DCD Release Note 38.9 PU0164

Fermi Modular Backup System Users Guide

Marc W. Mengel

September 17, 1996

Abstract

The Fermi Modular Backup System is an attempt to build a single set of backup, restore, and other utility scripts that can be used by the various UNIX flavors here at Fermilab. It uses simple to write, short "modules" to provide different tape rotation schemes, tape mount systems, archive formats, etc. and also provides for backing up remote machines, and/or using remote tape drives. This document describes the use of fmb version v6_5.

1 What's New

In fmb version v6_5 several things have been added or changed:

- New `fmb_summarize` command to run an fmb command, and mail the output to a user with a success/failure `Subject:` line.
- The fmb commands now have a `-O` option to turn on OCS tape allocation and statistics logging.
- More new tape rotation schemes:
  - `pool` keeps a pool of tapes and rotates through them, skipping tapes with 3 write failures.
  - `biweekly` does a two-week rotation and first monday of the month for a year saves.
  - `improved_weekly` does an alternate one-week rotation with monday saves for a month and first monday of the month saves for a year.
- A new option `-E` has been added to specify exception lists (`fmb_backup` with an archive format of `cpio` only).
- A new option `-A` has been added to specify that archive creation errors should not be considered retry-able.
- A script `fmb_subdirs` has been added to allow easy automatic generation of lists of subdirectories, to make it easier to back up areas like home directory areas with an archive per user.
- Improved error status reporting of `fmb_backup` and `fmb_set`

Contents

2 Terms

The following terms are used throughout this document and are important to understanding the rest of the document. Please read over these definitions before continuing with the rest of the document.

2.1 archive

An archive is a file containing an image of a file system tree in some known archive format, such as the output of the unix `tar, cpio`, or `dump` commands.

2.2 archive format

An archive format is a method for representing a file/directory tree on a flat file, as in the output of of the unix `tar, cpio`, or `dump` commands.

2.3 backup tape

A backup tape is a tape consisting of a header file, followed by a series of archive files. The header contains a tape name, a date on which it was written, and information about the rest of the tape, such as the block size of the archive files, the archive format used to build the archive, and origin of the file trees that are archived.

2.4 backup log

A backup log is a file which lists what filesystems were backed up on a given tape. This log can be used to find out what tape contains which file. FMB now also supports verbose backup log directories, where a record of each and every file backed up is kept, listed in a file named after the tape upon which it was written.

2.5 file tree

A file tree is a directory and all of its subdirectories, sub-sub-directories, etc. as well as all the files contained therein. It is generally named by a hostname, a colon, and a directory path from the root directory on that host.

2.6 mount scheme

A mount scheme is the mechanism by which tapes end up in tape drives. The Fermi Modular Backup utilities that deal with tapes will all request that the tape be placed in the appropriate drive using one of the available mount schemes. For example "opcom" is a mount scheme by which an operator on the VAXCluster is asked to mount a tape in a given tape drive.

2.7 rotation scheme

A rotation scheme is the rule by which backup tape [see 2.3]s are selected. A simple rotation scheme would be two tapes, an odd and an even tape, and each tape is used every other day.

2.8 tape label

a tape label is a name for a tape usually of the form `host.day.set` where the host name is the host writing the backup tape [see 2.3], the day is the name for today from a rotation scheme [see 2.7], and the set is an identifier to differentiate multiple backup tape [see 2.3]s written on the same host on the same day.

3 Common Tasks

3.1 Seeing what is on a tape

3.1.1 Getting a simple archive list

To check a tape and see what is on it, simply use `fmb_toc` [see 6.6] and specify a given tape drive and mount scheme [see 2.6]. If you are using a mount scheme [see 2.6] which requires a tape name, (i.e. an operator or jukebox tape mount) specify the tape name with the `-L` option.

For example:

```
% fmb_toc -t /dev/rmt0.1
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
%
```

3.1.2 Getting a full listing of one archive

To get a full listing of one archive on the tape, specify the archive name on the command line when issuing the `fmb_toc` [see 6.6] command. Specify the -v option [see 7.25] to get a "wide" listing of that archive. For example:

```
% fmb_toc -t /dev/rmt0.1 -v  server2:/usr
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archives:
    server2:/
```

```
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Printing table of contents of server2:/usr (file number 3)...
Finding file number 3...
40755  root        0  Jul 13 10:07:01 1993  .
40700  root        0  Feb 19 18:34:03 1991  lost+found
41777  sys         0  Sep  8 11:15:06 1993  tmp
40700  root        0  Oct 19 11:54:52 1992  tmp/lost+found
40755  upd         0  Mar  8 14:16:54 1993  tmp/mgr
40755  upd         0  Mar 17 00:30:05 1993  tmp/mgr/mkf
40755  upd         0  Aug 12 00:30:08 1993  tmp/bin
100644 upd      1362  May 12 09:58:11 1993  tmp/sgi.lis
40755  upd         0  Jul  9 00:30:10 1993  tmp/ups
40755  upd         0  Mar 21 00:30:05 1993  tmp/ups/toInfo
40755  root        0  Aug  8 00:30:07 1993  tmp/rbio
40755  upd         0  May  5 00:30:07 1993  tmp/lib
100644 root       11  Sep  5 04:39:00 1993  tmp/.fsrlast
40755  upd         0  Jun 25 10:51:41 1993  tmp/src
40755  upd         0  Jul  4 00:30:09 1993  tmp/src/fort
40755  e665        0  Jun  2 19:03:43 1993  tmp/newver.20677
40755  root        0  Sep  2 09:35:22 1993  lib
40755  root        0  Jun 19 11:50:09 1993  lib/inst
100644 root      564  Apr  6 11:55:30 1993  lib/inst/audio
100644 root     1907  Apr  6 11:55:30 1993  lib/inst/maint_nfs
100644 root   412798  Apr  6 11:55:24 1993  lib/inst/hist
...
%
```

Note that the archive listing lists path names relative to the archive name, rather than a full path name.

3.1.3 Getting a full listing of all archives

To get a full listing of all archives on the tape, specify the -v option [see 7.25] and no specific archive. Be warned, this usually generates lots of output! For example:

```
% fmb_toc -t /dev/rmt0.1 -v
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archives:
    server2:/
```

```
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Printing table of contents of server2:/ (file number 2)...
Finding file number 2...
40755  root        0  Jul 13 10:07:01 1993  .
40700  root        0  Feb 19 18:34:03 1991  lost+found
41777  sys         0  Sep  8 11:15:06 1993  tmp
...
Printing table of contents of server2:/usr (file number 3)...
Finding file number 3...
40755  root        0  Jul 13 10:07:01 1993  .
40700  root        0  Feb 19 18:34:03 1991  lost+found
41777  sys         0  Sep  8 11:15:06 1993  tmp
40700  root        0  Oct 19 11:54:52 1992  tmp/lost+found
40755  upd         0  Mar  8 14:16:54 1993  tmp/mgr
...
Printing table of contents of server2:/usr/products (file number 4)...
Finding file number 4...
40755  root        0  Jul 13 10:07:01 1993  .
40700  root        0  Feb 19 18:34:03 1991  lost+found
41777  sys         0  Sep  8 11:15:06 1993  tmp
...
%
```

3.2 Restoring Files

3.2.1 Restoring a Single File

Restoring a single file from a backup tape [see 2.3] is a fairly easy operation, although first of course you need to know which tape to look on. If your system has the history directory configured via `fmb_options` [see 6.8], you can simply use the `fmb_locate` [see 6.3] command and get a list of the last backup tape [see 2.3] to successfully backup the file `myfile` in the current directory, and do a restore to get the file back. If the weaker file logging options are used [see sections 7.26 and 7.17] you can use `fmb_find` [see 6.4 to find a tape that has archives of the same filesystem as the file you specify.

```
# fmb_restore -L server1.18.A -w myfile
Please insert tape server1.18.A in drive /dev/rmt0 and press Enter

Checking tape label
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archives:
```

```
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Restoring guest/myfile from server2:/usr into server2:/usr...
Finding file number 3...
Restoring...
#
```

The -L flag specifies a specific tape label [see 2.8], so that if your system defaults to using a tape jukebox or operator mounts the right tape will be requested. The -w option [see 7.27] tells fmb to put files where they came from. The default (if -w is not specified) is to unpack the file relative to the current directory; this is useful if you want to unpack a file or files in the /tmp directory.

