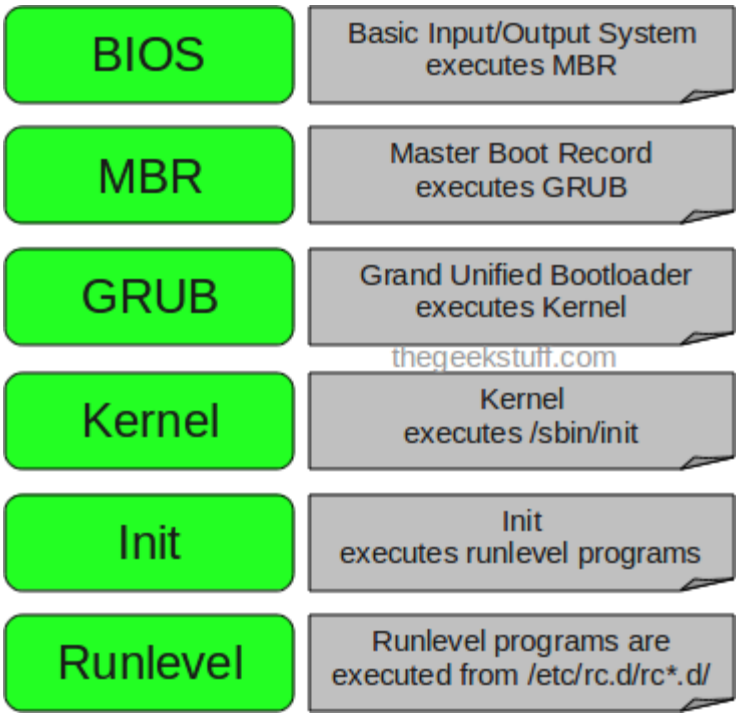
The following are the 6 high level stages of a typical Linux boot process.



1. BIOS

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it selects the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is **/boot/grub/grub.conf** (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda

default=0

timeout=5

splashimage=(hd0,0)/boot/grub/splash.xpm.gz

hiddenmenu

title CentOS (2.6.18-194.el5PAE)

          root (hd0,0)

          kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/

          initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

- Mounts the root file system as specified in the "root=" in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a 'ps -ef | grep init' and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
  - 0 – halt
  - 1 – Single user mode
  - 2 – Multiuser, without NFS
  - 3 – Full multiuser mode
  - 4 – unused
  - 5 – X11
  - 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
- Execute 'grep initdefault /etc/inittab' on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say "starting sendmail .... OK". Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
  - Run level 0 – /etc/rc.d/rc0.d/
  - Run level 1 – /etc/rc.d/rc1.d/
  - Run level 2 – /etc/rc.d/rc2.d/
  - Run level 3 – /etc/rc.d/rc3.d/
  - Run level 4 – /etc/rc.d/rc4.d/
  - Run level 5 – /etc/rc.d/rc5.d/
  - Run level 6 – /etc/rc.d/rc6.d/
- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
- For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

There you have it. That is what happens during the Linux boot process.

*******How to reset syour root passwd*******

1:-jst restart your system
2:-press up down arrow
3:-press e
4:-go to end of linux16 line
5:-jst give space and type rd.break
and remove that both words (console=tty console=ttys0)
6:-ctrl+x

 # mount -oremount,rw /sysroot
#chroot /sysroot
#passwd root
type new passwd:-redhat1234
retype same passwd:-redhat1234
#touch /.autorelabel
#exit
#exit
NFS/iSCSI/FC setup and troubleshooting)

What is the Network File System (NFS)?

The Network File System (NFS) is a way of mounting Linux discs/directories over a network. An NFS server can export one or more directories that can then be mounted on a remote Linux machine. Note, that if you need to mount a Linux filesystem on a Windows machine, you need to use Samba/CIFS instead.

When to use NFS and when to use Samba

Here are some examples of when to use Samba and when to use NFS:

Server O/S	Client O/S	Use Samba or NFS?
Linux	Linux	**NFS**
Windows	Linux	**Samba**
Linux	Windows	**Samba**
Windows	Windows	**Samba**

If you have a Linux server and a Linux client, those two should share data via NFS rather than Samba/CIFS.

What is Samba/CIFS?

Samba is a file sharing utility using the CIFS protocol, specifically aimed at allowing Windows and Linux servers to access the same file systems. The normal setup

is to have the data drives mounted on a Linux server and shared out using Samba and NFS. The Windows clients can then mount those drives using a Samba client and the Linux clients can mount them with NFS.

When to use Samba/CIFS

Samba should only be used to connect a server and a client if they running different Operating Systems (-i.e. Windows and Linux). If they are both running Linux - you should be using NFS instead.

What is a Firewall?

A firewall is simply a utility that manages network traffic access to and from a machine, whether a server or a desktop. Windows users will be familiar with the proprietary firewalls available, such as Symantec, Norton or Kaspersky which are vital for stopping unwanted processes getting access to their machine. The standard Linux firewall package is called iptables (-aka netfilter).

[A firewall] allows you to define rules which specify which traffic to allow or disallow.

There are three types of traffic that you can add a rule for-

- Inbound: traffic coming from the network and destined for your computer. This is the category to use to control external attacks and is therefore the most common category where controls are applied
- Outbound: traffic coming from your computer and destined for somewhere on the network. This is the category to use to control the resources that users on this computer can access (-normally used for things like stopping users of a PC connecting to various external services - e.g. a network printer or server containing sensitive data)
- Forward: traffic coming from outside your computer and destined for somewhere on the network, but passing through your PC. This is only really relevant if your machine is set up as a network router.

What is iscsi in Linux?

ISCSI is an Internet Protocol (IP)-based storage networking standard for linking data storage facilities. By carrying SCSI commands over IP networks, iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. iSCSI can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet, and can enable location-independent data storage and retrieval.

iSCSI allows clients (called Initiators) to send SCSI commands (CDBs) to SCSI storage devices (LinuxIOs) on remote servers. It is a popular SAN protocol, allowing organizations to consolidate storage into data center storage arrays while providing hosts (such as database and web servers) with the illusion of locally-attached disks. Unlike traditional Fibre Channel, which requires special-purpose cabling, iSCSI can be run over long distances using existing network infrastructure.

What is Swap Space?

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

Recommended System Swap Space

Amount of RAM in the System	Recommended Amount of Swap Space
4GB of RAM or less	a minimum of 2GB of swap space
4GB to 16GB of RAM	a minimum of 4GB of swap space
16GB to 64GB of RAM	a minimum of 8GB of swap space
64GB to 256GB of RAM	a minimum of 16GB of swap space
256GB to 512GB of RAM	a minimum of 32GB of swap space

How to check the disk space in linux?

df command - Shows the amount of disk space used and available on Linux file systems.

du command - Display the amount of disk space used by the specified files and for each subdirectory.

How to check the CPU utilization in Linux?

The "sar" command

sar -u 2 5t

This command will display CPU utilization 2 seconds apart, 5 times

Using TOP command

The top command provides dynamic view of CPU utilization. It displays system information as well as list of tasks currently managed by kernel. It also displays uptime, average load, physical and swap memory utilization.

Commands to check the Linux Version, Release name & Kernel version.

uname -a (Print all Information)

uname -r (Print the kernel name)

cat /proc/version

cat /etc/issue

cat /etc/redhat-release

tail /etc/redhat-release

What is Uptime in Linux?

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

How To Find The Process ID Of A Program And Kill It?

It often happens that you need to kill an unresponsive program.

ps aux | grep -i “name of your desired program”.

```
kill [signal or option] PID(s)
```

For a kill command a Signal Name could be:

Signal Name	Signal Value	Behaviour
SIGHUP	1	Hangup

```
SIGKILL                         9                               Kill Signal
SIGTERM                         15                              Terminate
```

Clearly from the behaviour above SIGTERM is the default and safest way to kill a process. SIGHUP is less secure way of killing a process as SIGTERM. SIGKILL is the most unsafe way among the above three, to kill a process which terminates a process without saving.

Customizable way of finding the PID of a process.

```
pidof mysqld
```

Another way to perform the same function is to execute ‘pgrep‘ command.

```
pgrep mysq
```

Sample Output

```
3139
```

To kill the above process PID, use the kill command as shown.

```
kill -9 3139
```

What is LVM and how to reduce and expand an LVM?

In Linux, Logical Volume Manager (LVM) is a device mapper target that provides logical volume management for the Linux kernel.

With LVM, a hard drive or set of hard drives is allocated to one or more *physical volumes*. LVM physical volumes can be placed on other block devices which might span two or more disks.

Steps to create LVM

- Creating physical volumes from the hard drives.
- Creating volume groups from the physical volumes.
- Creating logical volumes from the volume groups and assign the logical volumes mount points.

