**Unit 3: Process Management**

**Process Management:-**

Linux is a multitasking OS, i.e. many processes can be running at the same time. As well as there may be other users processes running on the system and the OS itself also be running various processes which it uses to manage everything.

A program in execution is called a process. There can be multiple processes

related with the same program and operating simultaneously without interfering with each other. The kernel is responsible for the management of the processes. It

determines the time and priorities that are allocated to processes so that more than

one process can share the CPU resources.

The kernel must keep track of the following data for each process on the system:

 See all the processes (user or system) that are running in system

 View the system resources consumed by the processes i.e.

 Locate a particular process and take specific action on it

 Change the priority levels associated with processes

 Kill or terminate the required processes if it has outlived (live longer then) its

usefulness or if it’s misbehaving (bad).

 Restrict the system resources available to processes etc

When you execute a program on your Linux system, the system creates a special environment for that program. This environment contains everything needed for the system to run the program. i.e. it creates a “process” on the system. Every process is assigned a unique serial number called its process id or PID.

**Two important attributes of a process are:**

 **The Process-Id (PID):**Each process is uniquely identified by a unique integer

called the PID that is allocated by the kernel when the process is born. The PID

can be used to control a process. E.g. to kill it.

 ***The Parent PID (PPID):*** The PID of Parent is also available as a process

attribute. When several processes have the same PPID, it often makes a sense to

kill its parent instead of all children separately.

Processes as a living being can born, can give birth, and also can die.

There are three types of processes:

 **Interactive*:*** Initiated by a shell and running in the foreground or background.

 ***Batch:*** a series of processes scheduled for execution at a specified point in time.

 **daemon:** Typically initiated at boot time to perform operating system functions

on demand, such as LPD, NFS, and DNS.

**Shell process :-**

A process is set up by the kernel. This process represents the login shell, which can be either sh(Bourne Shell), ksh(korn Shell), bash(Bourne Again Shell) or csh(C Shell). Any command you type in at the prompt is actually the standard input to the shell process. This process remains active until you log out, when it is killed by the kernel. The shell maintains a set of environment variables, like PATH and SHELL. The shell’s pathname is stored in SHELL, but its PID is stored in a special “variable “, $$. To know the pid of the current shell, type

**$ echo $$**

**Parents and Children:**

When you enter an external command at the prompt, the shell acts as the parent

process, which in turn starts the process representing the command entered. Since

every parent has a parent, the ultimate origin of any process can be traced back to the first process (PID 0) that is set up when the system is booted. It is similar to the root directory of the file system. A process can have only one parent. But a process can produce multiple child processes.

**A parent process can have two approaches for its child:**

 It may wait for the child to die so that it can spawn the next process. The death of the child is intimated to the parent by the kernel. Shell is an example of a parent

that waits for the child to terminate. However, the shell can be told not to wait for

the child to terminate.

 It may not wait for the child to terminate and may continue to spawn other

processes. init process is an example of such a parent process.

Every process has a parent. A process born from it is said to be child.

Example: When you run a command:

$ cat emp

This process will get started by shell process. It will remain active till the command is active. The shell (sh, ksh, bash, csh) is said to be its parent of cat. Every process have and parent, no process can be object.

$ cat emp | grep ‘director’

Sets up two processes for the two commands. In Linux multitasking nature permits a process to generate one or more children.

** Process status (ps):**

ps command is used to display some process attributes.ps can be seen as the process counterpart of the file system’s ls command. The Command reads through the kernel’s data structure and process tables to fetch the characteristics of processes. ps displays the processes associated with a user at the terminal. If you execute the command immediately after logging in. Like who ps also displays header information. It shows PID,TTY(terminal with which the process is associated),The cumulative processor time(TIME),CMD(and the process name)

Options:

|  |  |
| --- | --- |
| Full Listing(-f) | To get a detailed listing which also shows the parent of every process |
| Displaying Processes  of a User(-u) | System administrator uses it to know the activities of any  user |
| Displaying All User  Processes(-a) | Lists the processes of all users but doesn’t display the system processes. |
| System Processes  (-e or –A) | Displays the list of all processes along with the controlling  terminal (TTY), Time and CMD running on your system. |
| -x | Shows information about processes without terminals. |

**Examples :**

***$*  ps**

PID TTY TIME CMD

4245 pts/7 00:00:00 bash

5314 pts/7 00:00:00 ps

**$ ps -f // detailed listing of processes**

UID PID PPID C STIME TTY TIME COMMAND

Root 14931 136 0 08:37:48 ttys0 0:00 rlogind

Sartin 14932 14931 0 08:37:50 ttys0 0:00 -sh

sartin 15339 14932 7 16:32:29 ttys0 0:00 ps –f

The header includes the following information:

|  |  |
| --- | --- |
| **Column** | **Description** |
| **UID** | User ID that is Login name of the user |
| **PID** | Process ID. |
| **PPID** | Parent process ID (the ID of the process that started it). |
| **C** | CPU utilization of process. |
| **STIME** | Starting time of the process in hours, minutes and seconds |
| **TTY** | Terminal ID number |
| **TIME** | CPU time taken by the process. |
| **CMD** | The name of the command being executed |

Displays the list of all processes along with the controlling terminal (TTY), Time

and CMD running on your system.

**Multiple jobs in background and foreground**

You not only have control over a command’s input and output, but also over its

execution. You can run a job in the background while you execute other commands.