If you do not know what tape to restore from specifically, because a history log was not created, or you need a particular version of the file from several weeks ago, etc. the fmb_toc [see 6.6] command can be used to get a table of contents of the tape and see what is on it, and when it was written.

3.2.2 Restoring Multiple Files

You can restore several files from an archive by using wildcards to match filenames, but only if you're using cpio or tar archive formats [see 2.2]. Both of these archivers use asterisks ("*") to specify partial matches for names. For example if you wanted to get all of the files ending in a ".c" suffix from the /usr/fred/src/program directory on machine server2, you could simply issue the command:

```
# cd /tmp
# fmb_restore 'server2:/usr/fred/src/program/*.c'
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archive archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Restoring fred/src/program/*.c from server2:/usr into server2:/tmp...
Finding file number 3...
Restoring...
```

Note the single quotes around the filename with wildcards. The quotes are neccesary to stop the shell from attpempting to replace the "*.c" with a list of files.

You can also list multiple patterns on the command line, causing the restore to unpack files from several archives, although this will currently do one unwind per pattern, even if the patterns are on the same archive.

```
# cd /tmp
# fmb_restore 'server2:/usr/fred/src/program/*.c' 'server2:/usr/lib/*.a'
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archive archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Rewinding...
Restoring fred/src/program/*.c from server2:/usr into server2:/tmp...
Finding file number 3...
Restoring...
Rewinding...
Restoring lib/*.a from server2:/usr into server2:/tmp...
Finding file number 3...
Restoring...
```

3.2.3 Restoring Whole Directories

If you want to restore a directory and its contents, you usually need to specify the directory with a trailing '/' like the following:

```
# cd /tmp
# fmb_restore 'server2:/usr/fred/src/'
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archive archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Rewinding...
```

```
Restoring fred/src/ from server2:/usr into server2:/tmp...
Finding file number 3...
Restoring...
```

3.2.4 Restoring Whole Archives

Using the `fmb_restore` [see 6.7] command to restore an entire archive is also fairly easy to do; simply specify the archive name itself with a trailing "/" when doing the restore; for example to restore all of `server2:/usr`, you would do:

```
# cd /newusr
# fmb_restore server2:/usr/
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archive archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Restoring  from server2:/usr into server2:/newusr...
Finding file number 3...
Restoring...
```

3.2.5 Restoring Multiple Archives

Sometimes you have multiple partitions you need to restore from a backup tape [see 2.3], because they were on the same disk and it has been replaced, or for some other reason. In this case you probably want to use the -w option [see 7.27] to indicate that files should be put back where they came from, otherwise the archives you list will be unpacked in the current directory, one overtop the other. In the following example we unpack 3 partitions from the same tape with one `fmb_restore` [see 6.7] command, after using `fmb_toc` [see 6.6] to check where and when the tape was written and what was on it.

```
# fmb_toc
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archives:
```

```
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
# fmb_restore -w server2:/usr/ server2:/usr/products/ server2:/usr/products/IRIX/
Mount tape  on /dev/rmt0.1 and press Enter
Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archive archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Restoring  from server2:/usr into server2:/usr...
Already at file 1...
Restoring  from server2:/usr/products into server2:/usr/products...
Already at file 2...
Restoring  from server2:/usr/products/IRIX into server2:/usr/products/IRIX...
Already at file 3...
```

If you have really long lists of archives, (for example if you've been making separate archives of each user's home directory, etc.) you can redirect `fmb_toc` [see 6.6]'s output onto a file, edit the file to trim down to the list of archives you want, and do a restore with the -f file option [see 7.14] to use the file list in the file, like this:

```
# fmb_toc > myfile
Mount tape  on /dev/rmt0.1 and press Enter

Thanks!
# ex myfile
: 1,5d
: 1,$p
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
: 2,$s/$/\//
: 1,$p
    server2:/
    server2:/usr/
    server2:/usr/products/
    server2:/usr/products/IRIX/
```

```
: w
"myfile" 4 lines...
: q
# fmb_restore -w -f myfile
Mount tape  on /dev/rmt0.1 and press Enter

Thanks!
Tape server1.18.A:
    written Tue Jan 26 11:30:50 CST 1993
    format cpio
    blocksize 16k
  archives:
    server2:/
    server2:/usr
    server2:/usr/products
    server2:/usr/products/IRIX
Restoring  from server2:/usr into server2:/usr...
Already at file 1...
Restoring  from server2:/usr/products into server2:/usr/products...
Already at file 2...
Restoring  from server2:/usr/products/IRIX into server2:/usr/products/IRIX...
Already at file 3...
```

3.3 Doing Occasional Archiving

If you want to make an archive of some particular set of files with fmb, either to store the data long term, or to send to someone else, etc. you simply need to choose a reasonable label for the tape, specify the files you want backed up, and use `fmb_backup` [see 6.2] to write the tape, and `fmb_restore` [see 6.7] to unpack it later. The backup log file will list what tapes the files were saved on for later retrieval.

```
# fmb_backup 'server1:/usr/people/fred/src/program' -L Archive1 -u
Mount tape Archive1 on /dev/rmt0.1 and press Enter

Thanks!
Checking tape label
Writing Label Archive1

        Archive Size Summary

Estim.  Actual  %Est  Name
------  ------  ----  ----
  1872    1888   99%  server1:/usr/people/fred/src/program
------  ------
  1872    1888 TOTAL
```

```
Unmount tape Archive1 on /dev/rmt0.1
```

4 Administrative Tasks

4.1 Disk Repartitioning

When rearranging disks it is often convenient to dump data to tape, then repartition your disks, and finally restore selective parts from tape back to the disk. fmb is handy for this purpose; you can use `fmb_backup` [see 6.2] to back up the data, listing several archives, and after repartitioning you can change directories to a new partition, and restore the archvies that go there, then change directories to the other and unpack those. Or you can back up the entire partition in one archive, and use wildcards to restore the portion of the files you want in the new partition scheme.

4.2 Setting up Nightly Backups

Setting up backups to run automatically is a fairly easy task, but does require a little forethought on larger systems. You must decide what data needs to be backed up, as well as a backup policy; i.e. how often things need to be backed up, and how long backup tape [see 2.3]s should be kept. Then you can get and prelabel tapes, and start regular backups. Let's examine these in more detail.

4.2.1 What needs to be backed up?

Deciding what data needs to be backed up is usually fairly easy. The majority of the system usually needs to be backed up, with the exception of explicit scratch or temporary areas (like `/tmp`). On systems with multiple machines and NFS mounts, local filesystems on the clients need to be backed up as well as the areas provided by NFS servers. Simply make a list of the hostnames and filesystems that should be backed up; something like:

```
client1:/
client1:/usr
client1:/var
client2:/
client2:/usr
client2:/var
server1:/
server1:/usr
server1:/var
server1:/usr/products/AIX
server2:/
server2:/usr
server2:/var
server1:/usr/products
server1:/usr/products/IRIX
```

and put this in a file ("mylist" for example).

4.2.2 When should it be backed up?

Deciding on a backup policy is a bit harder. Usually you need access to files at least a few days back, in case someone realizes they deleted a file a few days ago and that they need it restored. Similarly, it might be important for your system to be able to retrieve files from up to a year ago to see old versions of a software product, login records, etc. in case of disaster. fmb comes with two good rotation schemes [see 2.7] for tapes that implement simple policies. The weekly rotation scheme just has a different tape for each day of the week; this means that you generally have a weeks worth of backups on hand all the time; but anything more than a week old is gone. The monthly rotation scheme similarly rotates a tape for each day of the month, but also has a "first Monday of the month" tape for each month. This means you have all the tapes for the last month, but also have first of the month tapes for the last year; and can often retrieve files up to a year old. If these schemes are not sufficient for your purposes, you can build your own rotation scheme. See section 4.7 entitled "Adding New Rotation Schemes".

Another important aspect of your backup policy is deciding how tapes are going to get changed. Is someone going to put tapes in the tape drives during the day for nightly backups? Are operator mounts going to be requested? Is some sort of robotic jukebox going to be used? This lets you choose a mount scheme [see 2.6] and tape drives for your backups. fmb provides several mount schemes (see the description of the -m mount_type option [see 7.18]) but most local systems use the "quiet" mount scheme ; it assumes someone has already put the tape in the drive for it. If you need a mount scheme different from the ones provided, see the section "Adding New Mount Schemes".

