**Table of Contents**

**Linux System Administration**

**Unit 1 - System Initialization and System Services**

1. Boot Process of a Linux System 2
2. Kernel Initialization 4
3. Types of Kernels 4
4. Basic Commands to manage the Kernel 5

**Unit 2 - Users, Groups and Permissions Administration**

1. Types of Users 8
2. Configuration Files of User Administration 8
3. User Creation 10
4. Group Creation 11
5. Switching Users 12
6. Modifying Existing Users 13
7. Removing Users 13
8. File Permissions 14
9. Chage Policies 18

**Unit 3 - File System Administration**

1. File System 20
2. Creating a Partition 22
3. File System Managing Commands 25
4. RAID Configuration 31
5. Logical Volume Manager (LVM) 34
6. Disk Quotas 37
7. Access Control Lists (ACL's) 39

**Unit 4 - Process Management and Performance Tuning**

1. Process 41
2. Types of Process 43
3. Process States 44
4. Real Time Snapshots of Process 45
5. Tool to Monitor Linux Performance 47
6. System Services 51

**Unit 5 - System Networking Configuration**

1. Basic Commands 54
2. Networking Configuration Files 57

**Unit 1 - SYSTEM INITIALIZATION AND SYSTEM SERVICES**

**The aim of this tip is:**

* Describe the functions of the [BIOS](http://en.kioskea.net/contents/373-bios) from the boot process
* Describe the functions of the boot loader
* Give the list of functions performed by the kernel at boot
* Describe the functions of "init"
* Give the list of run levels (System V) and indicate the function of each
* Turn off and turn on a system at any level of performance

1. **Boot Process of a Linux system**
2. Initializing the BIOS
3. Boot loader
4. Kernel initialization
5. Starting from "init"
6. Loads the services
7. **Initializing the BIOS**

* The BIOS ([Basic Input/output System](http://en.kioskea.net/contents/373-bios)) is the interface between the hardware and software at a very basic level, it provides all the basic instructions used by the [operating system](http://en.kioskea.net/contents/664-operating-system).
* The BIOS begins by executing an auto-ignition test (POST), then it searches for devices.
* After the POST, a boot device is selected from a list that is configurable in the BIOS.
* The BIOS reads and executes the first physical sector of the boot media selected on the system, which is usually contained in the first 512 bytes of hard disk.

1. **Boot loader**

The boot loader is usually contained in the first sector of the disk and then read and executed by the BIOS. The storage space that reads the BIOS is not sufficient to contain all the boot loader but just a part sufficient enough to start the rest of the boot loader, which is usually contained in a configuration file stored elsewhere on the disc.

Hence the start is done in two steps:

* Launch via BIOS
* Launch a file under/boot

The boot loader is designed to load and run the system kernel. The standard boot loader is GRUB but we can also shift to LILO.

1. **Kernel initialization**

The kernel initialization includes:

* The detection and initialization of devices: any device drivers compiled into the kernel are called and try to locate their corresponding devices.
* Mounting the [root](http://en.kioskea.net/contents/328-linux-user-management) file system in read-only mode
* Loading the initial process "init"

The kernel initialization is very rapid, therefore very difficult to follow visually, so to check what happened during kernel initialization, it is possible to read the log that is stored under:

* /var/log/dmesg

Initialize "init"

* "init" is the main process, it will always have a PID value: 1.
* "init" reads its configuration from the /etc/inittab file. that contains the settings for the system at every level of execution.

1. **Run levels**

"init" defines the X level of performance standards for Linux:

* **Level 0**: Stop (not to be attributed to the initdefault)
* **Level 1**, S: single user mode (only the root user can log). Typically used for maintenance.
* **Level 2**: Multi-user mode without NFS [network](http://en.kioskea.net/contents/261-the-concept-of-networks)
* **Level 3**: full mode for multiple users including network
* **Level 4**: User Configurable duplicate but the level 3 by default.
* **Level 5**: X11 (including network)
* **Level 6**: Restart

At the /etc/inittab file level, control of run level is default, after the system is started following this format:

Id :x :initdefault :

Where x is the desired run level after boot.

1. **Loads the Run level (System V)**

The ability to change run level offers easy interaction with administrators, this allows you to switch between different levels of startup. Scripts services are in /etc/rc.d/init.d. Each run level correspond to a /etc/rc.d/rcX.d directory, where X is the run level.

**System Shutdown**

To stop the system, use commands like:

#Shutdown -h now   
#halt   
#poweroff   
#init 0

1. **Kernel Initialization**

The kernel is the essential center of a computer [operating system](http://searchcio-midmarket.techtarget.com/definition/operating-system), the core that provides basic services for all other parts of the operating system. A synonym is *nucleus*. A kernel can be contrasted with a [shell](http://searchenterpriselinux.techtarget.com/definition/shell), the outermost part of an operating system that interacts with user commands. *Kernel* and *shell* are terms used more frequently in [Unix](http://searchenterpriselinux.techtarget.com/definition/Unix) operating systems than in IBM mainframe or Microsoft Windows systems.

Kernel Activities:

* The Kernel task manager allows tasks to run concurrently.
* Managing the computer resources: Kernel allows the other programs to run and use the resources. Resources include i/o devices, CPU, memory.
* Kernel is responsible for Process management. It allows multiple processes to run simultaneously allowing user to multitask.
* Kernel has an access to the systems memory and allows the processes to access the memory when required.
* Processes may also need to access the devices attached to the system. Kernel assists the processes in doing so.
* For the processes to access and make use of these services, system calls are used.

1. **Types Of Kernels**

Kernels may be classified mainly in two categories

1. Monolithic Kernel
2. Micro Kernel
3. **Monolithic Kernels**

Earlier in this type of kernel architecture, all the basic system services like process and memory management, interrupt handling etc were packaged into a single module in kernel space. This type of architecture led to some serious drawbacks like 1) Size of kernel, which was huge. 2)Poor maintainability, which means bug fixing or addition of new features resulted in recompilation of the whole kernel which could consume hours

In a modern day approach to monolithic architecture, the kernel consists of different modules which can be dynamically loaded and un-loaded. This modular approach allows easy extension of OS's capabilities. With this approach, maintainability of kernel became very easy as only the concerned module needs to be loaded and unloaded every time there is a change or bug fix in a particular module. So, there is no need to bring down and recompile the whole kernel for a smallest bit of change. Also, stripping of kernel for various platforms (say for embedded devices etc) became very easy as we can easily unload the module that we do not want.

Linux follows the monolithic modular approach

1. **Micro kernels**

This architecture majorly caters to the problem of ever growing size of kernel code which we could not control in the monolithic approach. This architecture allows some basic services like device driver management, protocol stack, file system etc to run in user space. This reduces the kernel code size and also increases the security and stability of OS as we have the bare minimum code running in kernel. So, if suppose a basic service like network service crashes due to buffer overflow, then only the networking service's memory would be corrupted, leaving the rest of the system still functional.

In this architecture, all the basic OS services which are made part of user space are made to run as servers which are used by other programs in the system through inter process communication (IPC). eg: we have servers for device drivers, network protocol stacks, file systems, graphics, etc. Microkernel servers are essentially daemon programs like any others, except that the kernel grants some of them privileges to interact with parts of physical memory that are otherwise off limits to most programs. This allows some servers, particularly device drivers, to interact directly with hardware. These servers are started at the system start-up.

So, what the bare minimum that micro kernel architecture recommends in kernel space?

* Managing memory protection
* Process scheduling
* Inter Process communication (IPC)

1. **Basic Commands to manage the Kernel**

* **uname** - print system information.

***[root@server02 ~]# uname -a***

Linux server02.flashindia.com 2.6.32-279.el6.x86\_64 #1 SMP Fri Jun 22 12:19:21 UTC 2012 x86\_64 x86\_64 x86\_64 GNU/Linux

* **uname -r** : Print the kernel release

***[root@server02 ~]# uname -r***

2.6.32-279.el6.x86\_64

***[root@server02 ~]# uname -m*** (machine architecture)

x86\_64

**[*root@server02 ~]# uname -s*** (Define the OS)

Linux

***[root@server02 ~]# uname -o***

GNU/Linux

***[root@DBtest2 ~]# cat /etc/redhat-release***  
Red Hat Enterprise Linux Server release 4 (Tikanga)

***[root@DBtest2 ~]# /usr/bin/lsb\_release --d***  
 Description: Red Hat Enterprise Linux Server release 4 (Tikanga)  
 ***[root@DBtest2 ~]# cat /etc/issue***  
 Red Hat Enterprise Linux Server release 5.5 (Tikanga)  
 Kernel \r on an \m

* **lsmod** – List Modules that Loaded Already

lsmod command will list modules that are already loaded in the kernel as shown below.

***# lsmod***  
Module Size Used by  
ppp\_deflate 12806 0   
zlib\_deflate 26445 1 ppp\_deflate  
bsd\_comp 12785 0

* **insmod** – Insert Module into Kernel

insmod command will insert a new module into the kernel as shown below.

***# insmod*** /lib/modules/3.5.0-19-generic/kernel/fs/squashfs/squashfs.ko  
  
**# lsmod** | ***grep*** "squash"  
squashfs 35834 0

* **modinfo** – Display Module Info

modinfo command will display information about a kernel module as shown below.

***# modinfo*** */*lib/modules/3.5.0-19-generic/kernel/fs/squashfs/squashfs.ko  
filename: /lib/modules/3.5.0-19-generic/kernel/fs/squashfs/squashfs.ko  
license: GPL  
author: Phillip Lougher   
description: squashfs 4.0, a compressed read-only file system  
srcversion: 89B46A0667BD5F2494C4C72  
depends:   
intree: Y  
vermagic: 3.5.0-19-generic SMP mod\_unload modversions 686

* **rmmod** – Remove Module from Kernel

rmmod command will remove a module from the kernel. You cannot remove a module which is already used by any program.

# rmmod squashfs.ko

* **modprobe** – Add or Remove modules from the kernel

modprobe is an intelligent command which will load/unload modules based on the dependency between modules.

* **/proc file system**

The proc file system is a pseudo-file system which is used as an inter‐ face to kernel data structures. It is commonly mounted at /proc. Most of it is read-only, but some files allow kernel variables to be changed.

* /proc Directories with names as numbers

Do a ls -l /proc, and you’ll see lot of directories with just numbers. These numbers represents the process ids, the files inside this numbered directory corresponds to the process with that particular PID.

Following are the important files located under each numbered directory (for each process):

cmdline – command line of the command.

environ – environment variables.

fd – Contains the file descriptors which is linked to the appropriate files.

limits – Contains the information about the specific limits to the process.

mounts – mount related information

Following are the important links under each numbered directory (for each process):

cwd – Link to current working directory of the process.

exe – Link to executable of the process.

root – Link to the root directory of the process.

* /proc Files about the system information

Following are some files which are available under /proc, that contains system information such as cpuinfo, meminfo, loadavg.

/proc/cpuinfo – information about CPU,

/proc/meminfo – information about memory,

/proc/loadvg – load average,

/proc/partitions – partition related information,

/proc/version – linux version

**Unit 2 - USERS, GROUPS and PERMISSIONS ADMINISTRATION**

1. **Types of Users**

In Linux there are three type of users.

* **Super user or root user :**

Super user or the root user is the most powerful user. He is the administrator user.

* **System users :**

System users are the users created by the software or applications. For example if we install Apache it will create a user apache.

* **Normal users :**

Normal users are the users created by root user. They are normal users like John, Ramu etc. Only the root user has the permission to create or remove a user.

In linux systems every user will have a unique user ID. It is known as UID. The Range of UIDs will be as follows:

1. Root user UID will be "0"

2. Systems users UID will be "1 - 499"

3. Normal users UID will be "500 - 60000"

The range of MIN\_GID and MAX\_GID is specified in the file "/etc/login.defs".

