**Important links:**

<https://www.golinuxhub.com/2018/06/scenario-based-interview-question-beginner-experience-linux.html>

<https://svrtechnologies.com/linux-interview-questions/new-54-linux-admin-interview-questions-and-answers-pdf>

**Basic commands in Linux:**

|  |  |
| --- | --- |
| **Command**  **uname -a**  **hostname**  hostname -I | **Description**  Displays version of linux, kernel release date, processor type, etc.  Displays your hostname  Display IP address |
| **cat [filename]**  cat file 1 > file2  cat file 1 >> file 2  cat file1 file2  cat > file1 | Display file’s contents  Output is written to file2 instead of displaying on screen  Append output of file 1 at the end of content of file2  Concatenate the content of file1, file2 and display both content  file1 can be created |
| **cd /directorypath**  cd /  cd .. | Change to directory.  Change to root directory  Change to parent directory of current directory |
| **chmod [options] mode filename**  chmod u=rwx,g=rx,o=r myfile   * **4** stands for "read", * **2** stands for "write", * **1** stands for "execute", and * **0** stands for "no permission."   Chmod -R [option] name of directory  **less /etc/passwd. : finding user** | Change a file’s permissions.  Change permission of all files in given directory |
| **chown [options] filename**  **Ex.** Chown [user]:[group] filename  Chown -R [user]:[group] name of directory | Change who owns a file.  Change owner and group  Change ownership of all files in directory. -R will do recursive operation |
| **clear** | Clear a command line screen/window for a fresh start. |
| **cp [options] source destination**  cp -i source destination  cp -R source destination  cp -s file1.txt file2.txt | Copy files and directories.  It will ask if u want to overwrite the content to dest.  It will copy all files in source directory into destination.  It will create soft link file2 pointing to file1  **1. Hard Links**   * Each hard linked file is assigned the same Inode value as the original, therefore they reference the same physical file location. Hard links more flexible and remain linked even if the original or linked files are moved throughout the file system, although hard links are unable to cross different file systems. * ls -l command shows all the links with the link column shows number of links. * Links have actual file contents * Removing any link, just reduces the link count, but doesn’t affect other links. * We cannot create a hard link for a directory to avoid recursive loops. * If original file is removed then the link will still show the content of the file. * Command to create a hard link is: * $ ln [original filename] [link name]   **2. Soft Links**   * A soft link is similar to the file shortcut feature which is used in Windows Operating systems. Each soft linked file contains a separate Inode value that points to the original file. As similar to hard links, any changes to the data in either file is reflected in the other. Soft links can be linked across different file systems, although if the original file is deleted or moved, the soft linked file will not work correctly (called hanging link). * ls -l command shows all links with first column value l? and the link points to original file. * Soft Link contains the path for original file and not the contents. * Removing soft link doesn’t affect anything but removing original file, the link becomes “dangling” link which points to nonexistent file. * A soft link can link to a directory. * Link across filesystems: If you want to link files across the filesystems, you can only use symlinks/soft links. * Command to create a Soft link is:   $ ln -s [original filename] [link name] |
| **date [options**] | Display or set the system date and time. |
| **df [options]**  df -h | Display used and available disk space by filesystem.  Display above information in human readable form |
| **du [options]**  du -h  du -sh | Show how much space each file takes up in folder.  Display above information in human readable form  Displays summery of memory in whole folder or directory.  s stands for summery and h stands for human readable. |
| **file [options] filename** | Determine what type of data is within a file. |
| **grep [options] pattern [filesname]**  grep -i [options] pattern [filesname]  grep -v pattern  grep -r pattern \* | Search files or output for a pattern.  Search pattern without considering case sensitiveness.  Print all pattern except given pattern  Print pattern recursively on all files in directory |
| **kill [options] pid**  kill -KILL [PID] | Stop a process. If the process refuses to stop, use kill -9 pid.  If process didn’t get killed, then we use this command. |
| **less [options] [filename]** | View the contents of a file one page at a time. |
| **ln source [name of link]**  **OR link**  **ln -s source [name of link]** | Create a shortcut. Create hard link  Create soft link |
| **locate filename**  locate -c filename | Search a copy of your filesystem for the specified filename.  It will count number of entries of given filename |
| **lpr [options]** | Send a print job. |
| **ls [options]**  ls -l  ls -a  ls -i  ls /[name of directory] | List directory contents.  List with file permission and owner  List file including hidden files  List file with inode number  List the file in defined directory |
| **man [command]** | Display the help information for the specified command. |
| **mkdir [options] directory** | Create a new directory. |
| **mv [options] source destination** | Rename or move file(s) or directories. |
| **passwd [name [password]]** | Change the password or allow (for the system administrator) to change any password. |
| **ps [options]**  ps -f  ps -lf  ps -elf | Display a snapshot of the currently running processes.  Displays parent process ID and user ID along with PID  Displays priority and nice value along with above info  Displays information of all opened bash shell |
| **pwd** | Display the pathname for the current directory. |
| **rm [options] directory** | Remove (delete) file(s) and/or directories. |
| **rmdir [options] directory** | Delete empty directories. |
| **ssh [options] user@machine** | Remotely log in to another Linux machine, over the network. Leave an ssh session by typing **exit**. |
| **su [options] [user [arguments]]** | Switch to another user account. |
| **tail [options] [filename]** | Display the last *n* lines of a file (the default is 10). |
| **tar [options] filename**  **zip/unzip** | Store and extract files from a tarfile (.tar) or tarball (.tar.gz or .tgz).  Use **zip** to compress files into a zip archive, and **unzip** to extract files from a zip archive. |
| **top** | Displays the resources being used on your system. Press q to exit. |
| **touch filename** | Create an empty file with the specified name. |
| **who [options]** | Display who is logged on. |

**Sudo apt-get install [Name of app]** Installs app (Ubuntu, Debian)

Sudo apt-get remove [Name of app] Uninstall app

Sudo apt-get upgrade [Name of app] upgrade app

**Yum -y install [name of app]** Install app (RedHat)

**Free [option]** Display used and free memory in disk and buffer and cache used by kernel as well as swap memory.

Free -m Display in megabyte

Free -k Display in kilobyte

Free -t Display in terabyte

**Netstat [option]** To check list of listening programs

Netstat -tan To check status of active socket

Netstat – i To check traffic in each interface

Netstat -r To check kernel Ip routing table

**tcpdump -c 10 -i eth0/1 port 22** Capture 10 packets on eth0/1 on port 22

**More** View contents of file

**Pg** Display contents of file page by page

## **repquota**  used to check the number of files and

## disk space used and each user’s defined quota

**Nice** used for changing priority of the jobs.

**logrotate [option] filename** allows automatic rotation, compression, removal, and mailing of log files

**Uptime** Shows how long system is running and number of users logged in

**W** Shows how long system is running and number of users logged in along with their processes and login name, tty, remote host, login time

**Who** Shows **user name**, **date**, **time** and **host information**.

**Users** and **Whoami** Shows currently logged in users

**Sort** Sorts line of file in ascending or descending order

**Find** [Option] search file with given option

|  |  |
| --- | --- |
| **zcat**  **partprobe**    **lsblk**    **echo**  echo > [Folder]  echo $$  echo $PPID | View content of compressed file without need to uncompressed it.  Save the created partition without the need to reboot.    Show disk partition space and mount points (same as fdisk -l)  Displays content written above it.  Redirect content into the given folder  Displays process ID  Displays Parent process ID |

**pidof** Display process ID

**pgrep** Display process ID

**iostat** System monitoring tool

**sar** System monitoring tool

**Ifconfig** Displays information about system interface. Information includes IP and MAC address, Input and output packets with error and drops.

**Route** Displays routing table

**lsof** Displays list of open files

**Setfacl**:

To give permission to another user or group apart from user who owns the file we cannot use chown or chmod command. We will use setfacl command.

To set the permission for any user  
# setfacl -m u: username:permission /path/to/directory

**Ex**.To add an acl for user deepak with read and execute permission on mydata directory  
# setfacl -m u:deepak:r-x /mydata

To set the permission for any group  
# setfacl -m g:groupname:permission /path/to/directory

**Curl command:**

CURL is a tool to transfer data **from** or **to** a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP). Basically, CURL is used to download content from internet.

To save the content to a file all you must do is specify the minus o (-o) switch as follows:

curl -o <filename> <URL>

To get the command to run in the background you then need to use the ampersand (&) as follows:

curl -s -O <URL> &

You can download from multiple URLs using a single curl command. Capitol o (O) is used to download content of URL. It will create filename **posts** in directory.

curl -O http://www.mysite.com/page1.html -O http://www.mysite.com/page2.html

By default, the curl command returns the following information as it downloads a URL:

Total bytes, Received/Transferred byte, Average download speed, Average upload speed, Total time, Current speed

You can use curl to fill in an online form and submit the data as if you had filled it in online.

curl -d name=john email=john@mail.com www.mysite.com/formpage.php

**Fdisk and gdisk command:**

**It** is used command-line based disk manipulation utility. With the help of fdisk command you can view, create, resize, delete, change, copy and move partitions on a hard drive. We can use Fdisk command only with **root** privileges. We can only create **4 default** partition using Fdisk. Then if we more need partition we have create **extended** partition.

To view all disk partition in linux we use. It displays size and type of partition.

**# Fdisk -l**

To view size of partition we use.

**# Fdisk -s [Name of partition]**

To add, delete and print partition use following command with different options.

**#Fdisk /dev/fda**

Now we can use different options for various operations.

m: List all available options.

**a**: Toggle bootable flag.

**d**: delete partition.

**n**: add new partition.

**p**: Print partition table.

**q**: quit without saving the changes.

**w**: write the table to disk and exit.

By default, once create partition then we need to reboot the system. We we don’t want to reboot the system we use **partprobe** command to save the partition.

To format partition, we use following command:

**#mkfs .ext4 [name of partition]**

**Logrotate:**

Following are the key files that you should be aware of for logrotate to work properly.

**/usr/sbin/logrotate** – The logrotate command itself.

**/etc/logrotate.conf** – Log rotation configuration for all the log files are specified in this file.

If you want to rotate a log file (for example, /tmp/output.log) for every 1KB, create the logrotate.conf as shown below.

$ cat logrotate.conf

/tmp/output.log {

Weekly

size 1k

create 700 bala bala

rotate 4

compress

maxage 100

This logrotate configuration has following three options:

* size 1k – logrotate runs only if the filesize is equal to (or greater than) this size.
* create – rotate the original file and create the new file with specified permission, user and group.
* rotate – limits the number of log file rotation. So, this would keep only the recent 4 rotated log files.
* Compress- It will compress the rotated file
* Monthly- It will do logrotate operation every week.
* Maxage- It will delete logroate file after 100 days

**Find command:**

 To start searching the whole drive you would type the following:

find /

If however, you want to start searching for the folder you are currently in then you can use the following syntax:

find .

to search for a file called myresume.odt across the whole drive you would use the following syntax:

find / -name myresume.odt

* The first part of the find command is obviously the word find.
* The second part is where to start searching from.
* The next part is an expression which determines what to find.
* Finally the last part is the name of the thing to find.
* If you want to find all the empty files and folders in your system use the following command:

find / -empty

* If you want to find all of the executable files on your computer use the following command:
* find / -exec
* To find all of the files that are readable use the following command:
* find / -read
* When you search for a file you can use a pattern. For example, maybe you are searching for all files with the extension [mp3](https://www.lifewire.com/what-is-an-mp3-2438590).
* You can use the following pattern:
* find / -name \*.mp3

You have an entire folder full of music files in a bunch of different formats. You want to find all the**\*.mp3** files from the artist **JayZ**, but you don’t want any of the remixed tracks. Using a **find command** with a couple of **grep** pipes will do the trick:

# find . –name “\*.mp3” | grep –i JayZ | grep –vi “remix”

In this example, we are using find to print all the files with a **\*.mp3 extension**, piping it to **grep –i** to filter out and prints all files with the name “**JayZ**” and then another pipe to **grep –vi** which filters out and does not print all filenames with the string (in any case) “**remix**”.

**NMAP command:**

The primary Uses of **nmap** is:

* Determining open ports and services running on a host
* Determine the Operating System running on a host
* Determine Domain name for IP address.
* nmap 172.16.0.0/24
* Starting Nmap 5.21 (http://nmap.org) at 2015-06-23 09:39 EST
* Nmap scan report for 172.16.0.1
* Host is up (0.0043s latency).
* Not shown: 998 closed ports
* PORT STATE SERVICE
* 22/tcp open ssh
* 80/tcp open http
* 443/tcp open https

**-------------------------------------------------------------------------------------------------------------**

**How to schedule task in linux?**

**Crontab**- It utilizes cron daemon to schedule repetitive scheduling task.

Crontab have some switches:

Crontab -I – list cron table.

Crontab -e – Creates new cron table.

Crontab- r – Remove cron table.

**Cron job** is a specific set of execution instructions specifying day, time and command to execute. crontab can have multiple execution statements.