4.2.3 Making tape sets

Now that you have an idea what needs to be backed up, and the number of copies of each piece of data you have, we can figure out how many tape sets we'll need. First we need to know how big the total data set is, we can use the `fmb_max` [see 6.5] command to find that, like this:

```
# setup fmb
# fmb_max -f mylist
Max     Used    Directory
32768        20768   client1:/
301056        228532  client1:/usr
18513        10230   client2:/
244545        189126  client2:/usr
20475        17206   server1:/
301000        222510  server1:/usr
1002820        722950  server1:/usr/products/AIX
20475        17206   server2:/
301000        222510  server2:/usr
672256        483328  server2:/usr/products
1002820        722950  server2:/usr/products/IRIX

3917728        2857316        TOTAL
```

If you get lots of P̈ermission Denied" errors at this point, you need to check that you have `.rhosts` files set up properly so that you can rsh to the different machines involved.

So we have a total of 4G of data possible, of which 3G is actually in use. With low density 8mm tapes, this would mean we need 2 tapes per night to do a full backup; with a high density drive we could do just one. We'll use the low density case for our example, this means we need to do two backup tape [see 2.3]s each night, therefore two sets of backup tapes will be needed. We then need to split our list of data to be backed up into two sections of about 2G apiece. In our example it looks like server2's data alone is about 2G, so we can put all the server2 stuff on one tape, and everything else on the second one. We can now make files that describe our situation, one file with the server2 data:

```
server2:/
server2:/usr
server2:/var
server1:/usr/products
server1:/usr/products/IRIX
```

and one with the rest:

```
client1:/
client1:/usr
client1:/var
client2:/
client2:/usr
client2:/var
server1:/
server1:/usr
server1:/var
server1:/usr/products/AIX
```

and we can verify our sizes by using `fmb_max` [see 6.5] again, this time once on each dataset:

```
# setup fmb
# fmb_max -f mylist.A
Max     Used    Directory
20475        17206   server2:/
301000        222510  server2:/usr
672256        483328  server2:/usr/products
1002820        722950  server2:/usr/products/IRIX

1725651 1445994 TOTAL

# fmb_max -f mylist.B
Max     Used    Directory
32768        20768   client1:/
```

```
301056        228532  client1:/usr
18513        10230   client2:/
244545        189126  client2:/usr
20475        17206   server1:/
301000        222510  server1:/usr
1002820        722950  server1:/usr/products/AIX

1921177 1411322 TOTAL
```

Now we can make these files into complete options files for `fmb_backup` [see 6.2] by adding a set identifier to the file; just add a "`-s A`" line to the mylist.A file, and a "`-s B`" line to the mylist.B file, and install them in the `$FMB_DIR/fmb_files` directory.

4.2.4 Running a test backup

Next we should run a test backup, to make sure we have sufficient privledges (with .rhosts files, etc.) to actually run the backups. For the test, we'll specify our list of archives with the -f file option [see 7.14], the tape drive to write on with the -t tape option [see 7.23], and that the tape is unlabelled with the -u option [see 7.24].

```
# fmb_backup -t /dev/rmt/tps0d3nrnsv -f mylist.A -u
Mount tape server2.Mon.A on /dev/rmt/tps0d3nrnsv and press Enter

Thanks!
Checking tape label
Writing Label Archive1

        Archive Size Summary

  Estim.   Actual  %Est  Name
  ------   ------  ----  ----
  17290k   17206k   99%  server2:/
 222494k  222510k   99%  server2:/usr
 483312k  483328k   99%  server2:/usr/products
 722816k  722950k   99%  server2:/usr/products/IRIX
 -------  -------
1445832k 1445994k TOTAL

Unmount server2.Mon.A from /dev/rmt0.1

# fmb_backup -t /dev/rmt/tps0d3nrnsv -f mylist.B -u
Mount tape server2.Mon.B on /dev/rmt/tps0d3nrnsv and press Enter

Thanks!
Checking tape label
```

```
Writing Label Archive1

        Archive Size Summary

  Estim.   Actual  %Est  Name
  ------   ------  ----  ----
  20752k   20768k   99%  client1:/
 228532k  228532k  100%  client1:/usr
  10230k   10230k  100%  client2:/
 189126k  189126k  100%  client2:/usr
  17206k   17206k  100%  server1:/
 222510k  222510k  100%  server1:/usr
 722950k  722950k  100%  server1:/usr/products/AIX
 -------  -------
1411306k 1411322k TOTAL

Unmount server2.Mon.B from /dev/rmt0.1
#
```

4.3 Important tape drive considerations

Note that we used no-rewind, variable block device names for the tape drives. fmb WILL NOT WORK PROPERLY unless a no-rewind tape device is used. Only the last archive written will be on the tape if a rewind-on-close device is used. Also, it is recommended that a variable blocksize device be used where possible, and that "no swap" devices on SGI's be used for tape portability.

4.4 Option Files for Tape Options

It is usually simpler in the long run if you put archive lists in one configuration file, and tape drive options in another. For example if we put the options for a given tape drive in an option file, it makes a nice shorthand for dealing with that tape drive in the future:

file `drive1`

```
-t server2:/dev/rmt/tps0d3nrnsv
-b 16k
-m quiet
```

file `drive2`

```
-t server1:/dev/rmt0.1
-b 16k
-m quiet
```

Then we can do a backup by simply specifying the option file for the tape drive and the archive list:

```
# fmb_backup -f mylist.A -f drive1
```

4.4.1 Configuring cron to run backups nightly

Finally, we need to add appropriate crontab entries to the crontab file to get the backups run every night. If we want the A set to run on server2 every night, and the B set to run on server1, and both to run at midnight, we'd get an entry like:

```
0 0 * * * /usr/products/fmb/v5_2/fmb_backup -f drive1 -f mylist.A
```

on server2, and:

```
0 0 * * * /usr/products/fmb/v5_2/fmb_backup -f drive2 -f mylist.B
```

on server1. See the previous sections for the descriptions of the "drive2" and "mylist.B" files we are using here.

4.4.2 Other configuration options to consider

There are also a few other options you may want to configure for your backups. The archive format[see 2.2] (i.e. cpio, dump, tar, etc.) and logging options are ones you may want to set with the `fmb_options` [see 6.8] command so that you get backups in the format you want, and a good log of what data is backed up where. Note that dump format, while often the fastest backup format, is not appropriate for filesystems that may be active at the time they are backed up. See section 5 for a complete discussion of all the fmb options.

Also note that options set with `fmb_options` [see 6.8] are on a per-host basis; if you want the same options on multiple hosts you can either set them from there on each host, or change directories to the $FMB_DIR/`fmb_defaults` directory, and link the options file for one host to the name of another host; this way if you change one, the others linked to it change too.

4.5 Error Recovery for Important Backups

Now let us examine a situation where we have a machine with a tape carousel, and we want to do two machines' backups to tapes in the carousel, every night, a total of 5 tapes worth of data, but fall back on OPCOM operator mounts in the event that a tape is not in the carousel.

Since `fmb_set` [see 6.10] uses option files, we need to define first our option files that describe what we want backed up, and ones that describe what drives we want to use to back them up. Here are the first two; they specify cpio archive backup tapes [see 2.3] of the root and user partitions of fnsg01 and cdfsga:

File `01.A`:

```
-a cpio
-x
-h fnsg01
-s A
fnsg01:/
fnsg01:/usr
```

File `a.A`:

```
-a cpio
-x
-h cdfsga
-s A
cdfsga:/
cdfsga:/usr
```

We would fill out other, similar ones, specifying set names of 'B' or 'C' with the -s set option [see 7.22] as `-s B` or `-s C`, and other lists of file trees [see 2.5] to back up.