1. **Configuration files of User Administration**

There are three important files a user administrator should be aware of.

1. "/etc/passwd"
2. "/etc/shadow"
3. "/etc/group"

Each of the above mentioned files have specific formats.

1. **"/etc/passwd"**

The first line will be like this.

root:x:0:0:root:/root:/bin/bash

There are seven fields in it with each separated by ":"

Fields are as follows,

**User\_name:Pointer\_to\_Shadow\_file:UID:Comment:GID:Home\_Directory:Login\_shell**

1. User\_name is the name of the user.

2. Pointer to shadow file is the pointer to the "/etc/shadow" where the encrypted password for that user is stored.

3. UID is the user ID.

4. GID is the group ID for the user.

5. Comment is a field where we can add some info about that user. Suppose if the user is a group leader, we can specify it there.

6. Home\_dir denotes the path of users home directory. By default for root user it'll be "/root" and for normal user it'll be "/home/user\_name".

7. Default login shell will be "/bin/bash". If we want to change it to korn shell edit it to "/bin/ksh". If no login shell is required for that user then give

"/sbin/nologin"

1. **"/etc/shadow"**

Shadow file contains the user's encrypted password and password aging options.

The first line will be like this

**root:$1fdsfsgsdfsdkffefje:14757:0:99999:7:::**

The fields are as follows,

1. User\_name

2. Encrypted password

3. Days since that password was last changed.

4. Days after which password must be changed.

5. Days before password is to expire that user is warned.

6. Days after the password is expires that the user is disabled.

7. Days since the account is disabled.

8. A reserved field.

1. **"/etc/group"**

Contains information about groups in the system.

The first line will be like this

**root:x:0:root**

The fields are as follows.

1. Group\_name, the name of the group

2. The encrypted group password

3. GID, Group ID

4. User\_list, all the group member's user names. Separated by commas.

1. **User Creation:**

**/etc/default/useradd**

This file may contain the following parameters that provide default values for the [useradd(1M)](http://uw714doc.sco.com/en/man/html.1M/useradd.1M.html) command

**SHELL:** full pathname of user's login shell

**HOMEDIR:** Base directory in which to create user's home directory

**SKELDIR:** Directory that contains skeleton information (such as a .profile file) to be copied to the user's home directory

**GROUPID:** Default group ID

**INACT:** The maximum number of days allowed between uses of a login name

**EXPIRE:** The date on which a login name can no longer be used

**FORCED\_PASS:** A password generator, defined for the user

**AUDIT\_MASK:** Default user audit mask; valid only if the Auditing Utilities are installed and running on the system

In linux a user can be created with specific UID, GID, comment, Home directory and login shell. The options are as follows.

The command to add a user is #adduser or #useradd. Actually useradd is the real command and adduser is a soft link to the useradd command. But the usage of both are same.

A command to add a user with all the fields we mentioned before is as follows.

***#useradd -u UID*** -g GID/Group\_name -c COMMENT -d Home\_dir -s LOGIN SHELL User\_name

An example:

***#useradd -u 555*** -g linux -c Teamlead -d /teamleads/john -s /bin/ksh john

Prior to the executing of the above command you should create the group 'linux'.

You can also add -p for password and -G for secondary groups which we will see later.

If u want to add the password, u 've to give the password in encrypted form.

For example,

***#useradd -p*** encrypted\_password Joseph

1. **Group creation**

There are two kinds of users in linux. They are,

1. Primary group

2. Secondary group

When we create a user a group also will be created in the same name of the user. Suppose we creating a user abc with uid 540, then a group abc will be created with same gid. And if the user abc creates a file xyz, it's owner will be user abc and group will be group abc. That is abc is its primary group. That's normally all the files and directories created by a user belongs to its primary group. But what if the user needs access to the directories created by other groups? or a user has to supervise a number of groups? then comes the secondary group concept. All the other groups are added as the secondary groups of that user.

The command to add a group is #groupadd

***#groupadd group\_name***

***#groupadd -g 540 linux***

Suppose we want to create a user rahul with linux, java, hp, ibm as secondary groups, it can be done as follows

***#useradd -G*** linux,java,hp,ibm rahul

You have to specify all the secondary groups in single command, not one after one in different commands.

But you can appended the secondary groups to a user using the -a option with usermod command.

For example,

A user john is a member of groups linux and java. We can append the group ibm to him as follows.

***#usermod -a -G ibm john***

Checking the groups of a user

***#groups username***

Will list all groups that the user belongs to.

How to set a password for a group?

***#groupadd linux***

***#gpasswd linux***

the password will be saved in **"/etc/gshadow"**.

To change the name of a group

***#groupmod -n newname oldname***

1. **Switching users**

sometimes we may need to switch between users.

The command for switching is #su

1. ***#su***

Switches to root user. But only gets privileges.

2***. #su -***

Switches to root user. Gets privileges and home directory access.

1. ***su raju***

Switches to user raju

2. ***su - raju***

gets also home dir access of raju.

If you are logind as root user and switching to normal user, you won't be prompted for the password. But you'll be prompted for password if otherwise.

**Understanding /etc/login.defs**

The /etc/login.defs file defines the site-specific configuration for the shadow password suite. User administration is, I think, the basic but most important part of any system administrator's job. In production environment we have to do some security hardening related with user and groups. In Redhat Enterprise Linux 6 (RHEL6) we have a file called /etc/login.defs which have default password policy for users. Sometimes we have to change the password policy according to the organization's user policy, like minimum password length, password expiration period, etc.

We can apply password policy through graphical user administration tool, but I would suggest to use command line tool to change password policy. To change password policy through command line in Linux, we just have to edit /etc/login.defs file. Only root user can edit this file.

When useradd command used a user is created and then useradd binary now proceeds with the process of user creation and goes to the /etc/login.defs file to get following values from the file :

MAIL\_DIR /var/spool/mail

PASS\_MAX\_DAYS 99999

PASS\_MIN\_DAYS 0

PASS\_MIN\_LEN 5

PASS\_WARN\_AGE 7

UID\_MIN 500

UID\_MAX 60000

GID\_MIN 500

GID\_MAX 60000

CREATE\_HOME yes

**Description : -**

1. **MAIL\_DIR**: Directory where the user's mail will be stored.

2**. PASS\_MAX\_DAYS:** Maximum number of days for the validity of a password.

3. **PASS\_MIN\_DAYS:** Minimum number of days gap before a password can be changed again.

4. **PASS\_MIN\_LEN:** Minimum required length of a password.

5. **PASS\_WARN\_AGE:** Warning for password expiry to be given before the stipulated number of days.

6. **UID\_MIN:** Minimum value for automatic user id selection.

7**. UID\_MAX:** Maximum value for automatic user id selection.

8. **GID\_MIN:** Minimum value for automatic group id selection.

9. **GID\_MAX:** Maximum value for automatic group id selection.

10. **CREATE\_HOME:** Whether useradd should create home directories for users .

If we change these values in /etc/login.defs file, when new user is created, the user will hold the new values.

1. **Modifying existing users**

We can also modify the existing user with ***#usermod*** command.

for example,

***#usermod -u 555 -g linux -c Teamlead -d /teamleads/john -s /bin/ksh john***

We can change the login name of a user using the option -l

Syntax is as follows

***#usermod -l new\_name old\_name***

Locking and unlocking the users

***#usermod -L username***

Executing the above command will lock the user with username.

***#usermod -U username***

Executing the above command will unlock the user with username.

1. **Removing a user**

we can remove a user using ***#userdel*** command

For example,

***#userdel user\_name***

the above command will remove the user but not his home directory. This is for taking back up of the files from it in case needed.

***#userdel -r user\_name***

the above command will remove the user as well as user's home directory.

Deleting the password of a user or allowing password less login for a user

***#passwd -d u\_name***

**Group Commands:**

* ***gpasswd:*** administer the /etc/group file
* ***groupadd:*** Create a new group

Format: groupadd [-g gid [-o]] [-f] [-K KEY=VALUE] group

Example: groupadd accounting

* ***groupmod:*** Modify a group

Format: groupmod [-g gid [-o ]] [-n new\_group\_name] group

Example - Change name of a group: groupmod -n accounting nerdyguys

* ***groupdel:*** Delete a group

Example: groupdel accounting

1. **File Permissions**

Although there are already a lot of good security features built into Linux-based systems, one very important potential vulnerability can exist when local access is granted - - that is file permission based issues resulting from a user not assigning the correct permissions to files and directories. So based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary.

Basic File Permissions

Permission Groups

Each file and directory has three user based permission groups:

* ***owner*** - The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.
* ***group*** - The Group permissions apply only to the group that has been assigned to the file or directory, they will not affect the actions of other users.
* ***all users*** - The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

**Permission Types**

Each file or directory has three basic permission types:

* ***read*** - The Read permission refers to a user's capability to read the contents of the file.
* ***write*** - The Write permissions refer to a user's capability to write or modify a file or directory.
* ***execute*** - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

**Viewing the Permissions**

You can view the permissions by checking the file or directory permissions in your favorite GUI File Manager (which I will not cover here) or by reviewing the output of the *\"ls -l\"* command while in the terminal and while working in the directory which contains the file or folder.

The permission in the command line is displayed as: *\_rwxrwxrwx 1 owner: group*

* User rights/Permissions
* The first character that I marked with an underscore is the special permission flag that can vary.
* The following set of three characters (rwx) is for the owner permissions.
* The second set of three characters (rwx) is for the Group permissions.
* The third set of three characters (rwx) is for the All Users permissions.
* Following that grouping since the integer/number displays the number of hard links to the file.
* The last piece is the Owner and Group assignment formatted as Owner: Group. Modifying the Permissions

When in the command line, the permissions are edited by using the command *chmod*. You can assign the permissions explicitly or by using a binary reference as described below.

**Explicitly Defining Permissions**

To explicitly define permissions you will need to reference the Permission Group and Permission Types.

The Permission Groups used are:

* **u - Owner**
* **g - Group**
* **o or a - All Users**

The potential Assignment Operators are + (plus) and - (minus); these are used to tell the system whether to add or remove the specific permissions.

The Permission Types that are used are:

* **r - Read**
* **w - Write**
* **x - Execute**

So for an example, let's say I have a file named file1 that currently has the permissions set to \_rw\_rw\_rw, which means that the owner, group and all users have read and write permission. Now we want to remove the read and write permissions from the all users group.

To make this modification you would invoke the command: ***chmod a-rw file1***

To add the permissions above you would invoke the command: ***chmod a+rw file1***

As you can see, if you want to grant those permissions you would change the minus character to a plus to add those permissions.

**Using Binary References to Set permissions**

Now that you understand the permissions groups and types this one should feel natural. To set the permission using binary references you must first understand that the input is done by entering three integers/numbers.

A sample permission string would be chmod 640 file1, which means that the owner has read and write permissions, the group has read permissions, and all other user have no rights to the file. The first number represents the Owner permission; the second represents the Group permissions; and the last number represents the permissions for all other users. The numbers are a binary representation of the rwx string.

* **r = 4**
* **w = 2**
* **x = 1**

You add the numbers to get the integer/number representing the permissions you wish to set. You will need to include the binary permissions for each of the three permission groups.

So to set a file to permissions on file1 to read \_***rwxr***\_\_\_\_\_, you would enter ***chmod 740 file1***.

**Changing the File Permissions and Ownership:**

You can use chown and chgrp commands to change the owner or the group of a particular file or directory.

**chown Command**

The chown is the command changes the [owner](http://www.computerhope.com/jargon/o/owner.htm) and owning group of files.

