**Chapter 6 -Backup and Recovery**

**Introduction to various types of backup media**

**Why do you need backup?**

* Backups are needed in case a file or a group of files is lost. The reasons for losing files include
* Hardware failure like disk breaking,accidentally deleting wrong file and computer being stolen.
* Backups help in all the above situations. In addition, it may be good to have access to older versions of files, for example a configuration file worked a week ago, but since then it has been changed and nobody remembers how, its just not working anymore.There are other solutions, and they are good to have if you can afford them. These include
* redundant disks (RAID 1 or 5), so that one disk can break without loss of data,using an undelete system (or not making mistakes when deleting files locking up computers.
* These help, but if there is anything you do not want to lose on the computer, make sure there are backups and they can be restored.

**What to backup?**

* If there is room on the backup media, and time limits permit running backups long enough, it probably is wisest to back up everything. You may skip /tmp or other places where it is known there are only temporary files that nobody wants to backup.
* If space or time limits place restrictions, consider not backing up the following
* Files that come directly from a CD or other removable media. It may even be faster to copy them again from CD than restoring from backup media.
* Files that can be regenerated easily. For example, object files that can be made with make. Just make sure all the source files and compilers are backed up.
* If the Internet connection is fast, it may be easy enough to download files again. Just keep a list of the files and where to download them from.

**Backup devices and media**

You need some media to store the backups. It is preferable to use removable media, to store the backups away from the computer and to get "unlimited" storage for backups.

If the backups are on-line, they can be wiped out by mistake. If the backups are on the same disk as the original data, they do not help at all if the disk fails and is not readable anymore. If the backup media is cheap, it is possible to take a backup every day and store them indefinitely.

Floppy,Disk,Tapes,CD-R and CD-RW are the medias available for backup

**Planning a Backup**

Before doing a backup, plan it carefully. Consider

* Which files are irreplaceable without a back up. Irreplaceable files probably include those in users' home directories (including /root), and configuration files, such as those in the /etc/ directory.
* Which files are on removable drives, such as cd s or floppies. Since you probably do not need to back up removable drives, you might unmount them before doing a complete system backup.
* Which files can be easily replaced by installing a package or doing a selective install or upgrade of the operating system. You can save time and storage space by not including these files in a backup.
* Which files are unnecessary or dangerous to backup. For example, files in /tmp are probably unnecessary, while restoring some files that are in the /proc directory could crash the system.
* Whether to compress files using gzip or bzip2 . Compressing saves space, but adds another step to the backup. Also, while compression is generally reliable, it creates another stage at which the process can fail.
* Whether users are responsible for backing up their own files. Since only the root user has full permissions for all files on the system, usually backups are best done by the root user. However, if users back up their own files, you might omit backing up the home directory, or at least not back it up regularly.

**Choosing a Backup Tool**

Linux has several tools for backing up and restoring files

* **dump / restore :** Old tools that work with filesystems, rather than files, and can back up unmounted devices. Although you can easyly control what is backed up with dump by editing a single column in the /etc/fstab file, for some reason these utilities have fallen into disuse. Today, many distributions of Linux, including Debian, do not even include them by default. If you want to use dump and restore , you must install them yourself.
* **tar :** A standard backup tool, and by far the easiest to use. It is especially useful for backing up over multiple removable devices using the -M option.
* **cpio :** A very flexible command, but one that is hard to use because of the unusual way in which the command must be entered.
* **dd :** The dd command is one of the original Unix utilities and should be in everyone's tool box. It can strip headers, extract parts of binary files and write into the middle of floppy disks; it is used by the Linux kernel Makefiles to make boot images.
* **Mondo :** Mondo is reliable. It backs up your GNU/Linux server or workstation to tape, CD-R, CD-RW, DVD-R[W], DVD+R[W], NFS or hard disk partition. In the event of catastrophic data loss, you will be able to restore all of your data [or as much as you want], from bare metal if necessary. Mondo is in use by Lockheed-Martin, Nortel Networks, Siemens, HP, IBM, NASA's JPL, the US Dept of Agriculture, dozens of smaller companies, and tens of thousands of users.
* **Dar:** dar is a shell command that backs up directory trees and files. It has been tested under Linux, Windows, Solaris, FreeBSD, NetBSD, MacOS X and several other systems

Many commercial or free software back up tools are also available.

