatted media are detected, allowing substitution of formatted media. Random blocks in an archive can be deliberately overwritten (corrupted) without affecting bru’s ability to recover data from the rest of the archive. When I/O errors are detected, retries are performed automatically. Out of order sequencing on multi-volume archive reads is detected, allowing replacement with the correct volume.

DIRECTORIES

bru takes two actions with respect to directories that make creation and extraction of entire hierarchies of files more convenient and complete. These actions are automatic archiving of parent directories and automatic expansion of explicitly named directories.

Automatic archiving of parent directories means that when bru is given the complete pathname of a file to archive, it attempts to automatically archive all parent directory nodes necessary to fully restore the specified file. During extraction, any required directories which do not already exist are restored from the archive if possible, otherwise they are created with appropriate defaults for the current user. When bru reads it's list of files from the standard input, or when the -Pp option is given, this automatic archiving of parent directory nodes is suppressed. Note also that when creating archives with additional constraints on the selected files (such as use of the -n option), these parent directories may be excluded.

Automatic expansion of explicitly named directories means that when bru is given an explicit file name that names a directory node, not only is that directory node archived, but all files and subdirectories in that directory are archived. I.e., the entire file hierarchy rooted in the explicitly named directory is archived. When bru reads its list of files from the standard input, or when the -Pe option is given, this automatic expansion of directories is suppressed.

Note that incremental archives, archives created with the -Pp option, or archives created from a list of files supplied on the standard input stream, may not contain all of the necessary parent directories to replicate the original hierarchy and thus may result in creation of directories with the default attributes when files are extracted from the archive.

When bru reads the list of files from the standard input stream, the default values for the -P options are -PeFp, which turns off expansion of directories, turns on filter mode, and turns off automatic archiving of parent directories. This allows bru to be conveniently used to archive only filesystem nodes that are explicitly named on the input list.

When files are explicitly named on the command line (or default to '.'), the default values for the -P options are -PEfP, which turns on expansion of directories, turns off filter mode, and turns on automatic archiving of parent directories. This is typically the most convenient behavior for arguments given on the command line.

WILDCARDS

When reading archives, bru recognizes file name generation patterns in the same format as the shell. This allows greater flexibility in specifying files to be extracted, compared, or listed. As a special extension of shell-type expansion, the sense of the match is reversed for patterns that begin with '!'.

Note that the patterns may have to be quoted to prevent expansion by the shell. Also note that patterns are processed independently, without regard to any other patterns that may or may not be present. In particular, "/bin/a* /bin/b*" is equivalent to "/bin/[ab]*", but "/bin/!a* /bin/!b*" is equivalent to "/bin/*", not "/bin/![ab]*".

BYTE/WORD SWAPPING

While reading archives produced on other machines, bru automatically attempts to perform byte and/or word swapping as necessary.

REMOTE TAPE DRIVES

On 4.2 BSD systems, and System V systems that support networking, bru allows the use of remote tape drives for the archive device (via the -f option). A remote tape drive file name has the form

```
     system[.user]:/dev/???
```

where system is the remote system, the optional user is the login name to use on the remote system if different from the current user's login name, and /dev/??? is the tape drive to use (1600 BPI or 800 BPI, raw or blocked, rewinding or non- rewinding, etc.). In all cases, the user must have the appropriate permissions on the remote system. (See also the CAVEATS section, below.)

RUNNING FROM CRON

Sometimes it is convenient to run bru under conditions where there is no controlling terminal. This can be a problem if interaction is needed, such as when switching to a new volume. As an example, consider the case of running bru from cron, where the operator mounts a tape before leaving in the evening, and bru writes the first volume in the middle of the night. When returning in the morning, the operator wants to be able to mount a second tape if necessary, and instruct bru to continue.

If no interaction with the user is required, running from cron is no different than running directly from a terminal. However, when interaction is necessary there are basically two options; terminate, or find some way to communicate with the operator (or another program masquerading as the operator). The -B option provides for simple termination. The -I options provide for communication with an operator.

On systems that support fifos, a pair of fifos are used to send requests and receive replies. Before running bru, verifiy that these fifos are present if they are not then create the pair of fifos with the commands:

```
mknod /dev/bru.q p
mknod /dev/bru.r p
```

Then, add the arguments “Ib” or if you created any other name for your fifos then run the following command “-Iq,/dev/bru.q -Ir,/dev/bru.r” to the desired bru command line which ultimately gets executed undercron. The first time bru needs to communicate with an operator, it will open the two fifos, write a query to the bru.q fifo, and wait for a response from the bru.r fifo. A simple program provided with bru, called brutalk can be used to read the query and send a reply:

```
brutalk
brutalk /dev/bru.r
```

The brutalk program will continue to read queries and send replies until either bru exits, or a control-D (EOF) is typed at the terminal.

EXIT CODES

Bru always returns meaningful status as follows:

0	Normal exit, no errors or warnings.
1	Warnings (or interrupted).
2	Errors (or quit received).

Appendix D – Table of BRU Modes and Options

The following table is a list of the bru modes. See Appendix C, "The BRU Manual Page," for Full descriptions.

```
bru mode [control options] [selection options] [files]
MODE
  -ccreate a new archive with specified files
  -dfind differences between archived files and
          current files
  -eestimate media requirements for create mode
          (media size must be known)
  -gprint only information from archive header
  -gg     print only the list of files created during
          the -G option (for pre-17.0 archives)
  -hprint help information
  -iinspect archive for consistency and data
          integrity
  -tlist archive table of contents for files
  -xextract named files from archive
```

The following table shows which of the BRU options are valid for each mode. See Appendix C, "The BRU Manual Page," for Full descriptions.

```
MODES                  CONTROL OPTIONS
c,d,e,g,gg,h,i,t,x
*,*, , , , , , ,  -a Reset file access times after reads.
*,*,*, , , , ,*,* -b bufsize Set archive buffer size to bufsize bytes (scalable).
*,*,*, , , , , ,  -B Background mode. No interaction with operator
 , , , , , , , ,* -C Always chown extracted files to the user's uid/gid
*,*, , , , , , ,* -D On some systems, provides speedup via double buffering
*, , , , , , , ,* -F No checksum computations or checking (dangerous for backup)
*,*,*,*,*,*,*,*,* -Ib Reads and writes to default fifos.
*,*,*,*,*,*,*,*,* -Il,pathname Write verbosity information to pathname
*,*,*,*,*,*,*,*,* -Iq,fifo Write interaction queries to fifo
*,*,*,*,*,*,*,*,* -Ir,fifo Read interaction replies from fifo

c,d,e,g,gg,h,i,t,x
*,*,*, , , ,*, ,* -j Enables a progress output for files larger than 500K
*, , , , , , , ,  -l Suppress warnings about unresolved links
*, , , , , , , ,* -L string In Create (backup) mode, BRU Labels the archive with a given string from either the given string or from a file (63 character maximum). In Extract (restore) mode, BRU is forced to check the archive label and compare it to the given label. Can be disabled with -QL option.
*, , , , , , , ,  -m Limit directory expansions to the same mounted filesystem
```

*, , , , , , , , , -N level Use level for compression (default is 3). See -Z

*, , , , , , , , , -O Overwrite archive regardless of OVERWRITEPROTECT and RECYCLEDAYS settings

 ,*, , , , ,*,*,*-p Pass over archive files by reading rather than seeking.

, ,, , , ,*,*,*-Pa Turn off absolute to relative pathname conversion.

, ,, , , ,*,*,*-PA Turn on absolute to relative pathname conversion

, ,, , , , , , -Pe Turn off expansion of directories.

, ,, , , , , , -PE Turn on expansion of directories.

, ,, , , , , , -Pf Turn off filter mode (build internal file tree). For stdin input only.

, ,, , , , , , -PF Turn on filter mode (do not build internal tree).

, ,, , , , , , -Pp Turn off automatic archiving of parent directory nodes

, ,, , , , , , -PP Turn on automatic archiving of parent directory nodes

*, , , , , , , , , -QH Turns off placing small files into file headers (NOFILESINHDRS=YES)

 , , , , , , , , ,*-QR Disable SmartRestore

 , , , , , , , , ,*-QS Forces the translation of symbolic links during the -T translate on restore option

,, , , , ,*,*,*-QV Ignore the incorrect volume warning in a multi-volume operation

, , , , , , , , ,-r rawdev Enable backup or restore of a raw data partition

*, , , , , , , , , -R Exclude remotely mounted filesystems (NFS/RFS)

,,*, , , ,*,*,*-s size Specify the size of the archive media (K - kilobytes, M - Megabytes, G - Gigabytes). This applies to files, tapes, diskettes, etc.

, , , , , , , , ,-S size Turn on options to handle sparse files intelligently and set threshold to size.

 , , , , , , , , ,*-T filename Translates expression to new_expression as defined in filename

,,*,*, , , , , -U level Do not resolve directory levels greater than level below the current directory level

,,*,*,*,*,*,*,*-v Enable verbose mode (-vv, -vvv, -vvvv available for more verbosity)

,,*,*,*,*,*,*,*-V Print execution summary

, , , , , , , , ,-w Prompt for action and wait for operator response

, ,, , , , , , -Z Enable software compression (also see -N)