**Syntax**

***chown [OPTION]... [OWNER][:[GROUP]] FILE...***

***chown [OPTION]... --reference=RFILE FILE…***

Change the owner of a file

***# ls -lart tmpfile***  
-rw-r--r-- 1 himanshu family 0 2012-05-22 20:03 tmpfile  
  
***# chown root tmpfile***  
  
***# ls -l tmpfile***  
-rw-r--r-- 1 root family 0 2012-05-22 20:03 tmpfile

**Chgrp Command**

CHGRP(CHange GRouP) is one more command which is useful to change group associated to a file/folder from one group to other in a Linux box.

An example of how to use chgrp command is shown below.

***[root@RHEL2 chgrptest]# chgrp <new\_group\_owner> <file\_name>***

***[root@RHEL2 chgrptest]# ls –l***

total 0

-rw-r--r-- 1 root root 0 Jul 6 06:28 chgrptest

***[root@RHEL2 chgrptest]# chgrp engg chgrptest***

***[root@RHEL2 chgrptest]# ls -l***

total 0

-rw-r--r-- 1 root engg 0 Jul 6 06:28 chgrptest

**Example1**: Change group name: sales of a file to other group name:hrgroup.

***#chgrp hrgroup file1***

**Example2:** Give access permissions to a command so that the command can be executed by all users belonging to apache-admins

***chgrp apache-admins /etc/init.d/httpd***

**Example3:** Change group ownership all the files located in /var/apache to group: apache

***chgrp -R apache /var/apache***

**Example4:**Change group ownership forcefully

***chgrp -f apache /var/apache***

**Umask Command**

UMASK (User Mask or User file creation MASK) is the default permission or base permissions given when a new file (even folder too, as Linux treats everything as files) is created on a Linux machine. Most of the Linux distros give 022 (0022) as default UMASK. In other words, it is a system default permissions for newly created files/folders in the machine.

* The minimum and maximum UMASK value for a folder is 000 and 777
* The minimum and maximum UMASK value for a file is 000 and 666
* The default umask 002 used for normal user. With this mask default directory permissions are 775 and default file permissions are 664.
* The default umask for the root user is 022 result into default directory permissions are 755 and default file permissions are 644.
* For directories, the base permissions are (rwxrwxrwx) 0777 and for files they are 0666 (rw-rw-rw).

The octal umasks are calculated via the bitwise AND of the unary complement of the argument using bitwise NOT. The octal notations are as follows:

* 0 –Full permissions (Read, Write, Execute)
* 1 –Write and read
* 2 –Read and execute
* 3 –Read only
* 4 –Write and execute
* 5 –Write only
* 6 –Execute only admin
* 7 –No permissions

**Task:** Calculating The Final Permission For FILES for Root User:

You can simply subtract the umask from the base permissions to determine the final permission for file as follows:

**666 - 022 = 644**

File base permissions : **666**

umask value : **022**

subtract to get permissions of new file **(666-022) : 644 (rw-r--r--)**

**Task:** Calculating The Final Permission For DIRECTORIES for Root User:

You can simply subtract the umask from the base permissions to determine the final permission for directory as follows:

**777 - 022 = 755**

Directory base permissions : **777**

umask value : **022**

Subtract to get permissions of new directory **(777-022) : 755 (rwxr-xr-x)**

1. **Chage Policies:**

chage changes the number of days between password changes and the date of the last password change. This information is used by the system to determine when a user must change her password. The chage command is restricted to the root user, except for the -l option, which may be used by an unprivileged user to determine when her password or account is due to expire.

Password aging is a mechanism that allows the Linux system to enforce a certain lifetime for passwords and this is a good security practice. Password aging information can be configured with the Linux chage command. If the chage command is immediately followed by a username, the administrator will be interactively prompted for the password aging values.

|  |  |
| --- | --- |
| **Option** | **Description** |
| -m days | Specify the minimum number of days between which the user must change passwords. If the value is 0, the password does not expire. |
| -M days | Specify the maximum number of days for which the password is valid. |
| -d days | Specify the number of days since January 1, 1970 the password was changed. |
| -I days | Specify the number of inactive days after the password expiration before locking the account. If the value is 0, the account is not locked after the password expires. |
| -E date | Specify the date on which the account is locked, in the format YYYY-MM-DD. Instead of the date, the number of days since January 1, 1970 can also be used. |
| -W days | Specify the number of days before the password expiration date to warn the user. |

**To view the user’s aging information**

**Syntax: *$chage -l <user>***

Ex: - ***# chage -l yugandhar***

Last password change : Oct 31, 2014

Password expires : never

Password inactive : never

Account expires : never

Minimum number of days between password change : 0

Maximum number of days between password change : 99999

Number of days of warning before password expires : 7

**Unit 3 - File System Administration**

1. **File system**

A file system is the underlying structure a computer uses to organize data on a hard disk. If you are installing a new hard disk, you need to partition and format it using a file system before you can begin storing data or programs.

ext2, ext3 and ext4 are all file systems created for Linux. This article explains the following:

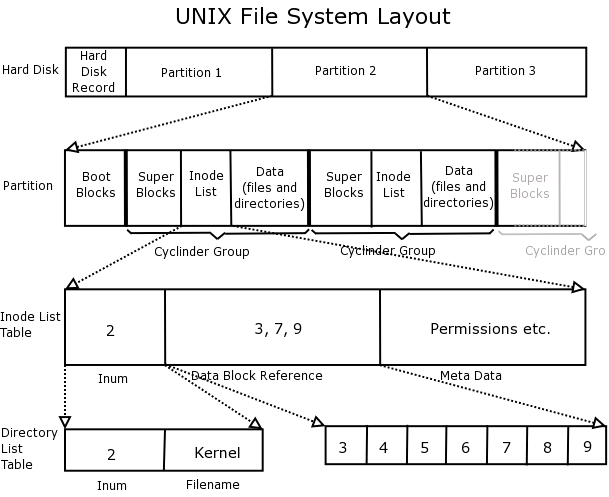
* High level difference between these file systems.
* How to create these file systems.
* How to convert from one file system type to another.
* **Ext2**
* Ext2 stands for second extended file system.
* It was introduced in 1993. Developed by Remy Card.
* This was developed to overcome the limitation of the original ext file system.
* Ext2 does not have journaling feature.
* On flash drives, usb drives, ext2 is recommended, as it doesn’t need to do the over head of journaling.
* Maximum individual file size can be from 16 GB to 2 TB
* Overall ext2 file system size can be from 2 TB to 32 TB
* **Ext3**
* Ext3 stands for third extended file system.
* It was introduced in 2001. Developed by Stephen Tweedie.
* Starting from Linux Kernel 2.4.15 ext3 was available.
* The main benefit of ext3 is that it allows journaling.
* Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.
* Maximum individual file size can be from 16 GB to 2 TB
* Overall ext3 file system size can be from 2 TB to 32 TB
* There are three types of journaling available in ext3 file system.
* Journal – Metadata and content are saved in the journal.
* Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default.
* Write back – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk.
* You can convert a ext2 file system to ext3 file system directly (without backup/restore).
* **Ext4**
* Ext4 stands for fourth extended file system.
* It was introduced in 2008.
* Starting from Linux Kernel 2.6.19 ext4 was available.
* Supports huge individual file size and overall file system size.
* Maximum individual file size can be from 16 GB to 16 TB
* Overall maximum ext4 file system size is 1 EB (Exabyte). 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte).
* Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3)
* You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it).
* Several other new features are introduced in ext4: multi block allocation, delayed allocation, journal checksum. fast fsck, etc. All you need to know is that these new features have improved the performance and reliability of the file system when compared to ext3.
* In ext4, you also have the option of turning the journaling feature “off”.

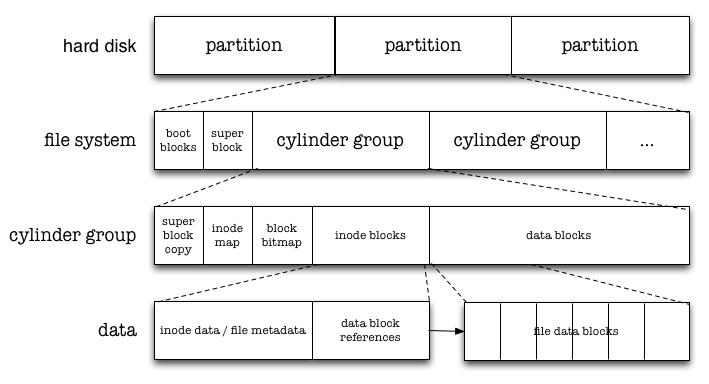
To find the file system type of the partition

**Syntax: *$blkid <hard disk/Partition>***

***# blkid /dev/sda1***

/dev/sda1: UUID="7906463d-f8f2-4799-96e2-4259c54b191e" TYPE="ext4"





1. **Creating a Partition**

* **Step1:(Device identification)**

First check the partition table using fdisk command. Using Fdisk command one can identify his device that whether it is your internal Hard disk or external hard disk.

**/dev/sd(a,b,c)----------->SCSI**

**/dev/hd(a,b,c)----------->IDE**

***#fdisk -l***

***[root@myvm1 ~]# fdisk -l***

Disk /dev/sda: 26.8 GB, 26843545600 bytes

255 heads, 63 sectors/track, 3263 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

**Device Boot Start End Blocks Id System**

/dev/sda1 \* 1 6 48163+ 83 Linux

/dev/sda2 7 515 4088542+ 83 Linux

/dev/sda3 516 3133 21029085 83 Linux

/dev/sda4 3134 3263 1044225 5 Extended

/dev/sda5 3134 3263 1044193+ 82 Linux swap / Solaris

we have seen that already partition up to /dev/sda5 has been created.

So now we have to create a new partition that will start from /dev/sda6

* **Step2: (Partitioning Device)**

See how to create a new Partition using fdisk tool. We do partition or you can say disks are partitioned to make a separate File System according our need.

***#fdisk /dev/sda***

***[root@myvm1 ~]# fdisk /dev/sda***

The number of cylinders for this disk is set to 3263.

There is nothing wrong with that, but this is larger than 1024,

and could in certain setups cause problems with:

1) software that runs at boot time (e.g., old versions of LILO)

2) booting and partitioning software from other OSs

(e.g., DOS FDISK, OS/2 FDISK)

**Command (m for help):**

Note: here we have taken sda because fdisk -l show us our hard disk is sda type not hda or hdb.

:n this will create a new partition

:l this will create a logical partition

: just press enter to take default cylinder value.

:+2000M this mean we want to create a partion of size 2gb approx ie 2000Mb.

:w write the changes and save it and exit

* **Step3:Usage of partprobe**

Why we use part probe in linux?

If You don't want to reboot your linux system for making update for the changes you have made in the partition table above just use partprobe command.

***#partprobe***

* **Step4:Making File System**

How to make a file system in Linux?

mkfs or mke2fs command is used to create file system in linux. So Create an ext3 file system.(Make File System so that it get recognized by OS)

**#mke2fs -j /dev/sda6**

or

**#mkfs -t ext3 /dev/sda6**

or

**#mkfs.ext3 /dev/sda6**

This will format the /dev/sda6 partition and create journal file system ext3 which can be recognized by our Linux operation System.

* **Step5:Mount point**

Now make a new directory and mount /dev/sda6 newly created partion on it. mount point are directories where file systems are mapped.

***#mkdir /new***

***#mount /dev/sda6 /new***

Now above command will mount /dev/sda6 on /new directory. So now whatever you write in new directory will be saved in newly created partition /dev/sda6

* **Step6:Verification on mount**

To verify whether the file system has been mounted or not.

***#df -h***

This will show you clearly that your /dev/sda6 is mounted on /new directory

* **Step7:**

Make File system permanent after reboot. To make this change exist after reboot or to make it permanent we have to make a entry in /etc/fstab file.

