Backup with dump and restore on Linux mini-HOWTO

FUKUSHIMA Osamu, <fuku@amorph.net>

2.0, 16 Aug. 2000

This document describes how to backup and restore your filesystems with the dump(8) and restore(8) utilities of BSD origin. Also covered is the basic operation of magnetic tapes with the mt(1) command to complement the simple yet powerful suite.

Contents

1 Introduction

The *dump*(8) and *restore*(8) commands have traditionally been used on the BSD systems to backup and restore filesystems. Dump backups a filesystem as a whole into an “archive”, and restore retrieves files from it. Although the archive may be created as a regular file on a regular filesystem, it is usually stored on an external backup device such as a magnetic tape. Some features are implemented in dump to support such devices.

1.1 Dump: Smart Archiver for Filesystems, not Files

There are other popular tools to manage backup archives such as *cpio*, *tar* and *afio*. These utilities handle files as the target to be archived and they are capable of excluding specific files and/or directories from the target. They can even create a single archive that contains files from multiple filesystems.

In contrast, dump handles a physical filesystem as an archiving target and the restore command usually uses the archive to restore the filesystem as it was dump’ed. Each file is managed by the i-node number and, basically, dump cannot exclude specific files from the archive (actually, you may do so in a different manner. See Section 3.1.1 (Excludes specific files from backup)).

Dump is indeed a simplistic and primitive tool, but it does come with a brilliant feature for incremental archiving. It identifies newly created or modified files after the previous backup and efficiently stores them to an archive very fast.

For example, suppose a file ‘foo’ was backed up during the last archiving and removed afterwards. On the next incremental archiving, dump puts the record in the archive as ‘there used to be a file foo, but it was removed’. If you use *tar* for your regular incremental backup tasks and attempt a full restoration one day, you may run out of the disk space by trying to restore a large number of files that had already been removed. With dump, you will never face such a problem.

In summary, it would be fair to say

- Cpio, tar or afio is suitable for archiving specific files or directories.
- Dump is suitable for archiving whole filesystems.

Just pick a right tool for you job. This documentation is on the dump and restore suite, and the following discussion focuses on the backup of filesystems, not files.

1.2 Simple is Not Just Beautiful for Restoration

Dump is not only an archiver (like tar), but also is a backup utility. Your Linux distribution might include a user-friendly backup utility, most likely a front-end to external archiving tools. I tried some of such utilities and found the nice-looking menu-driven backup configuration tools are indeed nice and handy.

In fact, I even considered switching to such utilities several times in the past, but I still use dump and restore today. One reason is, of course, that I am already familiar with dump and restore, and don’t want to learn new software. However, the main reason comes from my belief that ”the simpler, the more reliable”. The backup archives are often needed in a somewhat emergency situation and you may not even assume that the system itself shall boot. So, I believe a simple and less-demanding backup system suits better for the restoration tasks.

I made an emergency floppy disk with initrd myself. If you try to build one, you will soon realize that the disk space is fairly limited even with initrd. My diskette contains a generic kernel, which supports several machines around me, some driver modules, usual commands for system recovery (fdisk, fsck, ed, tar, ...) together with the mt command that we will discuss later, and of course the trusty little restore. I haven’t yet encountered the situation where this floppy is actually required but I’ve confirmed that the floppy does boot the system and drive the tape devices (both local and remote). To tell the truth, it wasn’t a pleasant experience for me to create this floppy disk, but I end up with getting a very good sense of assurance when I confirmed that the whole filesystem can be recovered with the single floppy disk.

I have no intention to deny the usability of other backup systems; I simply do not recommend an elaborate system for restoration, well, just in case for an emergency.

1.3 Hardware Device

These days, large hard drives are fairy affordable and you may want to use one as a backup medium. Hard drives are essentially maintenance-free and the archives on them will always be easily accessible.

If the amount of the backup target is not too large, using a HDD is not a bad idea at all. If you hesitate to buy a backup-specific device like a tape drive immediately, starting with a hard drive is also a good idea. When you get proper hardware, you can reuse the disk as a usual storage space.

However, if the capacity of the backup target is large, you should perhaps consider obtaining a backup specific device. Because a tape medium is generally removable, you can preserve the history of a series of backups at a modest cost.

The most popular medium for dump is a cartridge tape. On PCs, QIC (quarter inch cartridge), 4mm DAT, 8mm EXABYTE and DLT (Digital Linear Tape) are the most common.

I have been using an EXABYTE EXB-8505, an old SCSI1 tape drive. It takes a special tape cartridge similar to an 8mm videotape. Although such a system may not be optimal for personal use due to a high cost for the drive and media, it has been used in the industry for a long time and a high reliability is proven.

A 4mm DDS drive is a compact drive which has a similar mechanism to the 8mm EXABYTE. It is popular because of its affordability (you can find a brand-new one with 30,000 yen or so) and fairly large capacity. The DDS drive is quiet and easy to handle, although some claim that the lifetime of the head is rather short.

I would suggest you to choose a medium which can easily store the largest filesystem you.

You can find more general information about tape drives at

The Source of All Knowledge `http://www.paranoia.com/~vax/unix_tape.html`

and links given there.

In this document, I will use my EXB-8200 tape drive to explain how to create backup archives on tapes.

1.4 Linux Ports

1.4.1 dump and restore v0.3

There is the dump/restore command set for Linux ported by Remy Card (`card@linux.eu.org`) based on the one in 4.4BSD. Red Hat and Debian have their binary packages of dump-0.3. Source archive is available at:

ftp://tsx-11.mit.edu/pub/linux/packages/ext2fs/dump-0.3.tar.gz

There is no description in the LSM file about the compatibility with the 2.0.x (or later) versions of the Linux kernel, you should use it with no problem with these new kernels. Since dump depends on the structure of the ext2 filesystem, you need Ext2fs library to compile it by yourself. This library is also available from the directory of the above URL.