Appendix E - Sample `BRUTAB` ***File***

When completed, a typical `brutab` file will look like this. The first entry in the table represents the default archive device. If you do not specify a device on the command line (with the `-f` option), this is the archive device BRU will try to use.

```
#
#  *** GLOBAL BRUTAB PARAMETERS ****
#
#+OVERWRITEPROTECT=YES
#+ZBUFSIZE=500K
#+RECYCLEDAYS=14
#+MAXWRITES=200
#
#
#  FILE
#
#   brutab    bru data file for loadable device table
#
#
#  DESCRIPTION
#
#   Bru data file for loadable device table.
#   Note that the table MUST contain at least one entry
#   and the first entry is the default archive device.
#
#   Also, the table should contain an entry for the
#   standard input and output, with a name of "-".  This
#   entry SHOULD NOT be the first entry (or archives may
#   be inadvertently written to the user's terminal).
#
#   Entries consist of a device name field, followed by
#   one or more capabilities fields.  Entries may span
#   more than one line by escaping the newline at the end
#   of the line with a '\' character ('\' is last
#   character on line before newline).
#   All whitespace (tabs, blanks, newlines, formfeeds)
#   between fields is ignored.
#
#   The device name field must be the first field in the
#   entry and has the following form:
#
#          || ... |
#
#          ex:    /dev/rmt0|/dev/rmt1|/dev/rmt2
#
#   where each device has the same capabilities as the
#   other devices specified (a class of devices).
#
#   Each capability field is of the form:
#
#          =      or
#
#          ex:    size=640M   reopen  pwerr=EIO
#
#   Note that there can be no whitespace between the
#   capability name and the value.  Numeric values may be
#   given in absolute form or with a trailing scale factor
#   of:
#
#          b or B       Blocks (512 bytes)
#          k or K       Kilobytes (1024 bytes)
#          m or M       Megabytes (1024 * 1024 bytes)
#          g or G       Gigabytes (1024*1024*1024 bytes)
#
#   Currently used capabilities include:
#
# Name     Type         Meaning
# ————     —————        ————————————-
# bufsize  numeric             default I/O buffer size for this
#                       device (32Kb if omitted)
#                       (beware of shared mem limits)
# size     numeric             media size in bytes if known,
#                       zero if unknown or variable.
# seek     numeric             minimum seek resolution,
#                       zero if no seeks allowed
# reopen   boolean             close and reopen archive upon
#                       media switch
# noreopen boolean             no close and reopen archive upon
#                       media switch
# tape     boolean             archive device is a tape drive
# rawtape  boolean             archive device is a “raw” tape
#                       drive
# rawfloppy      boolean             archive device is a “raw” floppy
# norewind boolean             closing does not automatically
#                       rewind
#                       (“size” parameter should be
#                       zero)
# advance  boolean             read/writes advance media even
#                       when
#                       errors occur (most 9-track tape
#                       drives, few cartridge drives)
#                       WARNING: do not set if not true!
# qfwrite  boolean             query for OK on first write to
#                       device
# format    boolean     format media if necessary
# ignoreclose boolean   ignore errors from closing the
#                        archive
#                        device (bugs in some drivers)
#
################################################################
# Standard SCSI tape
# We set size=0 to allow the system to inform us when we
# reach the early warning (ENOSPC).
################################################################
/dev/st0 \
    size=0 seek=0 bufsize=32k shmseg=7 shmmax=200K \
    reopen rawtape tape rewind shmcopy

################################################################
#  Entry for “norewind” tape - notice that noautoscan and
#  norewind flags are set
################################################################
/dev/nst0 \
    size=0 seek=0 bufsize=32k shmseg=7 shmmax=200K \
    reopen rawtape tape norewind noautoscan shmcopy \
    fmtcmd="mt -f /dev/nst0 erase" \
    rfmcmd="mt -f /dev/nst0 fsf" \
    bfmcmd="mt -f /dev/nst0 bsfm 2" \
    retencmd="mt -f /dev/nst0 reten" \
    rewindcmd="mt -f /dev/nst0 rewind" \
    eodcmd="mt -f /dev/nst0 seod"

################################################################
#  3.5” 1.44M floppy drives
################################################################
/dev/fd0|/dev/fd0u1440 \
    size=1440K seek=2K bufsize=2k \
    format fmtcmd=”fdformat /dev/fd0u1440” reopen shmcopy

#===============================================================
#  This entry is a generic entry for stdin and stdout.  DO
#  NOT put this first or user’s will probably inadvertently
#  write archives to their terminals.
#===============================================================
- size=0 seek=0 bufsize=4k
```

Appendix F - BRU Execution Log

Every time BRU is executed, it writes a message to an execution log. The default log file is `/var/log/bruexeclog`.

Each entry in the execution log is a single line that contains the following information:

- date and time
- process ID of the BRU task
- ID of the user who started BRU
- message number
- message text

A line is written to the execution log when BRU is started and when BRU terminates. A line is also written to the log for each error or warning message. The execution log is designed to help you monitor your backups. Since it maintains a record of every time BRU was executed, you can tell how long it took to do a backup. Since it records all warnings and error messages, it is also useful for trouble-shooting. If a problem occurs while using BRU, the log will contain a record of what happened.

You can change the location of the execution log by setting the BRUEXECLOG environment variable. You should do this only if the default location is not satisfactory (because of limited space in that filesystem, for example).

The following two command lines will change the location of the execution log to `/etc/newbrulog`.

```
BRUEXECLOG=/etc/newbrulog
export BRUEXECLOG
```

The log file should not take up much disk space, since most users do not execute BRU more than once or twice a day. Even a year's worth of log entries should take less than 100K bytes of space. We do not recommend it, but you can disable the execution log by setting BRUEXECLOG to a null string (i.e., `BRUEXECLOG=""`).

NOTE: If you have created a root and a boot floppy for crash recovery, you may need to set `BRUEXECLOG=/dev/null`. BRU will attempt to write to or create a BRUEXECLOG file whenever you run any of the BRU commands. This can cause you a problem if you do not have room on the boot device to create and append to the file.

Execution logging will also be disabled if BRU is renamed (or a link is created) and executed as `Bru` (with a capital B).

IMPORTANT NOTE: If you are having difficulties with BRU, please save the information in the execution log file. This will help BRU Technical Support to identify and solve your specific problem.

BRU Execution Summary

If BRU's verbosity is specified as `-vvvv` or greater, an execution summary will be printed when BRU completes execution. If you wish to print the execution summary only, without listing all the files, the `-V` (capital V) option should be specified.

```
Here's a typical execution summary:
```

```
**** bru: execution summary ****
Started:                Mon Feb 27 12:59:05 1995
Completed:              Mon Feb 27 13:01:38 1995
Archive id:                   2f522f092adb
Messages:               0 warnings,  0 errors
Archive I/O:                  3690 blocks (7380Kb) written
Archive I/O:                  3690 blocks (7380Kb) read
Files written:          20 files (20 regular, 0 other)
Files read:                   20 files (20 regular, 0 other)
Files in headers:       2
Write errors:           0 soft,  0 hard
Read errors:                  0 soft,  0 hard
Checksum errors:        0
Min Compression:        41%
Avg Compression:        71%
Max Compression:        76%
Max Compressed Size:          2424613 bytes
Compression Overflows:  2
Difference count:       0
```

The fields are defined below:

`Started`: Date and time the process started.

`Completed`: Date and Time the process completed.

`Archive id`: A unique 12-character archive identifier.

`Messages`: A count of the total number of warning and error messages.

`Archive I/O`: Number of BRU blocks (2Kb) written to the archive device. This should be zero if you are just reading (extracting files or inspecting) a tape

`Archive I/O`: Number of BRU blocks (2Kb) read from the archive device. This should be zero if you are only writing to a tape (with no AUTOSCAN or inspection).

`Files written`: Number and type of files written to the archive. This should be zero if you are just reading a tape.

`Files read`: Number and type of files read from the archive device. This will be zero if you are just writing to a tape.

`Files in headers`: Number of small files stored in header.

`Write errors`: Number of errors counted when writing to the archive device. A ”soft” error indicates a problem detected by BRU. A “hard” error is a hardware error reported by UNIX that caused BRU to issue an error message.

`Read errors`: Number of errors counted when reading from the archive device.

`Checksum errors`: Total number of checksum errors detected when BRU was reading the archive (in AUTOSCAN, extract or inspection modes).

`Min Compression`: Minimum percent of compression for a file

`Avg Compression`: The average compression of all files in the current archive.

`Max Compression`: Maximum compression percentage.

`Max Compressed Size`: largest file compressed.

`Compression Overflows`: Number of files that were too large for the compression buffer (ZBUFSIZE).

`Difference count`: Total number of files that were different when the -d option was used.

Appendix G - The Shell Scripts: `fullbru` ***and*** `incbru`

Two commands, `fullbru` and `incbru`, allow you to do full and incremental backups. These commands are shell scripts that can be run from the command line, or from `cron` for unattended backups (see Chapter 8, “Advanced Uses,” “Running BRU from `cron`”). The standard `fullbru` command will back up all the files on your system. `incbru` will back up all the files that have changed since the last time `fullbru` was run. `fullbru` actually calls the `incbru` command with an argument of 0 (zero). The `incbru` script does all the work. The `incbru` script will perform incremental backups at levels from 0 through 9. The backup level is determined by an optional argument (a digit from 0 through 9) included on the `incbru` command line. If no argument is given, a level 1 backup will be performed. A level 1 backup will contain all files modified since the last full backup. A level 2 backup will contain all files modified since the last level 1 backup, and so on.

Here’s how to do an incremental backup. Either command will back up all the files that have changed since the last full backup was run:

Here’s how to do an incremental backup. Either command will back up all the files that have changed since the last full backup was run:

```
# incbru
```

or

```
# incbru 1
```

The following is a copy of the `incbru` script. It was written to be as simple as possible and may not work correctly on your system. Feel free to modify it to suit your needs.

```
# This script will perform different levels of
# incremental backups using bru.
#
# It will back up all files that have changed
# since doing a lower level backup.
#
# If no argument is specified, the script will do a
# level 1 backup and will back up all files since
# the last full (level 0) backup. If the
# level 0 backup does not exist, it will do
# a level 0 backup instead.
#
# Feel free to customize this script to handle
# your specific backup requirements. This script
# is designed to do basic backups of the entire
# system from the root directory.
#
BRUDEV=””  # name of backup device. use default
#           if none is specified
BRUOPTS=”-vvvv” # bru options
DIRS=”./”  # directories or filesystems to
```

```
#           back up
LOGFILE=”” # log file
MAILTO=root # where to mail the results
case “$#” in
  0)
    LVL=1  # no argument, so do a level 1 backup
    ;;
  1)
    LVL=$1
    ;;
  *)
    echo “usage: incbru [level]”
    exit 1
    ;;
esac
#
# check for legal backup level
#
case $LVL in
  0|1|2|3|4|5|6|7|8|9)
    ;;
  *)
    echo “incbru: $LVL is an illegal backup level”
    exit 1
    ;;
esac
#
# if level 0, do a full backup
#
DATEFILE=””
if [ “$LVL” -eq 0 ]
then
  DATEFILE=””
else
  PREVLVL=`expr $LVL - 1`  # previous backup level
  echo “PREVLVL = $PREVLVL”
  #
  # find date marker file for previous level backup
  #
  i=0
  while [ “$i” -le “$PREVLVL” ]
  do
    FILE=/etc/BRULEVEL$i
    if [ -f “$FILE” ]
    then
      DATEFILE=$FILE
      if [ “/etc/BRULEVEL${PREVLVL}” = “$FILE” ]
      then
        break # found previous date marker file
      fi
    fi
    i=`expr $i 1`
  done
fi
if [ "$DATEFILE" = "" ]
then
  if [ "$LVL" -ne 0 ]
  then
    echo "incbru: No previous backup less than Level $LVL"
    LVL=0
  fi
  echo "Performing Full Backup (Level 0)"
else
  echo "Performing Level $LVL Backup"
  BRUOPTS="$BRUOPTS -n $DATEFILE"
fi
#
# set up bru command that will do the backup.
# use default device from brutab if none is
# specified
#
if [ "$BRUDEV" != "" ]
then
  BRUCMD="bru -c $BRUOPTS -f $BRUDEV $DIRS"
else
  BRUCMD="bru -c $BRUOPTS $DIRS"  # use default
fi      # device
#
# create a temporary date marker file. if the
# backup is successful, this will be renamed
#
MARKER=/etc/BRULEVEL${LVL}
TMP=$MARKER.err
TS=`date`
echo "$TS - Started BRU Level $LVL Backup\n\tcommand = '$BRUCMD'" >
$TMP
#
# change to the root directory and execute the
# bru backup. write output to log file if one
# is specified
#
BRUEXIT=2
cd /
if [ "$LOGFILE" != "" ]
then
  $BRUCMD > $LOGFILE 2>&1
  BRUEXIT=$?
else
  $BRUCMD 2>&1
  BRUEXIT=$?
```

```
fi
#
# check the exit code. if greater than or equal
# to 2, we’ve got a problem
#
TS=`date`
if [ “$BRUEXIT” -ge 2 ]
then
  echo “incbru: ERROR in BRU Backup”
  echo “$TS - ERROR IN BRU BACKUP, check log file: \
    ‘$LOGFILE’” >> $TMP
  mail $MAILTO < $TMP
  exit 2
else
  echo “BRU Level $LVL Backup successful”
  mv $TMP $MARKER
  echo “$TS - BRU Level $LVL Backup successful, \
    results in log   file: ‘$LOGFILE’” | mail $MAILTO
  exit 0
fi
```