***#vim /etc/fstab***

/dev/sda6 /new ext3 defaults 0 0

save and exit the file and that's it.

You have created a new partion /dev/sda6 and now it's working.

**File System Disk Checking commands**

* **df -** Disk usage on file system.

***# df -h***

**File system Size Used Avail Use% Mounted on**

/dev/sda1 455G 84G 348G 20% /

none 4.0K 0 4.0K 0% /sys/fs/cgroup

udev 1.9G 8.0K 1.9G 1% /dev

tmpfs 386M 1.2M 385M 1% /run

none 5.0M 0 5.0M 0% /run/lock

none 1.9G 312K 1.9G 1% /run/shm

none 100M 36K 100M 1% /run/user

* **du** - used to check the information of disk usage of files and directories on a machine.

**# du -sch /etc/**

15M /etc/

15M total

**# du -sch /mnt/\***

0 /mnt/narafile

42M /mnt/teamviewer\_linux.deb

42M total

s- display only a total for each argument

c- produce a grand total

h- human readable format.

1. **File System Managing Commands**

* **Inode:**

Every partition in GNU/Linux has its own file system. In a file system, a file is represented by an inode. I nodes are data structures that contain information about files that are created when a file system is created. Each file has an inode and is identified by an inode number in the file system where it resides. Every partition has its own set of inodes.

Each inode describes a data structure on the hard disk, storing the properties of a file, including the physical location of the file data. When a hard disk is initialized, a fixed number of inodes per partition is created. This number will be the maximum amount of files, of all types (including directories, special files, links etc.) that can exist at the same time on the partition. Usually 1 inode per 2 to 8 kilobytes of storage.

When a new file is created, it gets a free inode. The following information are stored in inode.

• Owner and group owner of the file.

• File type

• Permissions on the file.

• Date and time of creation, last read and change.

• Date and time this information has been changed in the inode.

• Number of links to this file

• File size

• An address defining the actual location of the file data.

The only information not included in an inode, is the file name and directory. These are stored in the special directory files. By comparing file names and inode numbers, the system can make up a tree-structure that the user understands.

If you want to the inode number associated with a file, use "ls" command use "-i" option.

**Syntax : *ls -i <filename>***

***# ls -il tecadmin.txt***  
1150561 -rw-r--r-- 1 root root 0 Mar 10 01:06 tecadmin.txt

* **tune2fs**

The "tune2fs" command is used by the system administrator to change/modify tunable parameters on ext2, ext3 and ext4 type file systems. To display the current values that are set you can use the tune2fs command with the "-l" option or use the dumpe2fs command.

***# tune2fs -l /dev/sda1***

tune2fs 1.42.9 (4-Feb-2014)

File system volume name: <none>

Last mounted on: /

File system UUID: 7906463d-f8f2-4799-96e2-4259c54b191e

File system magic number: 0xEF53

File system revision #: 1 (dynamic)

File system features: has\_journal ext\_attr resize\_inode dir\_index filetype needs\_recovery extent flex\_bg sparse\_super large\_file huge\_file uninit\_bg dir\_nlink extra\_isize

File system flags: signed\_directory\_hash

Default mount options: user\_xattr acl

File system state: clean

Errors behavior: Continue

File system OS type: Linux

Inode count: 30269440

Block count: 121072128

Reserved block count: 6053606

Free blocks: 97272054

Free inodes: 29942239

First block: 0

Block size: 4096

Fragment size: 4096

Reserved GDT blocks: 995

Blocks per group: 32768

Fragments per group: 32768

I nodes per group: 8192

Inode blocks per group: 512

Flex block group size: 16

File system created: Sat Nov 1 00:53:38 2014

Last mount time: Mon Jan 19 15:19:39 2015

Last write time: Mon Jan 19 15:19:39 2015

Mount count: 197

Maximum mount count: -1

Last checked: Sat Nov 1 00:53:38 2014

Check interval: 0 (<none>)

Lifetime writes: 549 GB

Reserved blocks uid: 0 (user root)

Reserved blocks gid: 0 (group root)

First inode: 11

Inode size: 256

Required extra isize: 28

Desired extra isize: 28

Journal inode: 8

First orphan inode: 28180553

Default directory hash: half\_md4

Directory Hash Seed: 8d7f6349-52cf-4fa0-8255-36545b424553

Journal backup: inode blocks

* **e2label**

e2label - Change the label on an ext2/ext3/ext4 filesystem

Syntax : ***e2label <labelname> <partition>***

**Ex:** ***e2label DISK1 /dev/sda6***

* **dumpe2fs**

dumpe2fs prints the super block and blocks group information for the filesystem present on device.

**Note:** When used with a mounted filesystem, the printed information may be old or inconsistent.

***# dumpe2fs /dev/sda1 | grep -i superblock***

dumpe2fs 1.42.9 (4-Feb-2014)

Primary superblock at 0, Group descriptors at 1-29

Backup superblock at 32768, Group descriptors at 32769-32797

Backup superblock at 98304, Group descriptors at 98305-98333

Backup superblock at 163840, Group descriptors at 163841-163869

Backup superblock at 229376, Group descriptors at 229377-229405

* **fsck**

fsck is used to check and optionally repair one or more Linux file systems. files can be a device name (e.g. /dev/hdc1, /dev/sdb2), a mount point (e.g. /, /usr, /home), or an ext2 label or UUID specifyer (e.g. UUID=8868abf6-88c5-4a83-98b8-bfc24057f7bd or LABEL=root). Normally, the fsck program will try to handle file systems on different physical disk drives in parallel to reduce the total amount of time needed to check all of them.

**Syntax : *fsck <option> <partition>***

**Ex: *fsck /dev/sda1***

***$fsck /dev/sdb1***  
fsck from util-linux-ng 2.17.2  
e2fsck 1.41.11 (14-Mar-2010)

* Pass 1: Checking inodes, blocks, and sizes
* Pass 2: Checking directory structure
* Pass 3: Checking directory connectivity
* Pass 4: Checking reference counts
* Pass 5: Checking group summary information  
   57217 inodes used (24.81%)  
   42 non-contiguous files (0.1%)  
   65 non-contiguous directories (0.1%)  
   # of inodes with ind/dind/tind blocks: 0/0/0  
   Extent depth histogram: 50910/20  
   293868 blocks used (31.88%)  
   0 bad blocks  
   1 large file  
   43327 regular files  
   7242 directories  
   59 character device files  
   26 block device files  
   0 fifos  
   509 links  
   6549 symbolic links (6187 fast symbolic links)  
   5 sockets  
  --------  
   57717 files
* **/etc/fstab file**

fstab is a [configuration file](http://www.debianhelp.co.uk/fstab.htm) that contains information of all the partitions and storage devices in your computer. The file is located under /etc, so the ful path to this file is /etc/fstab. /etc/fstab contains information of where your partitions and storage devices should be mounted and how.

***#cat /etc/fstab***

# **device-spec mount-point fs-type options dump pass**  
LABEL=/ / ext4 defaults 1 1  
/dev/sda6 none swap defaults 0 0  
none /dev/pts devpts gid=5,mode=620 0 0  
none /proc proc defaults 0 0  
none /dev/shm tmpfs defaults 0 0

* The first field (fs\_spec): This field describes the block special device or remote filesystem to be mounted.
* The second field (fs\_file): This field describes the mount point for the filesystem.
* The third field (fs\_vfstype): This field describes the type of the filesystem
* The fourth field (fs\_mntops): This field describes the mount options associated with the filesystem.
* The fifth field (fs\_freq): This field is used for these file systems by the dump(8) command to determine which file systems need to be dumped. If the fifth field is not present, a value of zero is returned and dump will assume that the filesystem does not need to be dumped.
* The sixth field (fs\_passno): This field is used by the fsck(8) program to determine the order in which filesystem checks are done at reboot time.
* **swap**

Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM.

According to Redhat, the size of swap space can be calculated by using the following formulae.

M = Amount of RAM in GB, and S = Amount of swap in GB, then

If M < 2

S = M \*2

Else

S = M + 2

Using this formula, a system with 2 GB of physical RAM would have 4 GB of swap, while one with 3 GB of physical RAM would have 5 GB of swap.

To check the current swap usage by using the commands “swapon –s” or “free”

**Example:**

***[root@RHEL2 ~]# swapon -s***

Filename Type Size Used Priority

/dev/sda3 partition 803240 0 -1

***[root@RHEL2 ~]# free -m***

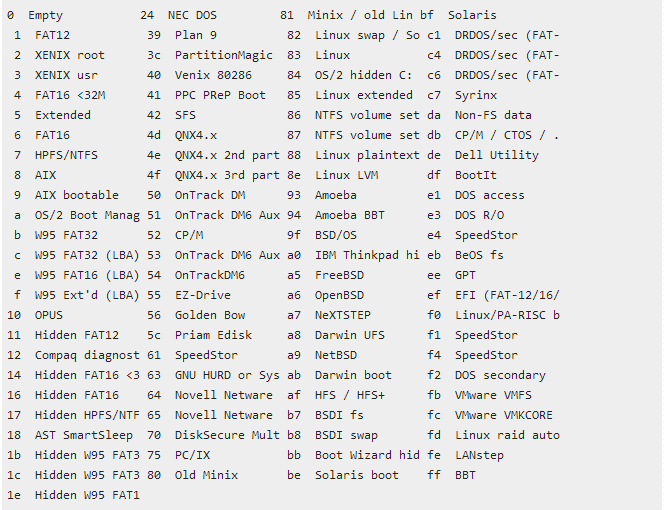
total used free shared buffers cached

Mem: 503 492 10 0 37 346

-/+ buffers/cache: 108 394

Swap: 784 0 784

List of Id’s:



**Create an additional swap partition and enabling swapping**

To Make Partition Swap:

**Step 1:**Create a new partition using fdisk. Set the partition’s system id as “82” (Linux Swap / Solaris).

You Know that how to make a partition.

**Step 2:** Create the swap filesystem on the raw partition using “mkswap” command.

***[root@RHEL2 ~]# mkswap /dev/sdd1***

Setting up swap space version 1, size = 518156 Kb

Enable swaping on the swap partition created now.

**Step 3:** ***swapon /dev/sdd1***

Update /etc/fstab to make use the new swap partition whenever the system is rebooted.

**To add a swap file:** Determine the size of the new swap file in megabytes and multiply by 1024 to determine the number of blocks. For example, the block size of a 64 MB swap file is 65536.At a shell prompt as root, type the following command with count being equal to the desired block size:

***dd if=/dev/zero of=/swapfile bs=1024 count=65536***

* if=/dev/zero : Read from /dev/zero file. /dev/zero is a special file in that provides as many null characters to build storage file called /swapfile1.
* of=/swapfile1 : Read from /dev/zero write storage file to /swapfile1.
* bs=1024 : Read and write 1024 BYTES bytes at a time.
* count=524288 : Copy only 523288 BLOCKS input blocks.

Change the permissions of the newly created file: ***chmod 0600 /swapfile***

Setup the swap file with the command: **mkswap /swapfile**

To enable the swap file immediately but not automatically at boot time: ***swapon /swapfile***

To enable it at boot time, edit /etc/fstab to include the following entry:

/swapfile swap swap defaults 0 0

The next time the system boots, it enables the new swap file. After adding the new swap file and enabling it, verify it is enabled by viewing the output of the command cat /proc/swaps or free.

