DL/CSE/TM40

technical memorandum Daresbury Laboratory

DL/CSE/TM40

UNIX FILE SYSTEM BACKUP AT DARESBURY LABORATORY

by

PAUL GRIFFITHS, SERC Daresbury Laboratory

DECEMBER, 1990

Science and Engineering Research Council

DARESBURY LABORATORY

Daresbury, Warrington WA4 4AD

ISSN 0144-5677

Technical Memorandum

UNIX file system backup at Daresbury Laboratory

P.Griffiths

Issue 1
24 October 1990

Abstract

This memorandum describes the procedures used at Daresbury Laboratory to backup a heterogeneous network of Unix based workstations and super computers. Detailed descriptions are given of the methods used to achieve automatic full and incremental backups of distributed file-systems onto tape using remote shells. Coverage is given to the set of support programs provided to interrogate backup cycles and ease the restoration of files, plus the integration of automated "Juke Box" devices into the backup mechanism.

Contents

List of Tables

List of Figures

1 Introduction

In November of 1987 Daresbury Laboratory bought its first Sun MicroSystems workstation; by June 1990 a total of 56 Sun workstations had arrived on site with every indication that the figure would continue to rise.

Because of the abundance of workstations, three Sun file servers, a Sparc 1, a 4/260 and a 4/370, were provided giving ~10.5Gbytes of online disk storage for workstation use. At roughly the same time several workstation/mini-supers arrived on site, each with their own disk drives and backup requirements. In total this gave us an estimated ~20Gbytes of disk. At this point we invested in an Exabyte tape drive to provide us with high capacity storage for disk backup. However, most computer manufactures provide backup mechanisms that assume the use of individual tapes per filesystem backup, thereby negating the advantage of the Exabyte. For this reason a backup program suite was developed incorporating the following features:

- Multiple backups on a single tape.
- Full and incremental backups freely intermixed.
- Executes separate, user configurable scripts for differing types of backup mechanisms.
- A single configuration file dictates the complete backup schedule.
- Automatic table of contents generation for backup tapes.
- Full support provided for JBL tape carousels.
- For ease of maintenance all programs written as shell scripts, also allows the backup software to be run on virtually any machine supporting `/bin/sh` and avoids the usual "recompile" problems when upgrading operating system software.

The system described here has been operating since September 1988, driven by a Sun 4/260 and writing to both single Exabyte unit and JBL enclosed drives. In the latter form it requires no human intervention in its day to day running.

Throughout this document the following notations and assumptions have been made:

- File pathnames, command names and shell procedures are signified using a `monospaced` font. For example, `rsh(1c)` indicates a unix command (in this case rsh); the notation `(1c)` is a reference to section 1c of the Unix manual pages accessible using the `man(1)` command.
- The reader has some prior knowledge of the Unix operating system.
- *fsd* is being used to backup a network of Unix machines.

2 Software

2.1 Overview

The backup program (*fsd*) provides a set of shell scripts to automate the issuing of backup commands from a central server to clients on a network using `rsh`, output being piped back to a local tape drive. When started *fsd* first creates a log file for the backup, followed by a tape initialisation sequence where a tape is requested and then checked for the correct label. Having completed its initialisation phase *fsd* next uses a file-system schedule database to work out which file-systems are to be backed up, when and on what machines. Once *fsd* has worked out its schedule backing up commences. *System specific* backup commands are issued using a set of scripts supplied in the release. Each script is monitored by *fsd* for successful completion before moving on to the next file-system. Backups are written in a sequential fashion onto the backup tape so that a single tape is sufficient to hold multiple backups from a collection of networked machines. When all the required backups have been completed *fsd* rewinds and unloads the tape, closes the logfile and terminates.

fsd is also capable of generating a TableOfContents database for all backups done using the utility `dump(8)`. Before unloading the tape *fsd* checks to see if a TableOfContents is required from it, and if so rereads the tape extracting a table of contents for each backup, appending each to a master database.

fsd is written to take advantage of the large storage capacity (2Gbytes) of Exabyte tape drives — using *fsd* on devices with only limited storage capacity really defeats its whole philosophy, namely that of packing a large number of backups onto a single tape in a well structured fashion.

2.2 Tags

One of the key features of *fsd* is the concept of a *tag*. Tags are used by *fsd* as a means of separating groups of backups (or rather file-systems) onto different sets of tapes. A tag may be viewed as a string used in filename construction in reference to a backup group. The following names are generated using tags:

${HOME}/scripts/*tag*.h	Header file containing tag dependant variables.
${HOME}/scripts/*tag*.exclude	Clients to exclude from this backup.
${HOME}/scripts/*tag*.file-db	The file-system database for this tag.
${HOME}/scripts/*tag*.pro-file-db	The file-system is processed daily and the results placed in this file.
${HOME}/scripts/*tag*.lock	The lock file created when *fsd* is running.
${HOME}/scripts/*tag*.lastcc	A temporary storage area for the return code of a backup.

Table 1 Tag dependent names (Continued ...)

${HOME}/logs/*tag*/	The directory where the log files for this tag are kept.
tag.tape-number	Tape volume label.

Table 1 Tag dependent names

fsd uses the hostname of the machine it is running on as a default tag, so if the hostname is merlin *fsd* uses the files:

${HOME}/scripts/merlin.h
${HOME}/scripts/merlin.exclude
${HOME}/scripts/merlin.file-db
${HOME}/scripts/merlin.pro-file-db
${HOME}/scripts/merlin.lock
${HOME}/logs/merlin/

and tapes with labels:

merlin.*tapenumber*

fsd uses its *first* argument to decide which tag to use — if this argument does not correspond to an installed tag then the default one is used. From the above example entering `fsd` causes the tag merlin to be used, however, entering `fsd gandalf` instructs *fsd* to use the tag gandalf. The advantage of tags becomes apparent when your daily backup requirement exceeds the storage limit of the backup media available thus making it desirable to split backups between two or more sets of tapes. Which backups are done to what tape is entirely dependent on the information you put in the file-system database for the relevant tag (see section 3.5 for a complete description of the file-system database).