Appendix H The* `brutalk` *Program

The following is a C language source code listing of the `brutalk` program. `brutalk` is a simple program that talks to a pair of fifos associated with a run of BRU. It uses the -I option and may be run under the control of the `cron` program. NOTE: A compiled version of this program is included in your `/bru` directory.

```
/*

brutalk.c simple program to talk to a pair of fifos for bru
DESCRIPTION
Simple-minded program to talk to a pair of fifos. Use as:
brutalk [-t tty] /dev/bru.r
or
brutalk [-t tty] /dev/bru.q /dev/bru.r
Normally brutalk will attempt to communicate with the user via /dev/
tty. The -t option can be used to select any tty type stream to open
in place of /dev/tty. We fork to create two processes. The child reads
from the fifo where queries are posted and writes to the terminal. The
parent reads the replies from the terminal and writes to the fifo
where replies are expected. There are basically three normal ways to
exit:
Before attempting to open explicitly named input and output fifos, we
set a timer. If the timer goes off because bru has not yet opened the
other end of the fifos, then we get a SIGALRM and exit. This basically
means that either bru has exited normally, or has not yet requested
any interaction.
Note that the timeout is only effective until bru opens the fifos for
the first time. If you answer one query using brutalk, then interrupt
brutalk, and come back and run it again later, it will wait until the
next query is available. The initial timeout can be avoided by using
the redirection form, since it is actually the shell that is opening
the fifos in that case. The child dies when the program posting
queries exits, closing the write side of the query fifo. The parent
gets notified that the child has died by a SIGCLD and itself exits.
If the user types a ^D (EOF) then the parent sends the child a kill
signal and exits. The original client at the other end of the fifos
continues to execute. If it knows how to deal gracefully with the fact
that we have exited (by closing and reopening the fifos), then we can
reattach at a later time to answer more queries.
This program is normally used to interact with bru when bru is
executed from cron by a command that includes the arguments:

    -I q,/dev/bru.q (send queries to fifo bru.q)
    -I r,/dev/bru.r (read replies from fifo bru.r)
    -I l,/tmp/brulog (send verbosity info here)
The main function returns 1 to satisfy lint and also to cover the
“impossible case” where exit returns rather than exiting.
AUTHOR
Fred Fish
```

```
NOTE: This code sample is intended as an example ONLY.  It will NOT
compile as it exists.  The finished version of brutalk is included
with your BRU distribution and can be found in the /bru directory.
*/

#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <sys/types.h>
#ifndef SIGCLD
#define SIGCLD SIGCHLD /* If no SIGCLD or SIGCHLD, we croak */
#endif
#if HAVE_PID_T
  typedef pid_t PID_T;
#else
  typedef int PID_T;
#endif
extern void perror ();
extern void exit ();
extern unsigned alarm ();
static void openfiles ();
static void passdata ();
static int rfifo;
static int wfifo;
static int ttyf;
int main (argc, argv)
int argc;
char *argv[];
{
  int optchar;
  char *infifo = NULL;
  char *outfifo = NULL;
  char *tty = NULL;
  extern int optind;
  extern char *optarg;
  while ((optchar = getopt (argc, argv, “t:”)) != EOF) {
    switch (optchar) {
      case ‘t’:
        tty = optarg;
        break;
      case ‘?’:
        exit (1);
        break;
    }
  }
  for ( ; optind < argc; optind) {
    if (infifo == NULL) {
      infifo = argv[optind];
    } else if (outfifo == NULL) {
      outfifo = argv[optind];
    } else if (tty == NULL) {
      tty = argv[optind];
    } else {
      fprintf (stderr, “brutalk: too many arguments\n”);
      fprintf (stderr,
        “usage: brutalk readfifo writefifo\n”);
      exit (1);
    }
  }
  if (tty == NULL) {
    tty = “/dev/tty”;
  }

  (void) signal (SIGALRM, exit);
  (void) alarm (5);
  openfiles (infifo, outfifo, tty);
  (void) alarm (0);
  passdata ();
  exit (0);
  return (1);
}

/*
Open the read and write fifos, and the terminal stream. Any failure is
fatal with an error message printed.
*/

static void openfiles (rname, wname, ttyname)
char *rname;
char *wname;
char *ttyname;
{
  if (rname == NULL) {
    rfifo = 0;
  } else if ((rfifo = open (rname, O_RDONLY)) == -1) {
    fprintf (stderr, “brutalk: can’t open ‘%s’”, rname);
    perror (“”);
    exit (1);
  }
  if (wname == NULL) {
    wfifo = 1;
  } else if ((wfifo = open (wname, O_WRONLY)) == -1) {
    fprintf (stderr, “brutalk: can’t open ‘%s’”, wname);
    perror (“”);
    exit (1);
  }
  if ((ttyf = open (ttyname, O_RDWR)) == -1) {
    fprintf (stderr, “brutalk: can’t open ‘%s’”, ttyname);
    perror (“”);
    exit (1);
  }
}

/*
Fork to create bidirectional data paths and then loop, passing data
through each path until a termination condition occurs.
*/

static void passdata ()
{
  int inbytes;
  int outbytes;
  int pid;
  char inbuf[1024];
  char outbuf[1024];
  if ((pid = fork ()) == -1) {
    perror ("brutalk: can't fork");
    exit (1);
  } else if (pid == 0) {
    while ((inbytes = read (rfifo, inbuf, sizeof(inbuf)))
      > 0) {
      (void) write (ttyf, inbuf, (unsigned) inbytes);
    }
  } else {
    (void) signal (SIGCLD, exit);
    while ((outbytes = read (ttyf, outbuf,
      sizeof (outbuf))) > 0) {
      (void) write (wfifo, outbuf, (unsigned) outbytes);
    }
    (void) kill ((PID_T) pid, SIGTERM);
  }
}
```

Appendix I The `bruxpat` ***File***

```
# This file is used by the -X option to provide
# an inclusion/exclusion list. For each
# pathname of a file selected for backup,
# each line of this file is examined for a
# pattern and that pattern is applied to the
# pathname. If the pattern matches, the
# appropriate action is taken (the pathname
# is accepted or rejected). If the pathname
# makes it through all the patterns, it is
# accepted.
#
# Each command line in the bruxpat file (the
# file you are now reading) consists of a
# control field and a pattern. The
# pattern is separated from the control
# field by whitespace. Control field
# characters are:
#
# i Include this pathname if pattern
#  matches. The pathname is accepted and
#  no further patterns are applied.
#
#       *** NOTE ***
# bru stops trying on the first pattern
#  match found and passes the filename.
#  Since it scans patterns in the order
#  listed, include patterns should
#  usually be listed before any exclude
#  patterns.
#
# x Exclude this pathname if pattern
#  matches. The pathname is rejected
# and no further patterns are applied.
#
# s The pattern is a shell-style wildcard
#  pattern except that '/' characters are
#  not treated as special characters.
#
# r The pattern is a regular expression
#  (the same as used by the "grep" command).
#
# l The pattern is a literal string.
#
#
# Include the C runtime startup file but
# exclude all other object files.
```

```
#
is */crt0.o
xs *.o
#
# Exclude all core files
#
xs */core
xs core
#
# Exclude all files and subdirectories in
# the temporary directories.
# Handle files specified with relative and
# absolute pathnames.
#
xs ./usr/tmp/*
xs /usr/tmp/*
xs ./tmp/*
xs /tmp/*
#
# the bruxpat file also tracks files that should
# be excluded from compression attempts (-Z).
# Files and patterns listed here are not
# compressed.
#
zs *.[zZ]
zs *.gz
zs *.arj
zs *.bz
zs *.bz2
zs *.mp3
zs *.gif
zs *.zip
```

Appendix J - Advanced* BRUTAB *Settings

This section of the user's guide explains the advanced settings of the brutab file. These settings are either numeric or boolean entries. **NOTE**: These are advanced settigs and should only be changed if you are completely competent in your devices specific responces. We do suggest that these settings remain un-changed in your BRUTAB file.

Name	Type	Meaning
`Advance`	`Boolean`	reads/writes advance media even when errors occur (most 9-track tape drives, few cartridge drives) **WARNING**: do not set if not true!
`ederr`	`numeric`	`errno` for end of device reached **NOTE**: "`ederr`" should be different than other **`err's`
`eject`	`boolean`	eject media after use (Macintosh style)
`format`	`boolean`	format media if necessary
`frerr`	`numeric`	`errno` for read of unformatted media
`fwerr`	`numeric`	`errno` for write of unformatted media
`maxbufsize`	`numeric`	maximum I/O buffer size for this device. Is set to 0 if unspecified or no limit.
`noautoscan`	`boolean`	disables automatic scanning of the archive after creation of an archive ****IF YOU SPECIFY THIS BOOLEAN YOU WILL DEFEAT BRU'S AUTOMATIC ERROR CHECKING OF YOUR ARCHIVE.**
`prerr`	`numeric`	`errno` for partial reads
`pwerr`	`numeric`	`errno` for partial writes
`shmall`	`numeric`	limit on total amount of `shm` used **NOTE**: applies only if "`-D`", double buffering, is specified when invoking `bru`.
`shmcopy`	`boolean`	tells `bru` that the archive device driver cannot do I/O directly from shared memory. **NOTE**: applies only if "`-D`", double buffering, is specified when invoking bru.
`shmmax`	`numeric`	limit on size of each shared memory segments **NOTE**: applies only if "`-D`", double buffering, is specified when invoking bru.

`Shmseg`	`numeric`	limit on number of shared memory segments **NOTE**: applies only if "`-D`", double buffering, is specified when invoking `bru`.
`wperr`	`numeric`	`errno` for write protected media
`zrerr`	`numeric`	`errno` for zero length reads
`zwerr`	`numeric`	`errno` for zero length writes.

Appendix K - Using `MOUNTCMD` ***and*** `UNMOUNTCMD`

`MOUNTCMD` & `UNMOUNTCMD` allow the user to specify external commands to handle devices. `MOUNTCMD` and `UNMOUNTCMD` definitions are placed in the `brutab` file or defined as environment variables. `MOUNTCMD=` is used to specify a command that will be called before BRU attempts to open a device for reading or writing. The `UNMOUNTCMD=` should be used to specify a command that will be called after BRU has finished reading/writing.

BRU passes four arguments to the commands specified by `MOUNTCMD` and `UNMOUNTCMD`. These arguments are described below in Table 1. In most cases, the commands specified will be shell scripts. However, BRU does not care what language is used to create these executable files as long as they are set to accept and act on the arguments listed below. Bellow you will find a reprint of the `mountcmd.sh` and `unmountcmd.sh` scripts supplied with BRU. This script like all other scripts supplied with BRU are to be modified by the user to fit their system.