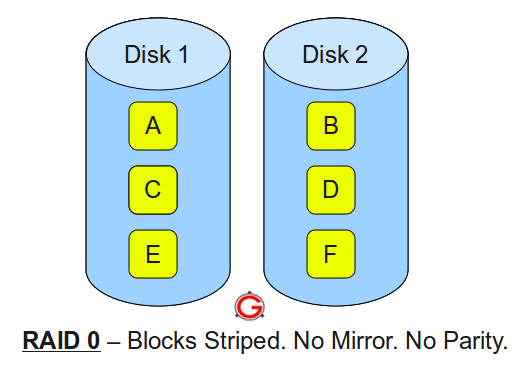
1. **RAID CONFIGURATION**

RAID stands for: redundant array of inexpensive drives or disks. By using raid, we can improve the storage capacity, disk performance and mainly for fault tolerance.

**Following are commonly used RAID levels :**

|  |  |  |  |
| --- | --- | --- | --- |
| **RAID level** | **Minimum hard disks** | **Suggested application** | **Notes** |
| RAID 0 - Striped Set without parity | 2 Hard disks | 1. Video Production and Editing  2. Image Editing  3. Any application requiring high bandwidth | Provides improved performance and additional storage but no fault tolerance from disk errors or disk failure. Any disk failure destroys the array, which becomes more likely with more disks in the array. |
| RAID 1 -MirroredSet (2 disks minimum) without parity. | 2 Hard disks | 1. Office application  2. Financial application  3. Payroll application etc | Provides fault tolerance from disk errors and single disk failure. Increased read performance occurs when using a multi-threaded operating system that supports split seeks, very small performance reduction when writing. Array continues to operate so long as at least one drive is functioning |
| RAID 5 | 3 Hard disks | 1. File and Application servers  2. Internet Web, E-mail servers  3. Intranet servers | Highest Read data transaction rate, Medium Write data transaction rate, Overall good (aggregate) transfer rate. drive failure requires replacement, but the array is not destroyed by a single drive failure. Upon drive failure, any subsequent reads can be calculated from the distributed parity such that the drive failure is masked from the end user. The array will have data loss in the event of a second drive failure and is vulnerable until the data that was on the failed drive is rebuilt onto a replacement drive |
| RAID 10 (nested RAID 1+0) | 4 Hard disks | 1. Database server (such as Oracle / MySQL / MS-SQL) which requiring high performance and fault tolerance | Provides fault tolerance and improved performance but increases complexity. |

**RAID LEVEL 0**



**Creation of RAID0:**

***$mdadm --create /dev/md0 --level 0 --raid-devices=2 /dev/sda6 /dev/sda7***

***$mkfs -t ext4 /dev/md0***

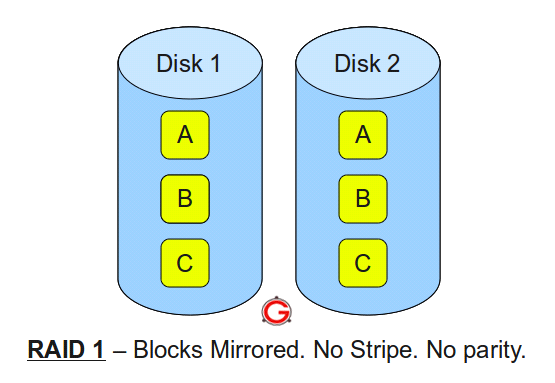
***$mkdir /raid0***

***$mount /dev/md0 /raid0***

**Verification:-**

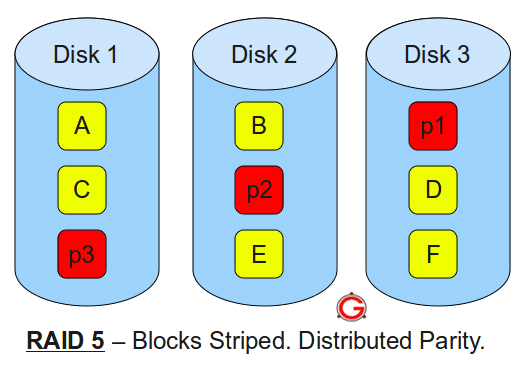
***$mdadm --detail /dev/md0 (OR) $cat /proc/mdstat***

**RAID LEVEL 1:**



***$mdadm --create /dev/md1 --level 1 --raid-devices=2 /dev/sda8 /dev/sda9***

RAID LEVEL 5:



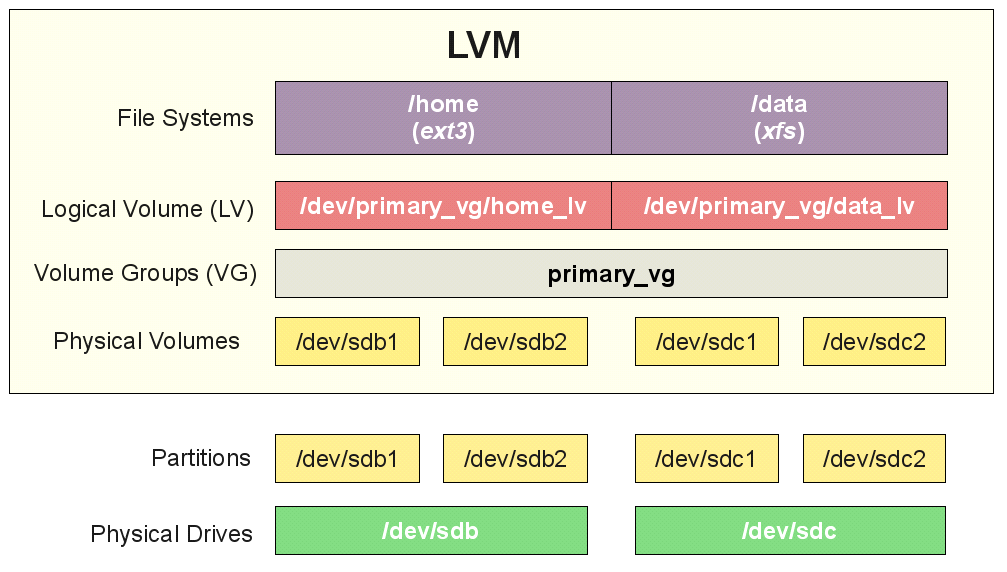
***$mdadm --create /dev/md5 --level 5 --raid-devices=3 /dev/sda10 /dev/sda11 /dev/sda12.***

1. **LOGICAL VOLUME MANAGER**

LVM is a method of allocating hard drive space into logical volumes that can be easily resized instead of partitions. With LVM the hard drive (or) set of hard drives are allocated to one or more physical volumes.

The physical volumes are combined into volume groups

Each volume group is divided into logical volumes which are assigned mount points such as /home and filesystem types such as ext3



**To configure LVM**

1)Create three LVM partitions

2)Convert them as physical volumes

3)Create volume groups from physical volumes

4)Create logical volumes from volume groups and assign mount points

**IMPLEMENTATION:**

***#fdisk /dev/sda***

:n

+500M

:t

<partition number>

:8e (Linux LVM)

:w (save & exit)

***#partx -a /dev/sda (update kernel)***

To convert LVM partitions as physical volumes

***#pvcreate /dev/sda<partition numbers>***

ex: ***pvcreate /dev/sda{9,10,11}***

To view physical volumes

***#pvdisplay***

To create volume group

***#vgcreate <vg name> <partitions>***

ex: ***vgcreate bsrtech /dev/sda{9,10,11}***

To view volume groups

***#vgdisplay***

To create a logical volume

***#lvcreate -L <+size> <vg name> -n <LV name>***

ex: ***lvcreate -L +300M /dev/bsrtech -n lv1***

To view logical volumes

***#lvdisplay***

To format logical volumes

***#mkfs.ext4 /dev/bsrtech/lv1***

Create a mountpoint and mount logical volume on it

***#mkdir /mysql***

***#mount /dev/bsrtech/lv1 /mysql***

***#cd /mysql***

To extend size of logical volume

***#umount <mountpoint>***

***#lvresize -L +<size> <lv name>***

ex:***lvresize -L +200M /dev/bsrtech/lv1***

To make filesystem for extended size

***#resize2fs <logical volume>***

ex: ***resize2fs /dev/bsrtech/lv1***

***#mount /dev/bsrtech/lv1 /mysql***

To reduce a logical volume

note: whenever we are reducing an LVM we have to take backup (More Details #man lvreduce)

***#mkdir /lvm-bkp***

***#cp -rf /mysql/\* /lvm-bkp***

***#lvreduce -L <-size> <LVM>***

ex: ***lvreduce -L -100M /dev/bsrtech/lv1***

To format LVM

***#mkfs.ext4 <logical volume>***

ex:***mkfs.ext4 /dev/bsrtech/lv1***

***#mount /dev/bsrtech/lv1 /mysql***

***#cp -rf /lvm-bkp/\* /mysql***

To remove an LVM

***#umount <mountpoint>***

***#lvremove <logical volume>***

ex:***lvremove /dev/bsrtech/lv1***

To extend volume group

1)create another LVM partition

2)convert into physical volume

***#vgextend <vg name> <partition name>***

ex:***vgextend /dev/bsrtech /dev/sda12***

To reduce volume group

***#vgreduce <vgname> <partition name>***

ex:***vgreduce /dev/bsrtech /dev/sda12***

To remove volume group

***#vgremove <vg name>***

ex:***vgremove /dev/bsrtec***

To delete physical volumes

***#pvremove <partitions>***

ex:***pvremove /dev/sda{9,10,11,12}***

1. **Disk Quotas**

Disk space can be restricted by implementing disk quotas which alert a system administrator before a user consumes too much disk space or a partition becomes full. Disk quotas can be configured for individual users as well as user groups. This makes it possible to manage the space allocated for user-specific files (such as email) separately from the space allocated to the projects a user works on (assuming the projects are given their own groups).

The quota RPM must be installed to implement disk quotas.

To implement disk quotas, use the following steps:

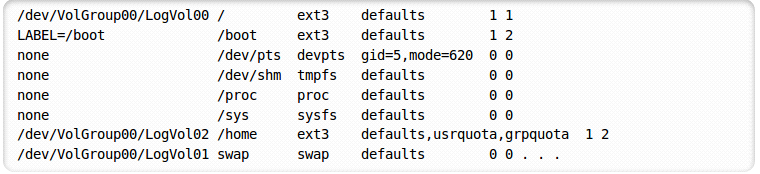
* Enable quotas per file system by modifying the /etc/fstab file.
* Remount the file system(s).
* Create the quota database files and generate the disk usage table.
* Assign quota policies.

Each of these steps is discussed in detail in the following sections.

Enabling Quotas:

Add the usrquota and/or grpquota options to the file systems that require quotas:

***# vim /etc/fstab***



**Remounting the Filesystem**

***#mount -o remount /home OR #mount -o remount <filesystem>***

**Creating The Quota Database Files**

After each quota-enabled file system is remounted run the quota check command. The quota check command examines quota-enabled file systems and builds a table of the current disk usage per file system. The table is then used to update the operating system's copy of disk usage. In addition, the file system's disk quota files are updated. To create the quota files (aquota.user and aquota.group) on the file system, use the -c option of the quota check command.

**Create quota files**

For example, if user and group quotas are enabled for the /home file system, create the files in the /home directory:

***# quotacheck -cug /home***

The -c option specifies that the quota files should be created for each file system with quotas enabled, the -u option specifies to check for user quotas, and the -g option specifies to check for group quotas. If neither the -u or -g options are specified, only the user quota file is created. If only -g is specified, only the group quota file is created.

After the files are created, run the following command to generate the table of current disk usage per file system with quotas enabled:

***# quotacheck -avug***

The options used are as follows:

***a*** -Check all quota-enabled, locally-mounted file systems

***v***-Display verbose status information as the quota check proceeds

***u***-Check user disk quota information

***g***-Check group disk quota information

After quotacheck has finished running, the quota files corresponding to the enabled quotas (user and/or group) are populated with data for each quota-enabled locally-mounted file system such as /home.