You can also cancel commands before they have finished executing. You can even

interrupt a command. A job is a name given to a group of processes that is typically

created by piping a series of commands using pipeline character. You can use job

control facilities to manipulate jobs. You can use job control facilities to,

 Transfer a job to the background (bg)

 List the active jobs (jobs)

 Bring jobs to the foreground (fg)

 Suspending & stopping a foreground job ([Ctrl-z])

 Kill a job (kill)

 **Running Jobs in the Background**

Background Processes run in the background and usually do not need of user input.

Example: Antivirus.

You execute a command in the background by placing an ampersand (&) at the end of the command. When you place a job in the background the shell displays, a ***user job number* i.e. *job ID*** and the ***system process number (PID)*** are displayed. The user job number placed in square bracket is the number by which the user references the job. The system process number is the number by which the system identifies the job.

***Example***: the command to print the file **mydata** is placed in the background:

$ **lpr mydata &**

[1] 534

**$ cat > a &**

[2] 548

**$ cat > b &**

[3] 554

The advantage of running a process in the background is that you can run other

commands; you do not have to wait until it completes to start another

**Advantages of Background Process (Jobs)**

 It reduces manual effort & automates the task.

 It can be scheduled as per user's choice.

 It reduces user interaction and can run effortlessly in the background without

user input

 Once you define a job or process in the background job, the user doesn't have to

worry about value input in the field. Thus, user confusion is also reduced.

 Ideal for time- consuming or resource intensive programs which can be scheduled

to run in the night (when system load is low).

**List the active jobs (jobs)**

To display the status of jobs in the current shell. Lists the jobs running in the

background, giving the *job number*.

|  |  |
| --- | --- |
| **Option** | **Description** |
| **-l** | Show process id's in addition to the normal information with list |
| **-p** | Show process id's only. |
| **-n** | Show only processes that have changed status since the last notification are printed. |
| **-r** | Restrict output to running jobs only. |
| **-s** | Restrict output to stopped jobs only |

**$ jobs**

[1] Running lpr mydata &

[2] - Running cat > a &

[3] + Running cat > b &

 **Bringing Jobs to the Foreground**

Foreground Processes run on the screen and need of input from the user. For example: Office Programs. Every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.

You can bring a job from the background to foreground just use **fg** command. If

only one job is in the background, the **fg** command only bring it to the foreground. But more than one job is in the background, you want to bring a certain job to the

foreground, execute jobs command which will show the job id and command.

You must use the job’s number (job ID) with the **fg** command. You place the job

number after the **fg** command, preceded by a **percent sign (%)**. A **bg** command,

usually used for interrupted jobs, places a job in the background.

***Example:*** the second job is brought into the foreground.

**$ jobs**

[1] Running lpr mydata &

[2] - Running cat > a &

[3] + Running cat > b &

**$ fg %2**

cat > a // bring a job background to foreground and its

executing in terminal

 **Suspending and Stopping foreground Jobs**

You can suspend a job and stop it with the **CTRL-Z** key. This command places your job to the side but it is restarted because job is not ended; it just suspended until you want to continue.

When you are ready for continue the execution of job, you can continue with the

job in either the foreground or the background using the **fg or bg** command.

 The **fg** command restarts a suspended job in the foreground.

 The **bg** command places the suspended job in the background.

You can send an already running foreground job to background. You cannot move a

currently running job directly into the background.

*Sending the current foreground job to the background using* ***CTRL-Z*** *and* ***bg*** *command*

 **Press ‘CTRL+Z’** which will suspend the current foreground job.

 Execute **bg** to make that command to execute in background.

***Example:* $ cat > progs**

**^Z // job suspend and stop by Ctrl-z**

**[4] + Stopped cat > progs**

**$ bg // after suspending job transfer into background**

**$ jobs // View all the background jobs**

[1] Running lpr mydata &

[3] - Running cat > b &

**[4] + Stopped cat > progs // suspended job add at background**

You can bring a background job to the foreground using **fg command**. When executed without arguments, it will take the most recent background job to the foreground.

$ **fg**

If you have multiple background jobs, and would want to bring a certain job to the

foreground, execute jobs command which will show the job id and command.

**Changing process priority with nice:-**

On a Linux machine, there are hundreds of processes that are continuously running

for some other tasks. Linux Kernel does a good job in between these processes and

allotting CPU to these processes i.e. kernel allocates CPU to all the process. Linux can **run a lot of processes at a time**, which can **slow down the speed of some high priority processes** and result in **poor performance**. To avoid this problem, you can set or change the **priority of processes as per your requirements by using *nice command.*** *This* ***priority is called Niceness in Linux*** and it has a **value between - 20 to 19**. The **default value** of all the processes **is 0**. A nice value of -20 represents highest priority and a nice value of 19 represent least priority for a process

Nice command will start a process with a user defined scheduling priority. If you

give a process a higher priority, then Kernel will allocate more CPU time to that

process.

***Why change the priority of process***

To adjust the nice value cause of following two reason:

 The first is when you have a process that is or may cause resource disagreement;

we would increase the processes niceness value.

 The second is when you want to increase the resources of a specific process in

order to decrease the run time or give a process higher priority. We would

decrease the processes nice value.

***Syntax: $* nice -n 'Nice value' process name**

**top** command displays processor activity of your Linux box and also displays tasks

managed by kernel in real-time. It will show processor and memory are being used and other information like running processes. This may help you to take correct action.

**$ top -u bcs // Display Specific User Process**

 **Finding the current nice value**

Before we start changing nice values I want to go over identifying what the current

nice values are.

**Determining the default nice value of new processes**

Different OS distributions can have different default values for new processes. The

simplest method to determine the default value is to simply run the nice command

with no arguments. By default nice will simply return the current niceness value.