A port is an application-specific or process-specific software construct serving as a communications endpoint and it is identified by its number such as TCP port number 80 . It is used by TCP and UDP of the Internet Protocol Suite. A port number is a 16-bit unsigned integer, thus ranging from 0 to 65535.

The port numbers are divided into three ranges:

1. Well Known Ports: those from 0 through 1023.
2. Registered Ports: those from 1024 through 49151
3. Dynamic and/or Private Ports: those from 49152 through 65535

Common Well Known Port Numbers

- 21: FTP Server
- 22: SSH Server (remote login)
- 25: SMTP (mail server)
- 53: Domain Name System (Bind 9 server)
- 80: World Wide Web (HTTPD server)
- 110: POP3 mail server
- 143: IMAP mail server
- 443: HTTP over Transport Layer Security/Secure Sockets Layer (HTTPDS server)
- 445: microsoft-ds, Server Message Block over TCP

What is DNS?

Short for Domain Name System (or Service or Server), an Internet service that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name www.example.com might translate to 198.105.232.4.

What is IP tables?

Iptables is a Linux command line firewall that allows system administrators to manage incoming and outgoing traffic via a set of configurable table rules.

Iptables uses a set of tables which have chains that contain set of built-in or user defined rules. Thanks to them a system administrator can properly filter the network traffic of his system.

Per iptables manual, there are currently 3 types of tables:

FILTER – this is the default table, which contains the built in chains for:

INPUT – packages destined for local sockets

FORWARD – packets routed through the system

OUTPUT – packets generated locally

NAT – a table that is consulted when a packet tries to create a new connection. It has the following built-in:

PREROUTING – used for altering a packet as soon as it's received

OUTPUT – used for altering locally generated packets

POSTROUTING – used for altering packets as they are about to go out

Daemon	Description
nfsd	The NFS daemon which services requests from NFS clients.
mountd	The NFS mount daemon which carries out requests received from nfsd.
rpcbind	This daemon allows NFS clients to discover which port the NFS server is using.

A daemon (also known as background processes) is a Linux or UNIX program that runs in the background. Almost all daemons have names that end with the letter "d". For example, httpd the daemon that handles the Apache server, or, sshd which handles SSH remote access connections. Linux often start daemons at boot time. Shell scripts stored in /etc/init.d directory are used to start and stop daemons.

They are the processes which run in the background and are not interactive. They have no controlling terminal.

How do I start / stop / restart daemons for the shell prompt?

```
service daemon-name-here start
```

```
service daemon-name-here stop
service daemon-name-here restart
```

How do I see list of all running daemons?

```
service --status-all
```

Normally, we can easily check the state of a network interface card like whether the cable plugged in to the slot or the network card is up or down in Graphical mode. What if you have only command line mode? Ofcourse, you can turn around the system and check for the cable is properly plugged in, or you can do the same easily from your Terminal. Here is how to do that. This method is almost same for Debian and RPM based systems.

How To: Linux Show List of Network Cards

I have two Ethernet cards on my laptop. One, eth0 is wired, and another, wlan0, is wireless.

Let us check the state of the eth0.

```
cat /sys/class/net/eth0/carrier
```

Sample output:

```
0
```

Or, use the following command to check the status.

```
cat /sys/class/net/eth0/operstate
```

Sample output:

```
down
```

What If I don't know the correct name of the installed interfaces?

One of the issue is, you may not know the names of the interfaces. In that case, run the following command. It will show all the installed interfaces, they generally follow the rule that wired interfaces start with the letter **"e"** and wireless start with a letter **"w"**.

```
ls /sys/class/net/
```

How do I display all available network interfaces names under Linux operating systems using bash shell prompt?

- **ip command** – It is used to show or manipulate routing devices, policy routing and tunnels.
- **netstat command** – It is used to display network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- **ifconfig command** – It is used to display or configure a network interface.

List Network Interfaces Using ip Command

```
$ ip link show
```

Sample outputs:

```
$ 1: lo:  mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0:  mtu 1500 qdisc mq state UP qlen 1000
    link/ether b8:ac:6f:65:31:e5 brd ff:ff:ff:ff:ff:ff
3: wlan0:  mtu 1500 qdisc mq state DOWN qlen 1000
    link/ether 00:21:6a:ca:9b:10 brd ff:ff:ff:ff:ff:ff
4: vboxnet0:  mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
5: pan0:  mtu 1500 qdisc noop state DOWN
    link/ether c2:10:fa:55:8e:32 brd ff:ff:ff:ff:ff:ff
6: vmnet1:  mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
```

```
    link/ether 00:50:56:c0:00:01 brd ff:ff:ff:ff:ff:ff

7: vmnet8:  mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000

    link/ether 00:50:56:c0:00:08 brd ff:ff:ff:ff:ff:ff

11: ppp0:  mtu 1496 qdisc pfifo_fast state UNKNOWN qlen 3

    link/ppp
```

Where,

1. lo – Loopback interface.
2. eth0 – First Ethernet network interface.
3. wlan0 – First Wireless network interface.
4. ppp0 – First Point to Point Protocol network interface which can be used by dial up modem, PPTP vpn connection, or 3G wireless USB modem./li>
5. vboxnet0, vmnet1, vmnet8 – Virtual machine interface working in bridge mode or NAT mode.

Show a Table Of All Network Interfaces Using netstat Command

```
$ netstat -i
```

```
Kernel Interface table

Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-
DRP TX-OVR Flg

eth0       1500 0   2697347      0      0 0       2630262      0
0      0 BMRU

lo        16436 0      2840      0      0 0          2840      0
0      0 LRU

ppp0       1496 0    102800      0      0 0         63437      0
0      0 MOPRU
```

```
vmnet1     1500 0          0      0      0 0             49      0
0       0 BMRU

vmnet8     1500 0          0      0      0 0             49      0
0       0 BMRU
```

ifconfig Command Example

```
$ /sbin/ifconfig -a
```

Sample outputs:

```
eth0      Link encap:Ethernet  HWaddr b8:ac:6f:65:31:e5

          inet addr:192.168.2.100  Bcast:192.168.2.255
Mask:255.255.255.0

          inet6 addr: fe80::baac:6fff:fe65:31e5/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          RX packets:2697529 errors:0 dropped:0 overruns:0
frame:0

          TX packets:2630541 errors:0 dropped:0 overruns:0
carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:2159382827 (2.0 GiB)  TX bytes:1389552776
(1.2 GiB)

          Interrupt:17



lo        Link encap:Local Loopback
```

```
          inet addr:127.0.0.1  Mask:255.0.0.0

          inet6 addr: ::1/128 Scope:Host

          UP LOOPBACK RUNNING  MTU:16436  Metric:1

          RX packets:2849 errors:0 dropped:0 overruns:0 frame:0

          TX packets:2849 errors:0 dropped:0 overruns:0
carrier:0

          collisions:0 txqueuelen:0

          RX bytes:2778290 (2.6 MiB)  TX bytes:2778290 (2.6 MiB)



ppp0      Link encap:Point-to-Point Protocol

          inet addr:10.1.3.105  P-t-P:10.0.31.18
Mask:255.255.255.255

          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1496
Metric:1

          RX packets:102800 errors:0 dropped:0 overruns:0
frame:0

          TX packets:63437 errors:0 dropped:0 overruns:0
carrier:0

          collisions:0 txqueuelen:3

          RX bytes:148532544 (141.6 MiB)  TX bytes:4425518 (4.2
MiB)
```

```
vmnet1    Link encap:Ethernet  HWaddr 00:50:56:c0:00:01

          inet addr:192.168.47.1  Bcast:192.168.47.255
Mask:255.255.255.0

          inet6 addr: fe80::250:56ff:fec0:1/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

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

          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)



vmnet8    Link encap:Ethernet  HWaddr 00:50:56:c0:00:08

          inet addr:172.16.232.1  Bcast:172.16.232.255
Mask:255.255.255.0

          inet6 addr: fe80::250:56ff:fec0:8/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

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

          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0

          Collisions: 0 txqueuelen:1000

          RX bytes:0 (0.0 B)  TX bytes: 0 (0.0 B)
```

What is inode number in Linux ?

Inode number also called index number, it consists of following attributes.

- File types (executable, block special etc)
- Permissions (read, write etc)
- UID (Owner)
- GID (Group)
- FileSize
- Time stamps including last access, last modification and last inode number change.
- File deletion time
- Number of links (soft/hard)
- Location of file on harddisk.
- Some other metadata about file.