Tags are created using the install script provided in the release — see section 3 for full details of the install script.

3 Installing fsd

3.1 Create a user-id

Before beginning installation it is necessary to create a user-id for the use of *fsd*. The reasons are twofold:

- To provide a permanent home for the sources and runtime created files.
- To provide *fsd* with a UID which can be used to grant access to raw devices both locally and on client machines. Obviously a common UID across all machines provides uniformity to the backup mechanism.

For example Daresbury uses the login id oper, UID of 10, group Operator with the raw filesystems in `/dev` having Operator group read permission.

Once a user-id is created set the PATH variable to be:

Bourne shell	C Shell
`PATH=${HOME}/bin:.:${PATH}`	`set path = (~/bin . $path)`

In both cases logout and login again so the new paths take effect.

3.2 Unpack fsd

The *fsd* distribution is in `tar(1)` format on Exabyte tape. To unpack it, login under the user-id created for *fsd* and enter the command:

```
tar xvf tape-drive
```

Here replace *tape-drive* with the drive where the release tape is mounted. Tar will unpack the following files:

Program/File Name	Use
`./fsd-install`	The installation script for *fsd*.
`./fsd.ps`	PostScript copy of this document.
`./bin/fsd`	The main backup program.
`./bin/fsdat`	Update the TableOfContents database from the contents of a backup tape.
`./bin/fsded`	Edit the file-system schedule database.
`./bin/fsdex`	Edit the exclude file.
`./bin/fsdlb`	Show what backups will be performed today.
`./bin/fsdrl`	Read the label on a backup tape.
`./bin/fsdtl`	Label a tape for use in the backup cycle.
`./bin/fsdts`	Scan a tape and report what's on it..

Table 2 Program Suite (Continued . . .)

`./bin/fsq`	Restore files from backup.
`./bin/fsr`	An interrogation program for the TableOfContents database; will also restore files using *fsg*.
`./scripts/shared.h`	A header file shared by all programs.
`./scripts/fsd.macros`	A collection Bourne shell macros loaded by *fsd* each time it runs - see the `sh(1)` man page for a complete description of macros.
`./scripts/BckScripts/*.script`	There are several similarly named files which perform the actual command required to backup a system. For example, SGIs are supplied with a backup facility known as bru, so there will be the script `./scripts/BckScripts/bru.script` containing the necessary commands to perform a bru backup locally or remotely.
`./scripts/awkscripts/awk.*`	Frequently used awk scripts.

Table 2 Program Suite

3.3 Install fsd

To install *fsd* enter the command `fsd-install` at the shell prompt. You will be presented with the following menu:

```
FSD Installation Script Version X.X

1) First time installation of fsd.
2) Install a user defined tag.
3) Quit

Please enter a choice from 1-3:
```

When installing *fsd* from scratch chose option 1 to create the default set of tag files. Option 2 allows you to create you own tag (for which you will be prompted) which can then be used as the *first* argument to *fsd* when started. Note! *fsd* must have been installed using option 1 before a user defined tag can be created using option 2.

Once a tag has been chosen ***fsd-install*** creates tag dependant files:

```
Created:  "${HOME}/dates/"
Created:  "${HOME}/logs/"
Created:  "${HOME}/logs/tag/"
Created:  "${HOME}/scripts/tag.file-db"
Created:  "${HOME}/scripts/tag.exclude"
```

If you are not running *fsd-install* for the first time some of the above messages will not appear.

Next *fsd-install* asks if you want to configure a JBL carousel for use in the backup mechanism:

```
Configure a JBL carousel into the cycle (y/n default n) :
```

Answering "n" continues with the next stage and leaves the JBL unconfigured for this tag — explanation of configuring the JBL (answering "y") is given in section 3.4. *fsd-install* now asks for the tape device to use:

```
Please enter the tape device fsd will write to.

Note!this must be a NO-REWIND RAW device
e.g.  "/dev/nrst1" (default /dev/nrst1) :/dev/
```

fsd-install offers completion of `/dev/`. The device chosen in this stage can be overridden at any time by setting the shell environment variable DEVICE before running *fsd*.

Next you are given the option to configure TableOfContents database generation for this tag — see section 3.8 for a description of the TableOfContents database. Answer `y` or `n` depending upon your preference:

```
Do you want to append to the TableOfContents database
at the end of each run (y/n default n) :
```

If you answered `y` to the TableOfContents database option *fsd-install* will now request the name of a directory it can use as temporary space when sorting the TableOfContents database. This directory must be writable by *fsd* and be of a reasonable size (~20Mbytes):

```
Please enter the name of a directory that fsd can use
as temporary sort space for the TableOfContents database
e.g.   /tmp :
```

The next stage of the install sets a timeout value in minutes for *fsd* to use when waiting for individual backups to complete. If this timeout value is reached *fsd* issues a kill signal to the child process running the backup. Doing this prevents backups that have failed and that are hung indefinitely causing *fsd* itself to hang. The value of 2 hours works well for the Sun backups at Daresbury. Consideration should also be given to backups that may run for more than 2 hours; for example backing up an Ardent Titan using `cpio` has been seen to run over 4 hours when the machine was heavily loaded! This is really a case for on-site tuning:


```
Please enter the timeout value in minutes before fsd
will kill a backup process (default 120 minutes) :
```