***Example:* $ nice // default nice value is 0**

0

**Determining the nice value of a current process**

The nice values of current processes are also pretty simple to find as they are visible in the ps command's long format.

 **Changing the nice value of a running process**

To change the nice value of a running process we will utilize the renice command. The usage is similar to nice however rather than supplying a command to run we will be supplying a process id

**Listing processes**

You have a process in the background; it would be helpful to display a list of the

processes, we can use either the **jobs** command or the more powerful **ps** command.

You can also cancel a job using the system process number, which you can obtain with the **ps** command. The **ps** command will display your processes, and you can use a process number to end any running process. The **ps** command displays a great deal more information than the jobs command does.

**The ps Command**

The *ps* (i.e., *process status*) [command](http://www.linfo.org/command.html) is used to provide information about the currently running [*processes*](http://www.linfo.org/process.html), including their*process identification numbers* (PIDs).

A process, also referred to as a *task*, is an *executing* (i.e., running) instance of a program. Every process is assigned a unique PID by the system.

The basic syntax of ps is

ps [options]

When ps is used without any options, it sends to [*standard output*](http://www.linfo.org/standard_output.html), which is the display monitor by default, four items of information for at least two processes currently on the system: the [*shell*](http://www.linfo.org/shell.html) and ps. A shell is a program that provides the traditional, text-only user interface in [Unix-like](http://www.linfo.org/unix-like.html) [operating systems](http://www.linfo.org/operating_systems_list.html) for issuing commands and interacting with the system, and it is *bash* by default on [Linux](http://www.linfo.org/linuxdef.html). ps itself is a process and it *dies* (i.e., is terminated) as soon as its output is displayed.

The four items are labeled PID, TTY, TIME and CMD. TIME is the amount of CPU (central processing unit) time in minutes and seconds that the process has been running. CMD is the name of the command that launched the process.

TTY (which now stands for *terminal type* but originally stood for *teletype*) is the name of the [*console*](http://www.linfo.org/console.html) or *terminal* (i.e., combination of monitor and keyboard) that the user [logged into](http://www.linfo.org/login_def.html), which can also be found by using the *tty* command. This information is generally only useful on a multi-user network.

A common and convenient way of using ps to obtain much more complete information about the processes currently on the system is to use the following:

ps -aux | less

The *-a* option tells ps to list the processes of all users on the system rather than just those of the current user, with the exception of *group leaders* and processes not associated with a terminal. A group leader is the first member of a group of related processes.

The *-u* option tells ps to provide detailed information about each process. The *-x* option adds to the list processes that have no controlling terminal, such as *daemons*, which are programs that are launched during *booting* (i.e., computer startup) and run unobtrusively in the background until they are activated by a particular event or condition.

In contrast to most commands, the hyphen preceding ps's options is optional, not mandatory. Thus, the following could be (and sometimes is) used in place of the above command:

ps aux | less

An alternative set of options for viewing all the processes running on a system is

ps -ef | less

The *-e* option generates a list of information about every process currently running. The *-f* option generates a listing that contains fewer items of information for each process than the *-l* option.

The processes shown by ps can be limited to those belonging to any given user by piping the output through *grep*, a [filter](http://www.linfo.org/filters.html) that is used for searching text. For example, processes belonging to a user with a username *adam* can be displayed with the following:

ps -ef | grep adam

The -l option generates a *long* listing, and when used together with the -e and -f options creates a table with 15 columns:

ps –efl

ps is most often used to obtain the PID of a malfunctioning process in order to terminate it with the *kill* command. For example, if the PID of a *frozen* or *crashed*program is found to be 1125, the following can usually terminate the process:

kill 1125

ps -ef or ps -efl can then be used to confirm that the process really has stopped. If it has not, then the more forceful *-9* option should be used, i.e.,

kill -9 1125

The *[pstree](http://www.linfo.org/pstree.html)* command is similar to ps in that it can be used to show all of the processes on the system along with their PIDs. However, it differs in that it presents its output in a *tree diagram* that shows how processes are related to each other and in that it provides less detailed information about each process than does ps.

**Kill a job (Canceling Jobs) : Premature termination of process**

If you want to cancel a job running in the background, you can force it to end with the kill command. The kill command takes as its argument either the user job number or the system process number. The user job number must be preceded by a percent sign (%). You can find out the job number from the **jobs command or ps command**.

***Example:*** the jobs command lists the background jobs; then job 2 is canceled:

**$ kill %3 // Use the user job number with preceding (%)sign**

**Or**

**$ kill 554 // Use the process number without any preceding (%)sign**

**$ kill -9 534 // kill -9 is used to forcefully terminate a process**

Terminated

Linux is normally used to kill a suspended or hanged process or process group. The kill is a internal command.

 **Premature Termination Of A Process**

It is an internal command used to kill the process. The command uses one o more

PID’s as its arguments, and by default uses the sigterm(15) signal.

**$ kill 111**

Here 111 is PID of a process to facilitate premature termination, the & operator

displays the PID of the process that runs in the background. If you don’t remember the PID the use ps command to know then kill.

If you want to kill more than one job either in the background or in different windows ,then all can be killed using single kill statements with their respective id’s

**$ kill 121 111 125 138 144**

**Disk management and System Administration**

**Disk Partitioning - RAID , LVM**

*Disk partitioning is dividing a hard disk drive into multiple logical storage units is called as partitions*

A partition is a contiguous set of blocks on a drive that are treated as an independent disk. Every partition has a type which identifies the kind of data

that it stores. There is a type for Linux file systems, a type for Linux swap space &

own partition type for its own file systems. Each hard disk can only contain four primary partitions. Because this is often not enough, it is possible for one of those four to be a special extended partition that can contain an unlimited number of logical partitions. If you make use of an extended partition, there is effectively no limit on the number that your hard disk can contain.