To check inode number of file use following command. The first field in output is an inode number of the file.

```
ls -il  myfile.txt

1150561 -rw-r--r-- 1 root root 0 Mar 10 01:06 myfile.txt
```

You can also search file with an inode number using find command. For example:

```
find /home/rahul -inum 1150561

/home/rahul/myfile.txt
```

Methods to Check Number of CPU Cores in Linux

```
grep -c ^processor /proc/cpuinfo
```

- Using nproc Command
- nproc – print the number of processing units available to the current process.

Using lscpu Command

lscpu – display information on CPU architecture and gathers CPU architecture information like number of CPUs, threads, cores, sockets, NUMA nodes, information about CPU caches, CPU family, model and prints it in a human-readable format.

Using /proc/interrupts file

This file /proc/interrupts contain information about the interrupts, like how many times processor has been interrupted.

```
cat /proc/interrupts | egrep -i 'cpu'
```

Let's check if the system is using swap space with free command

```
# free
total        used        free      shared  buff/cache   available
Mem:        3742792     2421060      433696      287376      888036
967000
Swap:             0           0           0
```

What is LDAP?

LDAP stands for Lightweight Directory Access Protocol and consists of a set of protocols that allows a client to access, over a network, centrally stored information (such as a directory of login shells, absolute paths to home directories, and other typical system user information, for example) that should be accessible from different places or available to a large number of end users (another example would be a directory of home addresses and phone numbers of all employees in a company).

Keeping such (and more) information centrally means it can be more easily maintained and accessed by everyone who has been granted permissions to use it.

What is the name of main configuration file for LDAP server?

slapd.conf

Which Configuration File Is Required For Ldap Clients?

ldap.conf

Default port for LDAP is 389
What is the default port of NFS server ?
By default NFS uses 2049 TCP port.
Samba runs on TCP ports 139 and 445 and UDP ports 137 and 138. If you want to run a Samba server, the firewall on that box would need to be open to allow those ports in.

Determine the date 60 days in the future.

date –d "+60days"

date -d "-1 day" >> date one day before

To give password to a user using echo command.
Echo "Password" |passwd --stdin Shashank

Password expires on a particular date
chage –E 2014-10-14 shashank(For user Shashank password will expire on 14 oct 2014)

Change the password policy for user anuj to require a new password change every 10 days.
chage –M 10 anuj
Now check the password policy for anuj using chage –l anuj command

Change the password policy for user Shashank that require to change their password on the first attempt.

chage –d 0 shashank

Home directory of anuj

/home/anuj

LSOF command

As we all know Linux/Unix considers everything as a files
(pipes, sockets, directories, devices etc). One of the reason to use lsof command is when a disk cannot be unmounted as it says the files are being used. With the help of this command we can easily identify the files which are in use.

1. List all Open Files with lsof Command

```
# lsof
COMMAND     PID      USER   FD      TYPE     DEVICE  SIZE/OFF       NODE NAME
init          1      root  cwd       DIR      253,0      4096          2 /
init          1      root  rtd       DIR     253,0       4096          2 /
init          1      root  txt       REG      253,0    145180     147164
/sbin/init
init          1      root  mem       REG       253,0   1889704     190149
/lib/libc-2.12.so
```

```
init         1      root    0u      CHR         1,3       0t0       3764
/dev/null
init         1      root    1u      CHR         1,3       0t0       3764
/dev/null
init         1      root    2u      CHR         1,3       0t0       3764
/dev/null
init         1      root    3r      FIFO        0,8       0t0       8449 pipe
init         1      root    4w      FIFO        0,8       0t0       8449 pipe
init         1      root    5r      DIR         0,10         0          1
inotify
init         1      root    6r      DIR         0,10         0          1
inotify
init         1      root    7u     unix 0xc1513880       0t0       8450 socket
```

Sections and it's values are self-explanatory. However, we'll review FD & TYPE columns more precisely.

FD – stands for File descriptor and may seen some of the values as:

1. cwd current working directory
2. rtd root directory
3. txt program text (code and data)
4. mem memory-mapped file

2. List User Specific Opened Files

lsof -u root |awk -F ' ' '{print $3 $6}'

lsof -u root |awk -F ' ' '{print $3":" $6}'

cat /etc/passwd |awk -F ':' '{print $1"--"$6}'

```
# lsof -u tecmint
COMMAND  PID    USER   FD   TYPE     DEVICE SIZE/OFF   NODE NAME
sshd    1838 tecmint  cwd    DIR      253,0     4096      2 /
sshd    1838 tecmint  rtd    DIR      253,0     4096      2 /
sshd    1838 tecmint  txt    REG      253,0   532336 188129 /usr/sbin/sshd
sshd    1838 tecmint  mem    REG      253,0    19784 190237 /lib/libdl-
2.12.so
sshd    1838 tecmint  mem    REG      253,0   122436 190247
/lib/libselinux.so.1
sshd    1838 tecmint  mem    REG      253,0   255968 190256
/lib/libgssapi_krb5.so.2.2
sshd    1838 tecmint  mem    REG      253,0   874580 190255
/lib/libkrb5.so.3.3
```

Find Processes running on Specific Port

```
# lsof -i TCP:22
COMMAND  PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    1471    root    3u  IPv4  12683      0t0  TCP *:ssh (LISTEN)
sshd    1471    root    4u  IPv6  12685      0t0  TCP *:ssh (LISTEN)
```

4. List Only IPv4 & IPv6 Open Files

In below example shows only IPv4 and IPv6 network files open with separate commands.

```
# lsof -i 4
COMMAND    PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
rpcbind   1203     rpc    6u  IPv4  11326      0t0  UDP *:sunrpc
rpcbind   1203     rpc    7u  IPv4  11330      0t0  UDP *:954
rpcbind   1203     rpc    8u  IPv4  11331      0t0  TCP *:sunrpc (LISTEN)
avahi-dae 1241   avahi   13u  IPv4  11579      0t0  UDP *:mdns
avahi-dae 1241   avahi   14u  IPv4  11580      0t0  UDP *:58600
# lsof -i 6
COMMAND    PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
rpcbind   1203     rpc    9u  IPv6  11333      0t0  UDP *:sunrpc
rpcbind   1203     rpc   10u  IPv6  11335      0t0  UDP *:954
rpcbind   1203     rpc   11u  IPv6  11336      0t0  TCP *:sunrpc (LISTEN)
rpc.statd 1277 rpcuser   10u  IPv6  11858      0t0  UDP *:55800
rpc.statd 1277 rpcuser   11u  IPv6  11862      0t0  TCP *:56428 (LISTEN)
cupsd     1346    root    6u  IPv6  12112      0t0  TCP localhost:ipp (LISTEN
```

Find Out who's Looking What Files and Commands?

Below example shows user tecmint is using command like ping and /etc directory .

```
# lsof -i -u tecmint
COMMAND  PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
bash    1839 tecmint  cwd    DIR  253,0    12288   15 /etc
ping    2525 tecmint  cwd    DIR  253,0    12288   15 /etc
```

List all Network Connections

The following command with option '-i' shows the list of all network connections 'LISTENING & ESTABLISHED'.

```
# lsof -i
COMMAND    PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
rpcbind   1203     rpc    6u  IPv4  11326      0t0  UDP *:sunrpc
rpcbind   1203     rpc    7u  IPv4  11330      0t0  UDP *:954
rpcbind   1203     rpc   11u  IPv6  11336      0t0  TCP *:sunrpc (LISTEN)
avahi-dae 1241   avahi   13u  IPv4  11579      0t0  UDP *:mdns
avahi-dae 1241   avahi   14u  IPv4  11580      0t0  UDP *:58600
rpc.statd 1277 rpcuser   11u  IPv6  11862      0t0  TCP *:56428 (LISTEN)
cupsd     1346    root    6u  IPv6  12112      0t0  TCP localhost:ipp
(LISTEN)
cupsd     1346    root    7u  IPv4  12113      0t0  TCP localhost:ipp
(LISTEN)
sshd      1471    root    3u  IPv4  12683      0t0  TCP *:ssh (LISTEN)
master    1551    root   12u  IPv4  12896      0t0  TCP localhost:smtp
(LISTEN)
master    1551    root   13u  IPv6  12898      0t0  TCP localhost:smtp
(LISTEN)
sshd      1834    root    3r  IPv4  15101      0t0  TCP 192.168.0.2:ssh-
>192.168.0.1:conclave-cpp (ESTABLISHED)
httpd     1918    root    5u  IPv6  15991      0t0  TCP *:http (LISTEN)
httpd     1918    root    7u  IPv6  15995      0t0  TCP *:https (LISTEN)
clock-app 2362   narad   21u  IPv4  22591      0t0  TCP 192.168.0.2:45284-
>www.gov.com:http (CLOSE_WAIT)
chrome    2377   narad   61u  IPv4  25862      0t0  TCP 192.168.0.2:33358-
>maa03s04-in-f3.1e100.net:http (ESTABLISHED)
chrome    2377   narad   80u  IPv4  25866      0t0  TCP 192.168.0.2:36405-
>bom03s01-in-f15.1e100
```

9. Search by PID

The below example only shows whose PID is 1 [One].

```
# lsof -p 1
COMMAND PID USER   FD   TYPE     DEVICE SIZE/OFF   NODE NAME
init      1 root  cwd    DIR      253,0     4096      2 /
init      1 root  rtd    DIR      253,0     4096      2 /
init      1 root  txt    REG      253,0   145180 147164 /sbin/init
init      1 root  mem    REG      253,0  1889704 190149 /lib/libc-2.12.so
init      1 root  mem    REG      253,0   142472 189970 /lib/ld-2.12.so
```

10. Kill all Activity of Particular User

Sometimes you may have to kill all the processes for a specific user. Below command will kills all the processes of tecmint user.

```
# kill -9 `lsof -t -u tecmint`
```

What is UMASK in Linux?