Next we need tape description files for tape drives in the carousel; something like:

File `car1`

```
-t /dev/rmt/tps1d6nrnsv
-m summus
```

File `car2`

```
-t /dev/rmt/tps1d7nrnsv
-m summus
```

And finally, when we don't find the tapes in the carousel, we want to do an OPCOM mount on the external 8mm drive:

File `ext`

```
-t /dev/rmt/tps0d7nrnsv
-m opcom
```

Then we would add a crontab entry to run `fmb_set` [see 6.10] to do the backups at night, like this one:

```
0 0 * * * /usr/products/fmb/v5_3/fmb_set  "01.A 01.B 01.C a.A a.B" "car1 car2" "ext"
```

This gives us the backup behavior we want of trying the carousel tape drives, and then retrying any failed backups on the external tape drive.

4.6 Adding New Mount Schemes

The requirements for a mount script are fairly simple; you need to write a script that takes arguments -u or -un or no arguments, as follows:

none The tape `${label}` should be mounted on the ${tape_path} tape drive, on the host reachable with ${tape_rsh}.

-u The tape `${label}` should be unmounted from the ${tape_path} tape drive, on the host reachable with ${tape_rsh}. The tape should be unloaded, with the command

```
${tape_rsh} "'${MT} ${tape_path} ${MTEJECT}'"
```

If the ${fmb_keep} environment variable is set, the script should do nothing when asked to unload.

-un The tape ${label} should be unmounted from the ${tape_path} tape drive, on the host reachable with ${tape_rsh}. The tape is assumed to have already been unloaded. If the ${fmb_keep} environment variable is set, the script should do nothing when asked to unload.

Users should note and copy the quoting conventions used in the existing scripts when using the $tape_rsh variable.

A good basic example is the $FMB_DIR/fmb_mount/manual script which asks the user to mount the tape.

4.7 Adding New Rotation Schemes

There are four scripts that should be used to provide a rotation scheme [see 2.7.

- makelabel uses the shell variables host, day, and set to make a label name, which it prints on its standard output.
- today prints today's day name, which becomes the day shell variable.
- list generates a list of all the day names in the rotation. This script is used by fmb_prelabel [see 6.9] to build a list of all the tape labels in a set of tapes.
- error is called with a label name (e.g. the one printed by makelabel to indicate that a write error occurred on the tape of that label name.

The scripts should be placed in a subdirectory of the $FMB_DIR/fmb_label directory, and the subdirectory name is the name of the rotation scheme.

4.8 Adding New Archive Formats

The three scripts for an archive format[see 2.2] called name are stored in $FMB_DIR/fmb_arch/name, and are called pack, unpack and toc. The easiest way to add a new archive format is to copy one of the existing sets of scripts (like the ones in $FMB_DIR/fmb_arch/cpio) and replace the actual commands in the script that do the creation/unpacking of the archives. This way you can avoid debugging the hideous quoting conventions and slightly messy argument passing conventions currently in use in the fmb scripts. If you have an archive format that needs to be invoked differently on different platforms, start with the $FMB_DIR/fmb_arch/dump scripts, which check the machine type of the (possibly remote) host and pick a different invocation string for each one.

If you need to write your own scripts altogether, you should make the pack script take an argument of the form user@host:path and generate an archive rooted at that pathname on that host on standard output, with relative pathnames. It should also echo "Exit code n" on the standard error to indicate the success or failure of the archive's creation.

The unpack script takes 2 arguments, the first is a pattern of files to unpack from an archive on the standard input; the second is an argument of the form user@host:path which indicates the directory the files should be unpacked into.

The `toc` command should take an archive on its standard input and generate a table of contents on its standard output.

All of the scripts for a given archive format[see 2.2] should take appropriate action if the `${fmb_verbose}` flag is set – i.e. give a longer or wider or more detailed listing if possible, list files as they are packed, etc.

Note that the pack script script should build an archive with path names relative to `${arch_path}` so that it can be unpacked somewhere else later.

The pack script should also look for the `${partial}` environment variable to be true or false, to decide if a partial or incremental backup should be performed.

The scripts should be placed in a subdirectory of the `$FMB_DIR/fmb_arch` directory, and the subdirectory name is the name of the archive format.

5 Error Handling

The fmb scripts generate very few error messages of their own, they mainly detect error returns from the various archive programs and `dd`, and proceed to display them with a wrapper saying at what point in the process the (apparent) error occurred.

5.1 Error recovery

fmb forges on through most errors, except for tape write errrors. It attempts to make sure some data gets written for each archive (even if its just a few blanks) so that file skipping based on the header information on the tape will be correct.

5.2 Format of error messages

Error messages from fmb generally look like:

```
*
fmb: error during fmb_backup on Tue Jun 15 20:00:08 CDT 1993
Write of ... to tape failed, dd output was:
    dd: write error: Invalid argument
    131281+2038 records in
    66187+1 records out

    Exit code 2
```

The "fmb: error...Write of..." part is the fmb wrapper that says when the error occurred, and what fmb was trying to do, in this case write a particular archive to tape. It then gives the actual error message that was generated by the program, and the exit code with which the program exited (0 is a normal exit, anything else indicates some sort of failure.) To find out what a particular error message means, you need to consult the manual pages for `dd(1), cpio(1), tar(1), dump(1)` or `restore(1)`, to help interpret the error messages that they have generated. For tape write errors, you sometimes need to chase down documentation on tape errors on that platform with `"man 4 rmt"` on AIX, `"man 7 mtio"` and `"man 7 jagtape"` on IRIX, `"man 4 mtio"` and `"man 4 st"` on SunOS.

5.3 Redirecting error messages

The -n option [see 7.19] can be used on most of the fmb programs to specify a filter or program to send error messages to. This can be used to mail just the error messages to someplace special, or write them on a log file, etc. so that error conditions can be alerted somewhere and kept separate from the normal backup output. The `fmb_mail` script is reccomended for mailing errors somewhere, because it also puts the error message in the backup output as well as mailing it somewhere. You might want to use `tee -a logfile` to put error messages on a log file and keep it in the backup output. If you have an error command that does not repeat the error on standard output, the backup output will not contain the error message, it will only go where your program sends it.

5.4 "Permission Denied" errors

"Permission Denied" errors are almost always lack of permission to do an `rsh` command to another host, although they may indicate lack of permission to change directories to a directory to be backed up, or to read a raw disk device for `dump` archives or write on tape devices. Make sure you can rsh to the host as the user who actually runs the backup, and that you have read permssion on the disk and write permission on the tape.

5.5 "Exit code 141" from cpio

This is an error that occurs when an archive doesn't fit on the tape, and the version of `dd` on the machine writing the tape doesn't consider it an error to hit end of file on the output. In this case `dd` simply exits silently, and cpio gets a SIGPIPE, and exits with an error code of 141. This means you need to rearrange which archives are on which tapes, so that everything will fit.

5.6 "dd: write error: Invalid argument"

This usually occurs when you are using a fixed-block-size tape device, fmb tries to block the data up to a particular block size, but at the end of an archive it often isn't. You should use a variable-block-size device for fmb backups. Under AIX this usually means changing the block size to zero in SMIT.

5.7 "Archive of ... failed" with no other error

Sometimes `cpio` or `dump` will exit with non-zero exit codes when everyhing basically worked fine. When this occurs, fmb will report this as an error, but have no error output for the program.

5.8 "archive of ... was a different size than estimated"

fmb also considers it an error if the amount of data written to tape is signifigantly different from an estimated amount it generates from using `du` and `df`. This is occaisonally due to files with "holes" in them (like SunOS core files, or some database files) and not really a problem. Other times it's due to permission problems reading individual files, or disk read errors. If there is no error other than the size difference, and the archive is bigger than estimated, the error message is probably spurious.

A common cause is large numbers of small files, or large numbers of symlinks, which generally cause the size of cpio or tar archives to be different than the estimate. Since du and df count disk blocks, and small files and symlinks commonly use less than a quarter of a disk block, the estimate given by du or df is often wrong, and fmb must try to "fudge" the result for this factor.

A less common cause of these differences if you're backing up with cpio or tar is files being "hidden" by a mount of another filesystem; for example if there is a file in the root filesystem called /usr/data/bigfile but there is a filesystem mounted over /usr/data, then the file bigfile is effectivly hidden, but still takes up space in the root filesystem. This usually happens because a filesystem was not mounted at some point, and data got written in the root filesystem instead of the one it should have been written in.

6 Commands Description

This section discusses the various commands in the fmb package, what options affect them, and where to find examples of their use elsewhere in the document.