There are many benefits of having multiple partitions on a disk. To separate the

operating system and program files from user files. Many Linux users prefer having

the /home directory on a separate partition. It is also common to have several

partitions on a hard drive, each of which stores an operating system. This enables

users to install and run multiple operating systems on a computer without using

virtualization.



When you are changing or reformatting any existing partitions on your system be

careful. Because of file system data will be loss & your system unbootable.

Disk partition is open opportunities for storage reliability as well as organizing access to your hard disk. Linux provides two methods for better managing your hard disks

**1) Logical Volume Management (LVM)**

**2) Redundant Arrays of Independent Disks (RAID)**

**Redundant Arrays of Independent Disks (RAID)**

RAID is a way of storing the same data in different places on multiple hard disks.

These multiple hard drives are treated as a single hard drive. RAID is a method for combining multiple partitions on different disks into one large virtual device, also

known as a RAID array. They include recovery information that allows you to restore your files should one of the drives fail. RAID can provide data protection. RAID is best suited to desktops and servers that hold multiple hard drives and require data recovery. It provides the best protection against hard drive failure and is considered a necessity for storage-intensive tasks such as enterprise, database, and Internet server operations. It can also provide peace of mind for smaller operations, providing recovery from hard disk failure.

RAID is a method of storing data across several disks to provide greater

performance and redundancy and a several hard disks that are treated as one virtual disk, where some of the disks are used as real-time mirrors, duplicating data. RAID then efficiently stores and retrieves data across all these disks, instead of having the operating system access each one as a separate file system. This allows greater flexibility in adding or removing as well as implementing redundancy in the storage system to provide greater reliability. When you place data on multiple disks, I/O operations can overlap in a balanced way, improving performance. Because having multiple disks increases the mean time between failures (MTBF), storing data redundantly also increases fault acceptance.

RAID can be implemented on a hardware or software level. Linux software RAID

supports five levels (linear, 0, 1, 4, 5, and 6).RAID can provide faster access for

applications that work with very large files, such as multimedia, database, or graphics applications.

Booting from a RAID Device you can create RAID devices from which you can

also boot your system. Your Linux system will be configured to load RAID kernel

support and automatically detect your RAID devices. The boot loader will be installed on your RAID device, meaning on all the hard disks making up that device. You use the *mdadm* tool to manage and monitor RAID devices. The ***mdadm*** tool is an all purpose means of creating, monitoring, administering, and fixing RAID devices. You can run commands directly to create and format RAID disks. It also runs as a daemon to monitor and detect problems with the devices.

 **Purpose of RAID**

RAID is used to increase the logical capacity of storage devices used, improve

read/write performance and ensure redundancy in case of a hard disk failure. All these needs can be addressed by other means, usually more expensive than the RAID configuration of several hard disks. The adjective Inexpensive used in the name is not without a reason.

 **Advantages of RAID**

The major pluses of RAID are the cost and flexibility. It is possible to dynamically

adjust to the growing or changing needs of a storage center, server performance or

machine backup requirements merely by changing parameters in software, without physically touching the hardware. This makes RAID more easily implemented than

equivalent hardware solutions.

Improved performance can be achieved by buying better, faster hard disks and

using them instead of the old ones. This necessitates spending money, turning off the machine, swapping out physical components, and performing a new installation. RAID can achieve the same with only a new installation required. In general, advantages include:

 Improved read/write performance in some RAID configurations.

 Improved redundancy in the case of a failure in some RAID configurations.

 Increased flexibility in hard disk & partition layout.

 **Disadvantages of RAID**

The problems with RAID are directly related to their advantages. For instance, while RAID can improve performance, this setup necessarily reduces the safety of the implementation. On the other hand, with increased redundancy, space efficiency is reduced. Other possible problems with RAID include:

 Increased wear of hard disks, leading to an increased failure rate.

 Lack of compatibility with other hardware components and some software, like

system imaging programs.

 Greater difficulty in performing backups and system rescue/restore in the case of

a failure.

 Support by operating systems expected to use the RAID.

**Logical Volume Manager (LVM)**

LVM is a virtual layer that sits between the operating system and your storage

devices, i.e., hard disks. It creates groups of your hard disks, called Volume Groups

(VGs). Within these VGs, you can create partitions, called Logical Volumes (LVs). The benefits of LVM over regular file systems include: dynamic resizing of VGs, and live snapshots of partitions without unmounting them. LVM is a method for organizing all your hard disks into logical volumes. The storage capabilities of several hard disks into a single logical volume. Your system then sees one large storage device. LVM is possibly the most effective way to add hard drives to your system, creating a large accessible collection of storage. LVM provides flexibility.



With LVM, you have no longer to keep track of separate disks and their

partitions, trying to remember where files are stored on what partitions located in

what drive. Partitions and their drives are combined into logical file systems that you can attach to your system directory tree. You can have several logical file systems, each with their own drives and or partitions. LVM has the added advantage of letting you implement several logical file systems on different partitions across several hard drives.

For easier hard disk storage management, you can set up your system to use the

Logical Volume Manager (LVM), creating LVM partitions that are organized into

logical volumes to which free space is automatically allocated. Logical volumes provide a more flexible and powerful way of dealing with disk storage, organizing physical partitions into logical volumes in which you can easily manage disk space. Disk storage for a logical volume is treated as one pool of memory, though the volume may in fact contain several hard disk partitions spread across different hard disks. Adding a new LVM partition simply increases the pool of storage accessible to the entire system.