When user create a file or directory under Linux or UNIX, she create it with a default set of permissions. In most case the system defaults may be open or relaxed for file sharing purpose. For example, if a text file has 666 permissions, it grants read and write permission to everyone. Similarly a directory with 777 permissions, grants read, write, and execute permission to everyone.

Default umask Value

The user file-creation mode mask (umask) is use to determine the file permission for newly created files. It can be used to control the **default file permission for new files**. It is a four-digit octal number. A umask can be set or expressed using:

- Symbolic values
- Octal values

Procedure To Setup Default umask

You can setup umask in **/etc/bashrc** or **/etc/profile** file for all users. By default most Linux distro set it to 0022 (022) or 0002 (002). Open /etc/profile or ~/.bashrc file, enter:

```
# vi /etc/profile
```

OR

```
$ vi ~/.bashrc
```

Append/modify following line to setup a new umask:

```
umask 022
```

Save and close the file. Changes will take effect after next login. All UNIX users can override the system umask defaults in their /etc/profile file, ~/.profile (Korn / Bourne shell) ~/.cshrc file (C shells), ~/.bash_profile (Bash shell) or ~/.login file (defines the user's environment at login).

Explain Octal umask Mode 022 And 002

As I said earlier, if the default settings are not changed, files are created with the access mode 666 and directories with 777. In this example:

1. The default **umask 002** used for normal user. With this mask default directory permissions are 775 and default file permissions are 664.

2. The default **umask for the root user is 022** result into default directory permissions are 755 and default file permissions are 644.
3. For directories, the **base permissions** are (rwxrwxrwx) 0777 and for files they are 0666 (rw-rw-rw).

In short,

1. A umask of **022** allows only you to write data, but anyone can read data.
2. A umask of **077** is good for a completely private system. No other user can read or write your data if umask is set to 077.
3. A umask of **002** is good when you share data with other users in the same group. Members of your group can create and modify data files; those outside your group can read data file, but cannot modify it. Set your umask to **007** to completely exclude users who are not group members.

But, How Do I Calculate umasks?

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

- **Octal value** : Permission
- **0** : read, write and execute
- **1** : read and write
- **2** : read and execute
- **3** : read only
- **4** : write and execute
- **5** : write only
- **6** : execute only
- **7** : no permissions

Now, you can use above table to calculate file permission. For example, if umask is set to 077, the permission can be calculated as follows:

Bit	Targeted at	File permission
0	Owner	read, write and execute
7	Group	No permissions
7	Others	No permissions

To set the umask 077 type the following command at shell prompt:

```
$ umask 077
$ mkdir dir1
$ touch file
$ ls -ld dir1 file
```

Sample outputs:

```
drwx------ 2 vivek vivek 4096 2011-03-04 02:05 dir1

-rw------- 1 vivek vivek    0 2011-03-04 02:05 file
```

What is NAT?

A. Network Address Translation (NAT) is designed for IP address conservation. It enables private IP networks that use unregistered IP addresses to connect to the Internet. NAT operates on a router, usually connecting two networks together, and translates the private (not globally unique) addresses in the internal network into legal addresses, before packets are forwarded to another network.
As part of this capability, NAT can be configured to advertise only one address for the entire network to the outside world. This provides additional security by effectively hiding the entire internal network behind that address. NAT offers the dual functions of security and address conservation and is typically implemented in remote-access environments.

A public IP address is an IP address that can be accessed over the Internet. Like postal address used to deliver a postal mail to your home, a public IP address is the globally unique IP address assigned to a computing device. Your public IP address can be found at What is my IP Address page. Private IP address, on the other hand, is used to assign computers within your private space without letting them directly expose to the Internet. For example, if you have multiple computers within your home you may want to use private IP addresses to address each computer within your home. In this scenario, your router gets the public IP address, and each of the computers, tablets and smartphones connected to your router (via wired or wifi) gets a private IP address from your router via DHCP protocol.

Experienced in atleast one of the Protocol (SNMP, FTP, CORBA etc).
Experience in troubleshooting of Files system, Disk Management and start-up scripts on Linux servers.

Reboot
Any of the following commands will reboot the system from the command line.

```
# reboot
```

```
# shutdown -r now


# init 6
```

Shutdown

```
# shutdown -h now


# init 0
```

Cd command in linux-

Your present working Directory is “/usr/local/lib/python3.4/dist-packages/ ”, change it to “/home/avi/Desktop/ ”, in one line command, by moving up in the directory till ‘/’ then using absolute path.

```
avi@tecmint:/usr/local/lib/python3.4/dist-packages$ cd
../../../../../home/avi/Desktop/




avi@tecmint:~/Desktop$
```

How to set Host-name in Linux.

```
hostnamectl set-hostname serverX/desktopX.example.com
ip addr >> Check IP address of the system.
Hostname –i >> It will show IP address only.
```

To configure static IP address in RHEL / CentOS / Fedora, you will need to edit:

```
/etc/sysconfig/network
```

```
/etc/sysconfig/network-scripts/ifcfg-eth0
```

Where in the above "ifcfg-eth0" answers to your network interface eth0. If your interface is named "eth1" then the file that you will need to edit is "ifcfg-eth1".

```
NETWORKING=yes

HOSTNAME=node01.tecmint.com

GATEWAY=192.168.0.1

NETWORKING_IPV6=no

IPV6INIT=no
```

Next open:

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

Make sure to open the file corresponding to your network interface. You can find your network interface name with **ifconfig -a command**.

In that file make the following changes:

You will only need to edit the settings for:

DNS1 and DNS2
GATEWAY
HOSTNAME
NETMASK
IPADDR

```
DEVICE="eth0"
```

```
BOOTPROTO="static"

DNS1="8.8.8.8"

DNS2="4.4.4.4"

GATEWAY="192.168.0.1"

HOSTNAME="node01.tecmint.com"

HWADDR="00:19:99:A4:46:AB"

IPADDR="192.68.0.100"

NETMASK="255.255.255.0"

NM_CONTROLLED="yes"

ONBOOT="yes"

TYPE="Ethernet"

UUID="8105c095-799b-4f5a-a445-c6d7c3681f07"
```

Remove oracle from Linux server:

Find the Oracle home directory by looking around /u01/app/Oracle or browsing /etc/oratab and then rm -rf starting there.

How to Install an RPM Package

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

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

RPM command and options
-i : install a package
-v : verbose for a nicer display
-h: print hash marks as the package archive is unpacked.

How to check dependencies of RPM Package before Installing

Let's say you would like to do a dependency check before installing or upgrading a package. For example, use the following command to check the dependencies of BitTorrent-5.2.2-1-Python2.4.noarch.rpm package. It will display the list of dependencies of package.

```
[root@tecmint]# rpm -qpR BitTorrent-5.2.2-1-Python2.4.noarch.rpm
```

RPM command and options
-q : Query a package
-p : List capabilities this package provides.
-R: List capabilities on which this package depends
How to Remove an RPM Package Without Dependencies
The –nodeps (Do not check dependencies) option forcefully remove the rpm package from the system. But keep in mind removing particular package may break other working applications.