Dump has some limitations because of its dependency on the ext2 filesystem:

One is that it cannot properly restore the file compressed by the e2compr, which is the extension to the ext2 filesystem developed by Peter Moulder (`reiter@netspace.net.au`). More concretely, on-the-fly access to the compressed data will not be available since the cluster bit, which tells the data is compressed or not, is not recovered.

For e2compr, therefore, you have to choose the backup system except for dump currently.

Another limitation is that dump version 0.3 does not support the multivolume backup (at least officially). Hence if the size of the backup media is smaller than that of the filesystem, it is possible you cannot restore it properly.

1.4.2 dump and restore v0.4 Beta

In January 1996, new dump-0.4 beta version was released for testing. It was ported from the dump in 4.4BSD-Lite2. This series was maintained and distributed by Remy Card until 0.4b4, but left unmaintained for a few years after that.

In September 1999, Stelian Pop (`pop@cybercable.fr`) took over the development, and is still been working for the official release. Now the beta version for this is available at:

http://dump.sourceforge.net/

Most notably, this version supports a multivolume backup. It also incorporates various feature enhancement and bug-fixes to version 0.3, which include (as of 2000/07/05):

- Import new features and bugfixes achieved on FreeBSD-3.1-RELEASE

- Apply patches by Red Hat, Debian and SuSE
- Bugfixes for the multivolume archive handling
- Support remote backup over remote shells other than rsh
- Support ext3 filesystem
- Support kerberos
- Fix buffer overflow
- Invoking command when dump is finished (-F)
- Read the list of file to be restored (-X)
- Support GNU readline on interactive restoration

It is known that the dump-0.4b series before 0.4b15 have a security hole. Please make sure to install 0.4b16 or later. As of the English translation of this document, 0.4b32 is employed in a Debian distribution.

1.4.3 Device files

The Linux kernel provides the drivers for the tape devices. Please build the proper driver for your device when compiling your kernel. You may also use loadable module, if you prefer.

Then, check the device files.

```
% ls -l /dev/*st[0-9]
crw-rw-rw-   1 root     disk       9, 128 Oct  5  1995 /dev/nst0
crw-rw----   1 root     disk       9, 129 Oct  5  1995 /dev/nst1
crw-rw-rw-   1 root     disk       9,   0 Oct  5  1995 /dev/st0
crw-rw----   1 root     disk       9,   1 Oct  5  1995 /dev/st1
```

There should be two kinds of device files: `/dev/nst?` and `/dev/st?` (if not, make them with MAKEDEV command). st? are "auto-rewind" devices, which rewind the tape after the command is invoked to the driver, and nst? are "no-rewind" devices. Which to use is your choice, but I prefer the no rewind ones. In this document, `/dev/nst0` is used as the target device. When the target device is chosen, you may want to create the symlink to it named "`/dev/tape`". With this, you can omit the device name on the command lines of mt and others.

```
% cd /dev; ln -s nst0 tape
```

If you intend to use the tape drive for the backup only, you should consider limiting the access to it. To do this, remove the read/write permissions for 'Others'. In the above example, the first tape drive is accessible to normal users, and the second drive is backup purposes only, to which the access is prohibited except for the owner and the users belonging to the 'disk' group.

2 The mt command

The `mt` command is a utility to manipulate tape drives. With mt, you can rewind/forward/position the tape, as well as check the drive status. It is a must-have tool if you want to use dump/restore with tape drives. If possible, it is a good idea to prepare a tape for training purposes, and practice around with it. Some commands of mt are drive-dependent, so please read the manual carefully to know which commands are available for your drive.

For Linux, mt-st is widely used. It is an mt variant ported from BSD to Linux by Kai Makisara (`Kai.Makisara@metla.fi`). You can get the source package from:

ftp://tsx-11.mit.edu/pub/linux/sources/sbin/mt-st-0.4.tar.gz

If your tape drive is a kind of DDS autoloader like Archive Python 28849 SCSI DDS drive, MTX program by Leonard N. Zubkoff (`lnz@dandelion.com`) is useful. Its source archive is available at:

http://www.dandelion.com/Linux/

The usage of MTX is explained 2.2 (later) in this chapter.

2.1 Mt operations

In this section, I show the mt features by showing its actual usage.

Insert a tape (for practice purpose, if possible) into your drive. After the tape has been loaded, let us confirm the tape status. `mt status` command can be used to do this. Here is an example:

```
% mt status
SCSI 1 tape drive:
File number=0, block number=0.
Tape block size 1024 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (41010000):
 BOT ONLINE IM_REP_EN
```

First of all, look at the bottom line. This means that the drive has a tape loaded, and the status BOT indicates that the drive head is at the beginning of the tape. Next word "ONLINE" indicates that the tape drive is ready to be operated (by mt). The drive status must be "ONLINE" before read / write operations. Next, see the third line. It shows that the current file number is zero. File number zero corresponds to the beginning of the tape, and is incremented as passing the End-Of-File (EOF) marks on the tape.

(In the following explanation, I use the ascii-art figures in the plain text version. Please use fixed width font if possible.)

```
  V (<--- Tape Head)
 /===================================================================/
| B<--- BOT Mark)                                   ->   Tape End   /
 \=================================================================/
                           V: Tape Head  B: BOT Mark  E: EOF Mark
```

Normally, you don't have to set tape density and tape block size parameters, because these will be automatically set to suit your drive. If you want to read/write the tape on other OS's also, you may want to set these parameters explicitly for portability. If your drive supports compression feature and you want to use it, you have to pass the "compression" flag explicitly to the drive by mt.