Assigning Quota for User or Group

***#edquota <username>***

***#edquota -g <group\_name>***

To assign the grace period

***#edquota -t***

Enabling & Disabling Quotas:

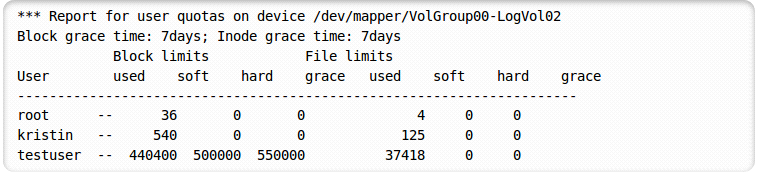
***# quotaoff -vaug***

***# quotaon -vaug***

***# quotaon -vug /home***

Reporting on Disk quotas

***#repquota /home***



To view the disk quota usage for all partitions.

***#repquota -a***

1. **ACLs (Access Control Lists)**

Access control lists are the advanced file permissions. By using the ACL we can assign the user to access or deny accessing a file or a directory.

**User Class**

The conventional POSIX permission concept uses three classes of users for assigning permissions in the file system: the owner, the owning group, and other users. Three permission bits can be set for each user class, giving permission to read (r), write (w), and execute (x).

**ACL**

The user and group access permissions for all kinds of file system objects (files and directories) are determined by means of ACLs.

**Default ACL**

Default ACLs can only be applied to directories. They determine the permissions a file system object inherits from its parent directory when it is created.

**ACL Entry**

Each ACL consists of a set of ACL entries. An ACL entry contains a type, a qualifier for the user or group to which the entry refers, and a set of permissions. For some entry types, the qualifier for the group or users is undefined.

There are four categories of ACLs per file:

* For an individual user,
* For a user group,
* Via the effective rights mask
* For users not in the user group associated with the file.

To Enable the ACL permissions to the partition is as follows:

***# vi /etc/fstab***

LABEL=/ / ext3 defaults, acl 1 1

:x (save and exit)

We should remount the '/' filesystem to take effect.

***# mount -o remount, rw /***

To display the permissions as of your choice.

***# getfacl /root/test/acltest.txt***

To set read and execute permissions on the file.

***# setfacl -m u:<username>:r-x /root/test/acltest.txt***

To remove the ACL on the file.

***# setfacl -x u:<username> /root/test/acltest.txt***

Set all permissions for user johny to file named "abc":

***# setfacl -m "u:johny:rwx" abc***

Check permissions

***# getfacl abc***

***# file: abc  
# owner: someone  
# group: someone***user::rw-  
user:johny:rwx  
group::r--  
mask::rwx  
other::r--

Remove all extended ACL entries:

***# setfacl -b abc***  
Check permissions

***# getfacl abc***

***# file: abc  
# owner: someone  
# group: someone***

user::rw-  
group::r--  
other::r--

**IV . Process Management & Performance Tuning**

1. **Process**

A process is an executing (i.e., running) [instance](http://www.linfo.org/instance.html) of a [program](http://www.linfo.org/program.html). Processes are also frequently referred to as tasks. Processes carry out tasks within the operating system. A program is a set of machine code instructions and data stored in an executable image on disk and is, as such, a passive entity; a process can be thought of as a computer program in action.

Linux is a multi-user and multi-tasking operating system(seemingly, discussed later in the article). A Linux process is a program in execution on a Linux system. Therefore, whenever a program is executed, a new process is created. A process also consumes resources like the file system, memory or other CPU resources. This gives rise to the need of process management in Linux.

**Identifier for Linux Processes**

In Linux, every process has a unique process Identifier(ID) associated to it. A process ID( i.e. PID) is a number which is uniquely assigned as soon as the process is created. The PID’s are allocated sequentially as the processes are being created. However, it generally starts from 2, as PID=1 is reserved for ‘init process. As we always expect, there is a maximum limit to the PID value. In a system, the way one can get to know the maximum limit to PID is

***$ cat /proc/sys/kernel/pid\_max***

PID : 32768

Listing the Process:

The Linux command used to view list of processes is ‘ps’ which means ‘process status’ (Some authors also interpret it as ‘process snapshot’).

***$ps***  
  
 **PID TTY TIME CMD**  
  
1779 pts/0 00:00:00 bash  
  
2176 pts/0 00:00:00 ps

**PID** – The process Identifier which is ‘1779’ for ‘bash’ and ‘2176’ for ‘ps’.

**TTY** – stands for terminal-type and is the name of the console/terminal, the process is associated to.

**TIME** – The CPU time since the process has started. It is confusing that why the CPU time for ‘bash’ process is ‘00:00:00’? This is because, CPU time is the time for which the process is being executed by the processor. However, when bash runs commands, lets say ‘ls command’ , a child process ‘ls’ is spawned and whatever execution and cpu utilization takes place, goes under the ‘ls’ process and not ‘bash’ Bash process is just the parent process.

**CMD** – Command run to create the process.

**List All processes**

***$ps -e | more***

e - select all process

$ps -ef | more  
  
**UID PID PPID C STIME TTY TIME CMD**  
  
root 1 0 0 Mar08 ? 00:00:00 /sbin/init  
root 2 0 0 Mar08 ? 00:00:00 [kthreadd]  
root 3 2 0 Mar08 ? 00:00:01 [ksoftirqd/0]

**UID** – The User ID,is the username of the user, which owns the process.

**PID** – Already discusses the Process ID.

**PPID** – It is the Parent Process ID

**C** – The CPU usage and scheduling information. The value is incremented with every tick of the system clock, however degraded by the scheduler by dividing it by two in every second. Therefore, A higher value indicates CPU intensive process.

**STIME** – The start time of the process.

**TTY** – The terminal type associated with the process. If this value is ‘?’, then it means the process is not associated with any terminal. These are daemon process, which we shall be discussing in the next section.

**TIME** – The cumulative CPU time since the process is running.

**CMD** – The command which launched the process.

Process can be represented as a tree (hierarchical) structure in Linux. To view the complete tree structure, linux provides a command – pstree. It gives an interesting output, showcasing the first ‘init’ process and the other processes spawned out of it.

pstree  
  
init─┬─NetworkManager─┬─dhclient  
  
 │ ├─dnsmasq  
  
 │ └─2\*[{NetworkManager}]  
  
 ├─accounts-daemon───{accounts-daemon}  
  
 ├─acpid  
  
 ├─at-spi-bus-laun───2\*[{at-spi-bus-laun}]

1. **Types of Processes**

Although there is no standard classification of types of processes in Linux. The segregation could be in interactive and non-interactive processes, foreground and background processes or daemon or batch processes. It can also be classified based on the status of the processes such as as zombie processes. It is good enough if we comprehend all these various terminologies in the linux system.

1. **Interactive Process:** An interactive process is one which needs user’s interaction while it is active. For example, when we launch a vi-editor, it is an interactive process. Another example could be the telnet command. Hence, the interactive processes have to be associated to a terminal.
2. **Foreground Process:** A process is a foreground process if it is in focus and can be given input from the standard input. It blocks the shell until the foreground process is complete. When we run our commands on the terminal, they generally run as foreground processes. They block the terminal until it is complete. Although most of our linux commands are quick enough for us to even realize that.
3. **Background Process:** Background processes are ones, that are running, but in the background, not taking any user input from the terminal. It doesn’t block the terminal, and allows us to use the terminal irrespective of the background process is complete or not. They key-character to make any new process to be run in background is ‘&’.

$ ./wait\_process &  
[1] 2534

1. **Daemon Process:** A daemon process in Linux is also one of its kind which runs in background. However, what is different here is, daemon processes are not associated to any terminal in any way. Therefore such processes don’t take interact with the user. A widespread example of daemon process is a server service of any kind. For example, if we consider a mail server, it just have to listen to the relevant ports and respond with its protocol routines on receiving packages. So, such kind of processes can be run as daemon processes, independent of terminal and user interaction.
2. **Zombie Processes:** When a process finishes execution, it will have an exit status to report to its parent process. Because of this last little bit of information, the process will remain in the operating system’s process table as a zombie process, indicating that it is not to be scheduled for further execution, but that it cannot be completely removed (and its process ID cannot be reused) until it has been determined that the exit status is no longer needed.
3. **Process States**

Linux processes generally go through six major states, which are listed below:

**1. Running or Runnable ( R )** – A running state has a broader concept here. Running always does not mean utilizing the CPU. Even while a process is ready to run, the state is running state.

Hence, there are two sub-states, when the process is queued in the ready queue to run and when the process is actually being executed, it is in the executing sub-state as has been scheduled by the scheduler.

**2. Stopped (T)** – If a running process receives a stop signal, it is moved to the stopped state. A process can also be in stopped state if it has been halted by a trace while debugging. .

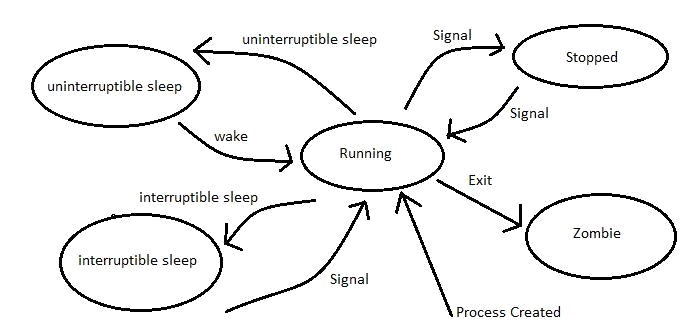
**3. Uninterruptible sleep (D)** – It is a sleeping state, process has been blocked. Mostly, process goes into an uninterruptible sleep during an IO operation.

**4. Interruptible sleep (S)** – It is a sleeping state i.e. a blocking state where the process is waiting for an event to occur.

**5. Zombie/Defunct state(Z)** – It is the process state in which process has been terminated but not reaped by its parent process.

**6. Dead (X)** – A process never reaches this state, as as soon as it is dead, it is gone.

A simplified life cycle of a Linux process is illustrated through following diagram:



1. **Real time snapshots of processes**

The ‘ps’ command that we just discussed in the previous section, evinces the active process list at a particular moment when the command is executed. In many cases, it is a need of the hour to view dynamic real time running of the processes. For such circumstances, linux comes with the ‘top’ command.

***$top***

top - 10:16:34 up 1 day, 17:36, 1 user, load average: 1.51, 0.99, 0.84  
Tasks: 136 total, 1 running, 135 sleeping, 0 stopped, 0 zombie  
Cpu(s): 28.8%us, 3.3%sy, 0.0%ni, 67.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st  
Mem: 507536k total, 498516k used, 9020k free, 10488k buffers  
Swap: 521212k total, 187996k used, 333216k free, 91572k cached  
  
 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND   
  
2129 ubuntu 20 0 727m 258m 10m S 27.1 52.1 1 13:12.30 firefox   
  
 918 root 20 0 116m 35m 2128 S 3.3 7.1 7:33.90 Xorg   
  
1772 ubuntu 20 0 89352 5264 2940 S 0.7 1.0 0:08.13 gnome-terminal

Then following rows list all the details of all the launched processes at real time. What each detail means is:

**PID** – The process ID

**USER** – The user which owns the process

**PR** – Process priority value

**NI** – The [nice](http://mylinuxbook.com/linux-processes-part1/<href=http://en.wikipedia.org/wiki/Nice_(Unix)>) value of a process.

**VIRT** – The virtual memory used by the process

**RES** – Physical memory used

**SHR** – Shared memory of the process

**S** – The status of the process where **S – Sleeping : R – Running : Z- Zombie**

**%CPU** – Percentage CPU utilization of the process

**%MEM** – Percentage memory usage of the process

**(TIME+)** – The total activity time of the process

**COMMAND** – The command used to launch the process.

**Terminating the processes**

* **kill:** The linux system allows its users to terminate any process, of course with due considerations to access permissions. The linux user can terminate a process using command ‘kill’.