 **LVM advantages**

 Use any number of disks as one big disk.

 Have logical volumes **extended over several disks**.

 Create small logical volumes and resize them "dynamically" as they get more

filled.

 Resize logical volumes regardless of their order on disk. It does not depend on

the position of the LV within VG, there is no need to ensure surrounding

available space.

 **Resize/create/delete** logical and physical volumes online. Filesystems on them

still need to be resized, but some support online resizing.

 **Online migration** of LV being used by services to different disks without

having to restart services.

 for normal operations LVM is as efficient as RAID

 **LVM disadvantages:**

 There is no official support in most other OS (FreeBSD, Windows).

 Additional steps in setting up the system more complicated.

 If you want to have a dual boot environment with Windows you have to create

a dedicated old fashion partition which is not managed by LVM.

 If you combine a lot of disks in one VG the probability of a failure of the VG

increase by every disk you add. If one disk fails than usually the whole VG

fails. You have to setup a RAID together with LVM in order to get rid of this

advantage.

 You shouldn’t use a whole disk as a PV (e.g. /dev/sda) .because other

operating systems like Windows may not detect the LVM and view the disk

as empty. That way it's possible to overwrite the LVM PV by accident.

Difference between **RAID and LVM**

|  |  |
| --- | --- |
| **RAID** | **LVM** |
| RAID is used for redundancy. | LVM is a way in which you partition the hard disk logically and it contains its own advantages. |
| A RAID device is a physical grouping of disk devices in order to create a logical presentation of one device to an Operating System for redundancy or performance or a combination of the two. | LVM is a logical layer that that can be  manipulated in order to create and, or  expand a logical presentation of a disk  device to an Operating System. |
| RAID is a way to create a redundant or striped block device with redundancy using other physical block devices | LVM usually sits on top of RAID blocks  or even standard block devices to accomplish the same result as a partitioning, however it is much more flexible than partitions. You can create multiple volumes crossing multiple  physical devices, remove physical  devices without losing data, resize the  volumes, create snapshots, etc |
| RAID is either a software or a hardware technique to create data storage redundancy across multiple block devices  based on required RAID levels | LVM is a software tool to manage large  pool of storage devices making them  appear as a single manageable pool of  storage resource. LVM can be used to  manage a large pool of what we call  Just-a-bunch-of-Disk (JBOD) presenting  them as a single logical volume and  thereby create various partitions for  software RAID. |
| RAID is NOT any kind of Data backup solution. It’s a solution to prevent one of the SPOFs (Single Point of Failure) i.e. DISK failure. By configuring RAID you are just providing an emergency  substitute for the Primary disk. It  NEVER means that you have configured DATA backup. | LVM is a disk management approach  that allows us to create, extend, reduce,  delete or resize the volume groups or  logical volumes. |

**disk related Management Tools Fdisk, Parted etc**

**Purposes for Disk Partitioning :**

An operating system like Windows / Linux can be installed on a single, unpartitioned hard disk. The ability to divide a hard disk into multiple partitions offers some important advantages. If you are running Linux on server consider following facts:

 **Ease of use** - Make it easier to recover a corrupted file system or operating

system installation.

 **Performance** - Smaller file systems are more efficient. You can tune file system

as per application such as log or cache files. Dedicated swap partition can also

improve the performance (this may not be true with latest Linux kernel 2.6).

 **Security** - Separation of the operating system files from user files may result

into a better and secure system. Restrict the growth of certain file systems is

possible using various techniques.

 **Backup and Recovery** - Easier backup and recovery.

 **Stability and efficiency** - You can increase disk space efficiency by formatting

disk with various block sizes. It depends upon usage. For example, if the data is

lots of small files, it is better to use small block size.

 **Testing** - Boot multiple operating systems such as Linux, Windows and

FreeBSD from a single hard disk.

**Management Tools: fdisk, parted**

Linux provides a variety of tools for creating and managing file systems. If you add

new hard disk partitions, create CD images, and format floppies. To use a new hard

drive, you will have to first partition it and then create a file system on it. You can use either **parted or fdisk** to partition your hard drive. To create the file system on the partitions, you use the **mkfs** command, which is a front end for various file system builders

**fdisk:-**

It is used for creates and deletes partitions. **fdisk** on the command line with the device name of the hard disk you are partitioning also you can use to create your Linux partition. But you use careful fdisk because it can exactly erase entire hard disk partitions and all the data on those partitions if you are not use careful command. fdisk for creating partitions on the hdb hard drive.

**$ fdisk /dev/hdb**

The partitions have different types that you need to specify. Linux fdisk is a line

oriented program

**Command Action**

**l** List known partition types

**m** List commands

**n** Add a new partition

**p** Print the partition table

**q** Quit without saving changes

**t** Change a partition’s system ID

**w** Write table to disk and exit

**$ fdisk -l**

**$ fdisk /dev/sdb**

**Entering an m will display the program menu:**

**Command: p**

Disk /dev/sdb: 16 MB, 16006656 bytes

1 heads, 31 sectors/track, 1008 cylinders

Units = cylinders of 31 \* 512 = 15872 bytes

**Command (m for help): w**

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x partitions, please see the fdisk manual page for additional information. Syncing disks.

**Command (m for help): t**

Selected partition 1

Hex code (type L to list codes): 83

Changed system type of partition 1 to 83 (Linux)

**Parted**

Parted is a disk partitioning and partition resize program. This command allows

managing hard disk partitions, creating new partition, and deleting old partition,

resize, move and copy of partition. The utility of parted command allows to users view the existing partition table, change the of existing partition and add partition from free space.