These hardware specific parameters are strongly dependent on the drive you use. Please refer to the mt(1) manual page (items on defsetblk, setblk, defcompression, datcompression and compression), and the manual of your drive.

If "mt status" outputs an error message as follows, chances are that the link /dev/tape doesn't point to the device file of your drive correctly.

```
/dev/nst0: No such device or address
```

In this case, try other tape-device files by -f option. After finding the right one, fix the link to point to it.

Now you can try writing some files to your tape. Create a directory for practice in an appropriate place. Generate six dummy files (from file-01 to file-06) by `touch` command.

```
(tcsh)% foreach num (01 02 03 04 05 06)
foreach? touch file-$num
foreach? end
(tcsh)% ls -l
-rw-r--r--   1 fuku     users           0 Nov 21 01:10 file-01
-rw-r--r--   1 fuku     users           0 Nov 21 01:10 file-02
-rw-r--r--   1 fuku     users           0 Nov 21 01:10 file-03
-rw-r--r--   1 fuku     users           0 Nov 21 01:10 file-04
```

```
-rw-r--r--   1 fuku     users           0 Nov 21 01:10 file-05
-rw-r--r--   1 fuku     users           0 Nov 21 01:10 file-06
```

Then, write these files to the tape with tar, one by one.

```
% tar cf /dev/tape file-01
```

If you see no errors, it should have worked. Let's see mt status.

```
% mt status
SCSI 1 tape drive:
File number=1, block number=0.
Tape block size 1024 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (81010000):
 EOF ONLINE IM_REP_EN
```

Looks fine. Since one EOF mark has been written on the tape, the file number is incremented by one. Because `/dev/tape` is `/dev/nst0` in this case, which is no rewind device, the head position is at the EOF of the file just written. And the drive is ready to write next data.

```
            V
 /==================================================================/
| B[file-01]E                                                      /
 \================================================================/
                             V: Tape Head  B: BOT Mark  E: EOF Mark
```

Then write rest of the files at once.

```
(tcsh)% foreach num (02 03 04 05 06)
foreach? tar cf /dev/tape file-$num
foreach? end
```

Again, confirm the status.

```
% mt status
SCSI 1 tape drive:
File number=6, block number=0.
Tape block size 1024 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (81010000):
 EOF ONLINE IM_REP_EN
```

All files have been properly written. The drive head position is at the end of the files just written, as shown in figure 3.

```
                                                              V
 /==================================================================/
| B[file-01]E[file-02]E[file-03]E[file-04]E[file-05]E[file-06]E    /
 \================================================================/
                           V: Tape Head  B: BOT Mark  E: EOF Mark
```

It is important to know that each file consists of two parts, a file content and the EOF mark. If you write a file successfully, these two parts are generated automatically. When reading the file, set the tape head at the EOF of the previous file so that you can read the file from the first block. And if you want to add a file to the tape, you must set the head at the EOF of the last file in this tape. In other words, the EOF mark of the file is also a start position of the next file. If you write data from middle of some file, of course you will lose whole contents of it.

As the next practice, let's read a certain file from the tape which contains multiple files sequentially. Firstly, consider extracting file-03 from the tape to which we just wrote six files. You have to move the head to where the target file is recorded.

This can be done as shown below. First, rewind the tape completely, and then go to the proper position.

```
% mt rewind
```

```
  V
 /==================================================================/
| B[file-01]E[file-02]E[file-03]E[file-04]E[file-05]E[file-06]E    /
 \================================================================/
                           V: Tape Head  B: BOT Mark  E: EOF Mark
```

file-03 is written at the position of file number 2. Now the head is at the beginning of this tape (BOT), so you have to skip two EOFs to go to file-03.

```
% mt fsf 2
```

`mt fsf` command skips specified numbers of EOFs and goes to the starting block of the next file. `fsf 2` means that the head should be moved to the starting position of the file, which is two files ahead of the current position.

```
% mt fsf 2
```

```
                      V
 /==================================================================/
| B[file-01]E[file-02]E[file-03]E[file-04]E[file-05]E[file-06]E    /
 \================================================================/
                           V: Tape Head  B: BOT Mark  E: EOF Mark
```

```
% mt status
SCSI 1 tape drive:
File number=2, block number=0.
Tape block size 1024 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (81010000):
 EOF ONLINE IM_REP_EN
```

Status says that the head is at the EOF of the file number 2 (where the file-02 is archived), and is also the starting point of file-03. Let's look the content of this file by tar:

```
% tar tf /dev/nst0
file-03
```

It is file-03, as expected. Let's see tape status.

```
% mt status
SCSI 1 tape drive:
File number=2, block number=10.
Tape block size 1024 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (1010000):
 ONLINE IM_REP_EN
```

Note that EOF is not shown in this status. `Tar` program usually reads an archive until its own "end of file" mark, and stops. This "end of file" is DIFFERENT from the EOF of the tape.

```
                                V
 /==================================================================/
| B[file-01]E[file-02]E[file-03 F]E[file-04]E[file-05]E[file-06]E  /
 \================================================================/
                           V: Tape Head  B: BOT Mark  E: EOF Mark
```

In figure-6, F (blue mark) is the tar's "end of file" mark. Note that this is still within the recorded block of the file. If you try to read next block from this position, tar immediately finds EOF mark and silently quits without reading any files. If you want to read the next file, do this command:

```
% mt fsf
```

to skip one EOF mark. Please remember this behavior, since it is slightly confusing.