**The usage syntax**

***kill [ -signal | -s signal ] pid ...***

The ‘kill’ linux command sends a signal to the specified process. Which process to send the signal is specified by its PID. There are standard numbers assigned to each set of signals. We can get the information of what number is corresponding to which signal using the same ‘kill’ command through ‘-l’ option.

**$ kill -l**  
1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP 6) SIGABRT 7) SIGBUS

8) SIGFPE 9) SIGKILL 10) SIGUSR1 11) SIGSEGV 12) SIGUSR2 13) SIGPIPE

14) SIGALRM 15) SIGTERM 16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP 21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ  
26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR 31) SIGSYS

34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3 38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8 43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13

48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13

52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9

56) SIGRTMAX-8 57) SIGRTMAX-7 58) SIGRTMAX-6 59) SIGRTMAX-5

60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2 63) SIGRTMAX-1

64) SIGRTMAX

* **pgrep:** looks through the currently running processes and lists the process IDs which match the selection criteria to stdout. All the criteria have to match.

For example***: $ pgrep -u root sshd***

* **pkill:** pkill will send the specified signal (by default SIGTERM) to each process instead of listing them on stdout

**Syntax :** ***$pkill <signal> <Processname>***

* **pidof**: find the process ID of a running program

Syntax: **$pidof <process>**

**Prioritize the Process:**

* **nice:** run a command with modified priority. nice command is used to run the given command with its scheduling priority adjusted. Priority range goes from -20 (highest priority) to 19 (lowest priority).

Examples: ***$ nice – Prints the current priority value.***

***$ nice ls*** — Increment the priority value of the ls command by 10 (Default value) and run.

***$ nice -n 5 ls*** — Increment the priority value of the ls command by 5 and run.

***# nice -n -2 ls*** — Decrement the priority value of the ls command by -2 and run.

**NOTE:** Incrementing the priority value will reduce the priority level and vice versa. Priority range is -20 (high) to 19 (low). .

* **renice :** renice is used to alters the scheduling priority of one or more running processes, priority range goes from -20 (highest priority) to 19 (lowest priority).

**Examples:**

***$ renice +1 123 — Increment the priority value of a process, which process ID is 123.***

***# renice +1 123 – Decrement the priority value of a process, which process ID is 23. (Root only can decrement ).***

***$ renice +1 -p 123 -p 200 — Same as above. Here no. of process is 2.***

***$ renice +1 -u sbharathi – Increment the priority value of all processes, which is owned by a user (sbharathi).***

***$ renice +1 -g backup — Increment the priority value of all processes, which is owned by a***

***group (backup).***

NOTE: Incrementing the priority value will reduce the priority level and vice versa. Priority range is -20 (high) to 19 (low).

* **JOBS:** jobs utility in linux will show you about background running jobs and stopped (Ctrl -Z) jobs.

***$jobs : list out all background running and stopped jobs***

***$fg %1 (jobid) : To run the background running job into foreground***

***$bg %2 (job id) : To run the foreground running jobs into background.***

1. **Tools to Monitor Linux Performance**

* **free command:** The Linux “free” command gives information about total used and available space of physical memory and swap memory with buffers used by kernel in Linux/Unix like operating systems

***# free -m***

total used free shared buffers cached

Mem: 3859 3053 806 188 200 1230

-/+ buffers/cache: 1621 2237

Swap: 3998 223 976

* **vmstat command:** vmstat reports information about [processes](http://www.computerhope.com/jargon/p/process.htm), [memory](http://www.computerhope.com/jargon/m/memory.htm), [paging](http://www.computerhope.com/jargon/p/paging.htm), [block](http://www.computerhope.com/jargon/b/block.htm) [IO](http://www.computerhope.com/jargon/i/io.htm), [traps](http://www.computerhope.com/unix/utrap.htm), [disks](http://www.computerhope.com/jargon/d/disk.htm) and [cpu](http://www.computerhope.com/jargon/c/cpu.htm) activity. The first report produced gives averages since the last [reboot](http://www.computerhope.com/jargon/r/reboot.htm). Additional reports give information on a sampling period of length delay. The process and memory reports are instantaneous in either case.

***# vmstat***

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----

r b swpd free buff cache si so bi bo in cs us sy id wa st

1 0 22824 885368 205684 1255900 0 0 10 61 71 89 21 4 75 0 0

**Procs:**

|  |  |
| --- | --- |
| r | The number of processes waiting for run time. |
| b | The number of processes in uninterruptible [sleep](http://www.computerhope.com/unix/usleep.htm). |

**Memory:**

|  |  |
| --- | --- |
| swpd | the amount of virtual memory used. |
| free | the amount of idle memory. |
| buff | the amount of memory used as buffers. |
| cache | the amount of memory used as [cache](http://www.computerhope.com/jargon/c/cache.htm). |
| inact | the amount of inactive memory. (-a option) |
| active | the amount of active memory. (-a option) |

**Swap:**

|  |  |
| --- | --- |
| si | Amount of memory swapped in from disk (per second). |
| so | Amount of memory swapped to disk (per second). |

**IO:**

|  |  |
| --- | --- |
| bi | Blocks received from a block device (blocks per second). |
| bo | Blocks sent to a block device (blocks/s). |

**System:**

|  |  |
| --- | --- |
| in | The number of [interrupts](http://www.computerhope.com/jargon/i/interrup.htm) per second, including the clock. |
| cs | The number of context switches per second. |

**CPU:** (values expressed in percentages of CPU time)

|  |  |
| --- | --- |
| us | Time spent running non-kernel code. (user time, including [nice](http://www.computerhope.com/unix/unice.htm) time) |
| sy | Time spent running kernel code. (system time) |
| id | Time spent idle. Prior to [Linux](http://www.computerhope.com/jargon/l/linux.htm) 2.5.41, this includes IO-wait time. |
| wa | Time spent waiting for IO. Prior to Linux 2.5.41, included in idle. |
| st | Time stolen from a virtual machine. Prior to Linux 2.6.11, unknown. |

* **iostat command :**

Iostat without any argument displays information about the CPU usage, and I/O statistics about all the partitions on the system as shown below.

**$ iostat**  
Linux 2.6.32-100.28.5.el6.x86\_64 (dev-db) 07/09/2011  
  
avg-cpu: %user %nice %system %iowait %steal %idle  
 5.68 0.00 0.52 2.03 0.00 91.76  
  
Device: tps Blk\_read/s Blk\_wrtn/s Blk\_read Blk\_wrtn  
sda 194.72 1096.66 1598.70 2719068704 3963827344  
sda1 178.20 773.45 1329.09 1917686794 3295354888  
sda2 16.51 323.19 269.61 801326686 668472456  
sdb 371.31 945.97 1073.33 2345452365 2661206408  
sdb1 371.31 945.95 1073.33 2345396901 2661206408  
sdc 408.03 207.05 972.42 513364213 2411023092  
sdc1 408.03 207.03 972.42 513308749 2411023092

* **System Activity Report (sar) Command :** sar you can monitor performance of various Linux subsystems (CPU, Memory, I/O..) in real time. You can also collect all performance data on an on-going basis, store them, and do historical analysis to identify bottlenecks explains how to monitor the following Linux performance statistics using sar.

Collective CPU usage

Individual CPU statistics

Memory used and available

Swap space used and available

Overall I/O activities of the system

Individual device I/O activities

Context switch statistics

Run queue and load average data

Network statistics

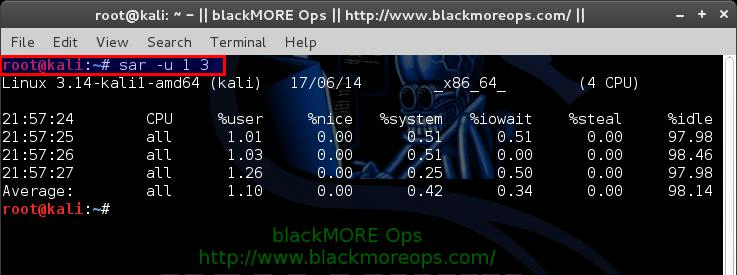
Report sar data from a specific time

CPU Usage of ALL CPUs (sar -u)

This gives the cumulative real-time CPU usage of all CPUs. “1 3″ reports for every 1 seconds a total of 3 times. Most likely you’ll focus on the last field “%idle” to see the cpu load.

**#root@kali:~# sar -u 1 3**  
Linux 3.14-kali1-amd64 (kali) 17/06/14 \_x86\_64\_ (4 CPU)  
21:57:24 CPU %user %nice %system %iowait %steal %idle  
21:57:25 all 1.01 0.00 0.51 0.51 0.00 97.98  
21:57:26 all 1.03 0.00 0.51 0.00 0.00 98.46  
21:57:27 all 1.26 0.00 0.25 0.50 0.00 97.

Average: all 1.10 0.00 0.42 0.34 0.00 98.14



The simple sar command output is as shown below.

12:00:01 AM CPU %user %nice %system %iowait %steal %idle  
12:01:01 AM all 73.28 0.00 1.25 0.00 0.00 25.47  
12:02:01 AM all 7.83 0.00 0.44 0.00 0.00 91.73  
12:03:01 AM all 61.65 0.00 0.70 0.00 0.00 37.66  
12:04:01 AM all 57.85 0.00 0.82 0.00 0.00 41.34

* **uptime :** uptime - Tell how long the system has been running. uptime gives a one line display of the following information. The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

***# uptime***

12:12:55 up 1 day, 20:53, 2 users, load average: 0.13, 0.27, 0.32

* **lsof command :** lsof - list open files. lists on its standard output file information about files opened by processes for the following UNIX dialects.

***# lsof***COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME  
init 1 root cwd DIR 8,1 4096 2 /  
init 1 root txt REG 8,1 124704 917562 /sbin/init  
init 1 root 0u CHR 1,3 0t0 4369 /dev/null  
init 1 root 1u CHR 1,3 0t0 4369 /dev/null  
init 1 root 2u CHR 1,3 0t0 4369 /dev/null  
init 1 root 3r FIFO 0,8 0t0 6323 pipe

1. **SYSTEM SERVICES**

**Basic Commands:**

* **service Command:** The service command is used to stop, start, restart and check the status of a specified service.

***# service httpd stop  
# service httpd start  
# service httpd restart  
# service httpd status***

Use the following command to get the status of all services.

***# service --status-all***

To fully restart a service (a separate stop then start) use the following command.

***# service httpd --full-restart***

The auto-start toggles for services are also available using a text-based tool initiated from the command line using the setup command.

* **chkconfig command:** The chkconfig command performs a number of service management tasks. To enable and disable services to auto-start at reboot, use the following commands.

**# chkconfig httpd on  
# chkconfig httpd off**

The "--level" flag allows auto-start to be limited to specific run levels.

**# chkconfig --level 35 httpd on**

The "--list" option displays all the run levels a specific service is associated with.

**# chkconfig --list httpd**  
httpd 0:off 1:off 2:on 3:on 4:on 5:on 6:off

**Scheduling Jobs:**

* **at job:** The Linux “at” command also can be used for scheduling jobs. But using Linux “at” command, you can set the job run only once. The “at” jobs are spooled in the “/var/spool/at” directory and run at the specified time.

Schedule first job using at command

Below example will schedule “ls -l” command to be executed on next 9:00 AM once.

**# at 9:00 AM**at> ls -l  
at> ^d  
job 3 at 2013-03-23 09:00  
  
Use ^d to exit from at prompt.

2. List the scheduled jobs using at

When we list jobs by root account using at , it shows all users jobs in result. But if we execute it from non root account, it will show only that users jobs.