**$ parted /dev/had**

To use parted on the partitions in a given hard drive, if none of the partitions on that drive can be in use. If you wish to use parted on partitions located on that same hard drive as your kernel, you have to boot your system in rescue mode and choose not to mount your system files.

**Options:**

|  |  |
| --- | --- |
| **print** | Command to list all your partitions |
| **Mkpart** | To create a new partition, use the command with either primary or extended, the file system type, and the beginning and end positions. You can create up to three primary partitions and one extended partition |
| **Rm** | To remove a partition with the partition number. |
| **resize** | To resize a partition with the partition number and the beginning and end positions. |
| **move** | To move a partition using the move command |
| **help** | Lists all commands |

**Examples:**

**$ (parted) print**

Model: ATA WDC WD5000BPVT-7 (scsi)

Disk /dev/sda: 500GB

Sector size (logical/physical): 512B/4096B

Partition Table: msdos

**$ (parted) mkpart primary 106 16179**

**$ (parted) resize 9**

WARNING: you are attempting to use parted to operate on (resize) a file system. Support for performing most operations on most types of file systems will be removed in an upcoming release.

Start? [373GB]? 373GB

End? [500GB]? 450GB

**$ (parted) rm**

Partition number? 9

(Parted)

**Boot Loaders:-**

*A Boot loader is a program that helps load and install the operating system.* A boot

loader, also called a boot manager, is a small program that places the operating system (OS) of a computer into memory. The boot loader is the first software program that runs when a computer starts. It is responsible for handing over control of the system to the operating system. Typically, the boot loader will reside in the Master Boot Record (MBR) of the disk, and it knows how to get the operating system up and running. The two most common boot loaders are known as **LILO** (**LI**nux **LO**ader) and **GRUB** (**GR**and **U**nified **B**ootloader) is used in Linux.

GRUB is the most common boot loader that distribute with the newer

distributions of Linux and it also has a lot more features than LILO.*GRUB is default boot loader*. LILO is made for historical reasons only. Both LILO and GRUB can be configured to boot other non-native operating systems.

Using a boot loader offers several benefits including

 The ability to boot multiple operating systems

 The ability to pass parameters interactively to the kernel, which is useful for

disabling certain features in order to solve hardware problems

 The ability to load different kernels interactively, which is useful when you

deploy a new kernel because it is easy to revert to the working kernel.

**GRUB (GRand Unified Bootloader)**

GRUB is a boot loader package developed to support multiple operating systems

and allow the user to select among them during booting time. you have the ability to

load different versions of the Linux kernel as well as other operating systems that you have installed on your system. The task of selecting and starting up an operating system or kernel is managed by a boot management utility, the GRUB. This is a versatile tool, letting you load operating systems that share the same disk drive, as well as letting you choose from different Linux kernels that may be installed on the same Linux system. The actual configuration of GRUB is based on four files that are Three GRUB interfaces provide different levels of functionality:

 menu interface

 menu entry editor interface

 command- line interface

**The GRUB configuration file**

 **/boot/grub/menu.lst -** This file contains all information about partitions or

operating systems that can be booted with GRUB.

 **/boot/grub/device.map -** This file translates device names from the GRUB

and BIOS notation to Linux device names

 **/etc/grub.conf -** This file contains the commands, parameters and options

the GRUB shell needs for installing the boot loader correctly.

 **/etc/sysconfig/bootloader -** when configuring the bootloader with YaST

and every time a new kernel is installed. It contains configuration options (such

as kernel parameters) that will be added by default to the boot loader

configuration file.

GRUB uses a configuration file – found at /boot/grub/grub.conf – to create the list of

operating systems to display in its menu interface.

The configuration file enables you to choose a group of commands that you want

GRUB to execute.

You can use the same GRUB commands that you use in the command- line interface, as well as certain commands that are available for use only in the configuration file.

Commands that you can use only in the configuration file include

 *default –* to specify the default entry title name or number that GRUB should

load as the default

 *timeout –* to set the time limit in seconds before GRUB starts the default

operating system

 *splashimage –* to set the default GRUB image

 *title –* to group the commands used to boot a particular operating system under

a particular title in the GRUB menu

Other configuration file commands that you can use are

 password – to prevent unauthorized users from editing the menu option entries

 color – to set the menu color scheme

 hiddenmenu – to load the default entry without displaying the GRUB menu

Interface

In the GRUB configuration file, the commands that affect the menu interface are

placed at the top. These are followed by entries that affect the booting of each of the

operating systems or kernels listed.

default=0 # default to the first entry

timeout=10 # set the timeout to 10 seconds

plashimage=(hd0,0)/grub/splash.xpm.gz # the splash image displayed with

menu

You make your new entries in grub configuration file & grub will read automatically when you can have many kernels in the /boot directory and use GRUB to choose the one of them.

 **GRUB has the following features**

 Recognize multiple executable formats and Support non-Multiboot kernels

 Load multiples modules with a configuration file

 Provide a menu interface

 Have a flexible command-line interface

 Support multiple filesystem types

 Support network booting & remote terminals

 **GRUB Advantages**

The GRUB bootloader is more powerful than LILO and supports larger file systems.

It has a fully interactive command line interface. Users navigate a menu system where they can choose which operating system they want to load. GRUB supports more file systems than LILO including DOS FAT, BSD FFS and Linux file systems. If the configuration file is not configured properly, GRUB will revert to a command line interface rather than rendering the system unbootable.