Let's consider how to read the archive which has file-03 again, after you did "mt fsf" and the head is now at the EOF mark of it. The answer is searching the tape backward until the second EOF mark will be found. That is the beginning of this file.

```
                            V <====== V
   /===================================================================/
  | B[file-01]E[file-02]E[file-03]E[file-04]E[file-05]E[file-06]E    /
   \=================================================================/
                                 V: Tape Head  B: BOT Mark  E: EOF Mark
```

To do this, type:

```
% mt bsfm 2
```

`bsfm` is an extended command of mt, and some old mt doesn't implement it. In that case, you will have to use bsf and fsf in sequence to achieve the desired operation. The detail is somewhat cumbersome so it is omitted here.

Next, we go into the operation to add a file after the last file in the tape. You can go to the EOF of the last file by `mt eod` command. However, this command might not work with certain drives, so you should test it beforehand. Even if it doesn't work, you can do the same by "fsf" command if you know how many files are written in this tape by logging your operations.

Finally, rewind the tape and eject it. This operation also depends on the kind of your drive, but usually the following command works:

```
% mt offline
```

Then the tape is rewinded if necessary, and ejected from the drive.

The tape operation with mt is not so easy and you have to be familiar with it. And as I'll mention in 3.4 (Log your dump records) below, there is no appropriate way to confirm the status of the recorded archives. Therefore, it is very important to keep track of your operations. Please prepare a tape for practice, try various mt commands and confirm their results.

2.2 MTX operations (tape library manipulation)

Some devices allow loading of multiple tapes. These include tape libraries such as Archive Python 28849-XXX, HP Surestore 12000, Quantum DLT 2500 XT. These drives usually have a dedicated cartridge magazine which holds multiple tapes. After inserting the cartridge filled with tapes into the drive, you can select any tape within the cartridge for reading and writing.

Some drives have switches for the media change, with which you can control them manually. But it is most likely to be irritating to operate, It may be hard to check if the loaded tape is the right one, or it allows changing the media only in the sequence as inserted in the cartridge magazine.

This situation can be avoided easily by using the MTX, which is maintained and distributed by Leonard N. Zubkoff (`lnz@dandelion.com`). MTX consists of a single binary command named mtx. As of writing, the current version is 1.1. Debian GNU/Linux (potato) has a binary package of MTX.

Even if the binary is not available, you can easily compile it from source if your kernel is 2.0.30 or later (for older kernels, you must slightly modify the header file as explained in the "LINUX KERNEL REQUIREMENTS" section of mtx.doc, which is included in the distribution archive). Depending on the location of your kernel source tree, the build may fail with no header file error. In that case, add -I/usr/src/linux/include (or the equivalent that works) to the CFLAGS macro in your Makefile.

```
CFLAGS = -O6 -m486 -fomit-frame-pointer -I/usr/src/linux/include
```

Now let's try to manipulate a tape library with MTX. We assume that the cartridge magazine is filled with tapes, and is inserted into the drive.

At this stage, every tape is staying in the magazine, and not loaded where the tape head is. If you check with mt here, "no tape found" status will be returned. Mtx also has the feature to show this status. The command line is exactly the same with mt, just add the device name and the optional command "status".

```
# mtx -f /dev/nst1 status
Data Transfer Element:  Empty
Storage Element 1:      Full
Storage Element 2:      Full
Storage Element 3:      Full
Storage Element 4:      Full
Storage Element 5:      Full
Storage Element 6:      Full
```

"Data Transfer Element" is the part which actually reads/writes the tape, and is Empty now. "Storage Element 1-6" shows the status of the cartridge magazine. The number is in sequence in the cartridge magazine, and called "storage number". The above example shows that all the elements of the magazine have the tape inserted.

Then try to load the tape to "Data Transfer Element". You can also use mtx for this purpose. Just specify the optional command "load" and the storage number.

```
# mtx -f /dev/nst1 load 1
Loading Storage Element 1 into Data Transfer Element...done
```

Now we have loaded the first tape. Check with the `status` command.

```
# mtx -f /dev/nst1 status
Data Transfer Element:  Full (Storage Element 1 Loaded)
Storage Element 1:      Empty
Storage Element 2:      Full
Storage Element 3:      Full
Storage Element 4:      Full
Storage Element 5:      Full
Storage Element 6:      Full
```

You can see the tape of storage number 1 has been loaded to the “Data Transfer Element”. You can also notice that the Storage Element 1 in the magazine has become empty.

At this stage, you can access the tape with mt as ordinary drives. When you want to return the tape to the magazine, just use the optional command “unload”.

```
# mtx -f /dev/nst1 unload 1
Unloading Data Transfer Element into Storage Element 1...done
```

You can load any tape of any storage number. Now, we load the tape of storage number 2.

```
# mtx -f /dev/nst1 load 2
Loading Storage Element 2 into Data Transfer Element...done
```

Looks fine. If you load another tape of different storage number, the one already loaded will be unloaded automatically, and the tape specified is loaded in place of it. You can also specify the storage in the magazine by ”next tape” or ”previous tape”. Issue the optional command “next” (note that you have the tape 2 loaded at the moment).

```
# mtx -f /dev/nst1 next
Unloading Data Transfer Element into Storage Element 2...done
Loading Storage Element 3 into Data Transfer Element...done
```

The tape of storage number 2 has been unloaded, and the number 3 has been loaded. If you want to load the previous tape again, you can use the optional command “previous”.

```
# mtx -f /dev/nst1 previous
Unloading Data Transfer Element into Storage Element 3...done
Loading Storage Element 2 into Data Transfer Element...done
```

You can find the status has been recovered.