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

How to Query a file that belongs which RPM Package
Let's say, you have list of files and you would like to find out which package belongs to these files. For example, the following command with -qf (query file) option will show you a file /usr/bin/htpasswd is own by package **httpd-tools-2.2.15-15.el6.centos.1.i686.**

```
[root@tecmint]# rpm -qf /usr/bin/htpasswd


httpd-tools-2.2.15-15.el6.centos.1.i686
```

How to Query a Information of Installed RPM Package
Let's say you have installed an rpm package and want to know the information about the package. The following -qi (query info) option will print the available information of the installed package.

```
[root@tecmint]# rpm -qi vsftpd

Name        : vsftpd                                Relocations: (not
relocatable)

Version     : 2.2.2                                 Vendor: CentOS

Release     : 11.el6                                Build Date: Fri 22 Jun 2012
01:54:24 PM BDT

Install Date: Mon 17 Sep 2012 07:55:28 PM BDT      Build Host:
c6b8.bsys.dev.centos.org

Group       : System Environment/Daemons           Source RPM: vsftpd-2.2.2-
11.el6.src.rpm

Size        : 351932                                License: GPLv2 with
exceptions

Signature   : RSA/SHA1, Mon 25 Jun 2012 04:07:34 AM BDT, Key ID
0946fca2c105b9de

Packager    : CentOS BuildSystem <http://bugs.centos.org>

URL         : http://vsftpd.beasts.org/

Summary     : Very Secure Ftp Daemon

Description :vsftpd is a Very Secure FTP daemon. It was written completely
from scratch.
```

Linux troubleshooting points-

Knowledge on storage is added advantage.
Hands on with NFS configure/ NFS issues
Hands on with Autofs issues like creating mount points / umounting and mounting / clearing stale mounts / soft mounts, hard mounts
Setting up of Crons for users

User acc and group acc managements
Hands on with VNC issues like setting up of vncservers and trouble shooting
Hands on experience with VMware ESX Server or VMware products
User shell management
Manage VI issues
Well verse with INIT process in Linux hosts
Hands on with FTP setup
L2 level trouble shooting in solaris and Linux env like performance tuning of host.
Should be able to identify stale and dead process on hosts by seeing the logs.
Hands on with AUTOFS configure/ AUTOFS issues
Config of apache knowledge
Log analysis of a Unix host
Hands on experience with VMware ESX Server or VMware products
Configure high available network connectivity, boding, administering and troubleshooting
Server upgrades /patching, configure yum repositories administer software install, configuration, removal etc
Excellent hands on Linux Logical Volume Management
Good exposure on configurations of network servers Like NFS, FTP, Samba, DHCP
Proficient in independently handling Server Installations and configurations.
Good exposure on configurations of network servers Like NFS, FTP, Samba, DHCP

What's CPU Socket: CPU socket or CPU slot is the connector on the motherboard that allows a computer processor to be connected to a motherboard. It's called a physical CPU (central processing unit).

What's CPU Core: Originally, CPUs had a single core but manufacturers added additional cores in to that to increase performance, that's why core came to picture. For example, a dual-core CPU has two central processing units, so it appears to the operating system as two CPUs.
like that, a quad-core CPU has four central processing units and an octa-core CPU has eight central processing units.

What's CPU Thread: Intel Hyper-Threading Technology uses processor resources more efficiently by enabling multiple threads to run on each core (each core can run two threads). It also increases processor throughput and improving overall performance on threaded software.
Refer the following details to understand this in real time.

```
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8


Socket(s):             2
```

The calculation is below: CPUs = Threads per core X cores per socket X sockets.
2 x 8 x 2 = 32

Using nproc Command

```
# nproc
32
```

Using /proc/cpuinfo file

/proc/cpuinfo is a virtual text file that contains information about the CPUs (central processing units) on a computer. We can get a number of CPUs by grepping processor parameter.

```
# grep -c ^processor /proc/cpuinfo
32
```

Using /proc/interrupts file
This file /proc/interrupts contains information about the interrupts, like how many times processor has been interrupted.

```
# cat /proc/interrupts | egrep -i 'cpu'
           CPU0       CPU1       CPU2       CPU3       CPU4
CPU5       CPU6       CPU7       CPU8       CPU9       CPU10
CPU11      CPU12      CPU13
CPU14      CPU15      CPU16      CPU17      CPU18      CPU19
CPU20      CPU21      CPU22      CPU23      CPU24      CPU25
CPU26      CPU27      CPU28
CPU29      CPU30      CPU31
```

Awk command in linux with examples-

`awk '{print "Welcome to awk command tutorial "}'`

`it will print below output-`

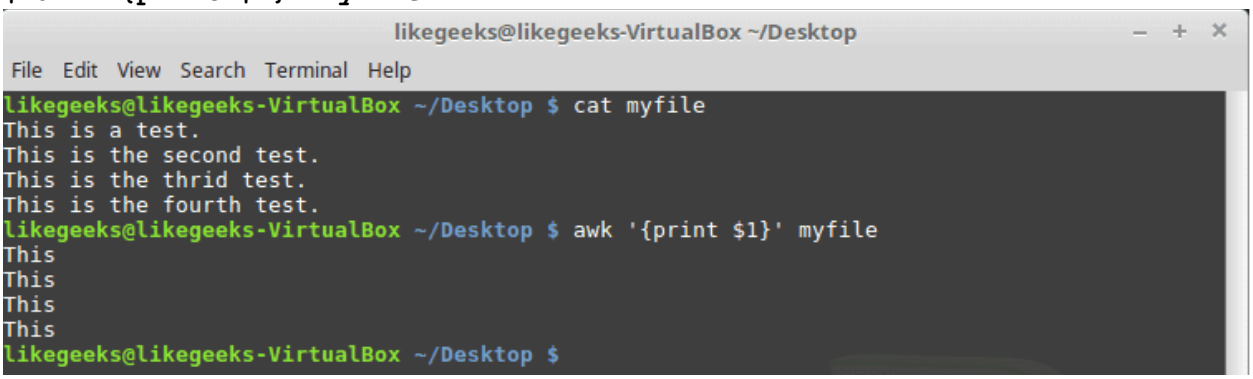
Welcome to awk command tutorial

Using Variables

With awk, you can process text files. Awk assigns some variables for each data field found:
$0 for the whole line.
$1 for the first field.
$2 for the second field.
$n for the nth field

The whitespace character like space or tab is the default separator between fields in awk.
Check this example and see how awk processes it:

$ awk '{print $1}' myfile



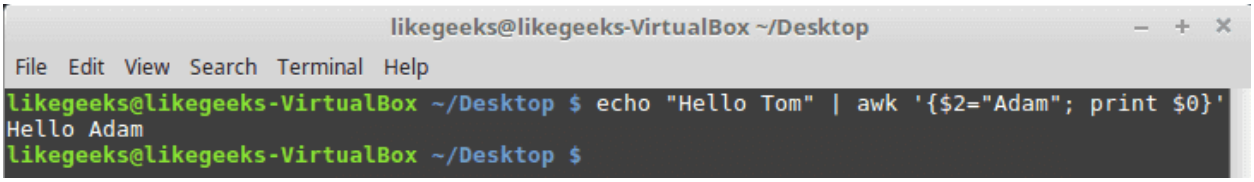
The above example prints the first word of each line.
Sometimes the separator in some files is not space nor tab but something else. You can specify it using -F option:

`$ awk -F: '{print $1}' /etc/passwd`

awk -F: '{print $1":" $2}' /etc/passwd

":">> This is used for field separator in file which is /etc/passwd

Using Multiple Commands
To run multiple commands, separate them with a semicolon like this:
$ echo "Hello Tom" | awk '{$2="Adam"; print $0}'



The first command makes the $2 field equals Adam. The second command prints the entire line.

```
echo "Hellow Tom" |awk '{$2="adam"; print $0}' >> file3.txt
File redirection >>output of echo will be written to file3.txt
```

Awk Preprocessing
awk 'BEGIN {print "The File Contents:"} {print $0}' myfile.txt

Awk Postprocessing
To run a script after processing the data, use the END keyword:
$ awk 'BEGIN {print "The File Contents:"}

{print $0}

END {print "File footer"}' myfile

awk 'BEGIN{print "The file content is:"} {print $0} END {print"This is end of Line"}' myfile.txt