Argument	**M/U**	**Description**
1	Both	Device Name
2	Both	Volume Number
3	Both	BRU Mode
4	M	Media size in Kbytes
4	U	Kbytes written/read

Table 1

Two example shell scripts follow - please understand that these are examples and should be used simply as templates for a functional media changing mechanism - these scripts do absolutely nothing as they appear here:

```
#!/bin/sh
##################################################################
#
#  mountcmd.sh   sample script for MOUNTCMD
#
# If the MOUNTCMD variable is set, this command will be called
# BEFORE BRU attempts to read or write a tape
#
# Please modify this script to suit your own needs
#
#
##################################################################

CMD=$0           # name of this command

DEV=$1           # 1st parameter is device name
VOL=$2           # 2nd parameter is expected volume number
MODE=$3          # 3rd parameter is BRU mode letter
```

```
MSIZE=$4          # 4th parameter is media size in Kbytes


case “$MODE” in
  c ) :
    # Replace the following lines with any commands that
    # should be called before BRU attempts to WRITE to a tape
    #
    echo “$CMD: volume $VOL on device $DEV (mode = $MODE)”
    #
    # Sample command to create a file containing label info
    #
    echo “VOLUME $VOL LABEL” > /tmp/labelfile

    RTN=$?        # set return code
    ;;
  [idtgx] ) :
    #
    # Replace the following lines with any commands that
    # should be called before BRU attempts to READ a tape
    #

    echo “$CMD: volume $VOL on device $DEV (mode = $MODE)”
    RTN=$?        # set return code
    ;;
esac

exit $RTN  # return 0 if successful

#!/bin/sh
################################################################
#
#  unmountcmd.sh  sample script for UNMOUNTCMD
#
# If the UNMOUNTCMD variable is set, this command will be
# called AFTER BRU is done reading/writing a tape.
#
# The UNMOUNTCMD will not be called unless the MOUNTCMD was
# specified. It is NOT possible to call the UNMOUNTCMD only.
# In most cases, the UNMOUNTCMD is not needed, as all the tape
# handling, jukebox commands, etc. can be done by the
# MOUNTCMD.
#
# Please modify this script to suit your own needs.
#
################################################################

CMD=$0            # name of this command

DEV=$1            # 1st parameter is device name
```

```
VOL=$2           # 2nd parameter is volume number
MODE=$3          # 3rd parameter is BRU mode letter
IOSIZE=$4  # 4th parameter is Kbytes written/read


case “$MODE” in
  c ) :
    #
    # Replace the following lines with any commands
    # that should be called after BRU is done WRITING a tape
    #

    echo “$CMD: volume $VOL on device $DEV (mode = $MODE)”
    RTN=$?       # set return code
    ;;
  [idtgx] ) :
    #
    # Replace the following lines with any commands      |
    # that should be called after BRU is done READING a tape
    #

    echo “$CMD: volume $VOL on device $DEV (mode = $MODE)”
    RTN=$?       # set return code
    ;;
esac

exit $RTN  # return 0 if succesful
```

Appendix L - BRU Messages

All BRU messages are associated with a number. The message number consists of a single letter followed by three digits. The first letter of the message number indicates the type of message (i.e.: `[I181]` is an Informational Message). The messages are listed below:

A - alert message

BRU is alerting the user of a problem. This problem or alert, is not as strong as an error or a warning message but none the less BRU will require a response. This is message is usually followed by a "`Q`" message

E - error message

BRU has found that the current operation has errors that it cannot overcome. Depending on the `MAXERRORS` setting in your `BRUTAB`, BRU will terminate with an execution summary exit code of 2.

I - informational message

Information regarding the current process. This type of message is echoed to the `/var/log/bruexeclog` file as an "`L`" message.

L - log message

Entry message to the `/var/log/bruexeclog` regarding the current process. Entry is a copy of what is sent to the screen with the "`I`" message. All entries are made with a date and time stamp.

Q - query message

BRU is waiting for a response to complete the current process.

W - warning message

BRU has found problems with the current operation. The errors where not serious but BRU was unable to resolve the issue. Depending on your setting for `MAXWARNINGS` in your `BRUTAB` file BRU will terminate the current process.

NOTE: You should investigate all WARNING and ERROR messages reported by BRU. These messages can result in you not being able to restore your data.

Many BRU messages are the result of errors returned by the UNIX system. For these types of errors, the UNIX error message is included as part of the BRU message. This portion of the message is described below:

`errno = code`, descriptioncode is the UNIX error number. Description is a brief text message associated with the error number.

The BRU messages are listed on the following pages. Following each message is an explanation and possible suggestions for corrective action.

```
[E001] specify mode (-cdeghitx)
```

The user ran BRU without specifying the mode. The user must specify a valid mode. See Appendix D for a list of BRU modes.

```
[W002] filename: can't open archive: errno = code,
description
```

BRU couldn't open the archive file filename for some reason. The reason is indicated by the UNIX error message. Make sure that the device actually exists.

Also see if another process (or an old BRU process) still has control of the device.

`[W003] filename: warning - close error on archive: errno = code, description`

BRU received an error from the operating system when BRU attempted to close the archive. The reason is indicated by the UNIX error message.

`[W004] warning - archive read error at block blocknum: errno = code, description`

BRU received an unrecoverable error when attempting to read an archive. The reason for the error is indicated by the UNIX error message. Whatever data was available at that location in the archive is unrecoverable.

`[W005] warning - archive write error at block blocknum: errno = code, description`

BRU received an unrecoverable write error while attempting to write an archive. The reason for the error is indicated by the UNIX error message. Whatever data that was to be written at that location in the archive has been discarded. Proper corrective action depends upon the situation and the specific file within which the error occurred. If this error occurred on the first block of an archive, the archive may be write protected, in which case, the `wperr` parameter is probably set incorrectly in the `brutab` file.

Another possibility is that the I/O buffer size is too large for the given device. Experiment with a smaller buffer size. Set the buffer size with the `bufsize` parameter in the `brutab` file or use the -b option to set it on the command line.

`[E006] seek error on archive: errno = code, description`

BRU received an unrecoverable seek error from the operating system on an archive file. The reason for the error is indicated by the UNIX error message. Make sure that your archive device actually has the capability to do random seeks to any location. Most tape drives do NOT have this capability. Check the value of the seek parameter in the `brutab` file. If your device does not have the ability to do random seeks, you must set `seek=0`.

`[E007] media size smaller than I/O buffer size!`

This error was caused when BRU detected a media size that was inconsistent with the I/O buffer size. The media size should be at least as large as the I/O buffer size. It is not possible to write to device if the media size smaller is smaller than the buffer size (i.e. BRU cannot write 1 megabyte of buffer data to a floppy with a media size of 720 kilobytes). Check the `brutab` file and make sure that the `bufsize` parameter is less than size.

`[W008] warning - buffer size bufsize exceeds maximum maxbufsize allowed for device`

Check the `brutab` file and make sure that `bufsize` is less than `maxbufsize`.

`[E009] can’t allocate bufsize archive buffer: errno = code, description`

This message was caused when BRU could not allocate an I/O buffer of the size requested. Try reducing the buffer size to a smaller value. This can be done with the `-b` command line option, or the `bufsize` parameter entry in the `brutab` file.

`[W010] filename: warning - block sequence error`

BRU detected an inconsistency in the ordering of blocks returned by the archive device on a read.

For example, BRU asked for blocks "`11, 12, 13, 14, 15`" and got blocks "`11, 12, 14, 15, 16`." This error may indicate a hardware problem. If this message occurs at the start of the 2nd volume (when reading a multiple volume archive), it probably means that the last few blocks of the previous volume were missing or were unreadable. This can occur if the media size parameter was set too large in the `brutab` file (i.e. `size=150M` for a 60 megabyte tape) during the backup and BRU attempted to write past the end of the previous volume. With some tape drives, it may also occur if the media size was set to zero (unknown). The problem can usually be avoided by setting the media size to a value that is certain to be less than the maximum capacity of the volume (i.e. set `size=149MT` for a 150 megabyte tape).

To detect and avoid this problem, always verify your archive immediately after the backup is performed. AUTOSCAN does this automatically for all devices that have an entry in the `brutab` file (unless `noautoscan` is set or you are using a `norewind` device).

`[W011] warning - file synchronization error; attempting recovery ...`

BRU was expecting to find a file header block while reading an archive, but instead found another type of block. This warning will occur if you started reading an archive at a volume other than the first, or skipped a volume in the middle of reading an archive. This error message can be suppressed with the `-QV` option flag. For more information on the optional flag see Chapter 7. BRU will scan each successive archive block looking for a file header block, and normal processing will resume from the first file header block found.

`[W012] filename: no file: errno = code, description`

The named file does not exist or part of the path name of the named file is not searchable given the current permission settings.

`[E013] filename: can't stat: errno = code, description`

The UNIX stat system call failed. This means that BRU was unable to obtain status information (ownership, access and modification times, link count, etc.) on the file. The reason for the error is indicated by the UNIX error message. Generally, this error is caused because the file is not accessible with the user's current permission settings. On networked systems, it may occur when trying to access remote (NFS-mounted) files. If this is the case, you should check your network permission settings.

`[E014] pathname path too big (1023 max)`

This error was caused when BRU detected a pathname longer than 1023 characters. No known UNIX system allows pathnames longer than 1023 characters, so this message may indicate that the filesystem is corrupted or that something else is seriously wrong.

`[E015] *** OBSOLETE MESSAGE NUMBER ***`

`[E016] filename: can't open: errno = code, description`

BRU could not open filename. The reason is given as part of the UNIX error message. In many cases, this is caused by insufficient file access permissions.

`[W017] filename: warning - file close error: errno = code, description`

BRU received an error when attempting to close the file filename. The reason for the error is given by the UNIX error message.

`[E018] filename: read error: errno = code, description`

BRU received an error while reading a file. The reason is indicated by the UNIX error message. This message means that the file was not backed up properly. It may be an indication of hard disk failure, a corrupted filesystem, a damaged file that is unreadable, or other system problems (like an NFS problem.)

`[W019] filename: warning - file was truncated`

The file filename was truncated while BRU was in the process of reading or writing it. Usually, another program has modified the file that BRU was reading. This can occur if you are attempting to back up database files and the database program is active. If BRU was creating an archive at the time, the archived file is padded with sufficient null characters to bring it back to the size it was originally (the size of the specified file when BRU began to archive it). This is the same size recorded in the file header block. If this message occurs while backing up, the data archived from filename is probably not correct (because the data was changed while BRU was reading it). Even though BRU may restore this file later without any warnings, the data could contain errors. If this message occurs while restoring an archive, it indicates that a problem occurred and that BRU was unable to restore data from the last part of the file.