**Crontab syntax:**  
A crontab file has five fields for specifying day, date and time followed by the command to be run at that interval.

|  |
| --- |
| \*     \*   \*  \*   \*  command to be executed  -     -    -   -  -  |     |     |   |    |  |     |     |   |    +----- day of week (0 - 6) (Sunday=0)  |     |     |   +------- month (1 - 12)  |     |     +--------- day of month (1 - 31)  |     +----------- hour (0 - 23)  +------------- min (0 - 59) |

### **Crontab Examples**

A line in crontab file like below removes the tmp files from /home/someuser/tmp each day at 6:30 PM.

30     18     \*     \*     \*         rm /home/someuser/tmp/\*

Changing the parameter values as below will cause this command to run at different time schedule below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| min | hour | day/month | month | day/week | Execution time |
| 30 | 0 | 1 | 1,6,12 | \* | — 00:30 hrs on 1st of Jan, June & Dec. |
|  | | | | | |
| 0 | 20 | \* | 10 | 1-5 | –8.00 PM every weekday (Mon-Fri) only in Oct. |
|  | | | | | |
| 0 | 0 | 1,10,15 | \* | \* | — midnight on 1st ,10th & 15th of month |
|  | | | | | |
| 5,10 | 0 | 10 | \* | 1 | — At 12.05,12.10 every Monday & on 10th of every month |
| : | | | | | |

**Commands related to networking:**

The procedure to turn off eth0 interface is as follows. Run:  
# ifdown eth0  
To turn on eth0 interface run:  
# ifup eth0  
See ip address info using the [ip command](https://www.cyberciti.biz/faq/linux-ip-command-examples-usage-syntax/):  
# ip a show eth0

## Debian / Ubuntu Linux restart network interface

To restart network interface, enter:  
sudo /etc/init.d/networking restart  
To stop and start use the following option (do not run them over remote ssh session as you will get disconnected):  
sudo /etc/init.d/networking stop  
sudo /etc/init.d/networking start

Assigning an IP Address and Gateway to interface on the fly.

**# ifconfig eth0 192.168.50.5 netmask 255.255.255.0**

### **DIG Command**

**Dig** query **DNS** related information like **A Record**, **CNAME**, **MX Record** etc. This command mainly use to troubleshoot **DNS** related query.

**# dig www.tecmint.com**; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.10.rc1.el6 <<>> www.tecmint.com

;; global options: +cmd

;; Got answer:

;; ->>HEADER<

### **NSLOOKUP Command**

**nslookup** command also use to find out **DNS** related query. The following examples shows **A Record** (**IP Address**) of **tecmint.com**.

**# nslookup www.tecmint.com**

Server: 4.2.2.2

Address: 4.2.2.2#53

Non-authoritative answer:

www.tecmint.com canonical name = tecmint.com.

Name: tecmint.com

Address: 50.116.66.136

### **10. ETHTOOL Command**

**ethtool** is a replacement of **mii-tool**. It is to view, setting speed and duplex of your **Network Interface Card**(**NIC**). You can set duplex permanently.

**# ethtool eth0**

Settings for eth0:

Current message level: 0x00000007 (7)

Link detected: yes

**-------------------------------------------------------------------------------------------------------------**

**What is Kernel:**

A kernel is the lowest level of software that interfaces with the hardware in computer. It is responsible for interfacing all applications to the physical hardware and allowing processes to get information from each other.

The kernel is highly involved in **resource management**. It must make sure that there is enough memory available for an ap+plication to run, as well as to place an application in the right location in memory. It tries to optimize the usage of the processor so that it can complete tasks as quickly as possible. It also aims to avoid deadlocks, which are problems that completely halt the system when one application needs a resource that another application is using.

Linux is **Monolithic kernel**. Kernel file is stored in **/boot** folder.

Kernel modules, also known as a loadable kernel module (LKM), are essential to keeping the kernel functioning with all your hardware without consuming all your available memory.

A module typically adds functionality to the base kernel for things like devices, file systems, and system calls. LKMs have the file extension .ko and are typically stored in the /lib/modules directory. Because of their modular nature you can easily customize your kernel by setting modules to load, or not load, during startup with by editing your /boot/config file.

**-------------------------------------------------------------------------------------------------------------**

**What is SWAP Partition?**

*Swap space* in Linux is used when the 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. Swap space is located on hard drives, which have a slower access time than physical memory.

Swap space can be a dedicated swap partition, a swap file. There is no need to be alarmed if you find the swap partition filled to 50%. The fact that swap space is being used does not mean a memory bottleneck but rather proves how efficiently Linux handles system resources. Also, a swapped-out page stays in swap space until there is a need for it, that is when it gets moved in (swap-in).

The memory which is going to be moved to swap partition will be dependent on “**swappiness**”, which is configurable. A **higher** swappiness means that items are more likely to be moved to the SWAP partition; a **lower** swappiness means that items are less likely to be moved to the SWAP partition.

swappiness can have a value of between 0 and 100

swappiness=0 tells the kernel to avoid swapping processes out of physical memory for as long as possible

swappiness=100 tells the kernel to aggressively swap processes out of physical memory and move them to swap cache

SWAP partition is used as the destination of your memory’s contents whenever you tell your system to hibernate. This means that without a SWAP partition, hibernation on Linux is impossible.

Swap should equal 2x physical RAM for up to 2 GB of physical RAM, and then an additional 1x physical RAM for any amount above 2 GB, but never less than 32 MB.

To check the current swappiness value  
$ cat /proc/sys/vm/swappiness  
60  
  
To change the value  
# echo 40 > /proc/sys/vm/swappiness  
To make the changes affect  
# sysctl -p

**-------------------------------------------------------------------------------------------------------------**

**Editor in Linux: (nano, vim)**

**vi** editor:

We use vi editor in following way.

1. **vi script.sh**

To insert text in editor

1. **i <text>**

exit the insert mode with **Exit**

Save the file and quit vi

1. **:wq**

**-------------------------------------------------------------------------------------------------------------**

## Directory structure

Directories are also known as folders because they can be thought of as folders in which files are kept in a sort of physical desktop analogy.

In Linux and many other operating systems, directories can be structured in a tree-like hierarchy. The Linux directory structure is well defined and documented in the [Linux Filesystem Hierarchy Standard](http://www.pathname.com/fhs/) (FHS). Referencing those directories when accessing them is accomplished by using the sequentially deeper directory names connected by forward slashes (/) such as /var/log and /var/spool/mail. These are called paths.

The following table provides a very brief list of the standard, well-known, and defined top-level Linux directories and their purposes.

| **Directory** | **Description** |
| --- | --- |
| / (root filesystem) | The root filesystem is the top-level directory of the filesystem. It must contain all the files required to boot the Linux system before other filesystems are mounted. It must include all the required executables and libraries required to boot the remaining filesystems. After the system is booted, all other filesystems are mounted on standard, well-defined mount points as subdirectories of the root filesystem. |
| /bin | The /bin directory contains user **executable files.** |
| /boot | Contains the static **bootloader** and **kernel executable** and configuration files required to boot a Linux computer. |
| /dev | This directory contains the device files for every hardware device attached to the system. These are not device drivers, rather they are files that represent each device on the computer and facilitate access to those devices. |
| /etc | Contains the local **system configuration files** for the host computer. |
| /home | Home directory storage for user files. Each user has a subdirectory in /home. |
| /lib | Contains shared library files that are required to boot the system. |
| /media | A place to mount external removable media devices such as USB thumb drives that may be connected to the host. |
| /mnt | A temporary mountpoint for regular filesystems (as in not removable media) that can be used while the administrator is repairing or working on a filesystem. |
| /opt | Optional files such as vendor supplied application programs should be located here. |
| /root | This is not the root (/) filesystem. It is the home directory for the root user. |
| /sbin | System binary files. These are executables used for system administration. |
| /tmp | Temporary directory. Used by the operating system and many programs to store temporary files. Users may also store files here temporarily. Note that files stored here may be deleted at any time without prior notice. |
| /usr | These are shareable, read-only files, including executable binaries and libraries, man files, and other types of documentation. |
| /var | Variable data files are stored here. This can include things like log files, MySQL, and other database files, web server data files, email inboxes, and much more. |

Table 1: The top level of the Linux filesystem hierarchy.

**What is Boot process in Linux:**

1. **BIOS:** Basic input output system.

It performs basic integrity check. It does POST check which is very basic things to start OS Ex. It checks if storage device is connected properly and checks whether video display is running.

It searches, loads and execute boot loader program.

It searches for boot loader in removable devices, hard drive, CD rom, Hard drive and SD card.

BIOS loads and execute MBR.

(we can change boot sequence by pressing key f2 or f12 depending on system.)

1. **MBR:** Master boot record. The MBR contains the data to let the system know about the partition on disk.

It is in first sector of bootable disk. Ex. /dev/had or /dev/sda

It is 512 bytes in size.

It has 3 components:

1. Primary boot loader info.
2. Partition table info.
3. MBR validation check

In last 2 bytes it contains information about GRUB. In short it loads and executes GRUB.

**3. GRUB**: Grand Unified Bootloader.

If you have multiple kernel image in system, then u can select which kernel image to be loaded and executed. It displays splash screen and wait for some time. If nothing is entered, then It will load default kernel image.

Grub configuration file is /boot/grub/grub.config

Grub has knowledge of file system. Grub loads and execute Kernel image.

**4. Kernel**: It mounts root file system as specified in GRUB configuration file. Kernel executes init program in system.

5. **Init (Initialization):** This. process decides Run Level. Then Run level decides which initial process to be loaded on start-up.

The function around init are starts with file **etc/inittab.**

It has information about scripts on every run level

Following are some Run level:

1. - halt (Shut down system)
2. Single user mode
3. Multiuser, without NFS (without networking)
4. Full multiuser mode (With Networking)
5. Unused (User defined)
6. X11 (Multi user with networking)
7. Reboot

6. **Run Level**: System now executes program depends on current run level. There are 7 run level directories.

Under etc/init.d/rc\*d direcotories you would see program that starts with S and K.

Run level 0 – /etc/init.d/rc0.d/

* Run level 1 – /etc/init.d/rc1.d/
* Run level 2 – /etc/init.d/rc2.d/
* Run level 3 – /etc/init.d/rc3.d/
* Run level 4 – /etc/init.d/rc4.d/
* Run level 5 – /etc/init.d/rc5.d/
* Run level 6 – /etc/init.d/rc6.d/

Program start with S used during startup. Program starts with K used during shutdown.

**-------------------------------------------------------------------------------------------------------------**

### **How to mount filesystem?**

Making the contents of a filesystem available to the OS and user is called mounting. In Linux, each filesystem is mounted at a point within the filesystem hierarchy, known as the mount point.

There are two main ways to mount a filesystem at boot: **manually** and **automatically**

1. The system filesystems, such as **/**and **/home**, are mounted automatically at boot, using information stored in the filesystem table **/etc/fstab.**

**2.** Mounting manually is done with the mount command:

**sudo mount /dev/sda4 /mnt/backup -t ext4**

The first two options are the device and mount point, -t specifies the filesystem type; if you omit this, auto is used.

Unmounting is done with the unmount command. This command also accepts multiple paths to unmount several at once:

**sudo umount /mnt/music /mnt/photos /dev/sda5**

If targeted device is busy, then we won’t be able to unmount it and will show error message.

To view mounted file system use following commands:

**#mount**

Information about mounted file system is stored in /etc/mtab folder. To view its content use following command:

**# cat /etc/mtab**

### **Creating filesystems**

After creating partition using fdisk or gdisk command, we need to make filesystem on them.

**#sudo mkfs.ext4 /dev/sdb1**

**-------------------------------------------------------------------------------------------------------------**

**TAR (Method of compressing file in linux):**

The standard archival program for Unix-like operating systems is Tar.

Compression programs work on one file or stream of data and produce one compressed file or stream, so this splits the job into two parts: **archival** and **compression.**

Use the following command to compress an entire directory or a single file on Linux. It’ll also compress every other directory inside a directory you specify–in other words, it works recursively.

tar -czvf name-of-archive.tar.gz /path/to/directory-or-file

Here’s what those switches mean:

* -c: **C**reate an archive.
* -z: Compress the archive with g**z**ip.
* -v: Display progress in the terminal while creating the archive, also known as “**v**erbose” mode. The v is always optional in these commands, but it’s helpful.
* -f: Allows you to specify the **f**ilename of the archive.

Let’s say you have a directory named “stuff” in the current directory and you want to save it to a file named archive.tar.gz. You’d run the following command:

tar -czvf archive.tar.gz stuff

Or, let’s say there’s a directory at /usr/local/something on the current system and you want to compress it to a file named archive.tar.gz.

tar -czvf archive.tar.gz /usr/local/something

For example, let’s say you want to compress /home/ubuntu, but you don’t want to compress the /home/ubuntu/Downloads and /home/ubuntu/.cache directories. Here’s how you’d do it:

tar -czvf archive.tar.gz /home/ubuntu --exclude=/home/ubuntu/Downloads --exclude=/home/ubuntu/.cache

You could archive an entire directory and exclude all .mp4 files with the following command:

tar -czvf archive.tar.gz /home/ubuntu --exclude=\*.mp4

tar also supports bzip2 compression. This allows you to create bzip2-compressed files, often named .tar.bz2, .tar.bz, or .tbz files. To do so, just replace the -z for gzip in the commands here with a -j for bzip2.

tar -cjvf archive.tar.bz2 stuff

## Extract an Archive

Once you have an archive, you can extract it with the tar command. The following command will extract the contents of archive.tar.gz to the current directory.

tar -xzvf archive.tar.gz

It’s the same as the archive creation command we used above, except the -x switch replaces the -c switch.