6.1 fmb_summarize

The fmb_summarize command takes a command line of the form

`fmb_summarize` *"mail address list" command*

runs the command, and mails the results to the users in the *mail address list*, with a summary subject line.

6.2 fmb_backup

The fmb_backup command will write one backup tape [see 2.3], composed of archives of various file tree [see 2.5]s on various systems. It assumes that the file trees given to it will fit on the tape specified; there is no provision for mounting a second tape to finish an archive, etc. It exits with a return code of 0 if the backup succeded; 1 if the tape was not available to be mounted, the label check failed, or if there were any errors writing on the tape that were not recoverable by retrying the writing of various archives.

If you want to setup automatic spillover-to-the-next tape backups, you need to use either a wrapper script that divies up the backup data to multiple tapes via directories, or a script to invoke via the-e option [see 7.13] which will generate a list of directories to backup for each tape.

See the Setting up Nightly Backups or Doing Occasional Archiving sections for examples.

The fmb_backup command exits with the following exit codes:

0 The backup was successful

1 The backup was unable to mount the tape

2 The backup had unrecovered tape write errors

3 The backup had unrecovered archive creation errors

4 The post-backup label check failed.

It uses the following options:

- -A command option [see 7.1]
- -C command option [see 7.2]
- -D drive option [see 7.3]
- -E exception-pattern option [see 7.4]
- -F option [see 7.5]
- -L label option [see 7.6]
- -O option [see 7.7]
- -P option [see 7.8]
- -R retries option [see 7.9]
- -a arch_format option [see 7.10]
- -b blocksize option [see 7.11]
- -d day option [see 7.12]
- -e command option [see 7.13]
- -f file option [see 7.14]
- -h host option [see 7.15]
- -k option [see 7.16]
- -l log option [see 7.17]
- -m mount_type option [see 7.18]
- -n option [see 7.19]
- -p option [see 7.20]
- -r rotation option [see 7.21]
- -s set option [see 7.22]
- -t tapedrive option [see 7.23]
- -u option [see 7.24]
- -v option [see 7.25]
- -V directory option [see 7.26]
- -x option [see 7.28]
- pathname arguments

6.3 fmb_locate

The fmb_locate command provides an interactive querey for searching a verbose log directory for tapes conataining a particular file or regular expression. For example to look for user mengel's .login file, you would:

```
% fmb_locate
   +------------------------------------------------------------------+
   |                Backup Log Search Criterea
   |                ------ --- ------ --------
   | 0) Exit Program
   | 1) Start Date:            1/1/1970
   | 2) End Date:              03/31/1994
   | 3) Filename Regex:
   | 4) NFS Server:
   | 5) Logfile Directory:     /usr/products/cluster_disk/fmb/logs
   | 6) Logfile Regex:         .*
   | 7) Ask on Each File:      true
   | 8) Start Search
   | 9) Summarize Results
   |10) Restore Individual Files
   |11) Restore Directories
   +------------------------------------------------------------------+
Enter choice number:
```

At this point you can enter a number from 0 to 7, and change a search critera, or enter 8 to start a search, or 10 to summarize search results. We can enter a 1, and change the date, and a 3 and set a file regular expression, like:

```
   +------------------------------------------------------------------+
   |                Backup Log Search Criterea
   |                ------ --- ------ --------
   | 0) Exit Program
   | 1) Start Date:            03/29/1970
   | 2) End Date:              03/31/1994
   | 3) Filename Regex:        mengel.*\.login
   | 4) NFS Server:
   | 5) Logfile Directory:     /usr/products/cluster_disk/fmb/logs
   | 6) Logfile Regex:         .*
   | 7) Ask on Each File:      true
   | 8) Start Search
   | 9) Summarize Results
   |10) Restore Files
   |11) Restore Directories
   +------------------------------------------------------------------+
Enter choice number: 8
```

Now when we start the search, we will be prompted whether we want to check each file in the backup log directory (unless we had set item 7 to be "false").

We can now summarize the search results by choosing item 10, and we get a result like:

```
Enter choice number:  9
Tapes found were:
        Fncase.30.E
Which were backed up on drives:
fndaut:/dev/nrst0
And files matched were:
fndaut:/t2/mengel/.login
```

We can now restore the file by choosing item 10, selecting the tape, file and tape drive from the lists provided.

```
Enter choice number:  10
Matches were found on the following tapes:
1       fndaub.24.C
2       fndaub.23.C
Which tape would you like to restore from [1..2]?  2
Backup tape drives are:
1       /dev/rmt/tps0d5nrnsv.8500
2       /dev/rmt/tps0d6nrnsv.8500
3       fncase:/dev/nrst8
4       fndaub:/dev/rmt/tps0d3nrnsv.8500
5       fndaub:/dev/rmt/tps1d2nrnsv.8500
6       fndaub:/dev/rmt/tps1d6nrnsv
7       fndaut:/dev/nrst9
Which drive would you like to restore on [1..7]?  3
fileserver:path matches:
1       fndaut:/t2/mengel/.login
Which file would you like to restore [1..1]?  1
About to execute:
  fmb_restore -w -L fndaub.23.C -t fncase:/dev/nrst8 fndaut:/t2/mengel/.login
This may overwrite any existing fndaut:/t2/mengel/.login are you sure? y
Please insert tape fndaub.23.C on drive /dev/nrst8 and press Enter

Thanks!
...
```

6.4 fmb_find

The fmb_find command will look in the current backup log [see 2.4] (if any) and print the tape label [see 2.8]s of the last backups to contain the files listed on its command line. It exits with a return code of 0 if it found a match, 1 otherwise. You can specify the -l log option [see 7.17] or the -V directory option to force a search in a particular logfile or verbose log directory.

See section 3.2.1 Restoring a Single file for an example.

Note that if only -l style logging is available, `fmb_find` [see 6.4] can only tell you that the archive that might contain the file is on a given tape, not whether the file is actually there.

However if -V style logging is enabled, you can use `fmb_locate` [see 6.3 to find the tape that actually contains the file.

It uses the following options:

- -e command option [see 7.13]
- -f file option [see 7.14]
- -l log option [see 7.17]
- -v option [see 7.25]
- -V directory option [see 7.26]
- pathname arguments

6.5 `fmb_max`

The `fmb_max` command will look at the sizes of the partitions of the data you have asked for, and gives you a total so that you can estimate if a backup will fit on a tape.

See section 4.2 Setting Up Nightly Backups for an exmple.

It uses the following options:

- -e command option [see 7.13]
- -f file option [see 7.14]
- pathname arguments

It assumes the "-x" option, with no override.

6.6 `fmb_toc`

The `fmb_toc` command will give a table of contents of a backup tape [see 2.3], listing either just the archives on the tape, as well as the date, etc. or a longer listing giving all of the individual files on the archives as well. It exits with a return code of 0 if it succeeds, 1 if it was unable to mount the tape, 2 if the label check failed, or 3 if there were errors reading the tape.

See the Seeing what is on a tape section for examples.

It uses the following options:

- -D drive option [see 7.3]
- -L label option [see 7.6]
- -e command option [see 7.13]
- -f file option [see 7.14]

- -k option [see 7.16]
- -m mount_type option [see 7.18]
- -n option [see 7.19]
- -t tapedrive option [see 7.23]
- -v option [see 7.25]
- pathname arguments

6.7 `fmb_restore`

The `fmb_restore` command allows files to be extracted from the archives on a backup tape [see 2.3]. The file requested is a filename qualified as `host:path` (or expanded into one by the option parser), this is then split by `fmb_restore` into an archive name, and a path relative to the archive name. (i.e. `host:archive/relative` is the name of the file) and the relative pathname file is extracted from that archive into the current directory, unless the -w option [see 7.27] is specified, in which case it goes into the directory it was backed up from. So if you are in the directory `/tmp` and you request to extract `fnsg01:/usr/lib/aliases` from a tape with an archive of `fnsg01:/usr` the file `lib/aliases` will be extracted from that archive, and end up as `/tmp/lib/aliases` on your current machine. It exits with a return code of 0 if it succeeds, 1 if it was unable to mount the tape, 2 if the label check failed, or 3 if there were errors reading the tape.

See Restoring Files [section
refrestoringfiles] for examples.

