Backups

Why Backup?

1. Software and Hardware failures are not an uncommon thing in the computer world. Any number of occurences can cause loss of valuable data.

a) Power failures

b) Natural Disasters

c) Hardware Failures

d) User Error

Types of backups

``Do nothing'' is not a computer program, but it is the most widely used backup strategy. There are no initial costs. There is no backup schedule to follow. Just say no. If something happens to your data, grin and bear it!

If your time and your data is worth little to nothing, then ``Do nothing'' is the most suitable backup program for your computer. But beware, UNIX is a useful tool, you may find that within six months you have a collection of files that are valuable to you.

``Do nothing'' is the correct backup method for /usr/obj, /usr/src and other directory trees that can be exactly recreated by your computer.

Dump

The traditional UNIX® backup programs are dump and restore. They operate on the drive as a collection of disk blocks, below the abstractions of files, links and directories that are created by the file systems. dump backs up an entire file system on a device. It is unable to backup only part of a file system or a directory tree that spans more than one file system. dump does not write files and directories to tape, but rather writes the raw data blocks that comprise files and directories.

Note: If you use dump on your root directory, you would not back up /home, /usr or many other directories since these are typically mount points for other file systems or symbolic links into those file systems.

dump has quirks that remain from its early days in Version 6 of AT&T UNIX (circa 1975). The default parameters are suitable for 9-track tapes (6250 bpi), not the high-density media available today (up to 62,182 ftpi). These defaults must be overridden on the command line to utilize the capacity of current tape drives.

It is also possible to backup data across the network to a tape drive attached to another computer with rdump and rrestore. Both programs rely upon rcmd(3) and ruserok(3) to access the remote tape drive. Therefore, the user performing the backup must be listed in the .rhosts file on the remote computer. The arguments to rdump and rrestore must be suitable to use on the remote computer.

It is also possible to use dump and restore in a more secure fashion over ssh.

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
    targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

Or using dump's built-in method, setting the environment variable RSH:

Example 16-2. Using dump over ssh with RSH set

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f
targetuser@targetmachine.example.com:/dev/sa0 /usr
```

tar

tar(1) also dates back to Version 6 of AT&T UNIX (circa 1975). tar operates in cooperation with the file system; tar writes files and directories to tape. tar does not support the full range of options that are available from cpio(1), but tar does not require the unusual command pipeline that cpio uses.

Most versions of tar do not support backups across the network. The GNU version of tar, which FreeBSD utilizes, supports remote devices using the same syntax as rdump.

Examples using tar on the local machine

We are now going to make a tar archive of our /etc directory where most of our configuration files live and store it in a directory named /u/backups on the same machine.

1. mkdir /u/backups

2.cd /

3. tar -cvf /u/backups/etc.tar /etc

Note: The -c option to tar tells it to create an archive, -v specifies verbose output and -f specifies the file to be either written to or read from.

You'll see quite a lot of output as tar creates the archive at this point.

Now we check whether our archive has actually been created.

4. cd /u/backups

5. ls

This now show us a new file in this directory named etc.tar

If we now wanted to restore this directory we can run

tar -xvf etc.tar

This will create a directory etc and unpack the contents that were backup up previously into /u/backups/etc

You will notice that the restore actually creates a new directory. This is because tar by default removes the leading / from the directories it is backup up in order not to overwrite the original files on your system when you choose to do a restore.

Using gzip with tar

The archives created with tar can become quite big and thus it is often useful to compress the achives using a utility like gzip or bzip2 or compress or any other compression utility.

1. cd /u/backups
2. gzip etc.tar
This will now give you an archive named etc.tar.gz in this directory.

This can however be shortened using tar itself. The GNU version of tar supports compressing the files using gzip as you create the archive. To do this you add a -z option to your tar command line. You can use this same option when unpacking to uncompress the arhcive first and then unpack it.
We will now recreate our arhcive using this new method.
1. rm /u/backups/etc.tar.gz
2. cd /
3. tar cvzpf /u/backups/etc.tgz /etc
4. You can then unpack this using:
cd /u/backups
tar -xvzpf etc.tgz
The -p option preserves the original file permissions and ownership when unpacking.

Backing up to other devices

Many times you need to keep your backup off the actual machine you're backing up since this backup would be useless in case of an actual hardware failure.
By default if tar is not run with the -f option to specify the destination , it will default to the first tape device on the system.
You can use the -f option to point to the name of the tape device on your system and tar will backup or restore from tape. You can choose to also backup to file and then write this file to CD or to a removable USB/firewire/SCSI or other storage media. You can find information on how to do this in the freeBSD handbook which is available online at http://www.freebsd.org/handbook

Backup to remote servers

GNU tar which comes with FreeBSD can also be used to backup to a remote filesystem or tape device using ssh.

In our example we can backup our etc filesystem to the noc server using
tar -czf - /etc | ssh -l stu1 noc.e0.ws.afnog.org dd of=/home/stu1/etc.tgz
Replace stu1 with your account on the noc server.
This can thus be used to backup to a tape device on a remote system.
To see if your backup was successful ssh to the noc machine and check your home directory to see if the file etc.tgz was created there.

Other Backup methods

There are a number of ways to do backups for example using dd to duplicate your entire disk block by block on another disk. However the source and destination disk should be identical in size or the destination must be bigger than the source.
Another way of doing this is using RAID mirroring and hot swappable disks. RAID is most useable when done at the hardware level rather than the software level. Other tools include:
cpio is the original UNIX file interchange tape program for magnetic media. cpio has options (among many others) to perform byte-swapping, write a number of different archive formats, and pipe the data to other programs. This last feature makes cpio an excellent choice for installation media. cpio does not know how to walk the directory tree and a list of files must be provided through stdin.

cpio does not support backups across the network. You can use a pipeline and ssh to send the data to a remote tape drive. cpio is what is used on most installation cds and by the rpm system in many linux distributions.

pax(1) is IEEE/POSIX®'s answer to tar and cpio. Over the years the various versions of tar and cpio have gotten slightly incompatible. So rather than fight it out to fully standardize them, POSIX created a new archive utility. pax attempts to read and write many of the various cpio and tar formats, plus new formats of its own. Its command set more resembles cpio than tar.

Large Scale Backup systems

There are a number of large scale backup systems out there for backing up many servers to a single tape library with hundreds of tapes and many tape drives. Some are commercial and other open source. one of the latter is called Amanda. Among the Commercial ones examples are arkeia and veritas netbackup. Amanda is open source and can be got form the FreeBSD packages or the ports system.

Amanda (Advanced Maryland Network Disk Archiver) is a client/server backup system, rather than a single program. An Amanda server will backup to a single tape drive any number of computers that have Amanda clients and a network connection to the Amanda server. A common problem at sites with a number of large disks is that the length of time required to backup to data directly to tape exceeds the amount of time available for the task. Amanda solves this problem. Amanda can use a ``holding disk'' to backup several file systems at the same time. Amanda creates ``archive sets'': a group of tapes used over a period of time to create full backups of all the file systems listed in Amanda's configuration file. The ``archive set'' also contains nightly incremental (or differential) backups of all the file systems. Restoring a damaged file system requires the most recent full backup and the incremental backups.

The configuration file provides fine control of backups and the network traffic that Amanda generates. Amanda will use any of the above backup programs to write the data to tape. Amanda is available as either a port or a package, it is not installed by default.