**LILO [Linux Loader]**

LILO is a boot loader for Linux. *LILO does not depend on a specific file system,*

*and can boot an operating system (e.g., Linux kernel images) from floppy disks and*

*hard disks.* One of up to sixteen different images can be selected at boot time. Various parameters, such as the root device, can be set independently for each kernel. *LILO can be placed either in the master boot record (MBR) or the boot sector of a partition*

At system start, only the BIOS drivers are available for LILO to access hard disks. For this reason, with very old BIOS, the accessible area is limited to cylinders 0 to 1023 of the first two hard disks. For later BIOS, LILO can use 32-bit "logical block

addressing" (LBA) to access practically the entire storage of all the hard disks that the BIOS allows access to.

LILO was the default boot loader for most Linux distributions in the years after the

popularity of loading. Today, most distributions use GRUB as the default boot loader.

LILO is that it two aspects: The boot loader and the LILO command. The LILO

command configures and installs the boot loader and updates it as necessary. The boot loader is the code that executes at system boot time and boot Linux or another

operating system.

**LILO Configuration Files:**

/***etc/lilo.conf :***

The *lilo.conf* file is typically located at */etc/lilo.conf*. Within *lilo.conf* there are

typically two section types. The first section, which defines the global options, contains parameters which specify boot location attributes. The second section(s) contain parameters associated with the operating system images to be loaded. The section type can be repeated for up to 16 different boot selections.

*/boot/*

LILO stores a number of files in the /boot/

*Master Boot Record :*

LILO can write a Master Boot Record on a device

At any time you change or rebuilt or move a kernel image. You need to rerun

LILO to rebuilt the map file and update it.

In the global section there is one section of options for each Linux kernel and for

each non Linux operating system that you want LILO to be able to boot. Each of these sections is referred as an image section, because each boots a different kernel image or other operating system. Each image section begins with an image.

image = /boot /vmlinr # Linux image

label = Linux # labels that appear at the boot pro mot.

root = /dev/hda2 # location of the root file system.

vga = ask # always prompt the user for vga mode.

read only # mount read only to run fsck for a file system

LILO is one of the most widely-used, well-tested and dependable Linux applications

ever written. Most experienced system administrators are well-versed in configuring the LILO and skilled enough to deal with any emergency situation

**Difference between GRUB and LILO**

|  |  |
| --- | --- |
| **GRUB** | **LILO** |
| GRUB supports network booting | LILO does not support network booting. |
| GRUB supports the use of MD5 password encryption | LILO only supports text passwords |
| GRUB supports an unlimited number of  boot entries | LILO supports only up to 16 different boot selections. |
| GRUB dynamically change configuration  file. it is much easier to correct a  misconfigured configuration file in GRUB | LILO must be written again every time you change the configuration file. |
| GRUB has a more powerful, interactive  command line interface. | LILO does not have an interactive command  interface |
| GRUB can boot operating systems from  external devices such as floppy disks and  hard drives | LILO can boot operating systems from  external devices such as floppy disks and  hard drives |
| GRUB is more complicated to use | LILO is simpler and easier to use |
| GRUB is the new default boot loader | LILO is the old default boot loader for Linux |
| Very good technical support | Low technical support |

**System administration**

Linux is providing too many users at the same time, providing an interface between

the users and the system with its resources, services, and devices. Users have their

own shells through which they interact with the operating system, but you may need to configure the operating system itself in different ways. You may need to add new users, devices like printers and scanners, and even file systems. Such operations come under the heading of system administration. The person who performs such actions is a system administrator or a superuser. Two types of interaction with Linux: regular users’ interactions, and those of the superuser, who performs system administration tasks such as changing system run levels, managing users, configuring printers, adding file systems, and compiling the kernel. You perform most of these tasks only rarely, such as adding a new printer or mounting a file system. Other tasks, such as adding or removing users, you perform on a regular basis. System administration such as system access by superusers, selecting the runlevel to start, system configuration files, and performance monitoring.

**Common administrative tasks**

The common administrative tasks are affecting users and group’s management.

 Creating & deleting a User Account

 Changing a User's Name

 Changing a User Account Password

 Configuring Group

 Creating & deleting a group

 Adding a member to a group

 Removing a member from a group

 Configuring Access to Shells

 Changing System Time and Date

 Changing file permissions and ownership

 Managing software & Scheduling Tasks

 System Runlevels: telinit, initab, and shutdown

**Identifying administrative files**

1. **/etc/passwd:** Keeps user account and password information. This file holds the

majority of information about accounts on the Linux system.

2. **/etc/shadow:** Holds the encrypted password of the corresponding account. Not

all the system supports this file.

3. **/etc/group:** This file contains the group information for each account.

4. **/etc/gshadow:** This file contains secure group account information

**Configuration and log files**

When you configure different elements of your system, such as users, applications,

servers, or network connections, you use configuration files kept in certain system

directories. Configuration files are placed in the /etc directory

**Administrative log files**

The main utilities for logging error and debugging messages for Linux are the

***syslogd*** and ***klogd*** daemons. General system logging is done by ***syslogd***. Logging that is specific to kernel activity is done by ***klogd***. Logging is done according to information in the ***/etc/syslog.conf*** file.

Messages are typically directed to log files that are usually in the ***/var/log*** directory. Here are a few common log files:

 **boot.log—**Contains boot messages about services as they start up.

 **Messages —**Contains many general informational messages about the system.

 **Secure —** Contains security-related messages, such as login activity.

 **XFree86.0.log or Xorg.0.log -** depending on which X server you are using,

contains messages about your video card, mouse, and monitor configuration

**Chkconfig**

The **chkconfig** utility is a command-line tool that allows you to specify in which

runlevel to start a selected service, as well as to list all available services along with

their current setting. The **Chkconfig** command tool allows configuring services start and stopping automatically in the **/etc/rd.d/init.d** scripts through command line.