It uses the following options:

- -D drive option [see 7.3]
- -L label option [see 7.6]
- -e command option [see 7.13]
- -f file option [see 7.14]
- -k option [see 7.16]
- -m mount_type option [see 7.18]
- -n option [see 7.19]
- -t tapedrive option [see 7.23]
- -v option [see 7.25]
- -w option [see 7.27]
- pathname arguments

6.8 fmb_options

The fmb_options command lets you set default options for future backups, restores, etc. on this host. The options set with this command are kept in the file $FMB_DIR/fmb_defaults/hostname for your host.

It uses the following options:

- -a arch_format option [see 7.10]
- -b blocksize option [see 7.11]
- -e command option [see 7.13]
- -f file option [see 7.14]
- -h host option [see 7.15]
- -l log option [see 7.17]
- -m mount_type option [see 7.18]
- -n option [see 7.19]
- -r rotation option [see 7.21]
- -s set option [see 7.22]
- -t tapedrive option [see 7.23]
- -V directory option [see 7.26]

For example

```
# fmb_options
Current default options are:
-a cpio (archive format)
-b 16k (block size)
-l /dev/null (log file)
-m manual (mount mechanism)
-n /bin/cat (notify program for errors)
-r monthly (tape rotation scheme)
-s A (set part of label)
-t dcdmwm:/dev/rmt0.1 (tape drive)
-V  (verbose log directory)
# fmb_options -t dcdmwm:/dev/rmt1.1 -l /var/msgs/backups
Current default options are:
-a cpio (archive format)
-b 16k (block size)
-l /var/msgs/backups (log file)
-m manual (mount mechanism)
-n /bin/cat (notify program for errors)
```

```
-r monthly (tape rotation scheme)
-s A (set part of label)
-t dcdmwm:/dev/rmt1.1 (tape drive)
-V  (verbose log directory)
#
```

The first invocation simply prints the current defaults, the second changes the tape drive and log file.

6.9 fmb_prelabel

The `fmb_prelabel` command lets you put tape label [see 2.8]s on a set of tapes for your rotation scheme [see 2.7] and generate a PostScript file to print labels on 1/2" by 4" laser printer label stock.

It uses the following options:

- -D drive option [see 7.3]
- -L label option [see 7.6]
- -a arch_format option [see 7.10]
- -b blocksize option [see 7.11]
- -d day option [see 7.12]
- -e command option [see 7.13]
- -f file option [see 7.14]
- -h host option [see 7.15]
- -k option [see 7.16]
- -m mount_type option [see 7.18]
- -n option [see 7.19]
- -r rotation option [see 7.21]
- -s set option [see 7.22]
- -t tapedrive option [see 7.23]
- pathname arguments

Specifying "-L label" prelabels just one tape with that label, otherwise fmb_prelabel lists the tapes in a full tape set, and asks you which ones you want to write.

For example, to make a weekly rotation batch of set "B" backup tape [see 2.3]s for host "kremvax", you would:

```
# fmb_prelabel -t /dev/rmt/tps0d1nrnsv -h kremvax -s B -r weekly
Building label list...
1 kremvax.Sun.B   3 kremvax.Tue.B   5 kremvax.Thu.B   7 kremvax.Sat.B
2 kremvax.Mon.B   4 kremvax.Wed.B   6 kremvax.Fri.B   8 ALL
Enter number(s)/range(s) [i.e. '5..10'] of label set to build:
8
Expanding ranges...
Enter a file to put PostScript labels in (or just hit Enter for none):
kremvax.ps
Press enter when tapes are ready to mount

Mount tape kremvax.Sun.B on /dev/rmt/tps0d1nrnsv and press Enter

Thanks!
4+0 records in
0+1 records out
Unmount tape kremvax.Sun.B on /dev/rmt/tps0d1nrnsv
Mount tape kremvax.Mon.B on /dev/rmt/tps0d1nrnsv and press Enter

Thanks!
4+0 records in
0+1 records out
Unmount tape kremvax.Mon.B on /dev/rmt/tps0d1nrnsv
Mount tape kremvax.Tue.B on /dev/rmt/tps0d1nrnsv and press Enter

...
```

6.10 fmb_set

The fmb_set command takes as arguments three lists of file names. The files so named should all be files containing command line arguments for fmb_backup [see 6.2]. The first set of files should contain options specifying file tree [see 2.5]s to be backed up, and what to call the tapes. The second set of files should contain options specifying tape drives and mount scheme [see 2.6]s we want to write backups on. The third set of files should specify tape drives and mount scheme [see 2.6]s we can fall back on if mounts fail from the previous set of files. fmb_set will backup the requested file tree [see 2.5]s on the requested tapes on the listed drives on a next available drive basis. It exits with a return code of 0 if it succeeds, or 1 if any of the backups failed.

See the Error Recovery for Important Backups section for an example.

It uses the following options:

- -O option [see 7.7]
- -d day option [see 7.12]
- -r rotation option [see 7.21]

6.11 fmb_cp

The fmb_cp command does remote and/or local copies by building archives of the source directories, and unarchiving them in the indicated destination.

It uses the following options:

- -a arch_format option [see 7.10]
- -e command option [see 7.13]
- -f file option [see 7.14]
- -m mount_type option [see 7.18]
- -p option [see 7.20]
- -v option [see 7.25]
- pathname arguments

Note that fmb_cp will update data relating to partial/full backups, as well as doing "partial" copies.

As an example, we'll copy files from the ups directory to /tmp/junk, to show that file modes, dates, etc. are all automatically preserved. This can be useful for moving home directories, etc.

```
# ls -l ups
total 8
-rw-r--r--   1 root     system        0 Dec  8 11:31 .lastfull
-rwxr-xr-x   1 mengel   g150        201 Dec  7 13:13 current.csh
-rwxr-xr-x   1 mengel   g150         26 Dec  7 13:13 setup.csh
-r-xr-xr-x   1 mengel   g150         32 Jul  7 1992  setup.ksh
-rwxr-xr-x   1 mengel   g150         32 Dec  7 13:13 setup.sh
drwxrwxrwx   2 mengel   g150        512 Dec  7 13:13 toman
-rwxr-xr-x   1 mengel   g150        133 Dec  7 13:13 uncurrent.csh
-rw-r--r--   1 mengel   system      139 Nov  8 12:14 unsetup.csh
-rw-r--r--   1 mengel   system      202 Nov  8 12:17 unsetup.sh
# ls -l /tmp/junk
total 0
# fmb_cp ups /tmp/junk
Converting "ups" to "dcdsv0:/usr/people/mengel/src/fmb/ups"
Converting "/tmp/junk" to "dcdmwm:/tmp/junk"

30 blocks

Exit code 0
#
30 blocks

Exit code 0
```

```
# ls -l /tmp/junk
total 64
-rw-r--r--   1 root     system         0 Dec  8 11:34 .lastfull
-rwxr-xr-x   1 mengel   g150         201 Dec  7 13:13 current.csh
-rwxr-xr-x   1 mengel   g150          26 Dec  7 13:13 setup.csh
-r-xr-xr-x   1 mengel   g150          32 Jul  7 1992  setup.ksh
-rwxr-xr-x   1 mengel   g150          32 Dec  7 13:13 setup.sh
drwxrwxrwx   2 mengel   g150         512 Dec  7 13:13 toman
-rwxr-xr-x   1 mengel   g150         133 Dec  7 13:13 uncurrent.csh
-rw-r--r--   1 mengel   system       139 Nov  8 12:14 unsetup.csh
-rw-r--r--   1 mengel   system       202 Nov  8 12:17 unsetup.sh
#
```

6.12 fmb_verify

This command simply mounts a tape in the fashion of fmb_backup [see 6.2] and unmounts it, reporting any problems. It is used to make sure a backup tape [see 2.3] can be accessed later.

For example, to check to see if the right backup tape [see 2.3] is present for the `dcdmwm.A` backup options file, use:

```
# fmb_verify -f dcdmwm.A -k
Checking tape label [see \ref{tapelabel}]
fmb: error during fmb_verify on Wed Dec  8 11:40:22 CST 1993
Expected label "dcdmwm.8.A", got label "BackupTest"
```

This way you can check that a backup that runs in the middle of the night will have the tape it needs while it is convenient to do something about it.