Lastly *fsd-install* asks for the electronic mail id's of users wanting to be informed of successful completion of a backup, or when errors occur:

```
Please enter the USER-ID of the administrator.  This
ID will receive messages from fsd on successful
completion of a backup, or when things go wrong.
You may enter a comma separated list of users for
multiple recipients e.g. "pg, root, daemon", or just
a single user as in "root".
Enter list (default "root"):
```

fsd-install now produces a completion message and exits:

```
Installation complete, to run enter:  "fsd [tag]"

You must now enter data into the file-system
database using fsded before running
any backups.
```

Once *fsd-install* has completed set the access permissions of the tape device to enable *fsd* to read and write to it; the following example assumes *fsd* is in group `operator` and that the tape device chosen is `/dev/nrst1` (this may have to be done as `root`):

```
chgrp operator /dev/nrst1
chmod g+rw /dev/nrst1
```

This completes the installation of *fsd*. It is now necessary to inform *fsd* which file-systems you want backed up before any attempt to use it is made. To do this add entries to the file-system database using the program *fsded*. When calling up an editor on the file-system database *fsded* first checks to see if the environment variable EDITOR is set, and if so uses that editor; if it isn't set `vi(1)` is used — see section 3.5 for a complete description of the file-system database. *fsded* will take a tag as its first argument and edit the appropriate database.

3.4 Configuring a JBL carousel into fsd

fsd supports the use of a Summus Juke Box Library (JBL) carousel. Successful configuration into the backup mechanism is achieved with the following steps:

- Install the `jbl(1)` utility in a directory appearing in *fsd's* PATH variable.
- Run *fsd-install*.
- Answer "y" when *fsd-install* asks if you want to configure the JBL carousel.
- *fsd-install* now asks for the JBL control port. Enter the device name to which the JBL control port is attached, e.g. `/dev/ttyb`:

```
Please enter the JBL control port (default /dev/ttyb)   /dev/
```

- *fsd-install* will now ask you to choose either JBL drive a or b — this is localised to the current *fsd* tag:

```
Please enter the drive number on the JBL to use (a/b)
```

- *fsd-install* will now reset the JBL tape number to 1. This means that when next run *fsd* will use the tape in slot 1 of the JBL; upon completion of each run *fsd* increments this counter. Running *fsd-install* resets this counter to 1:

```
JBL tape number reset to 1.
```

- When *fsd-install* asks for the tape device to use enter the name of the no-rewind raw device configured to the JBL drive chosen in the previous stage. No-rewind raw devices are found in the `/dev` directory with a `nr` prefix, e.g. `/dev/nrst1`.
- When *fsd-install* completes set the access permissions on the JBL control port to enable *fsd* to read and write to it. The following example assumes the control port chosen was `/dev/ttya` and that *fsd* runs under the user-id operator (note that only root can execute `chown`):

```
chown operator /dev/ttya
chmod 600 /dev/ttya
```

Once all these steps have been taken JBL configuration is complete. Note that it is perfectly legal to configure different tags to use different JBLs if you have more that one.

3.5 The file-system database

The file-system database is a sequentially organised file used by *fsd* to describe which file-systems are to be backed up, when, and at what level. Each entry in the file-system database specifies a day, a client machine, a file-system, a dump type and a dump level. Lines that start with a # (comment) are ignored. Fields are separated with one or more spaces or tabs. Specifically each field has the following properties:

DAY	Specifies the day which this entry refers to — in the range 1–14. If a carousel is configured days are in the range of 1–50. The * character is a simple wildcard matching all days.
MACHINE	The hostname of the cloient where the file-system physically resides.
FILE-SYSTEM	The file-system to be backed up. If using `dump(8)` file-system names are given as raw devices e.g. /dev/rsd0. Most other backup packages use mount points rather than devices, e.g. /usr.
DUMP TYPE	The type of backup package to use. More specifically the name refers to a script in the directory `./scripts/BckScripts`
DUMP LEVEL	The level of backup to do. Level 9 is the lowest incremental backup possible; level 0 is a full backup.

The day field has different meaning depending upon whether on not you are using a JBL carousel unit. See section 4.1 for a description of how the day field is used.

Here is an example database:

Day	Machine	File-System	Dump Type	Dump Level
*	dlsa	/dev/rsd0a	dump	9
*	dlsa	/dev/rsd0b	dump	9
*	dlsb	/dev/rxd0a	dump	9
*	dlsb	/dev/rxd0b	dump	9
*	dlsb	/dev/rxd0c	dump	9
1	dlsa	/dev/rsd0a	dump	0
1	dlsb	/dev/rxd0b	dump	5
2	dlsa	/dev/rsd0b	dump	0

An `awk(1)` script sequentially processes the database producing an output file containing records describing the backups to be done today.

An entry in the file system database with a day field of "*" causes a backup record to be created, but this entry will be overridden by any *later* entries encountered with a corresponding day/host/file-system tuple. Processing the above file for day 1 will produce the following output:

Client	File System	Dump Type	Dump Level
dlsa	/dev/rsd0a	dump	0
dlsa	/dev/rsd0b	dump	9
dlsb	/dev/rxd0a	dump	9
dlsb	/dev/rxd0b	dump	5
dlsb	/dev/rxd0c	dump	9

And similarly for day 2:

Client	File-System	Dump Type	Dump Level
dlsa	/dev/rsd0a	dump	9
dlsa	/dev/rsd0b	dump	0
dlsb	/dev/rxd0a	dump	9
dlsb	/dev/rxd0b	dump	9
dlsb	/dev/rxd0c	dump	9

Observe that the level 5 and level 0 entries have overridden the original level 9 entries.

This style of processing enables new file systems to be added quickly and makes for a less complicated structure.