**# at**  
3 2013-03-23 09:00 a root  
5 2013-03-23 10:00 a rahul  
1 2013-03-23 12:00 a root

|  |  |
| --- | --- |
| **Example** | **Meaning** |
| at noon | 12:00 PM September 18, 2001 |
| at midnight | 12:00 AM September 19, 2001 |
| at teatime | 4:00 PM September 18, 2001 |
| at tomorrow | 10:00 AM September 19, 2001 |
| at noon tomorrow | 12:00 PM September 19, 2001 |
| at next week | 10:00 AM September 25, 2001 |
| at next Monday | 10:00 AM September 24, 2001 |
| at fri | 10:00 AM September 21, 2001 |
| at OCT | 10:00 AM October 18, 2001 |
| at 9:00 AM | 9:00 AM September 19, 2001 |
| at 2:30 PM | 2:30 PM September 18, 2001 |
| at 1430 | 2:30 PM September 18, 2001 |
| at 2:30 PM tomorrow | 2:30 PM September 19, 2001 |
| at 2:30 PM next month | 2:30 PM October 18, 2001 |
| at 2:30 PM Fri | 2:30 PM September 21, 2001 |
| at 2:30 PM 9/21 | 2:30 PM September 21, 2001 |
| at 2:30 PM Sept 21 | 2:30 PM September 21, 2001 |
| at 2:30 PM 9/21/2010 | 2:30 PM September 21, 2010 |
| at 2:30 PM 21.9.10 | 2:30 PM September 21, 2010 |
| at now + 30 minutes | 10:30 AM September 18, 2001 |
| at now + 1 hour | 11:00 AM September 18, 2001 |
| at now + 2 days | 10:00 AM September 20, 2001 |
| at 4 PM + 2 days | 4:00 PM September 20, 2001 |
| at now + 3 weeks | 10:00 AM October 9, 2001 |
| at now + 4 months | 10:00 AM January 18, 2002 |
| at now + 5 years | 10:00 AM September 18, 2007 |

* Crontab: The Crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list. crontab stands for "cron table," because it uses the job scheduler [cron](http://www.computerhope.com/jargon/c/cron.htm) to execute tasks; cron itself is named after "chronos," the Greek word for time.

**Cron Format**

Cron format is a simple, yet powerful and flexible way to define time and frequency of various actions. nnCron make active use of cron format in both [classic](http://www.nncron.ru/help/EN/working/classic.htm) and [extended](http://www.nncron.ru/help/EN/working/extended.htm) modes.

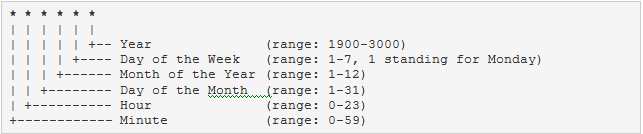
Traditional (inherited from Unix) cron format consists of five fields separated by white spaces:

<Minute> <Hour> <Day\_of\_the\_Month> <Month\_of\_the\_Year> <Day\_of\_the\_Week>

nnCron can use both traditional and "enhanced" version of cron format, which has an additional (6th) field: <Year>:

<Minute> <Hour> <Day\_of\_the\_Month> <Month\_of\_the\_Year> <Day\_of\_the\_Week> <Year>

A user can select the format he would like to use by selecting or unselecting the Year field checkbox on General tab in Options dialog (which can be opened by double-clicking the nnCron icon in system tray). By default, nnCron uses the enhanced format.



**Scheduling a Job For a Specific Time:** The basic usage of cron is to execute a job in a specific time as shown below. This will execute the Full backup shell script (full-backup) on 10th June 08:30 AM.

Please note that the time field uses 24 hours format. So, for 8 AM use 8, and for 8 PM use 20.

**#crontab -e**

30 08 10 06 \* /home/ramesh/full-backup

30 – 30th Minute

08 – 08 AM

10 – 10th Day

06 – 6th Month (June)

\* – Every day of the week

To allow the users and deny to set up a crontab , the files are

/etc/crontab.allow

/etc/crontab.deny

**V. SYSTEM NETWORKING CONFIGURATION**

A network consists of two or more computers that are linked in order to share resources (such as printers and CDs), exchange files, or allow electronic communications. The computers on a network may be linked through cables, telephone lines, radio waves, satellites, or infrared light beams.

1. **Basic Commands:**

* **ifconfig command** : ifconfig (interface configuration) command is use to initialize an interface, assign IP Address to interface and enable or disable interface on demand.

In Linux , we can call NIC as Ethernet cards.

NIC1 - eth0

NIC2 - eth1

NIC3 - eth2

NIC4 - eth3

***# ifconfig eth0***

eth0 Link encap:Ethernet HWaddr f0:1f:af:53:cd:c7

inet addr:192.168.0.104 Bcast:192.168.0.255 Mask:255.255.255.0

inet6 addr: fe80::f21f:afff:fe53:cdc7/64 Scope: Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:1081054 errors:0 dropped:0 overruns:0 frame:0

TX packets:890385 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:591893688 (591.8 MB) TX bytes:204316894 (204.3 MB)

Interrupt:20 Memory:f7c00000-f7c20000

Assigning the IP address, Netmask and Gateway.

***#ifconfig eth0 192.168.0.1 netmask 255.255.255.0 gw 192.168.0.254***

Enabling / Disabling interfaces:

***#ifup eth0***

***#ifdown eth0***

* **ethtool command:** The ethtool utility is a diagnostic tool that enables you to configure and diagnose problems with NICs. However, not all NICs support its use. This is the basic syntax for using the ethtool command: ethtool [options] interface.

**# ethtool eth0**

Settings for eth0:

Supported ports: [ TP ]

Supported link modes: 10baseT/Half 10baseT/Full

100baseT/Half 100baseT/Full

1000baseT/Full

Supported pause frame use: No

Supports auto-negotiation: Yes

Advertised link modes: 10baseT/Half 10baseT/Full

100baseT/Half 100baseT/Full

1000baseT/Full

Advertised pause frame use: No

Advertised auto-negotiation: Yes

Speed: 100Mb/s

Duplex: Full

Port: Twisted Pair

PHYAD: 2

Transceiver: internal

Auto-negotiation: on

MDI-X: off (auto)

Supports Wake-on: pumbg

Wake-on: d

Current message level: 0x00000007 (7)

drv probe link

Link detected: yes

* **mii-tool command:** The mii-tool utility enables you to display and manipulate the media-independent interface status. In its simplest form, you can use it to display the status of the interface’s link. This is the basic syntax for using the mii-tool command: mii-tool [options] [interface]

***# mii-tool***

SIOCGMIIREG on eth0 failed: Input/output error

SIOCGMIIREG on eth0 failed: Input/output error

eth0: negotiated 100baseTx-FD flow-control, link ok.

* **ping command:** PING (Packet Internet Groper) command is the best way to test connectivity between two nodes. Whether it is Local Area Network (LAN) or Wide Area Network (WAN). Ping use ICMP (Internet Control Message Protocol) to communicate to other devices. You can ping host name of ip address using below command.

**# ping 192.168.0.104**

PING 192.168.0.104 (192.168.0.104) 56(84) bytes of data.

64 bytes from 192.168.0.104: icmp\_seq=1 ttl=64 time=0.023 ms

64 bytes from 192.168.0.104: icmp\_seq=2 ttl=64 time=0.056 ms

^C

--- 192.168.0.104 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 999ms

rtt min/avg/max/mdev = 0.023/0.039/0.056/0.017 ms

* **fping command**: send ICMP ECHO\_REQUEST packets to network hosts

**# fping facebook.com**

facebook.com is alive

**# fping server02.flash.com**

server02.flash.com is unreachable

* **route command:** show / manipulate the IP routing table

**# route -n**

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 192.168.0.1 0.0.0.0 UG 0 0 0 eth0

192.168.0.0 0.0.0.0 255.255.255.0 U 1 0 0 eth0

* **traceroute command:** traceroute is a network troubleshooting utility which shows number of hops taken to reach destination also determine packets traveling path. Below we are tracing route to global DNS server IP Address and able to reach destination also shows path of that packet is traveling.

**# traceroute google.com**

traceroute to google.com (173.194.36.6), 64 hops max

1 192.168.0.1 1.993ms 0.822ms 0.780ms

2 202.88.156.1 18.478ms 14.969ms 14.850ms

3 202.88.156.66 14.765ms 15.067ms 14.743ms

4 202.88.156.61 14.727ms 15.210ms 15.234ms

5 123.63.30.130 26.559ms 24.702ms 25.232ms

6 103.29.44.15 24.943ms 25.053ms 24.184ms

7 103.29.44.12 26.486ms 23.878ms 24.415ms

8 72.14.211.134 26.153ms 24.465ms 24.759ms

9 72.14.233.204 25.927ms 27.836ms 28.527ms

10 216.239.50.170 42.782ms 45.042ms 49.643ms

11 209.85.241.187 48.612ms 50.825ms 49.749ms

12 173.194.36.6 50.582ms 51.132ms 48.000ms

* **netstat command:** Netstat (Network Statistic) command display connection info, routing table information etc. To displays routing table information use option as -r.

***# netstat -ntlp***

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 127.0.1.1:53 0.0.0.0:\* LISTEN 1849/dnsmasq

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 954/sshd

tcp 0 0 127.0.0.1:631 0.0.0.0:\* LISTEN 7603/cupsd

tcp 0 0 127.0.0.1:5939 0.0.0.0:\* LISTEN 2039/teamviewerd

tcp6 0 0 :::22 :::\* LISTEN 954/sshd

tcp6 0 0 ::1:631 :::\* LISTEN 7603/cupsd

* **ARP command:** ARP (Address Resolution Protocol) is useful to view / add the contents of the kernel’s ARP tables. To see default table use the command as.

***# arp -a***

? (192.168.0.1) at 78:54:2e:f3:69:8e [ether] on eth0

***# arp -v***

Address HWtype HWaddress Flags Mask Iface

192.168.0.1 ether 78:54:2e:f3:69:8e C eth0

Entries: 1 Skipped: 0 Found: 1

* **mtr command:** mtr combines the functionality of the traceroute and ping programs in a single network diagnostic tool.

***#mtr google.com***

***$ mtr --report google.com***  
HOST: ducklington Loss% Snt Last Avg Best Wrst StDev  
 1. inner-cake 0.0% 10 2.8 2.1 1.9 2.8 0.3  
 2. outer-cake 0.0% 10 3.2 2.6 2.4 3.2 0.3  
 3. 68.85.118.13 0.0% 10 9.8 12.2 8.7 18.2 3.0  
 4. po-20-ar01.absecon.nj.panjde 0.0% 10 10.2 10.4 8.9 14.2 1.6  
 5. be-30-crs01.audubon.nj.panjd 0.0% 10 10.8 12.2 10.1 16.6 1.7  
 6. pos-0-12-0-0-ar01.plainfield 0.0% 10 13.4 14.6 12.6 21.6 2.6  
 7. pos-0-6-0-0-cr01.newyork.ny. 0.0% 10 15.2 15.3 13.9 18.2 1.3  
 8. pos-0-4-0-0-pe01.111eighthav 0.0% 10 16.5 16.2 14.5 19.3 1.3  
 9. as15169-3.111eighthave.ny.ib 0.0% 10 16.0 17.1 14.2 27.7 3.9  
 10. 72.14.238.232 0.0% 10 19.1 22.0 13.9 43.3 11.1  
 11. 209.85.241.148 0.0% 10 15.1 16.2 14.8 20.2 1.6  
 12. lga15s02-in-f104.1e100.net 0.0% 10 15.6 16.9 15.2 20.6 1.7

1. **Network Configuration Files:**

/etc/sysconfig/network-script/ifcfg-eth0

/etc/sysconfig/network

/etc/hosts

/etc/resolv.conf