Besides the ones already described, there are other optional commands such as “first” (load the tape of smallest storage number in the magazine) and “last” (load the tape of largest storage number in the magazine). If you are using dump-0.4b17 or later, these commands can be invoked when the tape reached its end during the dump. It must be very useful for the backup of huge disk at once.

3 Backup with dump

3.1 Select filesystem to backup

In this section, I’ll give a step-by-step description of backup procedure using dump.

In general, you should backup ”all filesystems”. However, it is not necessary to dump the filesystems which don’t have to be retrieved on full-restoration. For example, if `/tmp` is an independent filesystem, it can be excluded immediately. Similarly, cache data of proxy daemon can also be omitted. As well, the spool, overview database and history of Netnews system may be excluded if you give up on them.

The rest of all the filesystems should be the target of the backup. For them, put ’1’ flag on the fifth field of their entry in `/etc/fstab` file. Here is an example:

```
# /etc/fstab
#
/dev/hda2  /                ext2  defaults,noquota         1 1
/proc      /proc            proc  defaults,noquota         0 0
/dev/sdb3  /tmp             ext2  defaults,noquota         0 2
/dev/sdb2  /var             ext2  defaults,noquota         1 3
/dev/sdd2  /usr             ext2  defaults,noquota         1 2
/dev/sdd1  /usr/local       ext2  defaults,noquota         1 3
/dev/sde1  /var/spool/news  ext2  defaults,noquota         0 4
/dev/sdc1  /var/spool/ov    ext2  defaults,noquota         0 3
/dev/hda3  /.d1             ext2  defaults,noquota         1 2
/dev/hdb1  /.d2             ext2  defaults,usrquota        1 4
/dev/sdb1  /.d3             ext2  defaults,ro,noquota      1 4
/dev/sda5  /home            ext2  defaults,usrquota        1 3
/dev/fd0   /mnt/floppy      ext2  defaults,noauto,noquota  0 0
/dev/sdb4   none            swap  sw                       0 0
```

In addition to `/tmp` and `/var/spool/news`, you may find that the fifth fields of `/proc` and `swap` are also 0. You may see 1 for these entries in some case, but they are by no means effective and may be even disgraceful, so fix them now with other modifications.

3.1.1 Excluding files not to be backuped

Depending on your partitioning, some backup target filesystems may include files which you don't want to backup. If you do wish to exclude them, you can mark their i-nodes with "no dump flag" with specific utilities. You may want to set this attribute for files updated frequently but no need to backup, such as proxy cache directory.

This can be done with lsattr(1) and chattr(1). Lsattr shows you the flag status of the file, and chattr sets it. For example,

```
# lsattr -d /var/spool/squid
------- /var/spool/squid
# chattr +d /var/spool/squid
# lsattr -d /var/spool/squid
------d /var/spool/squid
```

The files or directories with no dump flag (+d) are excluded from the target of dump. In the above example, the directory used by Squid cache system and its huge contents are marked so that they will not be dumped. Once the directory is marked with this flag, the files that will be created inside it will also have the same attribute.

Note that this specification is ignored by default on level 0 full dump, and works only at the incremental backups of level 1 and later. If you want this to work on level 0, you should specify h option on the dump command line.

3.2 Full backup

3.2.1 Stop the filesystem

Now we finally start the full backup.

Before the full backup, you may want to get down the system into single user mode, make all the filesystems static and fsck them to ensure that they are consistent. Since the full backup is performed rarely, I recommend you to do this at any rate.

Firstly, get into the single user mode with:

```
# init 1; exit
```

Then unmount the filesystems one by one, and check them.

```
# umount /usr/local; e2fsck -afv /dev/sdd1
```

(If it's annoying to unmount each of them, you can specify "`linux -b`" on bootup so that only the root file system will be mounted.)

Check all the filesystems to be backuped, presumably after unmounting them. Since you cannot unmount the root filesystem, you may want to remount it read-only, and check it with e2fsck.

```
# mount -r -n -o remount / ; e2fsck -afv /dev/hda3
```

After all the check is done, remount it again with read-write, so that dump can log the backup information:

```
# mount -w -n -o remount /
```

If you need `/usr` filesystem and others on backup, e.g. dump over the network, mount them again to make the network available.

3.2.2 Set up the drive

At this stage, we set up the backup media. As already described, I explain the operation for the EXABYTE EXB-8200 drive (SCSI) with 122m 8mm data cartridge as an example.

The first thing you should do is the cleaning of the tape head. When you load a cleaning tape into EXABYTE drive, the cleaning process runs automatically, and the tape will be ejected without rewinding. You should accustom yourself to do this, especially before the level 0 dump. It certainly reduces the error occurrence during the tape read and write.

Then load the new tape into the drive. Don't forget to label the tape beforehand, or you'd get corrupted as the number of tapes increases. It is enough to write down the sequence number of the tape. As will be explained later, I recommend to prepare a notebook to log the tape management information, rather than to write it on the label of each tape. After loading the tape into the drive, check the status with the mt command.

```
# mt status
SCSI 1 tape drive:
  :
  :
```

Depending on the kind of your drive, the output of this command differs. It is OK if you can confirm that the tape is loaded properly.