**Types of Backup**

There are different kinds of backups, the following lists some of them:

**Full Backup :-**

Full backup is the starting point for all other backups, and contains all the data in the folders and files that are selected to be backed up. Because full backup stores all files and folders, frequent full backups result in faster and simpler restore operations. Remember that when you choose other backup types, restore jobs may take longer.

**Advantages**

* Restore is the fastest

**Disadvantages**

* Backing up is the slowest
* The storage space requirements are the highest

**Incremental Backup :-**

Incremental backup means backing up everything that has changed since last full backup.

**Advantages**

* Backing up is the fastest
* The storage space requirements are the lowest

**Disadvantages**

* Restore is the slowest

**Differential Backup :-**

Differential seems to be another name for incremental.differential backup offers a middle ground by backing up all the files that have changed since the last full backup

**Advantages**

* Restore is faster than restoring from incremental backup
* Backing up is faster than a full backup
* The storage space requirements are lower than for full backup

**Disadvantages**

* Restore is slower than restoring from full backup
* Backing up is slower than incremental backup
* The storage space requirements are higher than for incremental backup

**Network Backup :-**

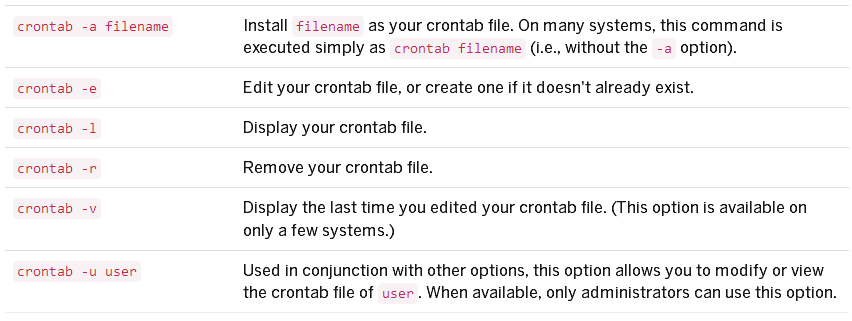
Network backup usually means backing up a client to a backup server, this means the client sends the files to the server and the server writes them to backup medium.

**Dump Backup :-**

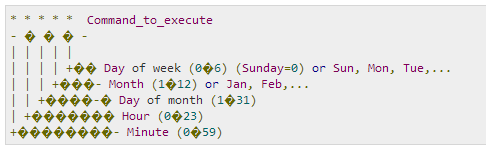
Dump backups are not ordinary file by file backups. The whole disk partition or file system is "dumped" to the backup medium as is. This means it is also necessary to restore the whole partition or file system at one go. The dump backup may be a disk image, which means it must be restored to a similar disk with same disk geometry and bad blocks in same places. Watch out for this.

**Automation of Jobs (Cronjob)**

Cron" is a time-based job scheduler in Unix-like operating systems.The **crontab** is a list of commands that you want to run on a regular schedule, **cron** is the system process which will automatically perform tasks for you according to a set schedule. Cron Jobs are used for scheduling tasks to run on the server. They're most commonly used for automating system maintenance or administration. The schedule is called the **crontab**, which is also the name of the program used to edit that schedule.