You may want to extract the contents of the archive to a specific directory. You can do so by appending the -C switch to the end of the command. For example, the following command will extract the contents of the archive.tar.gz file to the /tmp directory.

tar -xzvf archive.tar.gz -C /tmp

# **proc file system in Linux**

Proc file system is virtual file system created when system boots and is dissolved at time of system shut down.

It contains the useful information about the processes that are currently running, it is regarded as control and information Centre for kernel.

Below is snapshot of /proc from my PC.

**ls -l /proc**

total 0

dr-xr-xr-x 9 root root 0 Mar 31 21:34 1

dr-xr-xr-x 9 root root 0 Mar 31 21:34 10

dr-xr-xr-x 9 avahi avahi 0 Mar 31 21:34 1034

dr-xr-xr-x 9 root root 0 Mar 31 21:34 1036

dr-xr-xr-x 9 root root 0 Mar 31 21:34 1039

dr-xr-xr-x 9 root root 0 Mar 31 21:34 1041

dr-xr-xr-x 9 root root 0 Mar 31 21:34 1043

dr-xr-xr-x 9 root root 0 Mar 31 21:34 1044

 you can check that there is entry for every running process in /proc file system.

ls -ltr /proc/7494

Output:

total 0

-rw-r--r-- 1 mandeep mandeep 0 Apr 1 01:14 oom\_score\_adj

dr-xr-xr-x 13 mandeep mandeep 0 Apr 1 01:14 task

-r--r--r-- 1 mandeep mandeep 0 Apr 1 01:16 status

-r--r--r-- 1 mandeep mandeep 0 Apr 1 01:16 stat

-r--r--r-- 1 mandeep mandeep 0 Apr 1 01:16 cmdline

-r--r--r-- 1 mandeep mandeep 0 Apr 1 01:17 wchan

**-------------------------------------------------------------------------------------------------------------**

**System calls used for Process management:**  
Fork () :- Used to create a new process  
Exec() :- Execute a new program  
Wait():- wait until the process finishes execution  
Exit():- Exit from the process  
Getpid():- get the unique process id of the process  
Getppid():- get the parent process unique id  
Nice():- to bias the existing property of process

**-------------------------------------------------------------------------------------------------------------**

**What is Process?**

A process refers to a program in execution; it’s a running instance of a program.

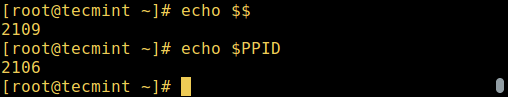
**Types of Processes**

There are fundamentally two types of processes in Linux:

* Foreground processes – these are initialized and controlled through a terminal session. In other words, there must be a user connected to the system to start such processes; they haven’t started automatically as part of the system functions/services.
* Background processes – are processes not connected to a terminal; they don’t expect any user input.

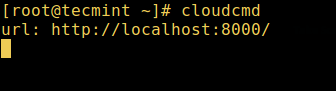
Using fork() and exec() Function, we can create new process.

A program is identified by its process ID (PID) as well as its parent processes ID (PPID), therefore processes can further be categorized into:

* Parent processes – these are processes that create other processes during run-time.
* Child processes – these processes are created by other processes during run-time.
* To find the process ID and parent process ID of the current shell, run:
* $ echo $$
* $ echo $PPID
* [](https://www.tecmint.com/wp-content/uploads/2017/03/Find-Linux-Parent-Process-ID.png)

run a program ex. cloudcmd, it will start a process in the system. You can start a foreground process as follows, it will be connected to the terminal and a user can send input it:

# cloudcmd

[](https://www.tecmint.com/wp-content/uploads/2017/03/Start-Linux-Interactive-Process.png)

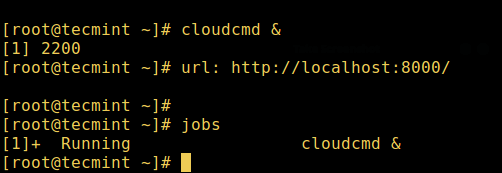
*Start Linux Interactive Process*

#### Linux Background Jobs

To start a process in the background (non-interactive), use the **&** symbol, here, the process doesn’t read input from a user until it’s moved to the foreground.

# cloudcmd &

# jobs

[](https://www.tecmint.com/wp-content/uploads/2017/03/Start-Linux-Process-in-Background.png)

*Start Linux Process in Background*

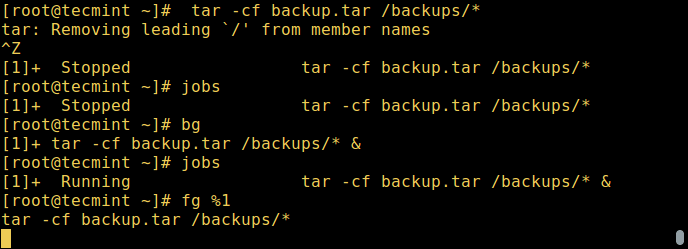
To continue running the above-suspended command in the background, use the **bg** command:

# bg

To send a background process to the foreground, use the **fg** command together with the job ID like so:

# jobs

# fg %1

[](https://www.tecmint.com/wp-content/uploads/2017/03/Linux-Background-Process-Jobs.png)

*Linux Background Process Jobs*

#### States of a Process in Linux

During execution, a process changes from one state to another depending on its environment/circumstances. In Linux, a process has the following possible states:

* **Running** – here it’s either running (it is the current process in the system) or it’s ready to run (it’s waiting to be assigned to one of the CPUs).
* **Waiting** – in this state, a process is waiting for an event to occur or for a system resource.
* **Stopped** – in this state, a process has been stopped, usually by receiving a signal. For instance, a process that is being debugged.
* **Zombie** – here, a process is dead, it has been halted but it’s still has an entry in the process table.

### **How to Control Processes in Linux**

Linux also has some commands for controlling processes such as **kill, pkill, pgrep** and **killall.**

$ pgrep -u tecmint top

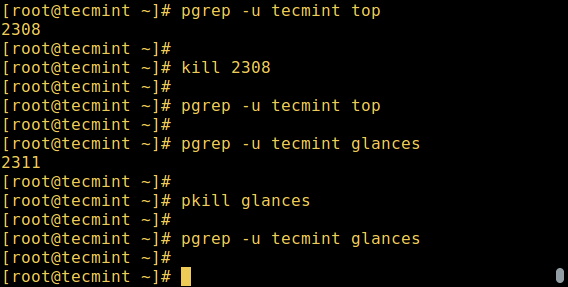
$ kill 2308

$ pgrep -u tecmint top

$ pgrep -u tecmint glances

$ pkill glances

$ pgrep -u tecmint glances

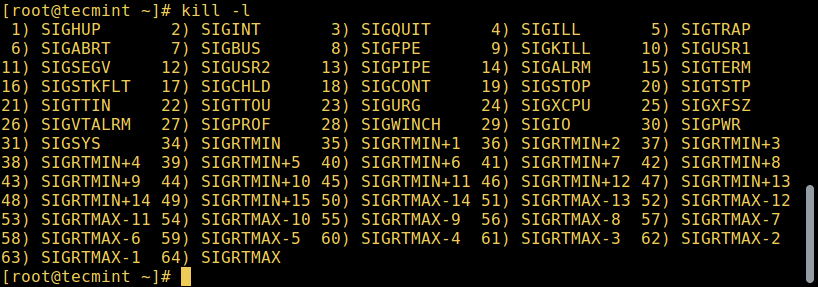
[](https://www.tecmint.com/wp-content/uploads/2017/03/Control-Linux-Processes.png)

*Control Linux Processes*

#### Sending Signals to Processes

The fundamental way of controlling processes in Linux is by sending signals to them. There are multiple signals that you can send to a process, to view all the signals run:

$ kill -l

[](https://www.tecmint.com/wp-content/uploads/2017/03/list-all-signals.png)

*List All Linux Signals*

To send a signal to a process, use the kill, pkill or pgrep commands we mentioned earlier on. But programs can only respond to signals if they are programmed to recognize those signals.

* **SIGHUP 1** – sent to a process when its controlling terminal is closed.
* **SIGKILL 9** – this signal immediately kills a process and the process will not perform any clean-up operations. (not safe)
* **SIGTERM 15** – this a program **termination** signal (kill will send this by default).

The following are kill commands examples to kill the Firefox application using its PID once it freezes:

$ pidof firefox

$ kill 9 2687

OR

$ kill -KILL 2687

OR

$ kill -SIGKILL 2687

If a process have too many instances and a number of child processes, we have a command ‘**killall**‘. This is the only command of this family, which takes process name as argument in-place of process number.

##### **Syntax:**

To kill all **mysql instances** along with child processes, use the command as follow.

# killall mysqld

$ killall firefox

#### Changing Linux Process Priority

The **kernel scheduler** is a unit of the kernel that determines the most suitable process out of all runnable processes to execute next.

By default, all the processes are considered equally urgent and are allotted the same amount of CPU time. The nice parameter is used to change priority. The Linux kernel then reserves CPU time for each process based on its relative priority value.

It ranges from **-** **20 to 19** and can take only integer values. A value of minus 20 represents the highest priority level, whereas 19 represents the lowest.

**Ex**. the following command line starts the process "large-job," setting the **nice** value to 12:

nice -12 large-job

To use higher priorities (negative nice values), administrator privileges are required.

You can change the priority of a job that is already running using **renice**. For example:

renice 17 -p 1134

This changes the nice value of the job with process id 1134 to 17. In this case, no dash is used for the command option when specifying the nice value.

### **Explain the terms suid, sgid and sticky bit?**

In addition to the basic file permissions in Linux, there are few special permissions that are available for executable files and directories.

**SUID :** If setuid bit is set, when the file is executed by a user, the process will have the same rights as the owner of the file being executed.

**SGID :** Same as above, but inherits group privileges of the file on execution, not user privileges. Similar way when you create a file within the directory, it will inherit the group ownership of the directories.

**Sticky bit** : Sticky bit was used on executables in linux so that they would remain in the memory more time after the initial execution, hoping they would be needed in the near future. But mainly it is on folders, to imply that a file or folder created inside a stickybit enabled folder could only be deleted by the owner.

**What is Init process?**

**Init** process is the parent process of all processes on the system and started by kernel, it’s the first program that is executed when the [Linux system boots up](https://www.tecmint.com/linux-boot-process/); it manages all other processes on the system.

The init process always has process ID of **1**.

init is centrally configured in the /etc/inittab file where the runlevels are defined. Depending on the entries in /etc/inittab, several **init scripts** are run all reside in the directory /etc/init.d.

There are two types of scripts in /etc/init.d:

Scripts Executed Directly by init

This is the case only during the boot process or if an immediate system shutdown is initiated (power failure or a user pressing Ctrl-Alt-Del). The execution of these scripts is defined in/etc/inittab.

Scripts Executed Indirectly by init

These are run when changing the runlevel and always call the master script **/etc/init.d/rc**, which guarantees the correct order of the relevant scripts.

### **What is initrd image and what is its function in the linux booting process?**

The initial RAM disk (initrd) is an initial root file system that is mounted prior to when the real root file system is available. The initrd is bound to the kernel and loaded as part of the kernel boot procedure. The kernel then mounts this initrd as part of the two-stage boot process to load the modules to make the real file systems available and get at the real root file system.

**What is Runlevel?**

When a Linux system boots, it launches the **init**processes. init is responsible for launching the other processes on the system. For example, when you start your Linux computer, the kernel starts init, and init executes the **startup scripts** to initialize your **hardware**, bring up **networking**, start your **graphical desktop**.

However, there isn’t just one single set of startup scripts init executes. There are multiple run levels with their own startup scripts – for example, one runlevel may bring up networking and launch the graphical desktop, while another runlevel may leave networking disabled and skip the graphical desktop.

Only one runlevel is executed when the system is booted.

0- halt (Shut down system)

1-Single user mode

2-Multiuser, without NFS (Without networking)

3-Full multiuser mode (With Networking)

4-Unused (User defined)

5-X11 (Multi user with networking)

6-Reboot

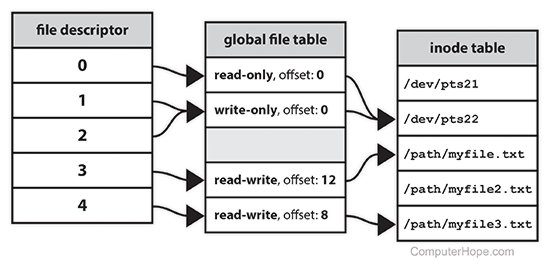
**-------------------------------------------------------------------------------------------------------------**

**What is file descriptor?**

A **file descriptor** is a number that uniquely identifies an **opened**[**file**](https://www.computerhope.com/jargon/f/file.htm) in a computer's [operating system](https://www.computerhope.com/os.htm).