Ouput will be like below-

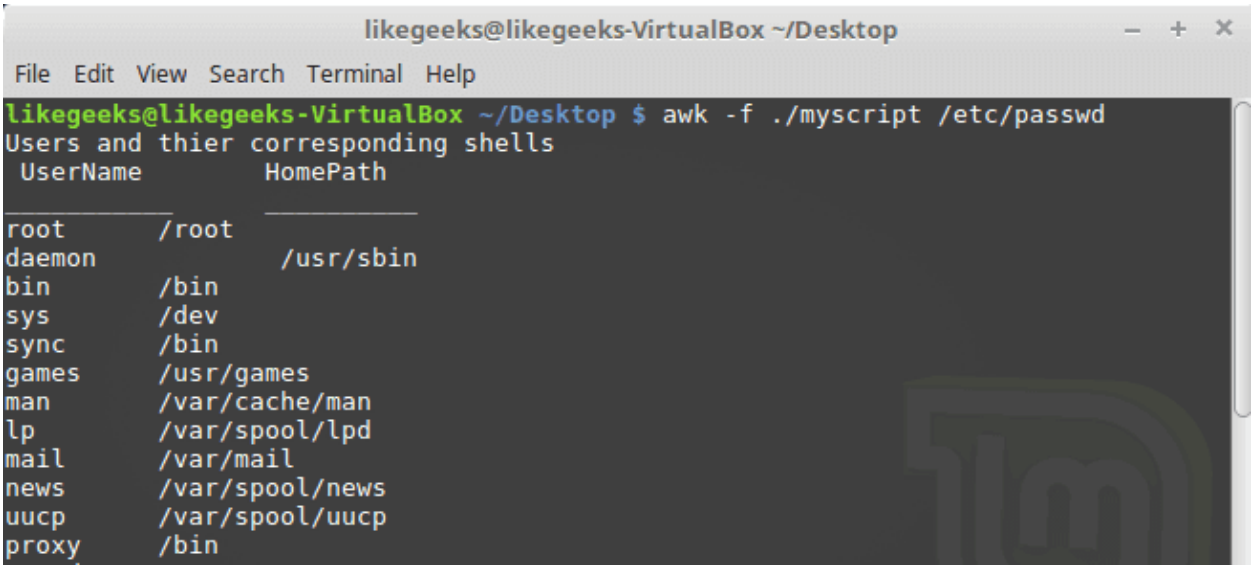
```
The file content is:
This is a test
This is a second test line
This is a third test line
This is a fourth test line
This is end of Line
```

This is useful, you can use it to add a footer for example.
Let's combine them together in a script file:

```
BEGIN {
print "Users and their corresponding home"
print " UserName \t HomePath"
print "_____ \t _____"
FS=":"
}
{
print $1 "  \t  " $6
}
END {
print "The end"
}
```

First, the top section is created using BEGIN keyword. Then we define the FS and print the footer at the end.

```
$ awk -f myscript   /etc/passwd
```



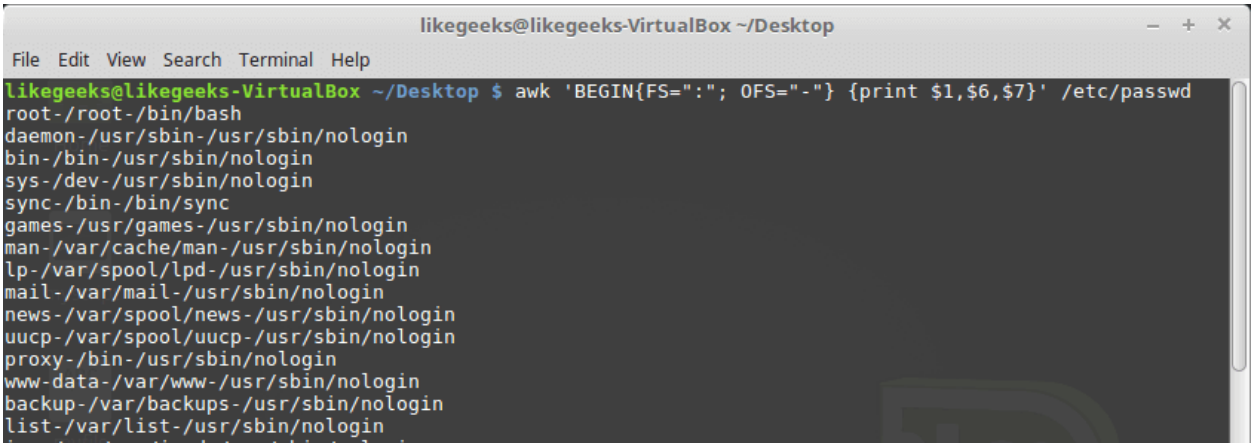
Built-in Variables
We saw the data field variables $1, $2 $3, etc are used to extract data fields, we also deal with the field separator FS.

But these are not the only variables, there are more built-in variables.

The following list shows some of the built-in variables:

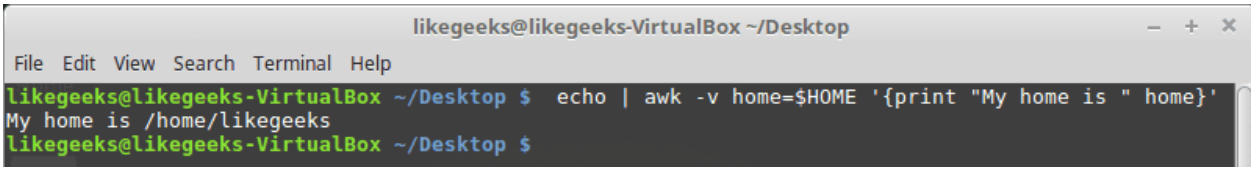
FIELDWIDTHS Specifies the field width.
RS Specifies the record separator.
FS Specifies the field separator.
OFS Specifies the Output separator.
ORS Specifies the Output separator.

```
$ awk 'BEGIN{FS=":"; OFS="-"} {print $1,$6,$7}' /etc/passwd
```



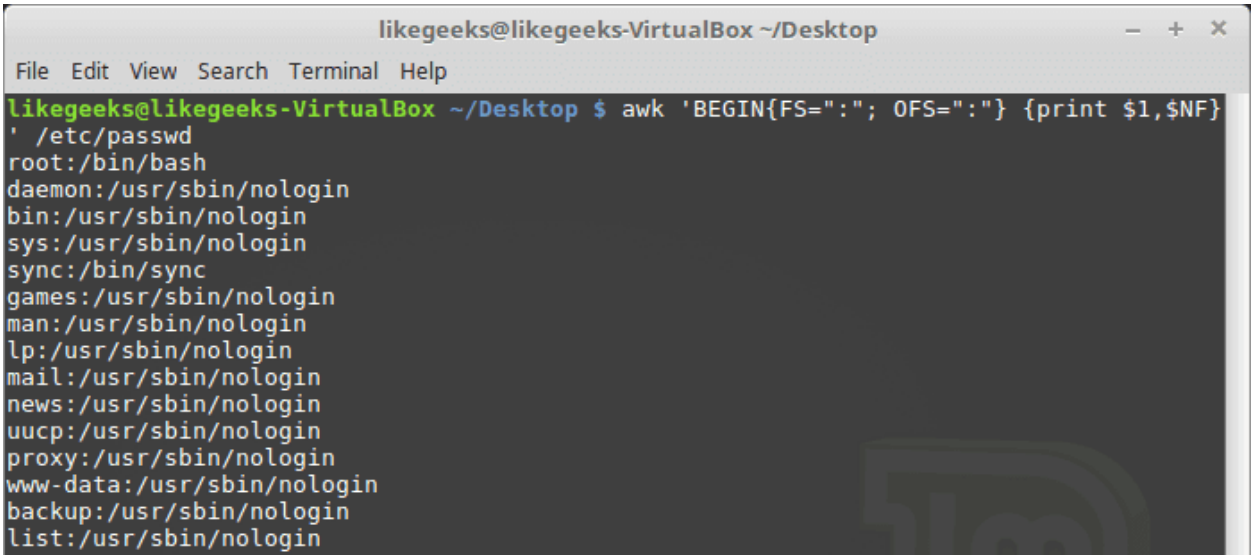
You can use bash variables without ENVIRON variables like this:

```
$  echo | awk -v home=$HOME '{print "My home is " home}'
```



The NF variable specifies the last field in the record without knowing its position:

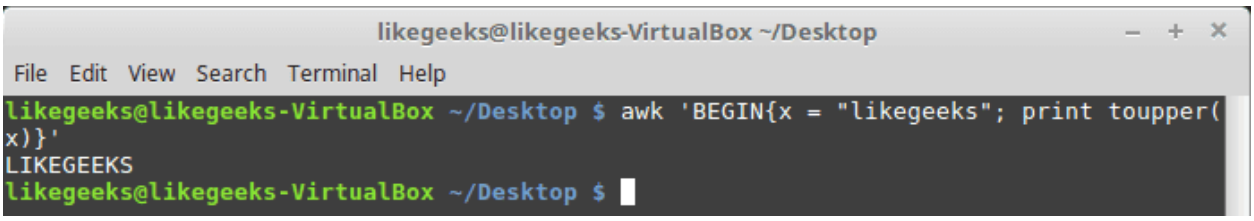
```
$ awk 'BEGIN{FS=":"; OFS=":"} {print $1,$NF}' /etc/passwd
```



User Defined Functions
You can define your function and use them like this:

String Functions
There are many string functions, you can check the list, but we will examine one of them as an example and the rest is the same:

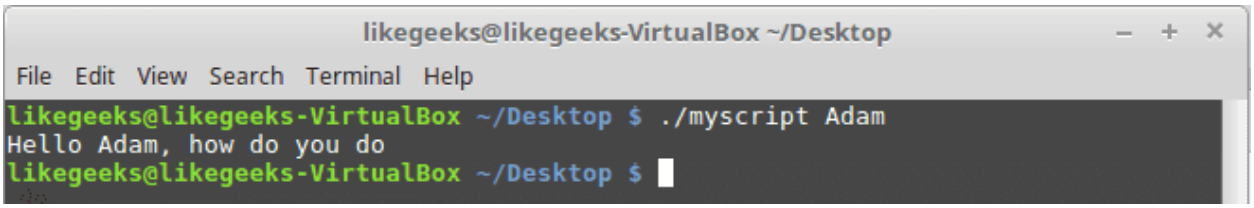
```
$ awk 'BEGIN{x = "likegeeks"; print toupper(x)}'
```



Shell Scripting

The parameters are not restricted to numbers, they could be strings like this:

```
#!/bin/bash
echo Hello $1, how do you do
```



Getting user input using the read command

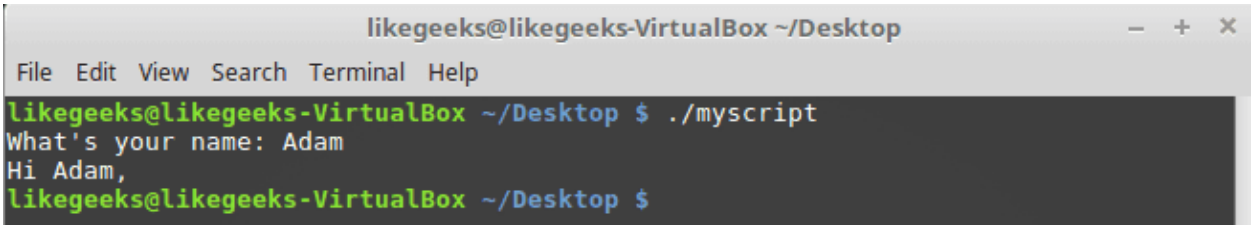
#!/bin/bash

echo -n "What's your name: "

read name

echo "Hi $name,"

The -n option is used to disable the newline, so you can type your text in the same line.



https://likegeeks.com/awk-command/
https://likegeeks.com/linux-bash-scripting-awesome-guide-part3/

Configure NTP server
Verify Server Time Sync
After NTP daemon has been started, wait a few minutes for the server to synchronize time with its pool list servers, and then run the following commands to verify NTP peers synchronization status and your system time.

```
# ntpq -p
# date -R
```

NTP service uses UDP port 123 on OSI transport layer (layer 4). It is designed particularly to resist the effects of variable latency (jitter). To open this port on RHEL/CentOS 7 run the following commands against Firewalld service.

```
# firewall-cmd --add-service=ntp --permanent
# firewall-cmd --reload
```

CPU utilization by

Ps -up |grep port number

Delete a File:

'rm' command is used in bash to remove any file. Create a file named 'delete_file.sh' with the following code to take the filename from the user and remove. Here, '-i' option is used to get permission from the user before removing the file.

```
#!/bin/bash
echo "Enter filename to remove"
read fn
rm -i $fn
```

Test if File Exist:

You can check the existence of file in bash by using **'-e'** or **'-f'** option. **'-f'** option is used in the following script to test the file existence. Create a file named, **'file_exist.sh'** and add the following code. Here, the filename will pass from the command line.

```
#!/bin/bash
filename=$1
if [ -f "$filename" ]; then
echo "File exists"
else
echo "File does not exist"
fi
```

Get Parse Current Date:

You can get the current system date and time value using **`date`** command. Every part of date and time value can be parsed using **'Y'**, **'m'**, **'d'**, **'H'**, **'M'** and **'S'**. Create a new file named **'date_parse.sh'** and add the following code to separate day, month, year, hour, minute and second values.

```
#!/bin/bash
Year=`date +%Y`
Month=`date +%m`
Day=`date +%d`
Hour=`date +%H`
Minute=`date +%M`
Second=`date +%S`
echo `date`
echo "Current Date is: $Day-$Month-$Year"
echo "Current Time is: $Hour:$Minute:$Second"
```

To fetch the customized date in linux

To custom your date format, **use a plus sign (+)**

$ date +"Day : %d Month : %m Year : %Y"

Day: 05 Day: 12 Month: 2013 Yea0072

https://ma.ttias.be/linux-date-format-change-the-date-output-for-scripts-or-commands/

To display your date and time with UTC format, use -u parameter

$ date –u
Thu 6 Dec 17:34:11 UTC 2018

Or just to display the current time:

```
$ date +"%T"
21:55:16
```

For instance, a simple YYYY-MM-DD representation:

```
$ date +"%Y-%m-%d"
2015-12-30
```

date +"%Y/%m/%d"
2018/12/06

Or a more complete example which follows the Apache log format of displaying dates (ie: *[30/Dec/2015:21:48:45 +0100]*).

```
$ date +"[%d/%b/%Y:%k:%M:%S %z]"
[30/Dec/2015:12:58:32 +0100]
```

%d = date in numerical form
%b = Month in Alphabetical form
%Y = Year in numerical form
%k = Hour in numerical form

date +"%d:%b:%Y"
06:Dec:2018

The date command also allows you pretty easy manipulation of the "current date". By default, date refers to "NOW". It'll show the current time or date when you execute the command. With the –d parameter you can also let it jump back & forth and show you a different date.

```
$ date +"%Y-%m-%d"
2015-12-30
$ date +"%Y-%m-%d" -d "8 days ago"
2015-12-22
$ date +"%Y-%m-%d" -d "next Sunday"
2016-01-03
$ date +"%Y-%m-%d" -d "last Friday"
2015-12-25
```

date +%D

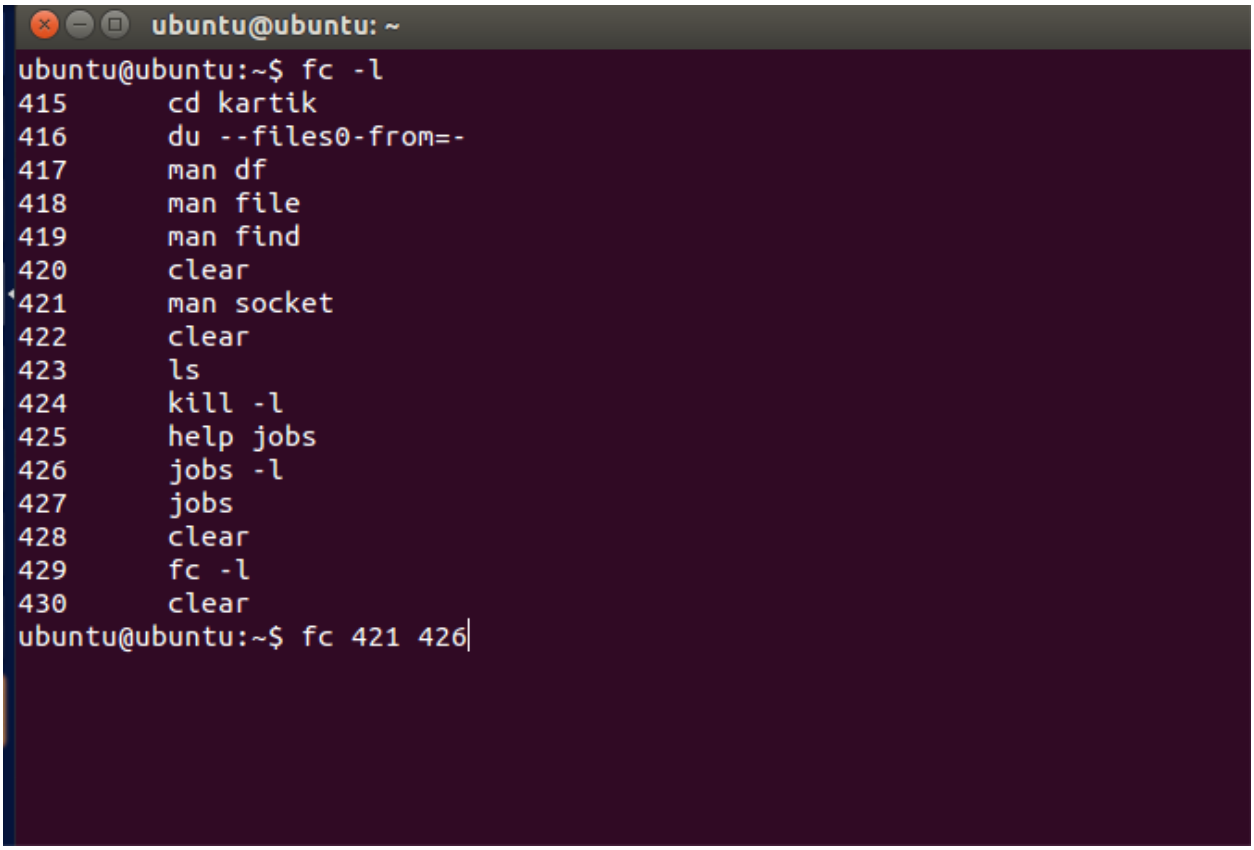
12/06/18

```
$ date +"%a %b %d %y"
Fri 06 Dec 2013
```

The "../" means to move the folder up one level. If you're buried deeper, say ~/Downloads/today/, you can still easily move that file with:
mv testfile ../../

Applications of fc command
fc command is the best way to edit the previously entered commands in the case of a minor mistake without re-writing the entire command syntax and argument again.
It can also be used to list the previously entered commands in the terminal which can be helpful in case you are working with some new commands.
fc command in a way lets you know the command history.