3.2.3 Dump

Now we can invoke dump. It is not so difficult to use, but has some peculiarity in its parameter specification. (v0.3 or later allows more flexible options, but we here stick to the conventional syntax.)

```
  USAGE: dump ‘option’ ‘parameter’ ‘filesystem’
- options
  0-9 : dump level
  B   : number of records per volume
  b   : blocksize per record (KB)
  h   : dump level below which the nodump attribute affects
  f   : output file (tape)
  d   : tape density
  n   : notify to the operator
  s   : tape length
  u   : update /etc/dumpdates
  T   : specify the date to record in /etc/dumpdates
  W   : print the filesystems to be dumped with marks
  w   : print the filesystems which need to be dumped
- parameters
  Specify the parameters corresponding to the options in sequence.
  For example, if the option is ‘‘sbf’’, the following parameters
  should be the order:
    dump sbf ‘tape length’ ‘blocksize’ ‘filesystem’ ‘output file’
- filesystem
  mount point or a device name of the filesystem to dump
```

Typical command line looks like:

```
# dump 0uf /dev/nst0 /home
```

The number as the first parameter is the dump level between 0-9. The ‘0’ here means the full backup, and the meaning of other numbers will be discussed later. The ‘u’ specifies that the `/etc/dumpdates` file will be updated so that the following incremental backup can use the record. `/etc/dumpdates` contains the time and the level of the dump. This will be referred to at the following invocation of dump for incremental backups or from w/W options.

If you dump to the tape, you may have to specify the blocksize and the tape length. When you encountered the situation: ‘the tape capacity should be enough but the tape reaches its end during the dump’, you may want to modify these parameters. Ad hoc workaround is to specify the over-estimated capacity with B option.

```
# dump 0uBf 2300000 /dev/nst0 /home
```

This example specifies that the tape capacity is 2.3GB (i.e. the number of records for that; 2.3GB divided by the blocksize), which is the slightly too large estimation. The options to specify the density or the tape length are prepared to teach the dump when to exchange the media. But most recent drives can detect the tape end properly, you should have no problem this way.

When dump is invoked normally, backup proceeds with some messages printed on the console, which include the estimated duration until the backup finishes. It is a good idea to leave this session foreground rather than to send it to background, until you are convinced the things work fine. If it reaches the tape end or if some error occurs, you will be requested to do some choices in the interactive session.

When the dump session finishes properly, you can dump another filesystem if the tape capacity remains enough.

3.3 Dump level and incremental backup

The backup behavior of dump is dependent on the specified dump level, which is an integer between 0-9. If not given, default value 1 is used. When invoked with some level, dump selects the files which have been updated since it ran with SMALLER level than the current one, and backups them.

For example, if level-5 backup is invoked after level-4 backup, the files updated after the level-4 backup will be recorded to this level-5 backup archive.

Using this, you can make incremental backups easily. If you run dump with:

```
Day 1    level 0
Day 2    level 1
Day 3    level 2
Day 4    level 3
Day 5    level 4
Day 6    level 5
Day 7    level 6
Day 8    level 7
Day 9    level 8
```

Then after the Day 2, you can backup updated files only. With this sequence, you can minimize the backup time.

If you like more elaborated method, “Tower of Hanoi sequence” is known for the dump level scheduling.

For example, proceed the dump sequence as follows.

```
0    -> 3 -> 2 -> 5 -> 4 -> 7 -> 6 -> 9 -> 8 ->
1(a) -> 3 -> 2 -> 5 -> 4 -> 7 -> 6 -> 9 -> 8 ->
1(b) -> 3 -> 2 -> 5 -> 4 -> 7 -> 6 -> 9 -> 8 -> Return to 0
```

You should renew the tape on each level 0 dump, and store them at safe place perpetually. For level 1, prepare tapes as many as the number of rotations (2 in the above example). For other levels, one tape is enough per level.

The merit of this method is that you can preserve the snapshots of the filesystem for a long period, and that you can save the time and the media necessary for each backup.

By the way, you may remember that we entered into the single user mode on getting the level 0 (full) backup. Should we also do the same way on each incremental backup? If you backup the system periodically, it should be difficult to stop the system every time. In my opinion, it is acceptable to backup the system on multi-user mode. It is your choice to choose the convenience or the robustness.

If critical maintenance (e.g. change the hardware) is scheduled after the incremental backup, however, you should consider the level 1 dump with single user mode, so that you can make sure the preservation of the system status prior to the system halt.

3.4 Log the dump records

As I mentioned in the section 3.2.2 (set up the drive), it is recommended to log the dump information on a destined notebook rather than on the tape label. It is because that the information is too much to record on the label.

This information is necessary when you restore the data. It is important to make clear that each filesystem is recorded which part of which tape (you may be upset when you have to restore the system :-)

You should maintain the detailed dump log, by preparing either the destined notebook or the folder to file printouts. Entries you may want to record are:

```
[Tape No.nnnn]        the serial number of the tape cartridge
File count:     the location of the data in the tape
Date:           date
Host:           hostname of the target machine
Filesystem:     name of the filesystem
Device:         device name of the filesystem
Level:          dump level
```

```
Blocks:        the number of blocks archived

Length:        the proportion of this archive in the tape
```

One of my records looks like:

```
[Tape No.0024]

Date:          Wed Nov 12 12:39:57 1996

File count:    1

Host:          nicorn

Filesystem:    /home

Device:        /dev/sda5

Level:         0

Blocks:        392556

Length:        0.16
```

It is not so hard to get such data by formatting the output of dump (using perl or the like). Once it works, all you have to do is print out the data and file it away.

It is also a good thing to keep other information like the output of ”fdisk -l”. If you have printouts of such data, you will find it helpful when you restore the system.

One more thing you may want to do is to record a list of archived files in a text file. To make this list, rewind the tape via mt command after each dump, invoke the ”restore -t” command and redirect its output. Save this list in a directory like `/var/spool/adm/dump_index/`, and you can easily grep this list to find out which archive has the file you want. This practice is also profitable to confirm that the archive just created can be read properly.