3.6 Configuring clients for fsd

Because *fsd* uses `rsh` exclusively when talking to clients some preliminary ground work must be done before the mechanism will work satisfactorily. Client configuration is accomplished by setting up a user-id for *fsd* to use when issuing remote shell requests to the client. This is a system dependent procedure and probably requires "root" access, however, as a rough guide:

- Create a password entry in the `/etc/passwd` file on the client. Interactive logins are not required and should be disabled by entering a '*' in the password field. The login name *must* be the same as that used by *fsd* in order for `rsh` to work correctly.
- Create a working directory for the user-id and change its ownership to that of the user-id. In this directory create a `.rhosts` file containing the one line "*fsd-host user-id*", substituting *fsd-host* with the name of the host running *fsd* and *user-id* with the user-id created for *fsd* on the host machine.
- The UID and GID should be set to some value which allows suitable (read) access to all file systems to be dumped; on Suns use a UID of 10, a GID of 5 (operator group) and grant operator group read access on the *raw* file systems (/dev/*). Most other machines will use a dump package which accesses files on an pathname basis and as such the only realistic option is to give the id root access (set the UID field to 0). *Remember to disable logins!*
- Add the host name and internet address of the client in the `/etc/hosts` file of the host running *fsd*, the yellow pages host database, the Internet domain name database, or some combination of the three.
- Again on the *fsd* host, in the `.rhosts` file located in the home directory of the *fsd* user-id, add the line "*client-host-name user-id*", substituting *client-host-name* with the host name of the client machine and *user-id* with the remote login name on the client machine. This entry is required by both `rsh` and `rmt`, *omitting it causes failure*.
- If any clients want to make use of `rdump` or `gnutar` via `rmt` you should ensure they are configured into the tape servers services list.

As an example *fsd* at Daresbury is set up in the following way:
The password entry on the *fsd* host (dlsa) is:

```
oper:password:10:5:Backups:/usr/etc/operdump:/bin/csh
```

On a Sun client the password entry is:

```
oper:*:10:5:Remote backups:/etc/operdump:/bin/csh
```

(group 5 is operator group).
Assuming that the *fsd* host is dlsa the `.rhost` file on the client contains the one line:

```
dlsa oper
```

Assuming that the clients host name is merlin the `.rhost` file on the *fsd* host includes the line:

```
merlin oper
```

On the client, devices in `/dev` have the permissions (for disk sd0):

```
brw-r----- 1 root operator Mar 3 1989 rsd0a
brw-r----- 1 root operator Mar 3 1989 rsd0b
brw-r----- 1 root operator Mar 3 1989 rsd0c
brw-r----- 1 root operator Mar 3 1989 rsd0d
brw-r----- 1 root operator Mar 3 1989 rsd0e
brw-r----- 1 root operator Mar 3 1989 rsd0f
```

thereby granting read access on raw devices to all those in the group operator. To set the permissions use the commands:

```
chgrp operator /dev/rsd0a...
chmod g+r /dev/rsd0a...
```

3.7 Setting the PATH

In distributed form *fsd* comes with a PATH of:

```
/etc:/usr/etc:/usr/ucb:/bin:/usr/bin:/usr/local/bin:
/usr/lbin:${HOME}:${HOME}/bin:.
```

This PATH is defined in `${HOME}/scripts/shared.h` and should be modified to reflect local conventions.

3.8 The TableOfContents database

The TableOfContents database is a set of files describing the current contents of backup tapes. If the TableOfContents option is configured at install time *fsd* will extract a table of contents from each backup done using dump and append each to the database. Once the whole tape has been read the TableOfContents is processed to remove redundant entries. Only the most recent copy of each file is recorded — earlier versions must be located via knowledge of the backup cycle in use.

If the TableOfContents database option has been configured you must have a directory named "`db`" (which may be a symbolic link) in *fsd's* home directory pointing to an area on disk large enough to hold it — at Daresbury a database size of ~20Mbytes is generated backing up ~56 file-systems. The database size will greatly depend upon the amount of files subject to backup and their frequency of modification. All references to the database should be made via this link.

The database provides an easy method of viewing the current contents of backup tapes. The program *fsr* is provided to query the database — see section 5.

Note! this database is not required for file retrieval — it just makes things a lot easier.

3.9 New tapes

For identification *fsd* writes volume labels on the tapes it uses. It therefore expects any tape it uses to have been first initialised with a volume label. When using new tapes you can either:

a. Label tapes using the *fsdtl* program supplied. *fsdtl* will take a tag as a first option and label tapes accordingly — when no tag is supplied the default one is used. If a JBL is configured use the `-jbl` option to label the complete set of tapes in one go.
b. Use the `-vt` (virgin tape) switch when starting *fsd* — see section 4.3 for a description of *fsd* options.

3.10 Running fsd

Operationally the backup procedure is very simple and requires only the minimum of manual intervention, that of mounting a tape in the Exabyte drive before running *fsd*. If a JBL unit is configured loading of the backup tape is done automatically by *fsd* allowing it to run completely autonomously.

fsd may be started in one of two ways:

1. Manually by entering the command *fsd* at the shell prompt.
2. Automatically using `cron(1)`. The crontab entry "30 23 * * * (fsd -b)" will start *fsd* daily at 23:30.

In both cases an initialisation phase is undertaken to check whether today's backup has been run, and if so *fsd* will terminate immediately. This avoids problems arising from running *fsd* manually during the day and also having cron run it in the evening.

4 Using fsd

4.1 Tape cycling

There are two approaches taken by *fsd* when cycling tapes. The first is a 14 day cycle used when JBL has not been configured. The current day of the month is taken and reduced to a number in the range 1–14 using the formula "whilst (day > 14) day = day - 14". In this respect 30 iterates down to 2 etc.