It uses the following options:

- -D drive option [see 7.3]
- -L label option [see 7.6]
- -e command option [see 7.13]
- -f file option [see 7.14]
- -h host option [see 7.15]
- -k option [see 7.16]
- -m mount_type option [see 7.18]
- -n option [see 7.19]
- -r rotation option [see 7.21]
- -s set option [see 7.22]
- -t tapedrive option [see 7.23]

- -u option [see 7.24]
- pathname arguments

6.13 `fmb_subdirs`

This command gives a list of contents of the directory it is given as an argument. For example if there is an area `host1:/home` with subdirectories `fred` and `mary` `fmb_subdirs host1:/home` will give the list

```
host1:/home/fred
host1:/home/mary
```

on its output. This is especially useful when used with the `-e` option (see 7.13) of command `fmb_backup` (see 6.2) as in:

```
fmb_backup -e "fmb_subdirs host1:/home" ...
```

to backup the home directories in `host1:/home` separately.

7 Options

The options you can specify on all of the various Fermi Modular Backup command lines are the same, although some options have no effect on some commands (i.e specifiying a tape drive for the `fmb_cp` [see 6.11] command has no effect). The options, along with what internal shell variables they set, are described in the following section.

7.1 -A command option

This option specifies that archive errors be treated as unrecoverable, so no retry attempts are made on them.

7.2 -C command option

This option lets you specify a pre- and post-command for a particular volume (it affects the next file tree mentioned). The command will be invoked as
command `"before"`
before the volume is backed up, and as
command `"after"`
afterwards.

7.3 -D drive option

This is a synonym of the -t tape option [see 7.23], maintained for backwards compatability.

7.4 -E exception-pattern option

For backups using the cpio archive format (see 7.10), this lets you specify a pattern which will be given to `egrep -v` to match files that should not be backed up. For example:

`fmb_backup -a cpio -E 'core|\.netscape/cache' fred:/home`
will backup the /home area on host fred, but not any core files or netscape cache directories.

7.5 -F option

This lets you define a particular volume as being a full backup (it affects the next file tree mentioned). This means you can have several partial backups and one or two full on the same tape, for example.

7.6 -L label option

This option speceifies the label name for the tape, overriding the construction of a `host.day.set` format label in `fmb_backup` [see 6.2] and `fmb_verify` [see 6.12]. This sets the `label` environment variable. It is used by all of the fmb commands to specify a particular tape to mount, and is required when using "cps", "opcom", or "juke" mount types on `fmb_toc` [see 6.6], `fmb_restore` [see 6.7], or `fmb_prelabel` [see 6.9]. It overrides the -s set option [see 7.22], the -d day option [see 7.12], and -h host option [see 7.14].

7.7 -O option

This option specifies that `ocs_init_stats` and `ocs_report_stats` should be used to collect tape statistics on this operation, regardless whether OCS is used for tape mounts, and that OCS be consulted about whether tape drives are useable.

This option affects: `fmb_backup` [see 6.2]. `fmb_restore` [see 6.7]. `fmb_set` [see 6.10].

7.8 -P option

This lets you define a particular volume as being a partial backup (it affects the next file tree mentioned). This means you can have several full backups and one or two partials on the same tape, for example.

7.9 -R retries option

This sets the number of tries `fmb_backup` will retry in the face of archive or tape write errors. This option affects: `fmb_backup` [see 6.2].

7.10 -a arch_format option

This option specifies the archive tool to use to create the archive files in the backup. On restores this option will be overridden by the archive format[see 2.2] written in the tape header. This sets the `arch_format` shell environment variable. The default is "cpio". Currently "dump", "cpio", "tar", and "gnutar" can be specified. This option only affects `fmb_backup` [see 6.2] and `fmb_cp` [see 6.11]. `fmb_toc` [see 6.6] and `fmb_restore` [see 6.7] set it internally based on the tape label on the tape.

7.11 -b blocksize option

This option specifies the size of blocks to be written to the output tape drive. On restores this option is overridden by the blocksize written on the tape label [see 2.8]. This sets the `blk` shell environment variable. The blocksize may be any notation acceptable to the `dd` command (i.e. `1k` to specify 1024 bytes). This option only affects `fmb_backup` [see 6.2] and `fmb_cp` [see 6.11]. `fmb_toc` [see 6.6] and `fmb_restore` [see 6.7] set it based on the tape label [see 2.8] on the tape

7.12 -d day option

This option specifies the day name for construction of tape label [see 2.8]s. The default for backups is the current day from the roation scheme. This sets the `day` environment variable, which is used to build the `label` environment variable if neccesary. This option only affects `fmb_backup` [see 6.2] and `fmb_verify` [see 6.12], if "-L" isn't set.

7.13 -e command option

This option executes a command, whose output is substituted into the command line as options. This was done because putting grave-quoted strings in command files doesn't work as expected. This option works on all fmb commands. See the discussion of precedence in the precedence of options section.

7.14 -f file option

This option specifies a file containing command line options. The options in the file are inserted into the argument list at this point. The file may contain newlines an spaces to facilitate easier reading, and comments delimited by '#' and end of line. The file can be a full pathname, or a plain file name, which will be looked for in the `fmb_files` sub-directory in the backup product directory. This option works on all fmb commands. See the discussion of precedence in the precedence of options section.

7.15 -h host option

This option specifies the host name for construction of tape label [see 2.8]s. The default for backups is the hostname on which the script is invoked. This sets the `host` shell variable which is used to build the `label` environment variable if neccesary. This option is used by `fmb_backup` [see 6.2], `fmb_verify` [see 6.12], and `fmb_prelabel` [see 6.9] via the `host` environment variable.

7.16 -k option

This option tells fmb to keep the tape in the drive at the end of the operation, rather than unloading it, which is the default. `fmb_toc` [see 6.6] turns on -k when "manual" or "quiet" mounts are being used. This affects all fmb commands that use tapes (i.e. everything except fmb_max, `fmb_cp` [see 6.11], `fmb_find` [see 6.4], etc.) This option can be overridden by specifying

`"--k"`

.

7.17 -l log option

This sets the log directory to keep the record of what files are backed up on which tape, as well as where to find that log if needed.

This option sets the `log` shell environment variable. It is used by `fmb_backup` [see 6.2] and `fmb_find` [see 6.4]. It can be overridden by using `-l /dev/null`

7.18 -m mount_type option

This option specifies the mechanism to be used to request that a tape be placed in the tapedrive. This sets the `mount_type` shell environment variable, and specifies a mount script name. Mount types included in the currrent release are:

Manual The user is prompted at his/her terminal to insert the tape, and is allowed to fail the mount, etc.

manual The user is prompted at his/her terminal to insert the tape.

cps Issues a `cps_tapemount` to a cps operator station.

opcom issues an OPCOM tape mount request to the central VAXCluster.

quiet does nothing; assumes a tape is already in the drive.

summus Mounts a tape from a Summus Systems tape carousel.

juke Mounts a tape with any "juke" reachable jukebox. Must be invoked with a jukebox name, i.e. `"juke sga"` to refer to the jukebox named `sga`.

xterm Puts a mount request (via `Manual`) up on a named X display in an xterm window. Must be invoked with a display name, as in `"xterm fnop8:0"` to refer to an X terminal named "fnop8".

The mount type may take an argument (as with juke) to specify which jukebox, opcom host, etc. to use. This is specified like:

```
# fmb_backup -m "juke sg01" ...

# fmb_backup -m "opcom fndcd" ...
```

The "juke" mount type always requires a jukebox name argument. (see DCD Release Note 39 for more information on the "juke" package).

This affects all fmb commands that use tapes (i.e. everything except fmb_max, `fmb_cp` [see 6.11], `fmb_find` [see 6.4], etc.)

7.19 -n option

This option specifies a program that error messages are to be piped through. This could be a command that inserted formatting strings to make a printer print in highlight mode, or make a terminal print inverse video, etc. Another common use is the `fmb_mail` script, which mails the error to a specified user, but also sends it to standard output so it still ends up in the backup output. It is possible to put a pipeline of several commands, like:

```
# fmb_backup -n "tee -a /local/logfile | fmb_mail fred" ...
```

The default is "/bin/cat". `fmb_backup` [see 6.2], `fmb_restore` [see 6.7] and `fmb_toc` [see 6.6] use this command.