3.5 Rdump

It is possible to dump over the network: you can use ”rdump”. It takes the same arguments as dump except for a remote hostname to send the dump data (actually rdump and dump share the one executable binary). In the following example, we dump the files in `/home` directory on the local machine, to a tape device on the remote machine named ”tapeserver”.

```
    # rdump 0uf tapeserver:/dev/nst0 /home
```

To dump over the network, you have to be able to rsh the remote system without entering a password. It is also required that the rsh can locate the `rmt` command binary in its path, since rdump uses it on the remote machine. The path to `rmt` is sometimes not set by default in some Linux distributions. You can check it by:

```
# rsh remotehost which rmt
```

If it doesn't tell the location of `rmt`, create a symbolic link of `rmt` in the command path in the environment of the rsh.

Usually, backup is done by root. But remote rsh via root without password may cause a serious security problem.

To reduce somewhat of the security risk, you should create a specific uid for your dump job. You can set it up in the following way.

- Using vipw, add a user named ”fsdump” on all hosts in your local network. Leave its password entry as ”`*`”.
- Set its home directory to `/etc/fsdump`.
- If you use Linux on the host, add the user fsdump to the ”disk” group (or to the equivalent group if you use other OS).
- Create `.rhosts` file in `/etc/fsdump` on the machine that has a tape drive, so that the user fsdump in your network can access to the machine without entering a password.
- Execute remote dump as user fsdump (use su). If you want to use cron, add this job to fsdump's crontab.

3.6 Store the tape

It is desirable to save the level 0 tape cartridge as long as possible. Needless to say, tape reel is very sensitive to heat, dusts, humidity and anything magnetic. You should keep it away from them, or you will not be able to read it out properly.

It is also recommended that the level 0 and level 1 tape cartridges should be kept in a separated room (and in a separated building, if possible) from the machine. This will protect you from losing both your machine and tapes in a single disaster like fire.

To provide the partial restoration service, you may want to keep level 2-9 tapes near the machine. Even in such cases, it is not good for security to leave them where anyone can touch.

The most desirable practice is to protect the tape cartridges in a (fireproof) cabinet and lock them up. It may sound exaggerated for you, but at least you should become aware of this kind of data protection. For instance, you should not write the information on the tape with which the bad person can guess the contents of it. It doesn't help once the tape is read, but it's still better not to help such a person at least. This will not be confusing if you record the dump logs in the notebook as described in 3.4 (Log your dump records).

If you are a paranoid, you may want to load the level 0 tape periodically and make sure that no read error should occur. But this is very hard to do, and affordable only to those who have plenty of time and power (I have never done it). Instead, I suggest you to perform level 0 (full) dump periodically and use a new tape cartridge each time.

3.7 Backup scheduling

Now, we can backup the system manually. But it is painful to continue the boresome work, so let's consider the way to do it automatically as much as possible. At the same time, you have to consider how often you should backup the system.

Backup schedule depends chiefly on how the system is being used. It also depends on how much time and power you can use for the backup, how much reliability the backup needs, and how much cost you can pay for it, etc. You have to determine your backup schedule taking all of these into accounts.

Ideally, you should backup the system once a day incrementally when you use it. If it's your private machine, it is acceptable to do it with a few days interval. On the other hand, if you're very cautious, you may want to backup most important filesystems every few hours.

One guideline is "Backup before the data exceed the tape capacity". If you don't have to change tapes during the backup, it is easier to automate the backup process.

Even if your tape has a plenty of space, consider backing up at least once a week. In most cases, newer data are more important. Data a month old are sometimes no more than the garbage.

It is also important when to run the backup. You should stop the system when you do the level 0 full backup. For other levels, it is also good to choose the time when the system load is low and the renewal of files seldom occurs (i.e. midnight).

Find out your own backup schedule considering and balancing these factors. And never forget cleaning the tape drives periodically.

4 Restore

"restore(8)" is a program to extract files from the archive created by "dump(8)", and to recover them on your target filesystem. It is possible either to recover the whole filesystem at once, or to choose the files to retrieve interactively. A piped combination of "dump" and "restore" can duplicate the contents of a filesystem on another filesystem. As rdump, there is an r-version of the restore, "rrestore", which can read the archive in the tape drive on a remote host.

```
   Usage: restore  ‘key’ [key-modifier]
Keys (summary):
```

`i`	`Interactive restoration of specified files`
`r`	`Extract the contents of the whole archive in the current directory`
`R`	`Resume interrupted full-restoration`
`t`	`List filenames on the backup archive`
`C`	`Compare the contents of the archive with the current filesystem`
`T`	`Specify the temporary directory`
`s`	`Specify the position of the archive in the tape`
`x`	`Only the named files are extracted from the archive`
`f`	`Specify the archive file`
`h`	`Extract only the directory when specified, rather than with its contents`
`v`	`verbose output`
`y`	`Do not query on error.`

Examples:

```
restore rf /dev/nst0
  (retrieve all the files from the archive in tape media,
   and write them under the current directory)
restore isf 3 /dev/nst0
  (do interactive restoration from the 3rd backup on the tape media)
rrestore tf  tapeserver:/dev/nst0
  (list filenames in the archive stored in the tape of the
   host ‘tapeserver’)
```

It is not so hard to use this but rather tricky, so you should be familiar with it before you have to use it. The explanation of basic operations is omitted here since you can read the manual page of "restore(8)" for this purpose.