`[W020] filename: warning - file grew while archiving`

The file filename grew in length while BRU was in the process of reading it. If BRU was creating an archive at the time, the archived file was truncated to it's original size (the size of the file when BRU started to read it). This is the size recorded in the file header block. This warning is commonly seen for log or database files, to which information is constantly being added. It can generally be avoided by backing up the system in single user mode (or by shutting down the database before doing a backup). If the file causing this message is not critical (i.e. log files like `/var/log/bruexeclog`) you may wish to exclude these files from the backup. This can be done by specifying a pattern in the `bruxpat` file and using the `-X` option.

`[W021] filename: warning - can't set user id: Not owner`

BRU attempted to extract filename which was stored with the `suid` bit set and the user running BRU was not the original owner of the file (and did not have superuser privileges).

`[W022] filename: warning - can't set group id: Permission denied`

BRU attempted to extract filename which was stored with the `sgid` bit set and the user running BRU was not the original owner of the file (and did not have superuser privileges).

`[E023] filename: can't exec: errno = code, description`

BRU could not execute the file filename for the reason given as part of the UNIX error message. Generally, this error occurs because filename does not exist, or it was not executable by the user running BRU.

`[E024] can't fork, try again: errno = code, description`

This error was caused when BRU couldn't execute a fork system call. The reason is indicated by the UNIX error message. Generally, this indicates the system is in serious trouble, or the per-user limit on processes has been exceeded.

`[E025] unrecognized wait return statcode`

The wait system call returned a status code of statcode which BRU was not able to understand. If this error occurs it may indicate that there is a problem with your system or that your version of BRU is incompatible with your current version of UNIX. If you have upgraded your operating system, you may also need to upgrade your copy of BRU.

`[E026] child interrupted: errno = code, description`

The child process which BRU was waiting for was interrupted. The reason for the error is indicated by the UNIX error message.

`[E027] filename: fatal error; stopped by signal sigcode`

The child process, filename, which BRU was waiting for was stopped by a UNIX signal. The reason is indicated by `sigcode`.

`[E028] filename: fatal error; terminated by signal sigcode`

The child process, filename, which BRU was waiting for was terminated by a UNIX signal. The reason is indicated by `sigcode`.

`[W029] filename core dumped`

BRU was waiting for a child process, filename, which terminated abnormally and dumped to core.

`[E030] inconsistent wait status wait code`

BRU received an unexpected return code of wait code from a UNIX wait system call. Usually, the child process which was being run has gone berserk in some manner. This error may indicate that there is a problem with your system or that your version of BRU is incompatible with your current version of UNIX. If you have upgraded your operating system, you may also need to upgrade your copy of BRU.

`[E031] can't set uid to userid: errno = code, description`

BRU received an error when trying to run the `setuid` system call. The reason is given by the UNIX error message. If this error occurs, it indicates a possible UNIX system bug or an internal bug in BRU.

`[E032] can't set gid to groupid: errno = code, description`

BRU received an error when trying to run the `setgid` system call. The reason is given by the UNIX error message. If this error occurs, it indicates a possible UNIX system bug or an internal bug in BRU.

`[W033] filename: warning - error count block checksum errors`

While reading an archive, BRU detected `errcount` number of checksum errors in the specified file. This message can occur if the archive was originally written with errors-possibly caused by a buffer size setting that is too large. Try setting the buffer size to a smaller value when creating the archive (i.e. set `bufsize=10K` in the `brutab` file or use the `-b` 10K option on the command line). When reading an archive, BRU normally attempts to read the archive with the write buffer size (it obtains the write buffer size from the archive header block stored at the start of the archive). The proper buffer size varies with the system and type of tape drive. In some cases, a tape written on one system (with a large buffer size like 64K) cannot be read properly on another system (which can only handle a small buffer size like 10K). If this is the case, you may be able to force BRU to read the tape by forcing the buffer size to smaller value (i.e. specify `-b 10K` as one of the command line options). Checksum errors may also be caused by hardware or tape problems. Try cleaning the heads on your tape drive. Try to retention the tape. Also, make sure that your tape cartridges are in good shape-tapes do not last forever. They should be rotated frequently and replaced on a regular basis. Make sure that you are using the proper kind of tape with your tape drive. Many tape cartridges look the same (especially 1/4" tapes), but have different densities. For instance, if you have a 150 MB tape drive, you should use 150 MB (DC6150) or 250 MB (DC6250) tape cartridges. You will get errors if you try to write to a 60 MB (DC600A) tape.

`[E034] internal bug in routine routinename`

BRU detected some sort of internal bug in the routine `routinename`. If this error occurs, it may possibly be a bug in BRU, or a hardware or kernel software problem. If it is not repeatable, it is likely to be a hardware or kernel bug. This message should be reported to TOLIS Group Tech Support.

`[E035] can't allocate byte_count more bytes: errno = code, description`

BRU ran out of memory for some reason. This error generally occurs when BRU tries to create a tree from a list of filenames read from the standard input, or when memory is very limited on the system due to hardware or CPU constraints.

`[E036] internal error in tree; pathname overflow`

`[E037] *** OBSOLETE MESSAGE NUMBER ***`

While building a file tree, BRU created a path which exceeded 1023 characters in length. No known UNIX system allows pathnames longer than 1023 characters, so this message may indicate that the filesystem is corrupted or that something else is seriously wrong.

`[E038] filename: seek error: errno = code, description`

BRU received an error when attempting to seek to a certain location in the file that it was reading or writing. The name of the file is indicated by filename. The reason for the error is given by the UNIX error message. This error is rare and usually indicates a hardware problem with the disk drive.

`[W039] warning - info block checksum error`

BRU detected a checksum error while reading the first block of an archive. The info block (archive header block) contains information about the archive which is of use to BRU, but not critical to reading or extracting files from the archive. Make sure that the archive you are attempting to read was actually written by BRU. This error often occurs when attempting to read a tape written by another program (like `tar` or `cpio`). It may also occur if you try to read a tape that is blank. If this is the ONLY warning or error message (and BRU appeared to work normally), it can usually be ignored. In this particular case, it means that BRU had trouble reading the first block of the archive, but was able to skip past the first block and read the rest of the archive normally.

`[E040] filename: write error: errno = code, description`

BRU received an error when attempting to write to the file filename. The reason for the error is shown by the UNIX error message. If this error occurs, it usually indicates a hardware problem with your hard disk. It could also indicate that the filesystem containing filename is out of space, or that the filesystem is write-protected (it may be mounted as “read-only” or you may not have write permission).

`[W041] filename: warning - error setting mode: errno = code, description`

BRU received an error when attempting to set the mode of filename. The error occurred when BRU was executing the system call `chmod`. The reason is indicated by the UNIX error message.

`[W042] filename: warning - error setting owner/group: errno = code, description`

BRU received an error from the operating system when attempting to set the owner id or group id of filename. The error occurred when BRU was executing the system call `chown`. The reason is indicated by the UNIX error message. On systems which support symbolic links, this error can occur when BRU attempts to set the owner/group id of a symbolic link which points to a file which does not exist. This can occur if the symbolic link filename is restored, but the file the symbolic link points to is not restored.

`[W043] filename: warning - error setting times: errno = code, description`

BRU received an error from the operating system when attempting to set the access and modification times of filename. The error occurred when BRU was executing the system call `utime`. The reason is indicated by the UNIX error message.

`[E044] filename: error making node: errno = code, description`

BRU received an error when attempting to create a special file system node, such as a FIFO, block special file, or character special file. The error occurred when BRU was executing the system call `mknod`. The reason is indicated by the UNIX error message. This message may occur when trying to restore special files and you do not have superuser privileges. On some systems, only the root user has the ability to create special files.

`[E045] filename1: can’t link to filename2: errno = code, description`

BRU received an error when attempting to make a hard link between filename1 and `filename2`. The reason is indicated by the UNIX error message. This message occurs when `filename2` already exists and cannot be overwritten by a link.

`[E046] internal error; inconsistent phys blk addrs`

`[W047] warning - missing archive header block; starting at volume volnum`

BRU couldn’t find an archive header block at volume number `volnum`. This warning is normal when BRU is asked to start reading an archive at some other volume than the first volume. For example, you will see this message if you immediately try to restore files from the 3rd tape of an archive (without reading through the 1st and 2nd tapes). If you want to start from a different tape other then the first tape. You must use the `-QV` option, this will suppress the error message.

`[W048] filename: warning - lost linkage: errno = code, description`

BRU could not preserve the linkage of two files. The reason is indicated by the UNIX error message. Generally, this error is seen when BRU ran out of memory when it attempted to allocate memory internally to maintain the linkage information of the specified file. In this case, the file would be archived as two separate, distinct files in the archive. Only the linkage information would be lost.

`[W049] filename: warning - linknum unresolved link(s)`

While archiving the file filename, BRU detected `linknum` number of unresolved links to filename. This error is generated when there is still another pathname which points to filename which does not appear in the archive. Usually, this message occurs when BRU is asked to archive a set of directories that contain files that have hard links to files located in other directories (that are not archived by BRU). This message can be disabled by specifying the `-l` option on the command line.

`[W050] ttyname: warning - can't open for interaction: errno = code, description`

BRU could not open the `tty` stream `ttyname` to interact with the user. The reason for the error is given by UNIX error message. This message may occur when attempting to run BRU in the background and the `-B` option (background mode) has not been specified. When run in the foreground, BRU attempts to use the `/dev/tty` device to communicate with the user. In background mode, `/dev/tty` is not available. In this case, the interaction pathnames can be specified with the `-Iq,queryfile` and `-Ir,replyfile` options on the BRU command line (this is normally used when running BRU with the `brutalk` program).

`[E051] date conversion error: date`

The string specified by date is not in the proper format or is not a legal date and time.

`[W052] warning - uname failed: errno = code, description`

BRU received an error when attempting to execute the `uname` system call. The reason is indicated by the UNIX error message.

`[W053] warning - label string too big`

BRU has a string length limit of 63 characters for a user specified label (used with the `-L` option). You must shorten the length of your label string.

`[E054] error - invalid uid/filename as -o argument: pattern`

BRU could not convert a given symbolic user name to the internal numeric form. This error usually occurs when the `-o` option is used and BRU cannot find username in the `/etc/passwd` file.

`[E055] error - illegal wildcard pattern: pattern, errmsg`

The wildcard matching pattern specified by pattern is not legal. The reason is indicated by `errmsg`.

`[E056] filename: can’t overwrite: errno = code, description`

The file filename could not be overwritten during extraction. The reason is indicated by the UNIX error message. In most cases, this message is due to a permissions problem.

`[W057] filename: can’t access for write: errno = code, description`

The file filename could not be accessed for write. The reason is indicated by the UNIX error message. In most cases, this message is due to a permissions problem.

`[W058] filename: can’t access for read: errno = code, description`

The file filename could not be accessed for read. The reason is indicated by the UNIX error message. In most cases, this message is due to a permissions problem.

`[W059] filename: warning - will not be contiguous: errno = code, description`

BRU was unable to create the file filename as a contiguous file. The reason is indicated by the UNIX error message. This message should only occur on systems that support contiguous files (like `Masscomp`).

`[W060] filename: warning - contiguous files not supported, extracted as a regular file`

The file filename cannot be restored as a contiguous file, so BRU will create a regular UNIX file instead.