7.20 -p option

Requests that a partial backup be performed, rather than a full backup. This means a backup of all the files in the specified file tree [see 2.5] newer than the last full backup. Some archive programs (e.g. `tar`) don't know how to do partial backups, and will ignore this option. This option sets the `partial` environment variable to `true` rather than the default `false`. Note that the cpio archive type scripts maintain the "last backup" information by creating the file ".lastfull" in the directory being backed up. This option can be overridden with

```
"--p"
```

. Only `fmb_backup` [see 6.2] and `fmb_cp` [see 6.11] pay attention to this option.

7.21 -r rotation option

Specifies the rotation scheme [see 2.7] for tape label [see 2.8]s to be used. Sets the `rotation` shell environment variable, which specifies a rotation module name. Currently the following rotation schemes are available:

monthly This has a tape for each day of the month, as well as a first monday of the month tape for each month.

biweekly This has a tape for each day of a two week cycle, as well as a first monday of the month tape for each month.

weekly This has a tape for each day of the week.

improved_weekly This has a tape for Tue-Sun, a tape for the 2nd-5th Monday of the month, and a first Monday of the month tape for Jan-Dec.

daily This has an Odd and Even tape which alternate based on the day of year being odd or even.

table This lets you set up a table mapping date and set name to a particular label string. The table file sits in `$FTT_DIR/ftt_label/table/tablefile`, and has a line beginnging with `#sets` which provides column headings, and then lines with the date in month/day/year format in the first column, and tape labels in successive colums, one for each set listed on the `#sets` line, respectively. Other lines beginning with `#` are ignored. A sample table looks like:

```
#sets           A        B        C
#dates        ---------------------------
10/03/1996    tape01   tape02   tape03
10/04/1996    tape04   tape05   tape06
10/05/1996    tape07   tape08   tape07
10/06/1996    tape10   tape11   tape12
```

pool This lets you set up a "pool" of tape labels which will be cylcled through in a least-recently-used fashion. Tapes with more than three write errors are removed from the running. A sample pool file looks like:

```
#Label Errs Uses Date Host Set
#----- ---- ---- ---- ---- ---
tape003 4 4 09/13/1996 dcdsv0 B
tape002 0 7 09/12/1996 dcdsv0 A
tape001 0 8 09/12/1996 dcdsv0 A
tape004 1 8 09/12/1996 dcdsv0 A
tape000 0 8 09/12/1996 dcdsv0 A
```

This option is used by `fmb_backup` [see 6.2], `fmb_verify` [see 6.12], and `fmb_prelabel` [see 6.9], if "-L" isn't set.

7.22 -s set option

Specifes the set name for the tape label [see 2.8]. Sets the `set` environment variable, which is used to build the `label` environment variable if neccesary. This option is used by `fmb_backup` [see 6.2], `fmb_verify` [see 6.12], and `fmb_prelabel` [see 6.9], if "-L" isn't set.

7.23 -t tapedrive option

This option specifies the backup device to write the backup to, or read the backup from. The drive can be in the format `user@host:path` to specify that an `rsh host -l user` command be executed to refer to the drive, or as `host:path` to specify an `rsh host` command, or simply as `path` to refer to a tape drive on the current system. using this option sets serveral shell environment variables for the various scripts:

tape_rsh the `rsh` command to execute.

tape_path the pathname of the drive

tape_mt the `mt` command to use on that host (i.e. `mt -t` on IRIX systems, or `tctl -f` on AIX systems).

This affects all fmb commands that use tapes (i.e. everything except `fmb_max` [see 6.5], `fmb_cp` [see 6.11], `fmb_find` [see 6.4], etc.)

7.24 -u option

This option says that the tape to be backed up onto is unlabelled, and that a backup should be written on it anyway (i.e. it forces the normal check for a valid tape label [see 2.8] to pass.) This option is not reccomended for unattended backups. It sets the `unlabelled` shell environment variable to the value `true` instead of the default `false`.

This option can be overridden with

`"--u"`

. This affects `fmb_backup` [see 6.2], and `fmb_verify` [see 6.12].

7.25 -v option

Verbose mode, says that file names should be printed as they are archived, and that full archive listings should be given on the `fmb_toc` [see 6.6] command. Sets the `verbose` shell environment variable to the value `true` instead of the default `false`.

This option can be overridden with

`"--v"`

. This affects `fmb_backup` [see 6.2], `fmb_toc` [see 6.6], and `fmb_restore` [see 6.7].

7.26 -V directory option

This specifies a directory to keep verbose logs in. Verbose logs are files with tape-label filenames containing full lists of files backed up. The logfiles are often compressed.

This option can be overridden with `-V ""` This affects `fmb_backup` [see 6.2], and `fmb_find` [see 6.4].

7.27 -w option

Says to put files where they came from on restores, rather than under the current directory.

This option can be overridden with

`"--w"`

. This affects `fmb_restore` [see 6.7].

7.28 -x option

Requests that mount points not be crossed when doing the backup. Some archive modules (e.g. `tar`) ignore this option, while others (e.g. `dump`) require it. This option sets the `nomount` shell environment variable to the value `true` instead of the default `false`.

This option can be overridden with

`"--x"`

. This affects `fmb_backup` [see 6.2].

7.29 pathname arguments

These arguments specify either pathnames of file tree [see 2.5]s to be backed up, or a file or files to be restored. They can be specified as `host:path` or `user@host:path` to refer to remote hosts, and to force access to the remote host as a particular user. These arguments are collected up into a single environment variable named `path_list`, but are also split up into:

arch_rsh the rsh command to execute to execute on the host

arch_path the pathname of the file tree [see 2.5]

before the archive command for each archive in a backup is called.

This affects all the fmb commands, except `fmb_verify` [see 6.12], and specifies directories/files to be backed up or restored, or archives to give tables of contents of.

7.30 precedence of options

Options are read left to right, with options further along overriding preceding ones. For example `fmb_backup -f file -L mylabel` [see 6.2] will do the backup as specifed in the file `file`, with the exception that the label `mylabel` will be used for the tape label [see 2.8]. Options set by the `fmb_options` [see 6.8] command are seen before (i.e. to the left of) any command-line option [see 7.17]s. Any option specified to an fmb program will override defaults set by `fmb_options` [see 6.8].

8 Scripts

In the backup product directory, there are numerous scripts that are called by the command scripts users and administrators normally use. They rely on the environment variables mentioned in the Options section of this document to operate, and often will not work if run directly from the command line.

8.1 `fmb_arch/$arch_type/pack`

The pack script for a given archive type, it uses the `arch_rsh` and `arch_path` variables to build an archive that it presents on its standard output. It should, when possible, pay attention to the `nomount` and `partial` environment variables.

8.2 `fmb_arch/$arch_type/unpack`

The unpack script for a given archive type, it takes a pattern list as arguments describing the files to restore, and unpacks them in the current directory from the archive on its standard input.

8.3 `fmb_arch/$arch_type/toc`

The toc script for a given archive format[see 2.2]: it reads an archive on stdin, and generates a table of contents on stdout.

8.4 fmb_check

This script reads and checks the label file from the current tape using the `tape_rsh`, `tape_mt`, `tape_path`, and `label` environment variables. It takes an optional `-f file` argument to specify a file into which to copy the tape label [see 2.8] so that information may be read from the label file. It succeeds if the label on the tape matches `label`, or if `label` is the empty string, or if the `unlabelled` environment variable is set.

8.5 fmb_defaults

This directory contains shell text files setting the default values of the various environment variables set by the command line arguments on a given host. These files are named for the host they refer to, updated by the `fmb_options` [see 6.8] command and sourced by all of the user executed commands.

8.6 fmb_rotation/$rotation/today

The script to get today's name for a given rotation scheme [see 2.7]; it prints today's label on stdout.

8.7 fmb_rotation/$rotation/list

The script to get all day names for a given rotation scheme [see 2.7]; it prints the list on stdout.

8.8 fmb_mount/$mount_type

The script for a given mount type, it mounts the tape specified by the `label` environment variable on the `tape_path` tape drive. With the `-u` option, it unmounts that tape.

8.9 fmb_parse

A common script for parsing command line options for all of the Fermi Modular Backup commands.

8.10 fmb_write

A script that generates a properly formatted tape label [see 2.8] on stdout.