JBL configuration into *fsd* causes a different approach to tape cycling. Here 50 tapes in the carousel are used in a round robin manner starting from tape 1 working upwards. Once tape 50 has been used the cycling starts again from tape 1. The tape number is only incremented upon the successful completion of *fsd*, if *fsd* fails to complete the same tape will be used again when *fsd* is next run.[1]

The above JBL cycling mechanism leaves tapes 51–54 unused. These 4 tapes are being reserved for a future release of *fsd* which will incorporate automatic tape head cleaning.

4.2 Temporarily excluding clients from fsd

The exclude file, defined as `${HOME}/scripts/tag.exclude`, is checked by *fsd* to see if it contains the hostname of the machine about to be backed up. If a correspondence is found then the dump is skipped and *fsd* continues. Host names appear in the exclude file one per line and should correspond to those given in the file-system database. Removing an entry is sufficient to reinstate a client in *fsd's* cycle.

To edit the exclude file for a *fsd* tag use the utility *fsdex*.

4.3 Fsd options

fsd accepts the following options:

-b
Run in the background (batch mode). Any problems normally requiring user interaction will cause *fsd* to exit condition code 3.

-dXX
Set the backup day to *XX*, (-d2 and -d02 are equivalent). If a JBL is configured XX refers to a slot number in the range 1–50.

-ffilename
Use *filename* as the file-system database instead of `${HOME}/scripts/tag.file-db`. This option can be used when a non-standard backup is required. For example, under some conditions you may wish to dump a single disk if there were signs that a crash was imminent.

-h
Give help on using *fsd*.

[1] See the Bugs section for a note on why two differing algorithms are used.

-lfilename
Use *filename* as the log file. Note this is a filename not a pathname and that the logfile is created in the tag specific log directory.

-ns
Do not sort the TableOfContents database. Useful for example when disk space is running low or when several *fsd's* are to be run back-to-back and thus only the last one need sort the database.

-notoc
Do not add to the TableOfContents database during this run.

-nojbl
Do not use the JBL carousel unit. If you want to use a tape drive external to the JBL this option will suppress the issuing of JBL commands. Use in conjunction with the *-D* option.

-ctape-label
Expect a tape whose volume label is *tape-label*. Stops *fsd* complaining that the wrong tape is mounted.

-vt
The tape in the tape drive is blank (virgin tape) — do not check the volume label just write a new one.

-Ddrive
Use *drive* as the tape device to write to instead of the one specified when *fsd* was installed. *drive* must be a fully qualified device name.

The following example will backup the file-systems listed in disk5.db, output to disk5.log, label the blank tape as disk5, use device /dev/nrst1, skip the TOC generation and run without issuing JBL commands, useful for example if disk5 is thinking of taking up smoking:

```
fsd -fdisk5.db -ldisk5.log -vt -tdisk5 -D/dev/nrst1\
-notoc -nojbl
```

4.4 The fsd program

This section describes the steps taken by the *fsd* program in order to perform its task.

The convention **[macro(xxx)]**[2] signifies that the task being undertaken is a macro loaded from `${HOME}/scripts/fsd.macros`.

1. Set version number, execute cd to home, clear the screen.
2. Load up the shell macros from the file:

   `${HOME}/scripts/fsd.macros`

[2] See the *sh* (1) manual page for a description of shell macros

3. Source the header file which defines variables common to all tags:

   `${HOME}/scripts/shared.h`

   [macro(Include_header_file)]
4. Source the header file specific to the current backup tag:

   `${HOME}/scripts/tag.h`

   [macro(Include_header_file)]
5. Check for the existence of a lock file — if found it will be reported to the user and an option to terminate given. In batch mode *fsd* will exit code 3. **[macro(Check_lock_file)]**
6. Set software traps for signals **sighup**, **sigint**, and **sigterm** to terminate gracefully. **sigquit** causes *fsd* to terminate immediately.
7. Evaluate arguments, see section 4.3.
8. Check to see if a logfile for today exists, and if so terminate. Logfile names are generated from a combination of the current day and month numerals. If today is the 7th March 1990 the logfile created will be 0307.log. At the end of each backup logfiles are compressed using `compress(1)` — if a compressed logfile for today exists we can assume today's backups have been run successfully. If a carousel is configured logfile name generation will be *XX*.log where *XX* refers to the slot number used in the carousel.
9. Move *uncompressed* logfiles to numbered backups. If today's logfile is 0507.log, and there exists the logfile 0507.log we can assume it comes from an uncompleted run, in which case it is renamed to 0507.log.0 — if 0507.log.0 exists rename it to 0507.log.1 etc. Uncompressed logfiles exist from previously run backups that fail, so renaming rather than deleting them provides a more accurate audit trail. **[macro(Check_old_logs)]**
10. Delete any log files corresponding to this day but *not* to this month. If today's logfile is 0608.log and 0508.log.Z exists (last month's logfile for the 8th in compressed form) then remove it. Note here that it is only when a tape is actually reused will any log files relating to it be deleted — useful in preserving log files where a 30 day month follows a 31 day month. **[macro(Open_logfile)]**
11. Check the tape drive for a ready condition. If the tape drive is usable, but no tape is loaded, *fsd* will wait till one is mounted (`"please mount a tape"` message is echoed to stdout). **[macro(Check_tape_drive)]**
12. Read the tape volume header into the variable LABEL and set the variable TAPENO to the volume serial number of the tape. **[macro(Read_tape_vol)]**
13. Check to see that TAPENO corresponds to the tape label expected — *tag.day*, where day is the day of the month. If the wrong tape is loaded it will be ejected and the user given the choice of either mounting a new tape or exiting *fsd*. If a new tape is mounted then stages 11 and 12 will be re-iterated, otherwise *fsd* will terminate with exit code of 0. In batch mode *fsd* will terminate with exit code 3 upon detection of the wrong tape. **[macro(Check_tape_volume)]**
14. Write the complete tape header to the log file and stdout, information is extracted from the LABEL variable set in stage 12. **[macro(Print_tapevol_header)]**