***Chkconfig has five different functions:***

 Adding new services for management

 Removing services from management

 Listing the current startup information for services

 Changing the startup information for services

 Checking the startup state of a particular service

**System administrator**

The duties of a system administrator are wide-ranging, and vary widely from one

organization to another. *Sysadmins* are usually charged with installing, supporting,

and maintaining servers or other computer systems, and planning for and responding to service outages and other problems. Other duties may include scripting or light programming, project management for systems-related projects.

**The system administrator is responsible for following things**

( Role of system administrator)

1. User administration (setup and maintaining account)

2. Maintaining system

3. Verify that peripherals are working properly

4. Quickly arrange repair for hardware in occasion of hardware failure

5. Monitor system performance

6. Create file systems

7. Install software

8. Create a backup and recovery policy

9. Monitor network communication

10. Update system as soon as new version of OS and application software comes out

11. Implement the policies for the use of the computer system and network

12. Setup security policies for users. A sysadmin must have a strong take hold of

computer security (e.g. firewalls and interruption detection systems)

13. Documentation in form of internal wiki

14. Password and identity management

**What is so special about the system administrator account?**

The root account has full (unrestricted) access, so he/she can do anything with system. For example, root can remove critical system files. In addition, there is no way you can recover file except using tape backup or disk based backup systems.

Many tasks for system administration can be automated using Perl/Python or shell

scripts. For example:

 Create new users

 Resetting user passwords

 Lock/unlock user accounts

 Monitor server security

 Monitor special services etc

**Security Enhanced Linux**

Security tools exist for protecting specific services as well as user information and

data. IT security is a vast subject and covers many areas— network security

(firewalls), data security (encryption, backup, etc), computing security (restricting

physical access, patching OS vulnerabilities, etc) and application security. No tool has been available for protecting the entire system at the administrative level.

Application software will have its defect and bugs. If multiple applications

running on the same OS have to be secured, the OS has to play a crucial role in

defining the confines of these applications. Security can only be achieved with better

underlying operating system security that can isolate applications and files used by

each, thus protecting the integrity of the entire system. In most organizations that

have implemented some form of security and have a security policy in place, the

weakest link in the security chain is the systems administrators.

**S**ecurity-**E**nhanced **Linux** (SELinux) is providing built-in administrative

protection for aspects of your entire Linux system. Instead of users to protect their

files or on a specific network program to control access, security measures are built

into the basic file management system and the network access methods. All controls

can be managed directly by a Linux administrator. Security-Enhanced Linux

(SELinux) is a project developed and maintained by the **National Security Agency (NSA).**

The NSA (National Security Agency) of the US developed SELinux, states

 *“The Security-enhanced Linux’s new features are designed to implement the*

*separation of information, based on privacy and integrity requirements.*

 preventing processes from reading data and programs, altering with data and

programs, avoiding application security mechanisms, executing unreliable

programs, or interfering with other processes in violation of the systems security

policy.

 They also help to lock up the potential damage that can be caused by malicious

or defective programs.

 They should also be useful for enabling a single system to be used by users with

opposed security authorizations to access multiple kinds of information with

differing security requirements without compromising those security

requirements.”

Installing and removing packages with rpm command

**RPM** (**Red Hat Package Manager**) is an default open source and most popular package management utility for **Red Hat** based systems like (**RHEL**, **CentOS** and **Fedora**). The tool allows system administrators and users to **install**, **update**, **uninstall**, **query**, **verify** and manage system software packages in **Unix/Linux** operating systems. The RPM formerly known as **.rpm** file, that includes compiled software programs and libraries needed by the packages. This utility only works with packages that built on **.rpm** format.

### Some Facts about RPM (RedHat Package Manager)

1) RPM is free and released under GPL (General Public License).

1. RPM keeps the information of all the installed packages under /var/lib/rpm database.

3)RPM is the only way to install packages under Linux systems, if you’ve

installed packages using source code, then rpm won’t manage it.

4)RPM deals with .rpm files, which contains the actual information about the packages such as: what it is, from where it comes, dependencies info, version info etc.

### There are five basic modes for RPM command

1. Install : It is used to install any RPM package.
2. Remove : It is used to erase, remove or un-install any RPM package.
3. Upgrade : It is used to update the existing RPM package.
4. Verify : It is used to verify an RPM packages.
5. Query : It is used query any RPM package

### How to Install an RPM Package

For installing an rpm software package, use the following command with **-i** option. For example, to install an rpm package called **pidgin-2.7.9-5.el6.2.i686.rpm**.

root@tecmint]# rpm -ivh pidgin-2.7.9-5.el6.2.i686.rpm

Preparing... ########################################### [100%]

1:pidgin ########################################### [100%]

##### **RPM command and options**

1. -i : install a package
2. -v : verbose for a nicer display
3. -h: print hash marks as the package archive is unpacked.

### How to Remove a RPM Package

To un-install an RPM package, for example we use the package name **nx**, not the original package name **nx-3.5.0- 2.el6.centos.i686.rpm**. The **-e (erase)** option is used to remove package.

[root@tecmint]# rpm -evv nx

### How to Remove an RPM Package Without Dependencies

The  **–nodeps** (**Do not check dependencies**) option forcefully remove the rpm package from the system. But keep in mind removing particular package may break other working applications.

[root@tecmint]# rpm -ev --nodeps vsftpd