4.1 Notes on restoration

On using the partial restoration interactively, it is not a good habit to restore them directly to the target directory. Instead, you should restore the files on an empty directory (by changing the current directory to it), and move them to proper locations accordingly. On full restoration, you should mount a formatted disk at first, move to that mount point, and invoke the restore command. Therefore, if you have to restore the root partition, you have to prepare a kind of rescue system.

You should be careful about the sequence of the archives to restore. Especially, on extracting the whole contents of the archives with r command, you MUST start with the level 0 archive.

It should also be noted that an archive with some level is ineffective if smaller level dump is taken after that. For example, consider the case when you follow the "Tower of Hanoi" sequence and choose the dump level as:

```
0 3 2 5 4 7 6 9 8
```

In this case, the level-3 archive becomes ineffective when the next level-2 dump is taken. Similarly, level-5 archive is ineffective after the next level-4 dump. Extremely, when level-0 dump is taken, all the existing archives with level-[1-9] become ineffective.

Therefore, on full-restoration, you should skip these ineffective archives. In the above example, you should choose the restoration sequence as

```
0 2 4 6 8
```

to obtain the latest status of the filesystem.

If you ignore this rule and try to restore the archives just following the "Tower of Hanoi" sequence, you will obtain "Incremental tape too high" error on the restoration of level-2 archive, and "Incremental tape too low" errors after that. Once you encounter one of these errors, you cannot complete the full restore by any means, and you must restart that restoration from the first step.

(The generation of ineffective archives does not mean the uselessness of the "Tower of Hanoi" sequence. It is still useful to preserve many snapshots of the filesystem for a long period with less backup media.)

Another point you have to consider is that the restore needs fairly large temporary space. Generally, you only have small amount of space when you boot the system with a kind of rescue disk for recovery. So you should prepare another filesystem for the temporary working space, in addition to the one to be restored. And mount the working filesystem on `/tmp`, or specify that temporary space with the T option of the restore command. It is frequently overlooked. Please be careful.

It is also confusing that dump/restore does not save/retrieve the boot block of the disk (or of the sort). After the full restoration, please check the boot configuration of the OS.

4.2 Full restoration recipe

With these points in mind, I summarize the procedure of the full restoration in case of trouble for your convenience.

- Check whether hardware is the problem, and solve it if necessary.
- Read the notebook of the dump log carefully, and figure out the restoration sequence. It is highly recommended to write down the sequence on paper and double-check it.

- Confirm that the tape is write-protected before setting it to the drive.
- Boot the system with the rescue floppy.
- Format the target disk and create the filesystem to restore.
- Mount the target filesystem on some directory.
- Prepare the temporary space used by restore.
- During the restoration of incremental archives in sequence, never erase the `./restoresymtable` file until the restoration is over.
- Reconfigure the boot method (lilo, etc.).
- After the restoration is finished successfully and you confirm everything is OK:
  - Erase `./restoresymtable`
  - Invoke the level 0 dump with a new tape (and preserve the old tapes for a while).

5 Is backup really necessary?

Now that, you really need the backup?

You may think reinstalling the system should be OK, but if you've been using Linux on daily basis for years, you must have achievements of your hacking effort, unfinished documents, or mails you received from your precious person, somewhere in your home directory. Once they are lost, then they cannot be restored even if you got your Linux reinstalled.

And yet, it may become a nightmare if you have modified your configuration files (most files in `/etc` directory are more or less changed), or have installed applications by yourself. After the reinstallation, you'll have to reconfigure the system, recompile/reinstall your applications to get back the system as it was.

You may also think that the ext2 filesystem is stable enough and hardly corrupted, so you don't need backup at all. It is true that the ext2 filesystem is very hard to fail. I've also never seen it crash because of the software problems.

But however cautious you may be, you cannot go through without facing troubles with files. On many occasions, careless users easily delete files. Or on rare occasions, the filesystem may become inconsistent, being damaged by hardware troubles. In my experience, a SCSI host adapter was broken and made inaccessible. This accident had brought to my hard disk some fatal errors which could not be repaired by e2fsck, so I had to rely solely on my backups to restore the all data.

Backup will consume your system resources, and its configuration is by no means fascinating or exciting. But you should think of it as a sort of insurance, personal but indispensable. As for real-life insurance, the specified sum is most reluctantly or sometimes not sufficiently paid out to you for some reasons or others. Compared to it, backup can be said to be far more faithful and reliable.

A Administrivia

A.1 Copyright notice

A.2 Disclaimer

A.3 Acknowledgement

I gratefully appreciate to NEC corporation (NEC solutions) for lending me devices when I wrote the section on MTX.

A.4 Feedback

Please send any questions, comments, or corrections to

`FUKUSHIMA Osamu <fuku@amorph.rim.or.jp>.`

A.5 History

```
1998/04/20: first release (revision 1.3).
1998/06/03: add a section on the permissions of device files.
            notes on MTX added.
1998/11/15: add some pointers to the information about tape drives.
1998/12/12: make clear that dump-0.3 could be run in kernel 2.0.x.
```

```
1999/02/28: mention the limitation of e2compr.

1999/03/10: second release (revision 1.4).

1999/05/23: add tips for the case where rmt command path is not in PATH.

1999/07/30: add cautions on system restoration.

1999/11/30: mention new 0.4 beta series by Stelian Pop.

2000/01/21: change the pointer to 0.4 beta

2000/08/01: add a section on MTX.
```

A.6 English version

English version of this document has been translated by the following JF Project members. Please contact jf@linux.or.jp if you have comments and/or suggestions about this document.

- Hitoshi Hayakawa
- Seiji Kaneko
- Takeo Nakano
- Taketoshi Sano
- Yuji Senda
- Yoshiyuki Yamashita
- Y.Hiro Yamazaki