When a program asks to open a file to the [kernel](https://www.computerhope.com/jargon/k/kernel.htm) of the operating system, it grants access, makes an entry in the **global file table**. The descriptor is identified by a unique non-negative [integer](https://www.computerhope.com/jargon/i/integer.htm), such as **0**, **12**, or **567**. At least one file descriptor exists for every opened file on the system.

The global file table entry contains information such as the [**inode**](https://www.computerhope.com/jargon/i/inode.htm) of the file, **byte**[**offset**](https://www.computerhope.com/jargon/o/offset.htm), and the **access restrictions**.



## **stdin, stdout, and stderr**

On a Unix-like operating system, the first three file descriptors, by default, are **STDIN**([standard input](https://www.computerhope.com/jargon/s/stdin.htm)), **STDOUT** (standard output), and **STDERR** (standard error).

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **File descriptor** | **Description** | **Abbreviation** |
| Standard input | **0** | The default data stream for input, for example in a command pipeline. In the [terminal](https://www.computerhope.com/jargon/t/terminal.htm), this defaults to keyboard input from the user. | **stdin** |
| Standard output | **1** | The default data stream for output, for example when a command prints text.  In the terminal, this defaults to the user's screen. | **stdout** |
| Standard error | **2** | The default data stream for output that relates to an error occurring. In the terminal, this defaults to the user's screen. |  |

**-------------------------------------------------------------------------------------------------------------**

**Process to new partition:**

To check current partition in system:

#lsblk

To create normal partition, write following command. Then follow all instructions and option accordingly

#fdisk /disk/sdb

Now we need to define LVM physical volume. Type following command

#pvcreate /dev/sdb

Now we will create volume group.

#vgcreate /vg1 /dev/sdb

Now we create logical volume

#Lvcreate -L +3G -n -lv1 -vg1

Create file system on LV:

mkfs -t ext4/dev/volume\_group/logical\_volume

Mounting logical volume:

mount -t ext4 /dev/volume\_group/logical\_volume /mount\_point

**-------------------------------------------------------------------------------------------------------------**

**What is inode?**

An inode stores basic information about a regular file, directory, or other file system object.

Each object in the filesystem is represented by an inode. Each file under Linux has following attributes:

=> File type (executable, block special etc)  
=> Permissions (read, write etc)  
=> Owner  
=> Group  
=> File Size  
=> File access, change and modification time  
=> Number of links (soft/hard)  
=> Access Control List (ACLs)

All the above information stored in an inode. Each inode is identified by a unique inode number within the file system.

You can use **ls -i** command to see inode number of file  
$ ls -i /etc/passwd  
Sample Output

32820 /etc/passwd

You can also use **stat** command to find out inode number and its attribute:  
$ stat /etc/passwdOutput:

File: `/etc/passwd'

Size: 1988 Blocks: 8 IO Block: 4096 regular file

Device: 341h/833d Inode: 32820 Links: 1

Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)

Access: 2005-11-10 01:26:01.000000000 +0530

Modify: 2005-10-27 13:26:56.000000000 +0530

**-------------------------------------------------------------------------------------------------------------**

**User management in Linux:**

A user or account of a system is uniquely identified by a numerical number called the UID (unique identification number). A super user can add, delete and modify a user account. The full account information is stored in the **/etc/passwd** file and password is stored in the file **/etc/shadow**

A user can be added by running the **useradd** command at the command prompt. After creating the user, set a password using the  
**passwd** utility, as follows:

# **useradd Shantanu**

# **passwd Shantanu**

**Locking and unlocking a user:** A super user can lock and unlock a user account. To lock an account, one needs to invoke *passwd* with the *-l* option.

|  |
| --- |
| # **passwd -l Shantanu** |

Unlocking password for user Shantanu

|  |
| --- |
| # **passwd -u Shantanu** |

**Linux group**  
Linux group is a mechanism to organize a collection of users. Like the user ID, each group is also associated with a unique ID called the GID (group ID). There are two types of groups – a primary group and a supplementary group. Each user is a member of a primary group and of zero or ‘more than zero’ supplementary groups. The group information is stored in **/etc/group** and the respective passwords are stored in the  file.

**Creating a group with default settings:** To add a new group with default settings, run the groupadd command as a root user, as shown below:

|  |
| --- |
| # **groupadd employee** |

If you wish to add a password, then type *gpasswd* with the group name, as follow:

|  |
| --- |
| # **gpasswd employee** |

**Ex**. We are adding new user Ironman to group Superhero by using following command.

**useradd -G Superhero Ironman**

Ex. If Ironman user already in system and we are adding then

**Usermod -G Superhero Ironman**

To check content of group file:

**Cat /etc/group**

**-------------------------------------------------------------------------------------------------------------**

**Linux Log Analysis:**

In linux log files are stored in **/var/log** directory. You will find different type of logs such as Authentication log, Kernel logs, boot logs, access logs.

Ex. **Sudo less boot.log**

Apart from less command we can use **cat, more, grep, tail** command to view log file.

Ex**. grep [username] /var/log/auth.log**

To view common log message, use the following command.

Ex. **less /var/log/message**

All the above logs are generated using **rsyslogd** service. It is a system utility providing support for message logging. Support of both internet and unix domain sockets enables this utility to support both local and remote logging. You can view its config file by tying the following command:  
# vi /etc/rsyslog.conf  
# ls /etc/rsyslog.d/

**-------------------------------------------------------------------------------------------------------------**

**Linux Resource utilization:**

**Vmstat command**: **vmstat** command reports information about processes, memory, paging, block IO, traps, and cpu activity.

**Vmstat:**

O/P

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

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

3 0 0 2485120 621952 2415368 0 0 0 0 0 1 32 0 68 0

#### (A) PROCS IS THE PROCESS-RELATED FIELDS ARE:

* r: The number of processes waiting for run time.
* b: The number of processes in uninterruptible sleep.

#### (B) MEMORY IS THE MEMORY-RELATED FIELDS ARE:

* 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.

#### (C) SWAP IS SWAP-RELATED FIELDS ARE:

* si: Amount of memory swapped in from disk (/s).
* so: Amount of memory swapped to disk (/s).

#### (D) IO IS THE I/O-RELATED FIELDS ARE:

* bi: Blocks received from a block device (blocks/s).
* bo: Blocks sent to a block device (blocks/s).

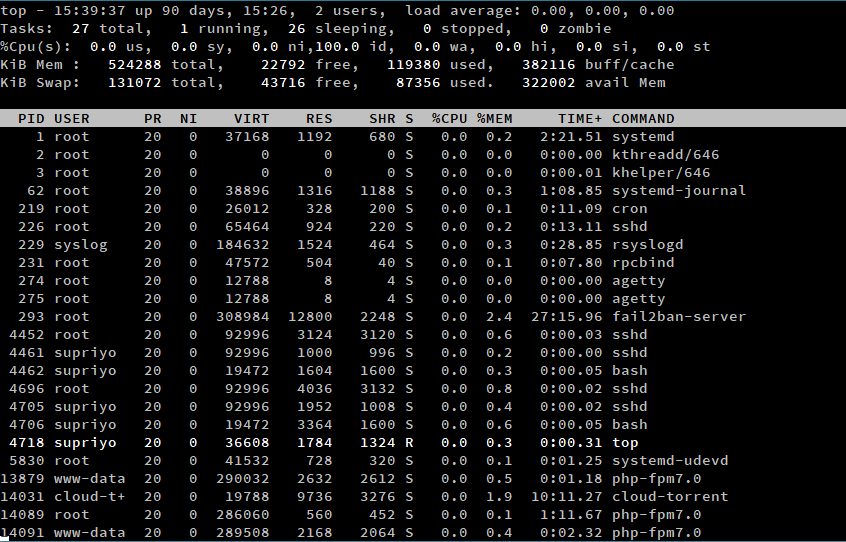
#### (E) SYSTEM IS THE SYSTEM-RELATED FIELDS ARE:

* in: The number of interrupts per second, including the clock.
* cs: The number of context switches per second.

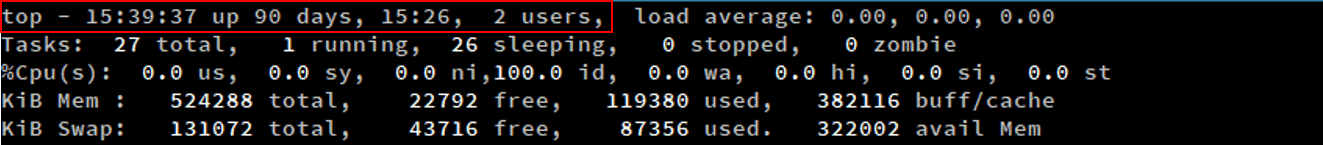
**TOP** command:

The top program provides a dynamic real-time view of a running system. It can display system memory usage, CPU usage, summery of tasks, system uptime and user session.

If you want to quit, simply press **“q”.**



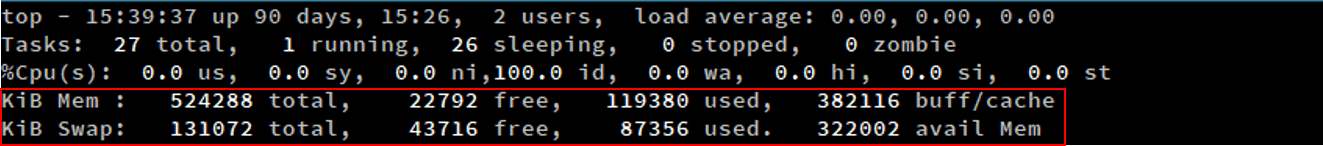
### **System time, uptime and user sessions**



At the very top left of the screen, top displays the current time. This is followed by the system uptime, which tells us the time for which the system has been running. For instance, in our example, the current time is “15:39:37”, and the system has been running for 90 days, 15 hours and 26 minutes.

Next comes the number of active user sessions. In this example, there are two active user sessions.

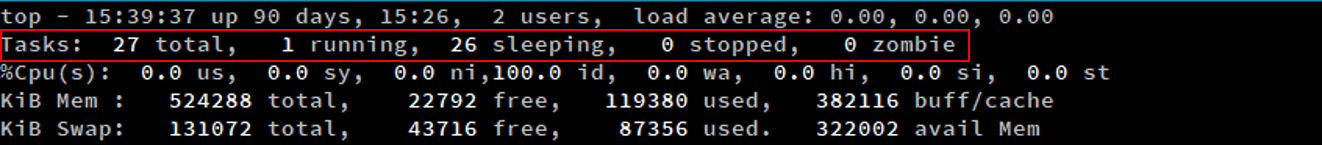
### **Memory usage**



The “memory” section shows information regarding the memory usage of the system. The lines marked “Mem” and “Swap” show information about RAM and swap space respectively.

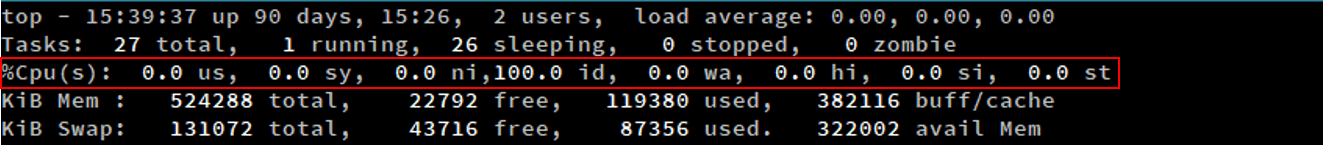
The Linux kernel also tries to reduce disk access times in various ways. It maintains a “disk cache” in RAM, where frequently used regions of the disk are stored. In addition, disk writes are stored to a “disk buffer”, and the kernel eventually writes them out to the disk. The total memory consumed by them is the “buff/cache” value.

### **Tasks**



The “Tasks” section shows statistics regarding the processes running on your system. The “total” value is simply the total number of processes. For example, in the above screenshot, there are 27 processes running.

### **CPU usage**



The CPU usage section shows the percentage of CPU time spent on various tasks. The us value is the time the CPU spends executing processes in userspace. Similarly, the sy value is the time spent on running kernelspace processes.

## **Understanding top’s interface: the task area**

* **PID**

This is the process ID, an unique positive integer that identifies a process.

* **USER**

This is the “effective” username (which maps to an user ID) of the user

**PR and NI**

The “NI” field shows the “nice” value of a process. The “PR” field shows the scheduling priority of the process from the perspective of the kernel.

**VIRT, RES, SHR and %MEM**

These three fields are related with to memory consumption of the processes. “VIRT” is the total amount of memory consumed by a process. “RES” is the memory consumed by the process in RAM, and Finally, “SHR” is the amount of memory shared with other processes.

* **S**

As we have seen before, a process may be in various states. This field shows the process state in the single-letter form.

* **TIME+**

This is the total CPU time used by the process since it started, precise to the hundredths of a second.

* **COMMAND**

The COMMAND column shows the name of the processes.