`[E061] can’t read both file list and archive from stdin!`

BRU was instructed to read both an archive and a list of files from the standard input stream. This error occurs when an illegal BRU command like the following is entered: `bru -x -f - -`

`[W062] warning - premature end of volume volnum`

When reading/writing an archive device, BRU encountered an end-of-file (or got an I/O error) before reaching the expected end of the archive. This message is often preceded by messages `[W004]` or `[W005]`. In this case, it may indicate a problem with the tape drive hardware, old or damaged tapes, or incompatible tape formats (i.e. trying to write to a 60MB tape cartridge on a 150MB tape drive).

`[W063] warning - media appears to be unformatted: errno = code, description`

When BRU first attempted to read/write to a device it received an error. The reason is indicated by the UNIX error message. If BRU receives an error on the first read or write to an archive device, and the error conditions match the values set in the `brutab` entry for unformatted media in this device, BRU will issue this warning message. When writing, if the format and `fmtcmd=` parameters are set for the device, this warning will be suppressed and BRU will attempt to format the media.

`[O064] *** OBSOLETE MESSAGE NUMBER ***`

`[W065] warning - using internal default device table`

BRU could not find the `brutab` file specified by the BRUTAB environment variable, or the default `brutab` file located in `/etc/brutab`. In this case, BRU used its internal `brutab`, which may not be correct for the current archive device.

`[I066] filename: not restored`

This is an informational message. BRU did not restore the file filename because the current file (on the disk) has a modification time that is newer than the file read from the archive. This is BRU's default method of restoring files. If you wish to overwrite all files, regardless of date, you should add the `-ua` option to the BRU command line.

`[W067] warning - media appears to be write protected or wrong density`

BRU received an error on its first attempt to write to an archive device. BRU has determined that it might be caused by media that is write-protected. The UNIX system may not return the proper error code, so it is not always possible for BRU to determine if the media is actually write-protected. BRU tries to determine the write-protect status by comparing the `errno` code returned by UNIX with the value of `wperr` (as specified for the device in the `brutab` file). If `wperr=0` (or is not set) then BRU must "guess" at whether the device is truly write-protected. In this case, it assumes that an error on the first write attempt is caused by write-protection, and issues the above message.

`[W068] filename: warning - not found or not selected`

The user specified a file on the command line which BRU did not find. The file filename may not exist or may be spelled incorrectly. If you are attempting to extract (restore) a file, make sure that filename EXACTLY MATCHES with the desired filename on the archive, including any beginning slashes. For example:

```
    /myfile DOES NOT MATCH./myfile.
```

`[W069] warning - may have to use -F option to read archive`

BRU encountered an archive which does not appear to have checksum. The archive may have been written with `-F` option (which is not recommended) and must be read with the same option. This message sometimes occurs when BRU attempts to read an archive that was written by another program, like tar or cpio. It can also occur when BRU has trouble reading a BRU archive due to bad tapes, dirty tape heads, hardware problems, incompatible tape formats, etc.

`[E070] interaction needed, aborted by -B option`

BRU was run with the `-B` option, indicating that it is running in background mode and that no user interaction is possible. It encountered a condition that required user interaction (like loading a new tape) and terminated. The `-B` option is normally set automatically when BRU is started in the background, so this message may occur even if `-B` was not explicitly specified.

`[E071] filename: error making directory: errno = code, description`

BRU received an error when attempting to create a directory. The reason is indicated by the UNIX error message. In most cases, this occurs when the user has insufficient permissions.

`[E072] filename: error reading symbolic link: errno = code, description`

BRU could not read a symbolic link for some reason. The reason is indicated by the UNIX error message.

`[E073] filename: symbolic links not supported`

While running on a system that does not support symbolic links, BRU encountered a symbolic link while comparing an archive in differences mode (`-d` option).

`[E074] filename: could not make symbolic link: errno = code, description`

While extracting the symbolic link filename, BRU was unable to create a symbolic link. The reason is indicated by the UNIX error message. This error will occur if your version of UNIX does not support symbolic links.

`[E075] filename: could not make fifo`

BRU tried to extract a FIFO (named pipe file) on a system which does not support FIFOs. Normally, BRU tries to create a regular file with the same name. In this case, the attempt to create a regular file was unsuccessful.

`[W076] warning - link of filename to dirname, dirname is a directory, no link made`

BRU was asked to create a symbolic link from filename to the directory `dirname`, on a system which does not support symbolic links. Since hard links to directories are not allowed by UNIX, this warning is issued and no link is made.

`[W077] warning - link of filename1 to filename2, filename2 does not exist`

BRU attempted to create a hard link from `filename1 to filename2 and filename2` does not exist. Generally, this message occurs when BRU is asked to do a partial restore and `filename2` is not present.

`[W078] warning - extracted fifo filename as a regular file`

BRU was asked to extract a FIFO named filename on a system which does not support FIFOs. It extracted filename as a regular file. The correctness or desirability of this behavior is subject to debate, which is why the warning is issued.

`[W079] filename: warning - linkcount additional link(s) added while archiving`

While BRU was archiving a file, there were `linkcount` additional links made to it. These additional links may or may not have been archived.

`[W080] *** OBSOLETE MESSAGE NUMBER ***`

`[E081] no default device in brutab file, use -f option`

BRU could not find a default device in the `brutab` file. The default device is always the first device entry in the `brutab` file.

`[E082] *** OBSOLETE MESSAGE NUMBER ***`

`[W083] warning - attempt to change buffer size from oldbufsize to newbufsize ignored (incompatible brutab entries)`

BRU detected different default buffer sizes when reading or writing to multiple devices (device cycling). The buffer size is not allowed to change between volumes of an archive. This error usually occurs at the start of the second device, when BRU reads the `bufsize` parameter for that device (from the `brutab` file) and discovers that the buffer size differs from the size used by the first device. To avoid this warning message, use the `-b` option to force a specific buffer size for all devices.

`[E084] double buffering I/O error, bytecount bytes read/ written: errno = code, description`

`[E085] problem setting up double buffering, using normal buffering`

Both of these errors indicate that BRU encountered a problem setting up the double buffering. Sometimes, reducing the I/O buffer size will remedy the problem.

`[E086] filename: media ejection failed: errno = code, description`

On systems which support ejection of archive media under software control, BRU may be configured to eject each media when it is done with the media. BRU encountered some sort of error while attempting to eject the media.

`[I087] filename: compressed version was larger, stored uncompressed`

When file compression is utilized via the `-Z` option, BRU will check to ensure that the compressed version of the file uses fewer archive blocks than the uncompressed version. If the compressed version will not result in any savings in archive space (it is larger than the normal file), then the uncompressed version will be archived instead.

`[E088] filename: decompression failed (errmsg)`

BRU received an error from when attempting to decompress a file. The reason is indicated by `errmsg`. The file filename has not been extracted properly and may contain errors.

`[W089] warning - estimate mode ignores compression`

BRU was told to use both the `-e` and `-Z` options simultaneously. Because of the large overhead in compressing files, and because there is no way to determine the compression ratio without actually doing the compression, BRU cannot estimate how much archive space is required for an archive when compression is enabled. Therefore, the `-e` option ignores possible savings due to compression.

`[W090] filename: warning - not deleted: errno = code, description`

BRU received some sort of error while attempting to delete (unlink) filename.

`[W091] filename: warning - compression failed, stored uncompressed`

BRU received an error from UNIX when BRU attempted to compress filename for storage (such as a filesystem temporary space overflow). BRU could not generate the compressed version of the file. Thus, the file was stored uncompressed.

`[E092] *** OBSOLETE MESSAGE NUMBER ***`

`[W093] warning - buffer size bufsize exceeds system imposed limit buflimit with double buffering`

While attempting to set up double buffering using System V Style shared memory support, BRU was asked to use an I/O buffer size which resulted in the double buffering buffers exceeding the system imposed shared memory limits. Try setting `shmmax` to a smaller value.

`[W094] warning - buffer size automatically adjusted to bufsize`

While attempting to set up double buffering using System V Style shared memory support, BRU was asked to use an I/O buffer size which resulted in the double buffering buffers exceeding the system imposed shared memory limits. The I/O buffer size was automatically adjusted downwards to the maximum size which the system could support.

`[E095] could not get shared memory segment: kilobytes: errno = code, description`

BRU was attempted a system call to `shmget` and was unable to get the requested amount of shared memory. The reason is indicated by the UNIX error message.

`[E096] could not attach shared memory segment: errno = code, description`

BRU made a system call to `shmget` which failed after BRU had already made a successful `shmget` call. The reason for the error is indicated by UNIX error message.

`[E097] could not allocate message queue: errno = code, description`

BRU could not allocate the memory needed to perform double buffering (`-D` option). The reason is indicated by the UNIX error message. Your system may not support shared memory, or the shared memory parameters (`shmmax`, `shmseg`, `shmall`) may not be set correctly in the `brutab` file.

`[E098] warning - don’t understand -I option badargs`

The string `badargs` was not recognized as a valid argument for the interaction option `-I`.

`[W099] warning - need more than segments shared memory segments`

BRU was not able to allocate enough shared memory segments. Try setting the `shmseg` parameter for the device in use to a lower value.

`[W100] warning - failed to move break value by {number of bytes} bytes: errno = code, description`

BRU made a system call to `sbrk` which failed. BRU was unable to adjust the break value. This message is only caused when BRU was previously able to adjust the break value to the desired place; or should be able to adjust the break value, such as when reducing the amount of memory used.

`[W101] warning - compression initialization failed, -Z suppressed`

BRU could not acquire sufficient memory to perform the requested file compression. Compression was not performed. Try reducing the number of bits in compression by using the `-N` option with a lower value.

`[W102] warning - unknown child died, pid pidnumX(expected pidnum), status statcode`

While waiting for a specific child process to exit, the wait system call returned to BRU the `pid` of another process, `pidnumX`, which exited with the status of `statcode`. This error should never occur. If this error occurs, it is usually indicative of a serious problem with the system.

`[E103] double buffer child died, status statcode`

The child process used by BRU for double buffering died unexpectedly. The reason is indicated by the UNIX status code `statcode`.

`[E104] warning - double buffer child error errcode`

The child process used by BRU for double buffering received some sort of fatal error, which the child process was able to recognize as unrecoverable. The reason for the error is indicated by `errcode`.

`[W105] warning - no double buffer child to reap: errno = code, description`

BRU was waiting for a double buffer child to exit and the wait system call failed for some reason which was unexpected in the parent process.

`[W106] warning - archive device may need “shmcopy” flag set in brutab entry`

On some systems, the device driver for a given archive device may not be able to do I/O directly to or from shared memory. BRU detects this condition when the first write to, or the first read from, a given device fails with UNIX error code `errno` set to `EFAULT`. BRU issues this warning message and automatically attempts to switch to a mode where the data is copied to or from a local buffer. This automatic switching generally succeeds on writes and fails on reads, which is why the suggested fix is printed as a warning message.