15. Write a new header to the tape. See section 6.2 for tape header formats. **[macro(Write_new_header)]**
16. Process the file-system database to produce a backup recipe for today; the file-system database is covered in detail in section 3.5. **[macro(Process_filesystem_database)]**
17. Execute the individual dumps. The following cycle is executed for each entry produced in stage 16:

    a. Read in the next line from the processed file-system database.
    b. Check the host name to see if it exists in the exclude file and if so loop back to stage (a) — see section 4.2 for a description of how to exclude clients from the backup mechanism. **[macro(Check_exclude_file)]**
    c. Write a backup header to the tape. **[macro(Write_dump_header)]**
    d. Execute the file `./scripts/RunChild` as a *background* process. The variable DUMP_TYPE, set in stage 17.a, is passed to RunChild to indicate the type of backup required. To perform the actual backup RunChild executes the script `./scripts/${DUMP_TYPE}.script`. *fsd* now sits in a timed loop watching the RunChild process; one of two things will happen:

       - The process completes naturally.
       - The process times out causing *fsd* to abort it using signal 3; `kill(1)` is used to send the signal.

       In both cases *fsd* continues with the next dump. **[macro(Run_monitor_child)]**

18. Write an end-of-data marker. **[macro(Write_eof)]**
19. If configured, add today's dumps to the TableOfContents database — applicable only to backups done using dump. **[macro(Do_toc_processing)]**
20. Compress the logfile. **[macro(Close_logfile)]**
21. Exit gracefully.

4.5 Dump scripts

Dump scripts vary according to which backup procedures they are driving, so the information covered here can only be a generalisation. However most scripts follow the same rules which, if adhered to, should ease their writing.

Each script is called with the following arguments:

1. Client where file-system physically resides.
2. File system to dump.
3. Level of dump.
4. Logfile name.
5. Device name to dump to.

Dump levels are defined as 0 being a full dump and 9 the lowest level incremental dump. Where you do not have a choice of incremental levels map 1–9 as just incremental. Dumps of networked machines use `rsh(1)` as the means of issuing remote commands. All scripts should source the common header file:

```
${HOME}/scripts/shared.h
```


The following example should act as good template for anyone wishing to write their own script:

```
#!/bin/sh
#
# Script for dump.
#
# Called with arguments CLIENT FILE-SYSTEM LEVEL LOGFILE DEVICE.
#
# Note the nasty frig necessary to capture the exit code of remote dumps,
# watch out for this one as the only exit code caught locally is from rsh(1).
#
. ${HOME}/scripts/shared.h

DUMP_FLAGS="ucbsdf"
LIST_FLAGS="M"

#
# For clarity strip out the arguments.
#
CLIENT=$1
FILESYS=$2
LEVEL=$3
LOGFILE=$4
DEVICE=$5

#
# The dump flags "56 5190 4100000" are relevant to a 2Gbyte Exabyte
# on SunOS 4.0.X.
# A block size of 56 is used in ALL scripts - Exabytes will pad out
# blocks to be a multiple of 8K thus a block size of 56 prevents tape
# wastage, plus 64K may fall foul of the max blockage on Sun SCSI.
#

if (test ${CLIENT} = ${HOST})
then
        #
        # Local backup.
        #
        echo "Backup procedure for ${FILESYS} is now commencing" | wall
        dump ${LEVEL}${DUMP_FLAGS} 56 5190 4100000 ${DEVICE} ${FILESYS}
        STATUS=$?
else
        #
        # Remote backup.
        #
        rsh -n ${HOST}\
          "echo Backup procedure for ${HOST}:${FILESYS} is now commencing\
            | wall"
        rsh -n ${HOST}\
          "( /etc/dump ${LEVEL}${DUMP_FLAGS} 56 5190 4100000 ${HOST}:${DEVICE}\
            ${FILESYS}; echo $? > /tmp/last_CC )"
        STATUS=`rsh -n ${HOST} "cat /tmp/last_CC"`
fi

#
# Log the effort.
#
echo "*** dump ended with Condition Code of ${STATUS}." >>${LOGFILE}

#
# Save the exit code in a file - this will be
# picked up by the parent and passed on to fsd.
#
echo ${STATUS} > ${CC_SAVE_FILE}

#
# Exit with dump status.
#
exit ${STATUS}
```

Supplied with this release are scripts to drive the following backup mechanisms:

Name	Description
dump	Suns and most BSD derived systems.
cpio	System V architectures.
bru	Silicon Graphics.
mac	Apple macs running A/UX.

Table 3 Supplied backup scripts (Continued ...)

wbak	Apollos.
tar	Any systems having to resort to tar for their backup requirements.
GNU-tar	GNU's version of tar which incorporates an incremental backup mechanism - preferable over tar.

Table 3 Supplied backup scripts

All scripts will be found in the directory `./scripts/BckScripts/` with the extension `.script` appended to the name they are referenced by in the file-system database. See the relevant man pages for documentation.

Some backup mechanisms require a previous backup date to be supplied to them. For this purpose the directory `${HOME}/dates/` is created by *fsd-install* and used by some scripts as a common save area for dump dates (see `${HOME}/scripts/BckScripts/bru.script`).

4.6 Environment variables

The following table lists the environment variables used by the *fsd* suite:

Variable	Description	Programs using variable	Default
DEVICE	Defines the tape device used for backup/restore operations.	fsdat, fsded, fsdex, fsdrl, fsdtl, fsdts. fse.	As defined when *fsd* was installed
TERMTYPE	Terminal type output is being displayed upon. Used to clear screen etc.	All programs. *Note! fsded, fsdex will fail if TERMTYPE not set.*	No default
PATH	For finding executables.	All programs.	Existing path
EDITOR	The editor to use when cahnges have to be made to tables.	fsdex, fsded.	vi

Table 4 Environment variables

5 Support programs

[*tag*] indicates that the program being described will take an optional tag as the first argument.