**-------------------------------------------------------------------------------------------------------------**

#### ****16. Analyzing Linux Server Performance****

The best way to improve efficiency of a system is to target bottlenecks that result in limiting overall speed. They usually can be identified by knowing the specifications of the system, but there are some basic indications:

* Sometimes the computer becomes slow when big applications such as Open office and Firefox are running at the same time. Then there is more of a chance of insufficiency of the amount of RAM.
* If boot time is slow and applications take a lot of time to load the first time they are launched, but run fine afterwards, then the hard drive may be working too slowly.
* Lower CPU usage if the CPU load is consistently high – even when RAM is available. CPU load time can be monitored in many ways, for instance with an independent[monitoring tool that analyzes CPU](https://www.monitis.com/sign-up).

**-------------------------------------------------------------------------------------------------------------**

**Iptables:**

There are 3 types of built in chains in Iptables

* INPUT: Packet coming into the PC.
* FORWARD: Packets passing through PC.
* OUTPUT: Packet leaving out of PC.

There are commonly used switches in Iptables.

-s: Source address

-d: Destination address

-p: Protocol

-j: Action

-P: Specify default policy for chain

-L: List chain rules

-A: Append rule to end of chain

-I: Append rule to start of chain

-i: Interface

Command to check Iptable rule in chain:

**iptables -L**

We can define rule which governs the traffic which we wish to allow first in Iptable. Then we can add catch-all at the bottom of these rule. Catch-all will block other traffic which is not previously allowed.

**EX**. Allow HTTP traffic for Apache web server over port 80 so it may service web request.

**Iptables -A INPUT -j ACCEPT -p tcp --destination-port 80 -i eth0**

**EX.** Allow FTP traffic for VSFTPD over port 21 to service FTP request

**Iptables -A INPUT -j ACCEPT -p tcp – destination-port 21 -i eth0**

Once we apply all rules to allow appropriate traffic we can apply **catch-all** rule to block traffic which don’t wish to allow. This rule must be applied at last

**Iptables -A INPUT -j DROP -p tcp -i eth0**

**-------------------------------------------------------------------------------------------------------------**

**Hard Links and Soft Links**

A link in UNIX is a pointer to a file.

Each **hard link** reference the same physical file location. Hard links more flexible and remain linked even if original file is deleted. Hard links are **unable to cross different file systems**. So, it is used within filesystem.

**Inode number will be same for both** hard link and original file.

ls -i command shows all the links with the link column shows number of links

**Symbolic link (Symlinks/Soft links)** are links between files. It is nothing but a shortcut of a file (in windows terms).

* You can delete the soft links without affecting the actual file or directory it is pointing to. The reason is because the **inode** of the linked file is different from that of the **inode** of the symbolic link. But if you delete the source file of the soft link, soft link of that file no longer works, or it becomes “dangling link” which points to nonexistent file. **Inode number will be** **different for both** soft link and original file.
* Soft link **can span across filesystem**.

**-------------------------------------------------------------------------------------------------------------**

**What is Daemon?**

A *daemon* is a type of [program](http://www.linfo.org/program.html) that runs unobtrusively in the background, rather than under the direct control of a user, waiting to be activated by the occurrence of a specific event or condition.

Daemons are recognized by the system as any processes whose ***parent process* has a PID of one**, which always represents the process *init*. Init *adopts* any process whose *parent* process *dies* without waiting for the *child* process's status. Thus, the common method for launching a daemon involves **forking** (i.e., dividing) once or twice, and making the parent processes die while the child process begins performing its normal function.

Some daemons are launched via [System V](http://www.linfo.org/system_v.html)init *scripts*, which are *scripts* that are run automatically when the system is booting up.

|  |  |
| --- | --- |
| [**init**](https://en.wikipedia.org/wiki/Init)[[1]](https://en.wikipedia.org/wiki/List_of_Unix_daemons#cite_note-verma2006-1) | The Unix program which spawns all other processes. As of 2016, for major Linux distributions, it has been replaced by [systemd](https://en.wikipedia.org/wiki/Systemd).[[2]](https://en.wikipedia.org/wiki/List_of_Unix_daemons#cite_note-2) |
|  |  |
| [**crond**](https://en.wikipedia.org/wiki/Cron)[[1]](https://en.wikipedia.org/wiki/List_of_Unix_daemons#cite_note-verma2006-1) | Time-based [job scheduler](https://en.wikipedia.org/wiki/Job_scheduler), runs jobs in the background. |
| [**dhcpd**](https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol) | Dynamically configure TCP/IP information for clients. |
|  |  |
| [**ftpd**](https://en.wikipedia.org/wiki/File_Transfer_Protocol)[[1]](https://en.wikipedia.org/wiki/List_of_Unix_daemons#cite_note-verma2006-1) | Services [FTP](https://en.wikipedia.org/wiki/File_Transfer_Protocol) requests from a remote system. |
| [**httpd**](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) | [Web server](https://en.wikipedia.org/wiki/Web_server) daemon. |
| [**inetd**](https://en.wikipedia.org/wiki/Inetd)[[4]](https://en.wikipedia.org/wiki/List_of_Unix_daemons#cite_note-stevens_fenner_rudocc2004-4) | Listens for network connection requests. If a request is accepted, it can launch a background daemon to handle the request, was known as the super server for this reason. Some systems use the replacement command [**xinetd**](https://en.wikipedia.org/wiki/Xinetd). |
|  |  |
| [**nfsd**](https://en.wikipedia.org/wiki/Network_File_System_(protocol))[[3]](https://en.wikipedia.org/wiki/List_of_Unix_daemons#cite_note-rosen_host_klee2006-3) | Processes [NFS](https://en.wikipedia.org/wiki/Network_File_System_(protocol)) operation requests from client systems. Historically each nfsd daemon handled one request at a time, so it was normal to start multiple copies. |
| **ntpd** | [Network Time Protocol](https://en.wikipedia.org/wiki/Network_Time_Protocol) daemon that manages clock synchronization across the network. **xntpd** implements the version 3 standard of NTP. |
|  |  |
| **sshd** | Listens for [secure shell](https://en.wikipedia.org/wiki/Secure_shell) requests from clients. |
|  |  |
| **swapper** | Copies process regions to [swap](https://en.wikipedia.org/wiki/Virtual_memory) space in order to reclaim physical [pages](https://en.wikipedia.org/wiki/Paging) of [memory](https://en.wikipedia.org/wiki/Computer_storage) for the [kernel](https://en.wikipedia.org/wiki/Kernel_(computer_science)). Also called **sched**. |
| [**syslogd**](https://en.wikipedia.org/wiki/Syslog) | System logger process that collects various system messages. |
| **syncd** | Periodically keeps the [file systems](https://en.wikipedia.org/wiki/File_system) synchronized with system memory. |
| [**systemd**](https://en.wikipedia.org/wiki/Systemd) | Replacement of [init](https://en.wikipedia.org/wiki/Init), the Unix program which spawns all other processes. |
| [**xfsd**](https://en.wikipedia.org/wiki/X_Font_Server) | Serve [X11](https://en.wikipedia.org/wiki/X_Window_System) [fonts](https://en.wikipedia.org/wiki/Font) to remote clients. |
|  |  |
|  |  |

.

**-------------------------------------------------------------------------------------------------------------**

**What is Zombie process?**

Zombies are basically the leftover bits of dead processes that haven’t been cleaned up properly.

When a process dies on Linux, the process’s status becomes **EXIT\_ZOMBIE** and the process’s parent is notified that its child process has died with the **SIGCHLD** signal. The parent process is then supposed to execute the **wait ()** system call to read the dead process’s exit status and other information. This allows the parent process to get information from the dead process. After wait () is called, the zombie process is completely removed from memory.

If a parent process isn’t programmed properly and never calls wait (), its zombie children will stick around in memory until they’re cleaned up.

the **top** command, and the **ps** command display zombie processes.

Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for **“Process Status”.** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options

Zombie process have PID assigned to it. So, if more zombie process are there then they may accommodate finite number of PID in system and will restrict other process to launch.

**How to kill zombie process?**

One way is by sending the SIGCHLD signal to the parent process. This signal tells the parent process to execute the wait() system call and clean up its zombie children. Send the signal with the **kill** command, replacing *pid* in the command below with the parent process’s PID:

#kill -s SIGCHLD pid

**-------------------------------------------------------------------------------------------------------------**

**How to check service in linux?**

If the service has an init script installed, you can use the service command to start, stop, and check the status of the service. The service command references a service by using its init script, which is stored in the **/etc/init.d**

A service can have any of the following statuses:

* start: The service has started.
* stop: The service has stopped running
* restart: The service is rebooting and will start after the process is complete

The following example shows how to check the status of httpd on CentOS using the service command.

$ sudo service httpd status

httpd is stopped

### Start the service

If a service isn’t running, you can use the service command to start it.

$ sudo service httpd start

Starting httpd:

**-------------------------------------------------------------------------------------------------------------**

**RAID** **(Redundant Array of Independent drives):**

Raid is just a collection of disks in a pool to become a logical volume.

RAID is managed using **mdadm** package

**RAID 0 (or) Striping**

In Raid 0 (Striping) the data will be written to disk using shared method. Half of the content will be in one disk and another half will be written to other disk.

Let us assume we have 2 Disk drives, for example, if we write data “TECMINT” to logical volume it will be saved as ‘T‘ will be saved in first disk and ‘E‘ will be saved in Second disk and ‘C‘ will be saved in First disk and again ‘M‘ will be saved in Second disk and it continues in round-robin process.

In this situation if any one of the drive fails we will lose our data. But while comparing to Write Speed and performance RAID 0 is Excellent. We need at least minimum 2 disks to create a RAID 0 (Striping).

1. High Performance.
2. Zero Fault Tolerance.

**RAID 1 (or) Mirroring**

Mirroring can make a copy of same data what we have. Assuming we have two numbers of 2TB Hard drives, total there we have 4TB, but because of mirroring we can see the 2TB of logical drive.

While we save any data, it will write to both 2TB Drives. Minimum two drives are needed to create a RAID 1 or Mirror. If a disk failure occurred, we can reproduce the raid set by replacing a new disk.

1. Good Performance.
2. Full Fault Tolerance.

**RAID 5 (or) Distributed Parity**

RAID 5 is mostly used in enterprise levels. **Parity** info will be used to rebuild the data. It rebuilds from the information left on the remaining good drives. This will protect our data from drive failure.

Assume we have 4 drives, if one drive fails and while we replace the failed drive we can rebuild the replaced drive from parity information. Parity information are Stored in all 4 drives, if we have 4 numbers of 1TB hard-drive. The parity information will be stored in 256GB in each driver and other 768GB in each drive will be defined for Users. RAID 5 can be survive from a single Drive failure, if drives fails more than 1 will cause loss of data’s.

1. Excellent Performance
2. Reading will be extremely very good in speed.
3. Fault tolerance

**RAID 6 Two Parity Distributed Disk**

RAID 6 is same as RAID 5 with two parity distributed system. Mostly used in many arrays. We need minimum 4 Drives, even if there 2 Drive fails we can rebuild the data while replacing new drives.

Very slower than RAID 5, because it writes data to all 4 drivers at same time. Will be average in speed while we are using a Hardware RAID Controller. If we have 6 numbers of 1TB hard-drives 4 drives will be used for data and 2 drives will be used for Parity.

1. Poor Performance.
2. Read Performance will be good.

**RAID 10 (or) Mirror & Stripe**

RAID 10 can be called as 1+0 or 0+1. This will do both works of Mirror & Striping.

Assume, we have 4 Number of drives. While I’m writing some data to my logical volume it will be saved under All 4 drives using mirror and stripe methods.

If I’m writing a data “TECMIN\*T” in RAID 10 it will save the data as follow. First “T” will write to both disks and second “E” will write to both disk, this step will be used for all data write. It will make a copy of every data to other disk too.

Same time it will use the RAID 0 method and write data as follow “T” will write to first disk and “E” will write to second disk. Again “C” will write to first Disk and “M” to second disk.

1. Good read and write performance.
2. Here Half of the Space will be lost in total capacity.
3. Fault Tolerance.

**What is Fstab?**

The configuration file /etc/fstab contains the necessary information to automate the process of mounting partitions. The fstab file can be used to define how disk partitions, various other block devices, or remote filesystems should be mounted into the filesystem.

In general, fstab is used for internal devices, CD/DVD devices, and network shares (samba/nfs/sshfs). Removable devices such as flash drives \*can\* be added to fstab.

1. [Device] [Mount Point] [File System Type] [Options] [Dump] [Pass]

|  |  |
| --- | --- |
| **fields** | **description** |
| <device> | The device/partition (by /dev location or UUID) that contain a file system. |
| <mount point> | The directory on your root file system (/mount) from which it will be possible to access the content of the device/partition. You may use any name you wish for the mount point, but you must create the mount point before you mount the partition. |
| <file system type> | Type of file system **Ex**. Vfat, ntfs, ext4, ext3, swap |
| <options> | Mount options of access to the device/partition **Ex**. Default, Sync, ro, rw |
| <dump> | This field sets whether the backup utility dump will backup file system. If set to "0" file system ignored, "1" file system is backed up.  Dump is seldom used and if in doubt use 0. |
| <pass num> | Fsck order is to tell fsck what order to check the file systems, if set to "0" file system is ignored.   1. 0 == do not check. 2. 1 == check this partition first. 3. 2 == check this partition(s) next |

**What is File system type in linux?**

### **Ext2 – Second Extended File System**

1. It was to overcome limitation of legacy Ext file system.
2. Maximum file size is **16GB – 2TB**.
3. Journaling feature is not available.
4. It’s being used for normally Flash based storage media like **USB Flash drive**, **SD Card** etc.