`[E107] filename: error - unrecoverable archive write error, some data lost: errno = code, description`

BRU received an unrecoverable write error while creating an archive, and all or part of the data was lost for filename. This message may be an indication of tape hardware problems, dirty tape heads, an improper BRU buffer size, tapes that need retensioning, or tapes that are simply worn out. On high-density tape drives, this message can occur when an attempt is made to write to a low-density tape. Usually this happens with 1/4” tape cartridges, which all look similar. For example, this error will occur when using a 150MB tape drive to write to a DC600A (60MB) or DC300 (30MB) tape cartridge. High-density tape drives can normally read low-density cartridges, but they cannot write to them.

`[W108] warning - media appears to be unformatted or write protected: errno = code, description`

This is a general warning which may appear on the first attempt to read or write an archive volume which is unformatted, or when an attempt is made by BRU to write to an archive which is write- protected. The reason for the warning is indicated by the UNIX error message. This warning may also occur if the backup device does not respond properly when BRU attempts to open the device for writing. BRU is “faked out” and thinks that the device is write-protected. This often occurs with on the first attempt to write to a SCSI device. Try repeating the command. If BRU works successfully, this message can be safely ignored.

```
[W109] warning - assuming end of volume volnum (unknown size)
```

BRU encountered an unrecoverable read or write error before reaching the end of an archive while reading or writing a volume of unknown size. BRU may have actually reached the end of the volume, or BRU may have simply reached a bad spot on the media, which BRU cannot proceed past. Because BRU does not know the media size, BRU has no way of knowing the difference, hence, the warning message. If no other warnings, or errors, occur, this warning is benign.

```
[W110] warning - found volume volnumX, expecting volnum
```

BRU was expecting to find volume `volnum` and it encountered a different volume. Remove the volume and replace it with the correct volume.

```
[O111] *** OBSOLETE MESSAGE NUMBER ***
```

```
[W112] warning - volume not part of archive created archivedate
```

BRU received the correct volume number, but the date of the volume differs from the current archive. Generally, this warning occurs when the wrong tape is inserted while attempting to extract an archive.

```
[A113] alert - all data currently on devicename will be destroyed
```

When the `brutab` entry for a device includes the `qfwrite` boolean value, this message will be issued on the first write to the first volume placed in that device, and BRU will wait for confirmation to continue. In devices which might share both mounted and unmounted media, this prevents inadvertently overwriting media which may have been left in the device by mistake.

```
[I114] *** OBSOLETE MESSAGE NUMBER ***
```

```
[A115] *** OBSOLETE MESSAGE NUMBER ***
```

```
[I116] *** OBSOLETE MESSAGE NUMBER ***
```

```
[I117] don’t know how to rewind archive device
```

BRU doesn’t know how to rewind the present archive device.

```
[A118] rerun with “-b bufsizek” argument
```

Re-run your BRU command with the specified buffer size

```
[Q119] action filename: please confirm [y/n/g]
```

BRU is waiting for confirmation of the given action. The `-w` flag was specified on the command line.

```
[Q120] query options [default: Option] >>
```

General message used to prompt user with various messages.

```
[A121] load volume volnum - press ENTER to continue on device device name
```

```
[W122] filename: warning - too large under current ulimit, not extracted
```

The size of filename exceeds the current `ulimit`. Set `ulimit` to a larger value and try again.

`[E123] ulimit call failed to set maximum file size limit to blkcount blocks`

BRU was unable to set the `ulimit` to a larger value.

`[W124] warning - no double buffering support included in this version`

Your version of BRU does not support double buffering. BRU reverts to normally buffered I/O.

`[W125] warning - shared memory does not appear to be working in your kernel`

Verify the settings in you kernel. Contact your UNIX provider for help.

`[E126] problem sending message to other process`

While in double-buffer mode, BRU could not communicate with the child process.

`[E127] problem receiving message from other process: errno = code, description`

While in double-buffer mode, BRU could not communicate with the child process.

`[W128] filename: warning - file contents changed while archiving`

BRU found that the file it was backing up had been changed. This can happen during live system backups. BRU does not lock files when reading. If a file is modified while BRU is reading it, this message will occur.

`[W129] *** OBSOLETE MESSSAGE NUMBER ***`

`[W130] warning - I/O error on first block`

BRU cannot read or write the first block of the archive. Verify that the given device has a tape in the drive and that the tape has not been ejected.

`[W131] warning - archive device may need “ignoreclose” flag set in brutab entry`

We have found that some tape drives require this setting in the `brutab` file. For additional information on this settings please refer to chapter 3 of this User’s Guide

`[W132] warning - media size automatically adjusted to size`

This message is issued when BRU starts writing to a new volume and has changed the media size to a value which is different than the size originally specified. This can occur if BRU encountered the end-of-tape sooner than expected. For example, the media size was specified as 150M, but BRU hit the end-of-tape (on the first tape) at 120 Mb. BRU will then ask for the next volume and adjust the media size (for the second tape) to a value slightly less than 120 Mb.

`[W133] warning - no entry for device devicename in brutabfile`

BRU could not locate an entry in your `brutab` file for the given device.

`[E134] internal error errcode - failed self consistency and portability checks`

This indicates that the BRU executable file has been damaged or the BRU version is not the correct one for your system. This usually occurs when BRU cannot successfully determine your timezone offset from GMT or receives a bad value for the current date and time from a `tzset` function call.

`[E135] path beginning with filename too large (maxlen characters max)`

The specified filename exceeds the system limit.

`[W136] filename1: warning - link to filename2 broken, saved as duplicate`

The character length of the link name `filename2`, was too large to store in the file header block.

`[W137] warning - wait failed: errno = code, description`

During double-buffering, an error occurred while waiting for a child process.

`[E138] error - unable to format device devicename`

BRU could not format the device. It may be write protected, or command specified by `fmtcmd=` failed when attempting to format the device.

`[E139] error - timed out during execution of pid pidnum`

BRU attempted to execute an external program and the process timed out.

`[E140] error - unable to read include/exclude pattern file: filename`

The specified include/exclude pattern contains an error. BRU could not read the file.

`[W141] warning - bruxpat pattern: pattern, errmsg`

The specified include/exclude pattern contains an error. The reason for the error is indicated by `errmsg`. Edit the `bruxpat` file to correct the problem.

`[W142] -E option ignored with -c, -e, -i or -t options`

This is an illegal command option for the given modes `( -c -e -i or -t)`

`[I143] rewinding volume [volnum] to begin autoscan`

BRU prints this informational message to let you know it is rewinding volume number `volnum` to begin the AUTOSCAN. Rewinding may sometimes take a substantial (several minutes) length of time, it depends on the speed of your tape drive.

`[I144] begin autoscan of volume [volnum]`

BRU prints this informational message to let you know it has begun the auto-scanning of volume number `volnum`. Auto-scanning may sometimes take a substantial (several minutes to over an hour) amount of time, depending on the speed of your archive device.

`[E145] autoscan checksum error at block blocknum (kbsize)`

BRU detected a problem while performing a checksum validation during the AUTOSCAN phase. It could be an indication of tape hardware problems, dirty tape heads, an improper BRU buffer size, tapes with the wrong density, tapes that need to be retensioned, or tapes that are simply worn out. If this message occurs at the beginning of a tape (`blknum` is less than 10), it may indicate that your tape drive (or device driver) contains a bug and returned control to BRU before it finished rewinding. BRU tried to start the AUTOSCAN, but was unable to read the tape (because it was still rewinding). Often this problem can be fixed by setting the `maxrewindtime` parameter for your device in the `/etc/brutab` file. A setting of `maxrewindtime=300` seems to work for most tape drives. Sometimes this message occurs with tape drives that are confused by BRU's overwrite-protect feature. If overwrite-protect is enabled, BRU attempts to read the tape before it tries to write. Some tape drives cannot handle this. If this is the case, edit the `/etc/brutab` file and disable the global `brutab` parameter "`#+ OVERWRITE PROTECT`" by removing the "+" sign or by deleting the line.

`[E146] unable to get memory (bytecount bytes) needed for autoscan buffer`

BRU is unable to allocate enough memory to create an AUTOSCAN buffer. This message usually indicates that the buffer size (the `bufsize` or `asbufsize` parameters in `/etc/brutab`) is too large. It may also be caused by a shortage of memory or by system problems.

`[W147] cannot do autoscan - device 'devicename' has "flagname" flag set in brutab`

This usually indicates that the norewind or noautoscan flags are set in the device entry in your `/etc/brutab` file.

`[E148] autoscan read error at block blknum (kbsize): errno = code, description`

BRU received an error when attempting to read an archive during the AUTOSCAN phase. The reason for the error is indicated by the UNIX error message. This message may be an indication of tape hardware problems, dirty tape heads, an improper BRU buffer size, tapes with the wrong density, tapes that need retensioning, or tapes that are simply worn out. Often, this message occurs along with `[E145]`. Refer to its description for more information.

`[W149] autoscan detected errors - media or hardware may be bad`

This message indicates that errors or warnings were detected during the AUTOSCAN phase.

`[I150] autoscan of blkcount blocks on volume [volnum], time, speed Kb/sec`

`[L151] autoscan of blkcount blocks on volume [volnum], time, speed Kb/sec`

`[E152] error - timed out trying to open filename`

BRU timed out while trying open a device or file.

`[E153] error - timed out trying to read`

BRU timed out while trying read form a device or file.

`[E154] error - timed out trying to write`

BRU timed out while trying to write to a device.

`[E155] error - memory fault (SIGSEGV)`

BRU received a `SIGSEGV` signal from your OS.

`[E156] error - memory fault (SIGSEGV) in child process`

BRU received a `SIGSEGV` signal from UNIX on one of its child processes.

`[E157] error - received terminate signal (SIGTERM)`

BRU received a `SIGTERM` signal from your OS.

`[E158] error - received quit signal (SIGQUIT)`

BRU received a `SIGQUIT` from your OS.

`[E159] error - received interrupt signal (SIGINT)`

BRU received a `SIGINT` signal from your OS.

`[E160] error - received hangup signal (SIGHUP)`

BRU received a `SIGHUP` signal from your OS.

`[E161] error - received strange signal (signame)`

BRU received an unexpected signal.

`[W162] warning - unable to open execution log file ‘logfilename’: errno = code, description`

BRU was not able to open the `bruexeclog` file. This could be due to a permissions problem or the location of the file is not what BRU expected. With every BRU command or operation BRU will attempt to make an entry to the `BRUEXECLOG` file.

`[L163] START (info), CMD = ‘cmdline’`

The entry in you `BRUEXECLOG` and to your screen shows that actual command that was started during your BRU process. It also shows you the release and version of BRU you are running.

`[L164] START - child process for double-buffering`

`[L165] FINISH - warncount warnings, errorcount errors, exit code = exitcode`

`[L166] starting volume volnum on device “devicename”`

`[L167] device = devicename, buffer = bufsizeK bytes, media size = size`

`[E168] *** OBSOLETE MESSSAGE NUMBER ***`

`[E169] error - bad argument for -T option`

The user entered BRU options in an illegal combination.

`[E170] error - illegal combination of mode options (-cdeghitx)`

The user entered BRU options in an illegal combination.

`[W171] warning - needs to be owned by root and have suid bit set`

`[W172] warning - cannot open device ‘devicename’ to do autoscan`

BRU could not open devicename to perform AUTOSCAN verification.