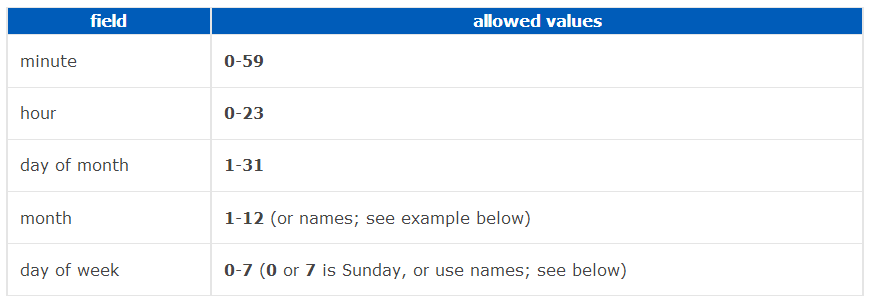


**Crontab Table Format :-**

****

Each entry in a crontab file consists of six fields, specifying in the following order:

minute(s) hour(s) day(s) month(s) weekday(s) command(s)



**Specifying multiple values in a field**

* The asterisk (\*) operator specifies all possible values for a field. e.g. every hour or every day.
* The comma (,) operator specifies a list of values, for example: "1,3,4,7,8".
* The dash (-) operator specifies a range of values, for example: "1-6", which is equivalent to "1,2,3,4,5,6".
* The slash (/) operator, can be used to skip a given number of values. For example, "\*/3" in the hour time field is equivalent to "0,3,6,9,12,15,18,21"; "\*" specifies 'every hour' but the "/3" means that only the first, fourth, seventh...and such values given by "\*" are used.

**EXAMPLES :-**

This cron job will run every minute, all the time:

\* \* \* \* \* [command]

This will run on Mondays, every hour (i.e. 24 times in one day, but only on Mondays):

0 \* \* \* 1 [command]

Division operator is also used. This will run 12 times per hour, i.e. every 5 minutes:

\*/5 \* \* \* \* [command]

To run Script sample.sh at 12.59 every day and supress the output

59 12 \* \* \* sh /usr/bin/sample.sh

To run sample.sh everyday at 9pm (21:00)

0 21 \* \* \* sh /usr/bin/sample.sh

To run sample.sh every Tuesday to Saturday at 1am (01:00)

0 1 \* \* 2-7 sh /usr/bin/sample.sh

To run sample.sh at 07:30, 09:30 13:30 and 15:30

30 07,09,13,15 \* \* \* sh sample.sh

What if you'd want to run something every 10 minutes? Well you could do this:

0,10,20,30,40,50 \* \* \* \* sh /usr/bin/sample.sh

But crontab allows you to do this as well:

\*/10 \* \* \* \* sh /usr/bin/sample.sh

Disable Email Notification.

\* \* \* \* \* >/dev/null 2>&1

To run /path/to/command five minutes after midnight, every day, enter:  
5 0 \* \* \* /path/to/command

Run script.sh at 2:15pm on the first of every month, enter:  
15 14 1 \* \* sh /usr/bin/sample.sh

Run script.php at 10 pm on weekdays, enter:  
0 22 \* \* 1-5 sh /usr/bin/sample.sh

Schedule a cron to execute at 2am daily.

0 2 \* \* \* sh /usr/bin/sample.sh

##### Schedule a cron to execute on every Sunday at 5 PM.

0 17 \* \* sun sh /usr/bin/sample.sh

Schedule a cron to execute on selected months.

\* \* \* jan,may,aug \* sh /usr/bin/sample.sh

***Multiple Commands with Double amper-sand(&&)***

@daily <command1> && <command2>

**Remote Backup with SCP**

From the remote server to which the LoadMaster backups will be sent:

Run the **ssh-keygen** command to generate the public/private rsa key pair.

Do not assign a passphrase (leave the value empty).

By default this will create the following files in the /home/user/.ssh/ directory:

**id\_rsa** (private key file) - this file will be uploaded to the LoadMaster.

**id\_rsa.pub** (public key file) - this value needs to be copied into the appropriate files on the remote host.

Run the **ssh-copy-id** command to copy the public key information into the “authorized\_keys” and “known\_hosts” files.

**ssh-copy-id user@server**

The /home/user/.ssh directory now has the following files.

authorized\_keys  id\_rsa  id\_rsa.pub  known\_hosts

Export the private key file (id\_rsa) from the server. This will be used for the **Remote Identity value** on the LoadMaster.

Be sure to create a backup directory on the server and enter this path under Remote Pathname on the LoadMaster. Ex: /home/user/LMbackups