### **Ext3 – Third Extended File System**

1. It has **journaling** feature, which is to improve reliability and eliminates need to check all file system after unclean shutdown.
2. Max file size **16GB – 2TB**.
3. Provide facility to upgrade from Ext2 to Ext3 file systems without having to back up and restore data.

### **Ext4 – Fourth Extended File System**

1. Backward compatibility.
2. Max file size **16GB to 16TB.**
3. Ext4 file system have option to Turn Off **journaling** feature.
4. Other features like **checksum**, Sub Directory Scalability, Multiblock Allocation.

#### JFS

The Journaled File System (JFS) was developed by IBM for AIX UNIX which was used as an alternative to system ext. JFS is an alternative to ext4 currently and is used where stability is required with the use of very few resources. When CPU power is limited JFS comes handy.

#### Btrfs

B-Tree File System (Btrfs) focus on fault tolerance, fun administration, repair System, large storage configuration and is still under development. Btrfs is not recommended for Production System.

**What is journaling?**

A system crashes, sometimes the [loss of data](http://www.linux.org/article/view/journal-file-system/1) occurs. Using a journal allows data recovery of files.

When a user submits a change to a file, the first thing the file system does is to mark the changes in a journal file. The size of the journal file is a set size which when full, older entries are overwritten.

If a crash occurs, the journal entries and files are compared. Data is written to the files that are in the journal, but not yet on the disk. The process recovers the data to its wanted state.  
There are three types of Journaling: writeback, ordered and data.

**1. writeback**  
Here, only the metadata is journaled, and data is written to the file on the disk. In a crash, the file system is recoverable, but the physical data can be corrupted.

**2. ordered (default)**

The physical data is written first before the metadata is journaled. The ordered mode allows the data and file system to be uncorrupted if a system crashes before the journal is written.

**3. data**  
In the data mode, the metadata and file contents are journaled. System performance can be poorer than the other two modes, but the fault tolerance is much better.

## **How secured is Linux? Explain.**

Security is the most important aspect of an operating system. Due to its unique authentication module, Linux is considered as more secured than other operating systems. Linux consists of PAM. PAM is Pluggable Authentication Modules. It provides a layer between applications and actual authentication mechanism. It is a library of loadable modules which are called by the application for authentication. It also allows the administrator to control when a user can log in. All PAM applications are configured in the directory "/etc/pam.d" or in a file "/etc/pam.conf". PAM is controlled using the configuration file or the configuration directory.

**What is demand paging?**

Demand paging allows that pages should only be brought into memory if the executing process demands them. This is often referred to as [lazy evaluation](https://en.wikipedia.org/wiki/Lazy_evaluation) as only those pages demanded by the process are swapped from [secondary storage](https://en.wikipedia.org/wiki/Secondary_storage) to [main memory](https://en.wikipedia.org/wiki/Main_memory). Contrast this to pure swapping, where all memory for a process is swapped from secondary storage to main memory during the process startup.

Commonly, to achieve this process a [page table](https://en.wikipedia.org/wiki/Page_table) implementation is used. The page table maps [logical memory](https://en.wikipedia.org/wiki/Logical_memory) to [physical memory](https://en.wikipedia.org/wiki/Physical_memory). The page table uses a [bitwise](https://en.wikipedia.org/wiki/Bitwise_operation) operator to mark if a page is valid or invalid. A valid page is one that currently resides in main memory. An invalid page is one that currently resides in secondary memory. When a process tries to access a page, the following steps are generally followed:

* Attempt to access page.
* If page is valid (in memory) then continue processing instruction as normal.
* If page is invalid, then a **page-fault trap** occurs.
* Check if the memory reference is a valid reference to a location on secondary memory. If not, the process is terminated (**illegal memory access**). Otherwise, we have to **page in** the required page.
* Schedule disk operation to read the desired page into main memory.
* Restart the instruction that was interrupted by the operating system trap.

**What is OOM killer? How to configure it?**

**https://www.oracle.com/technetwork/articles/servers-storage-dev/oom-killer-1911807.html**

When a server that's supporting a database or an application server goes down, the root cause of the issue can be traced to the system running low on memory and killing an important process to remain operational.

The Linux kernel allocates memory upon the demand of the applications running on the system. Because many applications allocate their memory up front and often don't utilize the memory allocated, the kernel was designed with the ability to over-commit memory to make memory usage more efficient. This over-commit model allows the kernel to allocate more memory than it has physically available. If a process utilizes the memory it was allocated, the kernel then provides these resources to the application. When too many applications start utilizing the memory they were allocated, the over-commit model sometimes becomes problematic and the kernel must start killing processes to stay operational. The mechanism the kernel uses to recover memory on the system is referred to as the out-of-memory killer or OOM killer for short.

If we want to make our oracle process less likely to be killed by the OOM killer, we can do the following.

echo -15 > /proc/2592/oom\_adj

We can make the OOM killer more likely to kill our oracle process by doing the following.

echo 10 > /proc/2592/oom\_adj

If we want to exclude our oracle process from the OOM killer, we can do the following, which will exclude it completely from the OOM killer. It is important to note that this might cause unexpected behavior depending on the resources and configuration of the system. If the kernel is unable to kill a process using a large amount of memory, it will move onto other available processes. Some of those processes might be important operating system processes that ultimately might cause the system to go down.

echo -17 > /proc/2592/oom\_adj

We can set valid ranges for oom\_adj from -16 to +15, and a setting of -17 exempts a process entirely from the OOM killer. The higher the number, the more likely our process will be selected for termination if the system encounters an OOM condition. The contents of /proc/2592/oom\_score can also be viewed to determine how likely a process is to be killed by the OOM killer. A score of 0 is an indication that our process is exempt from the OOM killer. The higher the OOM score, the more likely a process will be killed in an OOM condition.

**What is quota? How to control the limit of memory blocs and file to users.**

Quotas are used to limit the amount of disk space a user or group can use on the VPS.

## **Installing Quota**

The mount file fstab needs to be opened for editing using the following command:

**#sudo nano /etc/fstab**

The quotas are enabled by adding a usrquota and/or grpquota to the mounting options of the main hard disk

Both options can be independently added depending on the desired result.

/dev/VolGroup00/LogVol02 /home ext3 defaults,usrquota,grpquota 1 2

Save the file and enable the new mount options by remounting the file system as follows:

**#mount -o remount /**

The following command will create a new quotas file in the root directory of the file system.

**#quotacheck** **-cum /**

The command consists of the following three parameters:

1. The **c** parameter indicates the creation of a new file, overwriting any previous files.
2. The **u** parameter indicates that a new user index file should be created. To also create a group index file, add the g command in the previous command.
3. The m parameter indicates that no read-only mount of the complete file system is required to generate the different index files.

program.

## **Configuring Quotas For Different Users**

The user quotas are configured using the edquota command, followed by the desired user name or group name. The command will open the default configured text editor. In this guide, we assume that the user ftpuser should receive a quota of 10Mb. The command used is as follows:

**#edquota ftpuser**

Which opens the quota file for editing

Disk quotas for user ftpuser (uid 1001):

Filesystem blocks soft hard inodes soft hard

/dev/disk/by-label/DOROOT 8 10000 10240 2 0 0

The text editor shows 7 different columns:

1. Indicates the name of the file system that has a quota enabled
2. Indicates the amount of blocks currently used by the user
3. Indicates the **soft block** limit for the user on the file system
4. Indicates the **hard block** limit for the user on the file system
5. Indicates the amount of inodes currently used by the user
6. Indicates the **soft inode** limit for the user on the file system
7. Indicates the **hard inode** limit for the user on the file system

The blocks refer to the amount of disk space, while the inodes refer to the number of files/folders that can be used. Most of the time the block amount will be used in the quota.

The hard block limit is the absolute maximum amount of disk space that a user or group can use. Once this limit is reached, no further disk space can be used. The soft block limit defines the maximum amount of disk space that can be used. However, unlike the hard limit, the soft limit can be exceeded for a certain amount of time. This time is known as the **grace period**.

In the example above, a soft limit off 9,785Mb and hard limit of 10Mb are used. To see the quota in action an FTP/SFTP transfer can be started, where multiple files will be uploaded with a total size of 12 Mb for example. The FTP/SFTP client will indicate a transfer error, meaning that the user will be unable to upload any files. Of course, 10Mb isn't a meaningful quota. In this guide every user will get a soft limit of 976 Mb and a hard limit of 1Gb. The configuration looks as follows:

## **Generating Reports**

It is possible to generate a report from the different quotas. The following command is used:

**#repquota -a**

User used soft hard grace used soft hard grace

------------------------------------------------------------------------------------

root -- 1118708 0 0 37093 0 0

daemon -- 68 0 0 4 0 0

man -- 9568 0 0 139 0 0

www-data -- 2908 0 0 15 0 0

nobody -- 0 0 0 1 0 0

libuuid -- 24 0 0 2 0 0

Debian-exim -- 44 0 0 10 0 0

mysql -- 30116 0 0 141 0 0

ftpuser -- 8 1000000 1048576 2 0 0

## **Optional: Specify A Grace Period**

To give current users some time to reduce their files on the droplet, a grace period can be configured. This is the allowed time a user can exceed their soft limit, while still staying under the hard limit. The grace period can be expressed in seconds, minutes, hours, days, weeks or months.

edquota -t

The command gives the following output and specifies the different time unites that could be used. For this guide, a grace period of 7 days is used.

Grace period before enforcing soft limits for users:

Time units may be: days, hours, minutes, or seconds

Filesystem Block grace period Inode grace period

/dev/disk/by-label/DOROOT 7days 7days

**Linux system is too slow. How to solve this performance issue?**

We need to find bottlenecks. The main reason can be following:

* processor (CPU overloaded)
* memory (high memory usage)
* network (slower network channels)
* disk (poor performance of disk drives)
* a bug with application or kernel.
* hardware issue (need to upgrade hardware)

**1. CPU:**

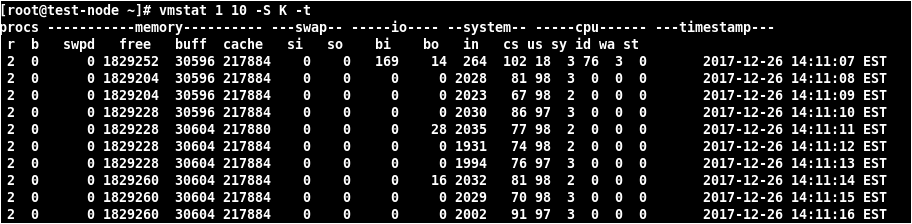
If any process is consuming more resources, then we need to reduce priority of process by doing **renice**. We many need to kill unnecessary process using **kill**.

**A**. First command to use is **uptime**. This is will give idea about load average. In this numbers shows average number of process waiting for CPU resources in last 1, 5, 15 min. If number is more than 1 then more resources are being used by process and some action needs to make.   
 

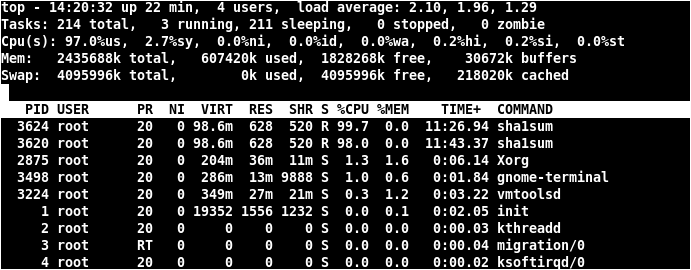
**B.** Use **top** command. It would show list of processes with CPU utilization and with priority value. Another command can be used is **vmstat**. If there is a spike in "%usr" value then this indicates that system has spent time in running a user space process, otherwise, if there is high usage percentage for "%sys" then it says that system was busy in running a kernel thread.

let’s spin up"sha1sum" command to create load on processor and see how it behaves now. I've started two of these processes in background. Now, you could see that the processor is busy in executing a user-space task as shown in the below snap: -

Using "vmstat" to print stats with 1 second interval of 10 samples and showing in kilo bytes with time stamps:



At this time, we come to know that processor is busy and there 2 tasks running now which are consuming high CPU cycles, so check what are those processes now using "top" command: -



**2. Memory (RAM):**

We will use **Free** command to analyses memory usage. If a system is showing high memory usage (almost no free memory left out) and swap space is being consumed, then it certainly indicates that system is under memory pressure.

Use **ps** command to find out top 10 processes consuming memory.

Check out page faults (major faults) using "sar -B" command, if there are major faults then it would certainly delay since page needs to be moved from disk to memory.

https://www.thomas-krenn.com/en/wiki/Linux\_Performance\_Measurements\_using\_vmstat

Example: To check the page faults happened between certain time stamp using sar command: -



**3. Hard disk drives:**

Slow disks would cause memory buffers being filled up which would delay all disk operations. CPU idle time would increase since CPU would be waiting for IO from disks.