5.1 fsdat(l)

fsdat (*fsd* add table of contents) will read a backup tape and update the TableOfContents database from the contents of the tape. This is usually only needed when *fsd* fails to do the job, or to recreate the database from scratch.

5.2 fsded(l) [*tag*]

fsded (*fsd* edit database) is used to call up an editor on the file-system database. The default editor is `vi(1)`; this can be overridden by setting the environment variable EDITOR to the name of a preferred editor.

5.3 fsdex(l) [*tag*]

fsdex (*fsd* edit exclude) is used to call up an editor on the exclude file. The default editor is `vi(1)`; this can be overridden by setting the environment variable EDITOR to the name of a preferred editor.

5.4 fsdlb(l) [*tag*]

fsdlb (*fsd* list backups) reports which file systems will be backed up today, arguments:

-dxx
Show the backups to be done on day *xx*.

5.5 fsdrl(l)

fsdrl (*fsd* read label) reads and reports the label on an *fsd* tape mounted in the tape drive.

5.6 fsdtl(l) [*tag*]

fsdtl (*fsd* tape label) labels tapes ready for use by *fsd*. A tape label takes the form of *tag.day*, where *tag* represents the default backup tag and *day* is the current day of the month. If a JBL is being used the carousel slot number is used instead of *day*, arguments:

-dday
Use *day* as the day field in the label. If a JBL is configured *day* represents a slot number in the range 1–50.

-llabel
Label the tape as *label* rather than the default.

-jbl
Label the first 50 tapes in the jbl. The tape in slot 1 will be labelled *tag*.01, the tape in slot 2 *tag*.02; this continues till all 50 tapes have been labelled.

`-noeject`
Rewind but do not eject the tape after it has been labelled — ignored when the `-jbl` option is used.

5.7 fsdts(1)

fsdts (*fsd* tape scan) will interrogate a tape in the tape drive and reports what backups are on it. *fsdts* will prompt for any information it needs.

5.8 fse(1)

fse (*f*ile *s*ystem *e*xtractor) is used to restore a file or files from backup. All information required by *fse* can be gained by running *fsr* first — I would recommend approaching restores in this manner. The following arguments are recognised by *fse*:

-bn
Tells *fse* which backup on the tape to restore from. *fse* will skip forward *n* backups on the tape before attempting a restore. If this argument is missing *fse* will prompt for it.

-ffilename
Restore *filename*. If this argument is missing *fse* will prompt for it. By default files are restored into the directory `/tmp` — you will be offered the option to change directory before any restors are done.

-sn
Tells *fse* the file mark on the tape where the required backup appears. *fse* will skip forward *n* file marks on the tape before attempting a restore. This option is used by *fsr* from information taken from the TableOfContents database.

-jbln
If you are using a JBL carousel the `-jbl` option specifies the slot number of the tape to load. If not supplied as an argument *fse* will prompt for the slot number.

fse isn't fussy about which tape is mounted — it's up to you to get it right. *fse* is only capable of restoring files from either dump or cpio archives. For any other archive type *fse* will copy the whole backup to disk thereby allowing it to be shipped to the appropriate client for processing by the utility originally writing the backup.

5.9 fsr(1)

fsr (*f*ile *s*ystem *r*estore) searches the TableOfContents database for an exact match to a filename it is given (for which it prompts). Output will be formatted to a human readable form. Once an entry is found *fsr* offers the option to restore it automatically using *fse*.

6 Specifications

6.1 Tape data formats

The format of a tape written by *fsd* is shown in Figure 1.

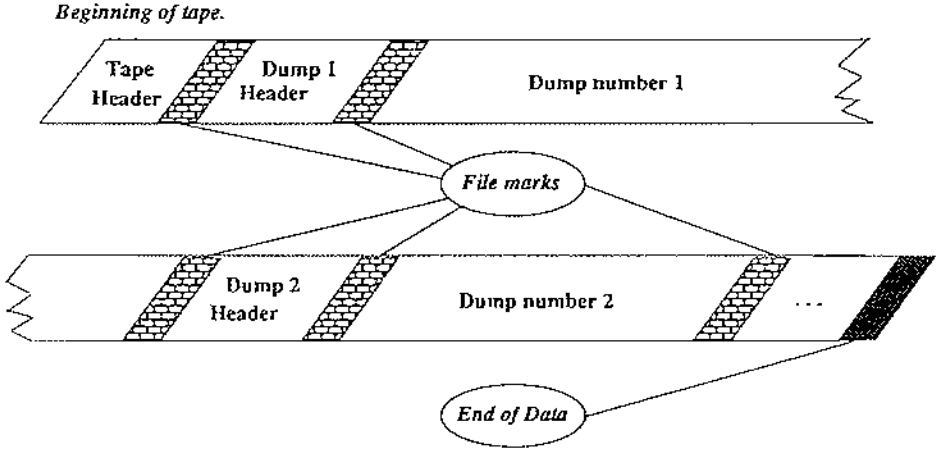

Figure 1 Fsd tape format

There are two differing header types used on a *fsd* tape. The first, (the **tape header**), identifies the backup volume and appears as a 512byte block at the beginning of each tape. The second, (the **backup header**), identifies individual backups on the tape and appears as a 512byte block before each dump. Backup headers are the only way of locating and identifying a dump on the tape. The exact formats of each header are described in Tables 5 and 6.