`[E173] error - exceeded warning count limit of maxwarn`

BRU exceeded the maximum number of allowed warnings and terminated. The maximum number of warnings can be changed with the `BRUMAXWARNINGS` environment variable.

`[E174] error - exceeded error count limit of maxerror`

BRU exceeded the maximum number of allowed errors and terminated. The maximum number of errors can be changed with the `BRUMAXERRORS` environment variable or the `BRUMAXERRORS GLOBAL BRUTAB` setting see Chapter 3.

`[E175] error - bad pattern match on: {/etc/bruxpat}, entry`

This would and should be superseded by `[W141]`.

`[E176] error - bad raw block size = blksize, cannot extract raw file 'filename'`

BRU cannot restore or extract the given raw file from the archive. The information specified in the BRURAW file has an incorrect block size defined.

`[W177] warning - specified size is too large, try setting size to newsize Kbytes`

BRU has determined that the size given for the device you are using as an archive deice is to high. BRU has suggested that you use a smaller size as the one specified in this warning message.

`[L178] rewinding volume [volnum] to begin autoscan`

BRU writes this message to the `bruexeclog` to time stamp when it began to rewind volume `volnum` prior to starting an AUTOSCAN. Rewinding may take a substantial (several minutes) amount of time.

`[L179] issued reset cmd 'cmdstring'`

BRU has reset the device as specified by the reset command you defined in the `BRUTAB` file

`[W180] warning - reset cmd error: errno = code, description`

BRU received a UNIX error message when attempting to reset the device with the given reset command as specified in the `BRUTAB` file

`[I181] read/wrote blkcount blocks on volume [volnum], time, speed Kb/sec`

This reports the number of blocks written/read during the current BRU process. This will also indicate the speed at which the process ran by showing you the time and speed in Kb/sec.

`[L182] read/wrote blkcount blocks on volume [volnum], time, speed Kb/sec`

This reports the number of blocks written/read during the current BRU process. This will also indicate the speed at which the process ran by showing you the time and speed in Kb/sec

`[W183] skipped autoscan of volume [volnum]: reason`

BRU prints this message to let you know it has skipped the AUTOSCAN of volume `volnum`. The reason is indicated as part of the message.

`[I184] waiting time seconds to finish rewind`

Reports that BRU is waiting a certain amount of seconds before attempting to complete the process

`[L185] waiting time seconds to do rewind`

Reports that BRU is waiting a certain amount of seconds before attempting to complete the process

`[L186] using ‘rshname’ as remote shell`

BRU will use the given file name as it’s remote shell

`[E187] unable to execute remote shell ‘rshname’`

BRU is unable to use the given filename as the remote shell. Verify that it exists and that you have the proper permission set.

`[E188] cannot find remote shell to execute`

BRU can not execute the remote shell. Verify it is in the correct location and that you have proper permissions set.

`[A189] filename is not a device`

You have attempted to write to a file name. BRU is reporting that this is not a device but a file.

`[A190] file filename already exists`

BRU is reporting that you are attempting to write to a file that already exists on the system.

`[E191] error - compression buffer too large, cannot allocate kbcountk bytes`

The given `ZBUFSIZE` in your `BRUTAB` file is too large for your system to support. You should change the `ZBUFSIZE` setting in your `BRUTAB` file to a smaller number then re-attempt your process.

`[E192] filename: compression error (errmsg), data is corrupted`

An error occurred while BRU was compressing filename, the reason for the error is indicated by `errmsg`. The file was not backed up properly and the archived file contains errors. Normally, this error is caused when BRU attempts to back up a file that was changing (like a database file). For information, refer to “Live System Backups” in Chapter 8. In a few rare occurrences, this error has been caused by disk- controller hardware failures or by corrupted filesystems.

`[E193] filename: decompression error (errmsg)`

An error occurred during the restore and decompression of a compressed file. The `errmsg` will indicate what the problem is with the file.

`[E194] filename: warning - file was not backed up: errno = code, {description}`

This message is issued when BRU encounters a problem on its first attempt to read filename. Often, this is caused by improper permissions. If filename is part of an NFS-mounted filesystem or the `/etc/export` file (and BRU is running as root), this message can occur if BRU is unable to access the file due to insufficient root permissions. You may need to modify your network security parameters by editing your `/etc/hosts.equiv` or `.rhosts` file. Configuring network security can get complicated (and all networks are different), so refer to your UNIX network documentation for details.

`[Q195] enter new device name [default: devicename] >>`

BRU is requesting that you enter a new device name or to continue the operation on the default device.

`[E196] error - attempt limit exceeded ... BRU terminated`

BRU has reached it's maximum numbers of errors given in the `BRUTAB` file.

`[E197] error - illegal device name ... BRU terminated`

BRU has attempted to write to a device that is not stated in the `BRUTAB` file. The setting in `BRUTAB` of `BRUTABONLY=YES` has forced this error. By changing the default value to NO for this setting you will be able to write to this device.

`[W198] try using a smaller buffer size (like bufsizeK)`

BRU is attempting to write to a device that is responding with errors. BRU has determined this write error as having to large of a buffer size setting set for the given device. By using the `-b` option in your command line and change the buffer size to a smaller size BRU might be able to complete the operation.

`[A199] OVERWRITE PROTECT: volume is from archive written on date`

BRU has found that the date of this archive is within it's overwrite protect setting.

`[A200] insert another volume and press ENTER to continue`

Query message asking for a different tape.

`[E201] user entered Q => QUIT in routine routinename`

A user entered `Q` to quit the operation on one of BRU query to the terminal.

`[I202] switching to next device`

This is an information message stating that BRU is switching to a different tape drive. This message is sent during a device cycling operation

`[I203] device cycling discontinued`

BRU has stopped device cycling.

`[W204] filename filename1 too long, changed to filename2`

This message occurs when BRU restores a file with a name that is longer than 14 characters (the maximum on older UNIX systems). BRU automatically renames the file and shortens it to 14 characters. If your system supports filenames longer than this, this behavior can be suppressed. Simply add the global `brutab` parameter "`#+ MAXFILENAMELEN=255`" to the beginning of the `/etc/brutab` file.

`[I205] filename is an existing directory`

The given file name in filename is a directory

`[W206] regular expression error, string`

`[E207] error - failed MOUNTCMD/UNMOUNTCMD of volume volnum on device device name (exit code = #)`

BRU attempted to run the specified `MOUNT` and `UNMOUNT` commands as was not able to complete the operation. The device and volume number is listed in the error message.

`[W208] warning - autoscan buffer size adjusted to newbufsizeKb`

BRU found that the given buffer size for the `AUTOSCAN` process had to be changed to complete the process. If you see this warning message you should adjust your `BRUTAB` buffer size setting to match the given size for the warning message.

`[W209] warning - skipped archive file filename`

BRU has skipped the file listed on the warning message.

`[W210] warning - could not rewind device`

BRU could not rewind the given device.

`[E211] unable to write nullcount end nulls`

`[E212] BRU terminated, media may be write protected or wrong density`

BRU could not continue to write to the device. The reason for this error could be that the configuration of the tape drive size was wrong or that you are using a different size tape on this device.

`[W213] could not read password file`

BRU could read your password file. This can be caused by a permissions problem. Try running the process again as root and see if that resolves the warning.

`[W214] could not read group file`

BRU could not read your group file. This can be caused by a permissions problem. Try running the process again as root and see if that resolves the warning.

`[I215] translating filename to filename`

This is a status message informing you that the filename is being translated to the new filename

`[L216] translating filename to filename`

This is a log entry you will not see this message echoed to the screen

`[W217] warning - filename translates to “null”`

BRU is reporting the given filename in your translate file is translating to null

`[E218] error - unable to read translate table file: filename`

BRU is reporting that it can read your translation file. Make sure that the file is in an ASCII format and is in the given directory stated in your `BRUTAB` file

`[E219] translation table error ‘%s’ : %s`

`[O220] *** OBSOLETE MESSAGE NUMBER ***`

`[E221] filename : cannot overwrite directory with file`

BRU can not overwrite the directory with a file.

`[E222] filename : cannot overwrite file with directory`

BRU can not overwrite the file with a directory.

`[E223] unable to inspect “norewind” device`

The given device has the “norewind” option set in BRUTAB. With this option set BRU can not rewind the device and perform the AUTOSCAN option.

`[E224] error - invalid -u argument “%s”`

Invalid argument with the -u option. Valid options are `( a b c d i p r f )`

`[E225] error - unable to execute MOUNTCMD=”filename”`

BRU can not run the given `MOUNTCMD` or it is attempting to run a command that is illegal.

`[W226] “device” : device open when UNMOUNTCMD was called (loc=%d)`

The device was open with a different request when BRU called the `UNMOUNTCMD`

`[W227] “filename”: warning - error setting owner/group on symbolic link: errno = number, {description}`

BRU could not set the owner/group information for the specified file. The UNIX error number and description, descried the problem.

`[E228] error - unable to read raw-partition file: “filename”`

BRU could not read the given BRURAW file. Make sure that the file has the correct permissions and is in the given directory.

`[E229] error - no raw-device description in bruraw table: “filename”`

There was no information provided in the `BRURAW` file.

`[W230] warning - invalid raw-device specified (& ignored): “filename”`

A invalid device was specified in the `BRURAW` file BRU has now ignored that device and is not backing up the given device.

`[W231] warning - problem reading label file: “filename”`

BRU could not read the given file to create a label from. Make sure that the file exist and that

`[W232] “filename”: warning - file may be locked`

BRU is reporting that the given file name might be a locked file. BRU can not read the file and this might cause BRU to terminate if you have set a low setting for the maximum number of warnings.

`[W233] warning - unable to read smart-restore pattern file: filename`

BRU is not able to read your given Smart-Restore file. Verify that the file is in the correct ASCII format and has the correct permissions set.

`[W234] warning - smart-restore parse error ‘%s’ on line ‘%s’`

An error occurred while parsing your `/etc/smartrestore` file.

`[W235] warning - problem creating archive catalog: filename`

BRU could create its archive catalog file in the given directory specified in the `GLOBAL BRUTAB BRUTEMP=directory`

```
[W236] *** OBSOLETE MEGSSAGE NUMBER ***
[W237] *** OBSOLETE MEGSSAGE NUMBER ***
[E238] error - invalid -Q argument: “%s”
```

The argument that you passed with the `-Q` option is not valid. Arguments for `-Q` must immediately follow the `-Q` (i.e.: `-QL`) and be separated from the other options on the command line by at least one white space character.

```
[E239] error - invalid -U argument: “%s”
```

The only arguments for `-U` are numeric (`0, 1, 2,` etc).

```
[E240] error - unable to read, tape may be incompatible
[W241] warning - cannot translate “%s” to “%s”
```

Attempts to translate filenames during a restore were not allowed. This is probably a permissions problem.

```
[A242] enter label for volume %d:
[E243] error - previous UNMOUNTCMD on device “%s” failed
[W244] warning - left %d temporary files, check removelog file “%s”
```

During a restore, BRU ran into a number of open/in use files. These files were copied to a temporary directory and the tape version was restored. You may remove the temporary files by running the shell script displayed.

```
[I245] “%s”: skipped file, %s
```

Index