**Vmstat** command could give disk stats with respective to blocks per second received and sent. Also, would show number of processes running or blocked.

**4. Network issue:**

Make sure that network drivers/firmware of system are updated. Network interface speed is matching with router/gateway speed.

The **netstat** can provide details about open network connections and stack statistics.

The **ethtool** command would be the ideal one when someone wish to see packet drops/loss at hardware level. This command could also be used to identify network card driver/firmware version.

To check network bandwidth and throughout between server and client use **netcat** command.

If there is a need to analyze issues at network packet level, then one could use **tcpdump** command to capture dump data to analyze further.

There are at times in-correct DNS configuration may also lead to a glitch in network performance.

**Restart the process:**

You can go to the init.d directory and restart process u want. In init.d directory we have startup scripts for every process in machine we can go into the directory and simply restart it.

**# cd /etc/init.d**

**#ls**

U will have number of process name and daemons responsible for that and then u can restart the script.

**# /etc/init.d restart**

**Customer can't ssh into their machine, what do you do?**

**1.** try using "ssh -v" so that you can see and paste more debug information.

2. I will check logs in /var/log/auth.log

3. check the contents of **/etc/ssh/sshd\_config** on the target machine - it is possible that your specific user is not permitted to log in remotely. Specific lines to check for:

PermitRootLogin no # should never allow remote root login

AllowUsers someusername # whitelist of users who are allowed to ssh to the machine.

4. I will check file permission on machine I am trying to login.

5. I will check if open ssh server is installed properly on target machine.

**You have plenty of space, but still cannot write on the drive, what could be the issue**

You are out of inodes. It's likely that you have a directory somewhere with

many very small files.

**How will you restrict IP so that the restricted IP’s may not use the FTP Server?**

We can block suspicious IP by integrating tcp\_wrapper. We need to enable the parameter “tcp\_wrapper=YES” in the configuration file at ‘/etc/vsftpd.conf’. And then add the suspicious IP in the ‘host.deny’ file at location ‘/etc/host.deny’.

##### **Block IP Address**

Open ‘/etc/hosts.deny’ file.

# vi /etc/hosts.deny

Add the IP address that you want to block at the bottom of the file.

# hosts.deny This file contains access rules which are used to

# deny connections to network services that either use

# the tcp\_wrappers library or that have been

# started through a tcp\_wrappers-enable

vsftpd:172.16.16.1

###### **You need to search for the string “Amazon” in all the “.txt” files in the current directory. How will you do it?**

**Answer:** We need to run the **find** command to search for the text “Amazon” in the current directory, recursively.

# find -­name “\*.txt” | xargs grep “Amazon”

###### **You want to send a message to all connected users as “Server is going down for maintenance”, what will you do?**

**Answer:** This can be achieved using the wall command. The **wall** command sends a message to all connected users on the sever.

# echo please save your work, immediately. The server is going down for Maintenance at 12:30 Pm

### **As the disk space utilization was so high in the server, the Administrator has removed few files from the server but still the disk utilization is showing as high. What would be the reason? df shows disk is full but du shows it still have memory.**

In Linux even if we remove a file from the mounted file system, that will still be in use by some application and for this application, it remains available. It’s because file descriptor in /proc/ filesystem is held open. Check for files on located under mount points. Frequently if you mount a directory (say a sambafs) onto a filesystem that already had a file or directories under it, you lose the ability to see those files, but they're still consuming space on the underlying disk

So, if there are such open descriptors to files already removed, space occupied by them considered as used. You find this difference by checking them using the "df" and "du" commands. While df is to show the file system usage, du shows we still have space. du works from files while df works at filesystem level, reporting what the kernel says it has available.

You can find all unlinked but held open files with:

**# lsof | grep /var | grep '(deleted)'**

This will list the filename which is open with the pid in which it is running. We can kill those Pids and which will stop these processes and will recover the disk space responsible for this file.

**Server is failed with hosting service. What can be troubleshooting steps?**

1. Check service logs.

2. Check process.

3. Check networking and iptables.

4. Check daemon messages.

5. Check crash file or dump file.

6. Check bugs with software version. So, check problem with certification and authentication.

**NFS (Network File System):**

**NFS** (**Network File System**) is basically developed for sharing of **files** and **folders** between **Linux**/**Unix** systems by **Sun Microsystems** in **1980**. It allows you to mount your local file systems over a network and remote hosts to interact with them as they are mounted locally on the same system.

##### **NFS Services**

The **NFS** server package includes three facilities, included in the **portmap** and **nfs-utils** packages.

1. **portmap** : It maps calls made from other machines to the correct **RPC** service (not required with **NFSv4**).
2. **nfs**: It translates remote **file sharing** requests into requests on the local file system.
3. **rpc.mountd**: This service is responsible for **mounting** and **unmounting** of file systems.

Port 111 (TCP and UDP) and 2049 (TCP and UDP) for the NFS server.

##### **Important Files for NFS Configuration**

1. **/etc/exports**: It’s a main configuration file of **NFS**, all exported **files** and **directories** are defined in this file at the **NFS Server** end.
2. **/etc/fstab**: To mount a **NFS directory** on your system across the **reboots**, we need to make an entry in **/etc/fstab**.
3. **/etc/sysconfig/nfs**: Configuration file of **NFS** to control on which port **rpc** and other services are **listening**.

### **Setup and Configure NFS Mounts on Linux Server**

To setup **NFS** mounts, we’ll be needing at least two **Linux**/**Unix** machines. Here in this tutorial, I’ll be using two servers.

1. **NFS Server**: nfsserver.example.com with IP-**192.168.0.100**
2. **NFS Client**: nfsclient.example.com with IP-**192.168.0.101**

Now we will check connectivity between server and client by using **Ping**.

##### **Installing NFS Server and NFS Client**

We need to install **NFS** packages on our **NFS Server** as well as on **NFS Client** machine. We can install it via “**yum**” (**Red Hat** Linux) and “**apt-get**” (**Debian** and **Ubuntu**) package installers.

[root@nfsserver ~]# yum install nfs-utils nfs-utils-lib

[root@nfsserver ~]# yum install portmap (not required with NFSv4)

Now start the **services** on both machines.

[root@nfsserver ~]# /etc/init.d/portmap start

[root@nfsserver ~]# /etc/init.d/nfs start

[root@nfsserver ~]# chkconfig --level 35 portmap on

[root@nfsserver ~]# chkconfig --level 35 nfs on

After installing packages and starting services on both the machines, we need to configure both the machines for file sharing.

### **Setting Up the NFS Server**

First, we will be configuring the **NFS** server.

##### **Configure Export directory**

For sharing a directory with **NFS**, we need to make an entry in “**/etc/exports**” configuration file. Here I’ll be creating a new directory named “**nfsshare**” in “**/**” partition to share with **client server**, you can also share an already existing directory with NFS.

[root@nfsserver ~]# mkdir /nfsshare

Now we need to make an entry in “**/etc/exports**” and **restart** the services to make our directory shareable in the network.

[root@nfsserver ~]# vi /etc/exports

/nfsshare 192.168.0.101(rw,sync,no\_root\_squash)

In the above example, there is a directory in **/** partition named “**nfsshare**” is being shared with client IP “**192.168.0.101**” with **read** and **write** (**rw**) privilege, you can also use **hostname** of the client in the place of **IP** in above example.

##### **NFS Options**

Some other options we can use in “**/etc/exports**” file for file sharing is as follows.

1. **ro**: we can provide **read only access** to the shared files i.e **client** will only be able to **read**.
2. **rw**: This option allows the **client server** to both **read** and **write** access within the shared directory.
3. **sync**: Sync confirms requests to the shared directory only once the **changes** have been committed.
4. **no\_subtree\_check**: This option prevents the **subtree** checking. When a shared directory is the subdirectory of a larger file system, **nfs** performs scans of every directory above it, to verify its permissions and details. Disabling the **subtree** check may increase the reliability of **NFS** but reduce **security**.
5. **no\_root\_squash**: This phrase allows **root** to **connect** to the designated directory.

### **Setting Up the NFS Client**

After configuring the **NFS** server, we need to **mount** that shared directory or partition in the **client** server.

##### **Mount Shared Directories on NFS Client**

First we need to find out that shares available on the remote server or NFS Server.

[root@nfsclient ~]# showmount -e 192.168.0.100

Export list for 192.168.0.100:

/nfsshare 192.168.0.101

Above command shows that a directory named “**nfsshare**” is available at “**192.168.0.100**” to share with your server.

##### **Mount Shared NFS Directory**

To **mount** that shared **NFS** directory we can use following mount command.

[root@nfsclient ~]# mount -t nfs 192.168.0.100:/nfsshare /mnt/nfsshare

The above command will mount that shared directory in “**/mnt/nfsshare**” on the client server. You can verify it following command.

[root@nfsclient ~]# mount | grep nfs

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw)

nfsd on /proc/fs/nfsd type nfsd (rw)

192.168.0.100:/nfsshare on /mnt type nfs (rw,addr=192.168.0.100)

The above mount command mounted the **nfs shared directory** on to **nfs client** temporarily, to mount an NFS directory **permanently** on your system across the **reboots**, we need to make an entry in “**/etc/fstab**“.

[root@nfsclient ~]# vi /etc/fstab

Add the following new line as shown below.

192.168.0.100:/nfsshare /mnt nfs defaults 0 0

### **Test the Working of NFS Setup**

We can test our **NFS server setup** by creating a **test file** on the server end and check its availability at **nfs client**side or vice-versa.

##### **At the nfsserver end**

I have created a new text file named “**nfstest.txt**’ in that shared directory.

[root@nfsserver ~]# cat > /nfsshare/nfstest.txt

This is a test file to test the working of NFS server setup.

##### **At the nfsclient end**

Go to that shared directory in **client server** and you’ll find that shared file without any manual refresh or service restart.

[root@nfsclient]# ll /mnt/nfsshare

total 4

-rw-r--r-- 1 root root 61 Sep 21 21:44 nfstest.txt

root@nfsclient ~]# cat /mnt/nfsshare/nfstest.txt

This is a test file to test the working of NFS server setup.

### **Removing the NFS Mount**

If you want to **unmount** that shared directory from your server after you are done with the file sharing, you can simply **unmount** that particular directory with “**umount**” command. See this example below.

root@nfsclient ~]# umount /mnt/nfsshare

You can see that the mounts were removed by then looking at the filesystem again.

Following command is used to check space in directory shared by nfs.

[root@nfsclient ~]# df -h -F nfs

You’ll see that those shared directories are not available any more.

##### **Important commands for NFS**

Some more important commands for **NFS**.

1. **showmount -e** : Shows the available **shares** on your local machine
2. **showmount -e** <server-ip or hostname>: Lists the available **shares** at the **remote** server
3. **showmount -d** : Lists all the **sub directories**
4. **showmount -a : Used by server and shows list of all client right now accessing sever.**
5. **nfsstat – s: Used by server to check current number and load on server by client.**
6. **nfsstat – c: Used by client r to check current number and load on client.**
7. **exportfs -v** : Displays a list of shares **files** and **options** on a server
8. **exportfs -a** : Exports all shares listed in **/etc/exports**, or given name
9. **exportfs -u** : Unexports all shares listed in **/etc/exports**, or given name
10. **exportfs -r** : Refresh the server’s list after modifying **/etc/exports**
11. **nfsstat -o all: To check current version of nfs**

**NIS (Network Information System):**

Network Information System (NIS) is designed to centralize administration of UNIX®-like systems such as Solaris™, HP-UX, AIX®, Linux, NetBSD, OpenBSD, and FreeBSD

NIS is a Remote Procedure Call (RPC)-based client/server system that allows a group of machines within an NIS domain to share a common set of configuration files. This permits a system administrator to set up NIS client systems to add, remove, or modify configuration data from a single location.

**Table 29.1. NIS Terminology**

| **Term** | **Description** |
| --- | --- |
| NIS domain name | NIS servers and clients share an NIS domain name. Typically, this name does not have anything to do with DNS. |
| [rpcbind(8)](https://www.freebsd.org/cgi/man.cgi?query=rpcbind&sektion=8&manpath=freebsd-release-ports) | This service enables RPC and must be running to run an NIS server or act as an NIS client. |
| [ypbind(8)](https://www.freebsd.org/cgi/man.cgi?query=ypbind&sektion=8&manpath=freebsd-release-ports) | This service binds an NIS client to its NIS server. It will take the NIS domain name and use RPC to connect to the server. It is the core of client/server communication in an NIS environment. If this service is not running on a client machine, it will not be able to access the NIS server. |
| [ypserv(8)](https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&manpath=freebsd-release-ports) | This is the process for the NIS server. If this service stops running, the server will no longer be able to respond to NIS requests so hopefully, there is a slave server to take over. Some non-FreeBSD clients will not try to reconnect using a slave server and they bind process may need to be restarted on these clients. |
|  |  |

There are three types of hosts in an NIS environment:

* **NIS master server**

This server acts as a central repository for host configuration information and maintains the authoritative copy of the files used by all the NISClients. The passwd, group, and other various files used by NIS clients are stored on the master server.