6.2 Tape header format

Tape header	
Field Name	*Description*
Tape volume	String, e.g. disa.24.
Date tape last used	Output from unix date(1) command e.g. Thu Jul 12 18:12:26 BST 1990.
Number of times tape used	Incremented each time the tape is used.
User-ID	Set to the user-id backup is running under.

Table 5 Tape Header (Continued . . .)

Version Number	The version number of the *fsd* writing the tape.
Start Day/Month/Year	The date *fsd* was started — used when generating TableOfContents entries.

Table 5 Tape Header

6.3 Backup header format

Backup Header	
Field Name	*Format*
File System dumped	String in the form of either /dev/sd0e or /usr.
Machine being backed up.	String, e.g. titan2.
Type of dump	String, e.g. cpio.
Backup level	Backups at level 0 are full dumps. level 1-9 backups are incremental, level 1 being nearest to a full dump.
Backup number	A count of the number of backups on the tape.
Date	Date of this backup in unix date format.

Table 6 Backup Header

7 Bugs and enhancements

7.1 Bugs

The following bugs/problems are known about in the current version of *fsd*:

1. A tag appearing as option to a command **must** be the first argument. This limitation will be removed in the next release.
2. TOC generation for inclusion in the TableOfContents database is limited to backups done using dump.
3. Leaving the carousel in the JBL unit exposes the backup tapes to disaster (e.g. fire).
4. All dumps are taken "multiuser" — this is not acceptable if absolute consistency must be guaranteed. We rely upon the frequency of out-of-hours backups to lend resilience.
5. The JBL control port is not locked by the JBL utilities, thereby leaving it susceptible to interference by other programs whilst *fsd* is running. The same is true of the Exabyte drives in the JBL — don't let more than two concurrent *fsds* (using drives a & b) start.
6. The JBL has to take a different approach to tape cycling because the original method is tightly bound to the 31 day max of a month and as such would not be able to use any JBL slots above 31. This inconsistency will be removed in future versions.

Email bug reports to:

P.Griffiths@UK.AC.Daresbury (on the JANET network)

Snail mail bug reports to:

P.Griffiths,
Daresbury Laboratory,
Keckwick Lane,
Warrington,
Cheshire,
WA4 4AD.

7.2 Enhancements

Future versions of *fsd* will incorporate the following enhancements:

1. Dumps done using `cpio` will be incorporated into the TableOfContents database.
2. Improvements on the tape cycling algorithm for the JBL, perhaps binary chop.
3. Use made of the LCD display on JBLs.
4. A menu driven *fsd* program suite integrator.
5. Automatic generation of a suitable PATH variable.
6. The tape cycling limit number (14) should be an installable parameter.

8 Glossary

Compress - Uncompress	A program employing a compression algorithm which allows files to be stored in a compressed format. Uncompress reverts compressed files back to their original form.
EOD mark	A unique recognisable mark put on a tape to indicate that no more data follows. An EOD mark is usually two consecutive file marks.
Exabyte tape	See Appendix 1.
File mark	A mark on tape that may be recognised by low level hardware as an end of file indication.
File-system	Describes a collection of files, usually hierarchically ordered, that reside on a disk partition.
Fsd client	A machine requiring its disks to be backed up by *fsd*.
Fsd server	The machine actually running *fsd*.
Full dumps	An image copy of a disk partition.
Host	A machine accessible on a network.
Hostname	A unique name identifiying a host on a network.
Incremental dumps	Only files which have been modified from a particular (usually supplied) date are backed up.
Virgin tape	A blank tape fresh out of the box - contains no written data whatsoever.
Yellow pages	Suns distributed database program. Generally there is one master database residing on a known server from which machines on a network can request information.

Appendix A Exabyte tape drives

An Exabyte tape drive consists of an 8mm tape transport mechanism with recording channel, servo, formatter, controller, interface electronics, software, and package parts all designed and produced by the Exabyte company. The product is derived from 8mm video recording technology, with performance improvements and many additional functions necessary for data processing purposes.

Helical scan recorders write very narrow tracks at an acute angle to the edge of tape in a diagonal pattern on the tape. In this way a track length is created which is several times longer than the width of the tape. This facilitates a very high tracks per inch (TPI).

No preformatting or other medium conditioning procedures are required for the 8mm data cartridges. During write operations the Exabyte records servo information and formatted user data blocks and performs a read-after-write check of the recorded user data. In the event of an error during the read back check error recovery procedures are performed without host intervention, and without the need to reposition the tape.

Exabytes operate either as streaming tape devices or as start/stop tape devices. If data is transferred to an Exabyte at a fast enough rate streaming will take place. If the necessary transfer rate to support streaming is not reached then stopping and starting will occur automatically as the data buffer empties and fills. Streaming requires a minimum of 246Kbytes per second.

Appendix B The Summus JBL — JukeBox Library

The summus JBL is an automated tape loading subsystem, consisting of two Exabyte 8mm tape drives and a Data Cartridge Carousel.

The system also has an internal power supply, an EIA RS232 port, two SCSI ports, for connection to the host, and a 1 line 40 character LCD panel that provides status information. A single JBL subsystem occupies a standard 19″ rackmount and is 10.5″ in height.

The removable Data Carousel holds 54 standard 8mm tape cartridges, giving a maximum capacity of 125Gbytes.

Six stepping motors handle all tape mounting to and from the carousel, using intelligent robotic electronics, controlled by the host computer via the onboard RS232 port. The carousel is rotated clockwise and counter clockwise by the main drive motor and is based upon the shortest route between the tape requested and the appropriate drive. A full rotation of the carousel takes 7 seconds, so the maximum seek time is 3.5 seconds for the requested tape to be loaded into the selected drive.

As the JBL uses standard 8mm Exabytes, it is thus compatible with all host hardware and software normally used with any stand-alone Exabyte drive.