**NIS slave servers**

NIS slave servers maintain copies of the NIS master's data files to provide redundancy.

**NIS clients**

NIS clients authenticate against the NIS server during log on.

This section describes a sample NIS environment which consists of 15 FreeBSD machines with no centralized point of administration. Each machine has its own /etc/passwd and /etc/master.passwd. These files are kept in sync with each other only through manual intervention. Currently, when a user is added to the lab, the process must be repeated on all 15 machines.

The configuration of the lab will be as follows:

| **Machine name** | **IP address** | **Machine role** |
| --- | --- | --- |
| ellington | 10.0.0.2 | NIS master |
| coltrane | 10.0.0.3 | NIS slave |
| basie | 10.0.0.4 | Faculty workstation |
| bird | 10.0.0.5 | Client machine |
| cli[1-11] | 10.0.0.[6-17] | Other client machines |

### **A.** **Configuring the NIS Master Server**

The copies of all NIS files are stored on the master server. The databases used to store the information are called **NIS maps**. In FreeBSD, these maps are stored in /var/yp/[domainname] where [domainname] is the name of the NIS domain.

it only needs to be enabled by adding the following lines to /etc/rc.conf:

nisdomainname="test-domain" 1

nis\_server\_enable="YES" 2

nis\_yppasswdd\_enable="YES" 3

|  |  |
| --- | --- |
| [1](https://www.freebsd.org/doc/handbook/network-nis.html#network-nis-co-domainname) | This line sets the NIS domain name to test-domain. |
| [2](https://www.freebsd.org/doc/handbook/network-nis.html#network-nis-co-server) | This automates the startup of the NIS server processes when the system boots. |
| [3](https://www.freebsd.org/doc/handbook/network-nis.html#network-nis-co-yppasswdd) | This enables the [rpc.yppasswdd(8)](https://www.freebsd.org/cgi/man.cgi?query=rpc.yppasswdd&sektion=8&manpath=freebsd-release-ports) daemon so that users can change their NIS password from a client machine. |

#### Initializing the NIS Maps

NIS maps (Database for NIS file) are generated from the configuration files in /etc on the NIS master, with one exception: /etc/master.passwd. This is to prevent the propagation of passwords to all the servers in the NIS domain. Therefore, before the NIS maps are initialized, configure the primary password files:

# **cp /etc/master.passwd /var/yp/master.passwd**

# **cd /var/yp**

# **vi master.passwd**

**Write following script to generate NIS MAPs**

ellington# **ypinit -m test-domain**

#### Adding New Users

Every time a new user is created, the user account must be added to the master NIS server and the NIS maps rebuilt. Until this occurs, the new user will not be able to login anywhere except on the NIS master. For example, to add the new user jsmith to the test-domain domain, run these commands on the master server:

# **pw useradd jsmith**

# **cd /var/yp**

# **make test-domain**

### **B. Setting up a NIS Slave Server**

To set up an NIS slave server, log on to the slave server and edit /etc/rc.conf as for the master server. Do not generate any NIS maps, as these already exist on the master server. When running ypinit on the slave server, use -s (for slave) instead of -m (for master). This option requires the name of the NISmaster in addition to the domain name, as seen in this example:

coltrane# **ypinit -s ellington test-domain**

### **C. Setting Up an NIS Client**

An NIS client binds to an NIS server using [ypbind(8)](https://www.freebsd.org/cgi/man.cgi?query=ypbind&sektion=8&manpath=freebsd-release-ports). This daemon broadcasts RPC requests on the local network. These requests specify the domain name configured on the client. If an NIS server in the same domain receives one of the broadcasts, it will respond to ypbind, which will record the server's address. If there are several servers available, the client will use the address of the first server to respond and will direct all its NIS requests to that server.

To configure a FreeBSD machine to be an NIS client:

1. Edit /etc/rc.conf and add the following lines in order to set the NIS domain name and start [ypbind(8)](https://www.freebsd.org/cgi/man.cgi?query=ypbind&sektion=8&manpath=freebsd-release-ports) during network startup:
2. nisdomainname="test-domain"

nis\_client\_enable="YES"

# **How to Use Kickstart to Install CentOS 7**

You can install CentOS 7 automatically with a Kickstart file. A Kickstart file has the answer to all the questions that the CentOS 7 installer asks when you manually install it. You can create a Kickstart configuration file with **Kickstart Configurator** and use it to install CentOS 7 automatically.

## Installing **Kickstart Configurator** on CentOS 7

**Kickstart Configurator** is a graphical application for creating a Kickstart configuration file. It is not installed by default on CentOS 7. You can easily install **Kickstart Configurator** from the App Store.

First search for **App Store** in the GNOME 3 Application Menu. You should see the following icon as marked in the screenshot below.

### Now Using **Kickstart Configurator** to Generate a Kickstart File

You can generate kickstart file and configure some paraments like

**Basic configuration**: Default language, Time zone, Keyboard, Root password

**Installation method:** Installation method, Installation source (CD-ROM, FTP, HTTP)

**Boot loader option**: Install type, GRUB option

**Partition Information**: U can add new partition and give appropriate resources

**Network configuration**: You can add network device

**Authentication:** We can specify how user will authenticate once installation completes

**Firewall configuration:**

**Display configuration:**

Now save the Kickstart configuration file to a USB drive as **ks.cfg**.

Now boot CentOS DVD on any machine where you want to install CentOS 7. Also insert the USB device where you have **ks.cfg** file stored.

Now press **<Esc>** button. You should see the following window.

https://linuxhint.com/wp-content/uploads/2018/04/p33.png

Now type in the following command and then press **<Enter>**:

linux ks=hd:sdb1:/ks.cfg

NOTE: Here **/dev/sda**is the hard drive where CentOS 7 should be installed and **/dev/sdb1**is the USB drive where you saved **ks.cfg** file.

# **What is Network Bonding? Types of Network Bonding**

Network bonding is a process of combing or joining two or more network interfaces together into a single logical interface. Network bonding offers performance improvements, load balancing and redundancy by increasing the network throughput and bandwidth. If one interface is down or unplugged the other one will work.

**Types of Network Bonding**

**1) mode=0 (balance-rr)**

This mode is based on Round-robin policy and it is the default mode. This mode offers fault tolerance and load balancing features. It transmits the packets in Round robin fashion that is from the first available slave through the last.

**2) mode-1 (active-backup)**

This mode is based on Active-backup policy. Only one slave is active in this band, and another one will act only when the other fails. The MAC address of this bond is available only on the network adapter part to avoid confusing the switch. This mode also provides fault tolerance.

**3) mode=2 (balance-xor)**

**4) mode=3 (broadcast)**

**5) mode=4 (802.3ad)**

**6) mode=5 (balance-tlb)**

**7) mode=6 (balance-alb)**

**Configure Network Bonding on CentOS**

1) Create the **bond file** (ifcfg-bond0) and specify the IP address, netmask & gateway.

*# vi /etc/sysconfig/network-scripts/ifcfg-bond0*

*DEVICE=bond0*

*IPADDR=192.x.x.x*

*NETMASK=255.255.255.0*

*GATEWAY=192.x.x.1*

*TYPE=Bond*

*ONBOOT=yes*

*NM\_CONTROLLED=no*

*BOOTPROTO=static*

2) Edit the files of **eth0 & eth1** and make sure you enter the master and slave entry.

*# vi /etc/sysconfig/network-scripts/ifcfg-eth0*

*DEVICE=eth0*

*HWADDR=08:00:27:5C:A8:8F*

*TYPE=Ethernet*

*ONBOOT=yes*

*NM\_CONTROLLED=no*

*MASTER=bond0*

*SLAVE=yes*

*# vi /etc/sysconfig/network-scripts/ifcfg-eth1*

*DEVICE=eth1*

*TYPE=Ethernet*

*ONBOOT=yes*

*NM\_CONTROLLED=no*

*MASTER=bond0*

*SLAVE=yes*

3) Create the Bond file(bonding.conf)

*# vi /etc/modprobe.d/bonding.conf*

*alias bond0 bonding*

*options bond0 mode=1 miimon=100*

4) Now Restart the network Service

*# service network restart*

5) To check the bond interface, use command:

*#  ifconfig bond0*

6) To verify the status of bond interface, use command:

*#  cat /proc/net/bonding/bond0*

**What is SELinux?**

It is an access control implementation and security feature for the Linux kernel. It is designed to protect the server against misconfigurations and/or compromised daemons.

It improves security on accessing services/files which improve security. **SELinux** is short form of Security Enhanced Linux.

## **SETTING OF SELINUX**

**SELinux is set in three modes.**

* **Enforcing** – SELinux security policy is enforced. IF this is set SELinux is enabled and will try to enforce the SELinux policies strictly
* **Permissive** – SELinux prints warnings instead of enforcing. This setting will just give warning when any SELinux policy setting is breached
* **Disabled** – No SELinux policy is loaded. This will totally disable SELinux policies.

**And SELinux is set in two levels**

* **Targeted** – Targeted processes are protected,
* **Mls**– Multi Level Security protection.

## GET SELINUX STATUS

**Example1**:Is SELinux enabled or not on your box? use below command to get the status.

**#getenforce**The output will be either “Enabled” or “Disabled”

**Example2**: To see SELinux status in simplified way you can use sestatus  
**#sestatus**

**Sample output:**  
SElinux status : enabled  
SELinux mount : /selinux  
Current mode : enforcing  
Mode from config file : enforcing  
Policy version : 21  
Policy from config file : targeted

From the above output we can see that SElinux is enabled and it is in enforced mode.  
and to see detailed status you can use **-b** option, this will give on which services SElinux is enabled and which services are disabled.

**Example3:**To get elobrated info on difference status of SELinux on different services use -b option along sestatus

**#sestatus -b**

**Sample output:**

[root@centos1 ~]# sestatus -b  
SELinux status: enabled  
SELinuxfs mount: /selinux  
Current mode: permissive  
Mode from config file: enforcing  
Policy version: 24  
Policy from config file: targeted

Policy booleans:  
abrt\_anon\_write off  
allow\_console\_login on  
allow\_corosync\_rw\_tmpfs off  
allow\_cvs\_read\_shadow off  
allow\_daemons\_dump\_core on  
allow\_daemons\_use\_tty on

## **DISABLING SELINUX**

**Example4**:How to disable SElinux

We can do it in two ways  
**1)**Permanent**way : edit /etc/selinux/config**

change the **status** of SELINUX from**enforcing** to **disabled**

SELINUX=enforcing

to

SELINUX=disabled

Save the file and exit.

**2)Temporary way :**Execute below command

**echo 0 > /selinux/enforce**

or

setenforce 0

## **ENABLING SELINUX**

**Example5:** How about enabling SELinux

**1)**Permanent**way: edit /etc/selinux/config**

change the **status** of SELINUX from**disabled to enforcing**

SELINUX=disabled

to

SELINUX=**enforcing**

Save the file and exit.

**2)Temporary way :**Execute below command

**echo 1 > /selinux/enforce**

**Virtual Hosting in Apache sever (version2.4):**

Virtual hosting is method of hosting multiple domain name on single server. This allows single server to share resources such as memory, process cycle.

1. We need to preinstall **LAMP server**.

2. Create 2 users ex. userA and userB and create directory for both users.

3. We need to create HTML file and write some markup text.

4. write **markup code** into the HTML file for both user using any editor (vi).

5. Go to apache directory /etc/apache. Then cd site-available. Here u will find full configuration file. 000-degfault-config.

6. copy that file to both user config. Ex. Cp 000-degfault-config userA

7. Now we will have to change **document root** in copied user file to user file root instead of www.

8. Now enable both site by commads #a2ensite userA.config.

9. Now reload apache and restart it. Commads #Systemctl reload apache and #service apache restart.

To stop apache webserver:

#/etc/init.d/apache2 stop

**Automount:**

Autofs also referred as Automount is a nice feature in linux used to mount the filesystems automatically on user’s demand. There are two ways available in linux by which we can mount the file system i.e. /etc/fstab and another one is Autofs. /etc/fstab is used to mount the filesystems automatically at when system bootsup and Autofs is also doing the same thing.

**Difference Between /etc/fstab and Autofs (AutoMount)**

As we know that **/etc/fstab** is used for permanent mounting of file systems but it will be useful only if you have less mount points connected to your /etc/fstab file But Autofs mounts the file systems on user’s demand.

Bydefault the mount point’s configured in **Autofs** is in unmounted state till the user access the mount point, once user try to access the mount point it will mount automatically and if user dont use the mount point for some time then it will automatically go to unmount state.

I am familiar with Linux file system and boot process. We currently have REHL 7.4

I installed and upgrade various Linux machines and diagnosis then Monitoring memory

Status

problems associated with DNS, DHCP and VPN.

I have worked on user management and giving permissions to specific users.

Then I have worked on configuring Iptables to restrict access in Linux servers.

I have configured and troubleshooted nfs related issues

<https://www.digitalocean.com/community/tutorials/how-to-use-traceroute-and-mtr-to-diagnose-network-issues>