Using first and last : Suppose you want to just edit a particular set of commands, in that case you can use first and last arguments as shown below :



How to do reboot of a linux system
Reboot:

- init 6
- shutdown -r now
- reboot

How to check the current run-level?
Check current runlevel: runlevel

How to enter into single user mode?
init 1 >> On a terminal a user can login at a time, In this init level duplicate sessions are not allowed.

How to shutdown a linux server?
init 0
shutdown -h now
-a: Use file /etc/shutdown.allow
-c: Cancel scheduled shutdown.
halt -p
-p: Turn power off after shutdown.
poweroff

A runlevel of "5" will boot the system into GUI mode using XDM and X-Windows. Booting to runlevel "3" (often called console mode) is often used by servers which do not need a graphical user interface.

How to check the maximum speed of NIC installed in Linux server?
dmesg |grep -i duplex

[root@host-1-97 ~]# **ifconfig**
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.1.97 netmask 255.255.255.248 broadcast 10.0.1.103
    inet6 fe80::f2ef:d36:29b:ff46 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:61:00:01:90 txqueuelen 1000 (Ethernet)
    RX packets 241 bytes 30906 (30.1 KiB)
    RX errors 13 dropped 0 overruns 0 frame 13
    TX packets 244 bytes 33992 (33.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 6 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
    ether 52:54:00:fb:c1:2c txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

If we want to check the information for a particular NIC only then

```
[root@host-1-97 ~]# ifconfig ens3
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 10.0.1.97  netmask 255.255.255.248  broadcast 10.0.1.103
      inet6 fe80::f2ef:d36:29b:ff46  prefixlen 64  scopeid 0x20<link>
      ether 00:0c:61:00:01:90  txqueuelen 1000  (Ethernet)
      RX packets 268  bytes 33520 (32.7 KiB)
      RX errors 13  dropped 0  overruns 0  frame 13
      TX packets 258  bytes 37124 (36.2 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

How to know why server keeps restarting?

We need to check the logs in /var/log/messages

Last time system booted?

```
$ who -b
         system boot  2013-08-01 17:56
```

Past reboots

If you're interested in seeing a more extensive list of previous reboots you can use the last command.

```
$ last reboot | less
reboot   system boot  2.6.35.14-106.fc Thu Aug  1 17:56 - 02:03 (7+08:06)
reboot   system boot  2.6.35.14-106.fc Thu Aug  1 09:41 - 17:55  (08:14)
reboot   system boot  2.6.35.14-106.fc Thu Jul 25 15:24 - 17:55 (7+02:31)
reboot   system boot  2.6.35.14-106.fc Thu Jul 18 18:05 - 15:23 (6+21:17)
```

Past system shutdowns & runlevel changes?

```
last -x |less
```

How to Troubleshoot Sudden Server Restarts

```
[root@host-1-97 ~]# tail -f /var/log/messages

[root@host-1-97 ~]# last reboot |head -1
reboot   system boot  3.10.0-514.10.2. Mon Dec 17 18:16 - 18:50  (00:33)
```

Command to check the Ethernet speed

```
$ ethtool eth0 | grep -i speed
```

Task: Find full or half duplex speed

You can use dmesg command to find out your duplex mode:

```
# dmesg | grep -i duplex
```

```
Touch -- -abc.txt
File based on special characters(-).
```

Now we want to open this -abc.txt file using vim it will not open for that purpose we need to open the file using below option-

Vim -- -abc.txt

How to create a file starting with :
touch ':abc.txt'

touch abcd.txt ':abcd.txt'
This command will two files named as abcd.txt and :abcd.txt at the same time.

Find all files ending with extension .txt and delete them at the same time.
```
find . -name '*.txt' -delete
```

find . -name '*.txt' -exec rm -rf {} \;
Find all files ending with .txt extension and after that remove that also.
Dot . simply represents the current directory

Find / -name abc
This command will search abc file in the entire system.

If you hit only find command then it will display the path names of all the files in the current directory and subdirectories also.

Find . -name abc.txt
Search file abc.txt in the current directory.

Find . -type f -name abc.txt
Find file type in the current directory. Here -type describes the type of content we want to search.

Now if we want to search the directory here then we need to enter below command.
Find . -type d -name abc

Find / -type d -name abc >> This command will find the directory abc in the entire system.

Find xyz named file under the abc directory.
Find /home/abc -name xyz.txt

Find all the files in current directory having 777 permission.
Find . -type f -perm 0777 -print

Find file abc.txt and remove it at the same time.
Find . -type f -name "abc.txt" -exec {} rm -rf \;

find / -type f -cmin 1
Find all the files that changed 1 min back.

Find / -mmin 1
Find all the files modified in last one minute.

Find all the files that were accessed 10 days back.
Find / -atime 10

Find all the files that were modified 10 days back.
Find / -mtime 10

Find all the files which are modified more than 10 days back and less than 20 days
Find / -mtime +10 –mtime -20

Find files in your system less than 1 MB
Find / -size -1M

Find files more than 1MB and less than 2 MB
Find / -size +1M –size -2M

Remove all the files having size in between 1MB to 2MB
Find / -size +1M –size -2M –exec rm –rf {} \;

WC command
l for number of lines.
w for number of words
C for couting number of characters.

Read command in shell scriting.

read –p examples

```
#!/bin/bash
#
read -p "Please Enter some words followed by ENTER: " vara varb varc
echo "vara contains $vara"
echo "varb contains $varb"
echo "varc contains any remaining words $varc"
```

```
#!/bin/bash
read -p  " Enter your full name:" Firstname Middlename Lastname
echo "Your Name is $Firstname $Middlename $Lastname"
```

Arguments

```
#!/bin/bash

# example of using arguments to a script
echo "My first name is $1"
echo "My surname is $2"
echo "Total number of arguments is $#"
```

---

rm -rf *.txt
This command will delete all the files in a folder ending with .txt extension.

Awk command with examples
Some data for examples

```
1) Amit     Physics   80
2) Rahul    Maths     90
3) Shyam    Biology   87
4) Kedar    English   85
5) Hari     History   89
```

Save it as ak.sh

You can instruct AWK to print only certain columns from the input field. The following example demonstrates this –

```
[jerry]$ awk '{print $3 "\t" $4}' marks.txt
```

awk '{print $3 "\t" $4}' awk.txt
This command will print below output

Physics 80
Maths 90
Biology 87
English 85
History 89

```
[jerry]$ awk '/a/ {print $0}' marks.txt
```

On executing this code, you get the following result –

Output

```
2) Rahul    Maths     90
3) Shyam    Biology   87
4) Kedar    English   85
5) Hari     History   89
```

Now if we want to know the marks of Rahul only then hit below command

awk '/Rahul/ {print $0}' awk.txt

We will get the below output

2) Rahul Maths 90

Counting and Printing Matched Pattern

Let us see an example where you can count and print the number of lines for which a pattern match succeeded.

```
[jerry]$ awk '/a/{++cnt} END {print "Count = ", cnt}' marks.txt
```

awk '/Rahul/{++cnt} END {print "Count = " , cnt}' awk.txt

Output of this command would be
Count = 1

In this example, we increment the value of counter when a pattern match succeeds and we print this value in the END block.

Difference between RHEL6 & RHEL7

Here we made some useful changes made in RHEL 7 over RHEL 6.

OS BOOT TIME
RHEL6: 40 sec
RHEL7: 20 sec

MAXIMUM SIZE OF SINGLE PARTITION
RHEL6: 50TB(EXT4)
RHEL7: 500TB(XFS)

BOOT LOADER
RHEL6: /boot/grub/grub.conf
RHEL7: /boot/grupb2/grub.cfg

PROCESSOR ARCHITECTURE
RHEL6: It support 32bit & 64bit both
RHEL7: It only support 64bit

HOW TO FORMAT OR ASSIGN A FILE SYSTEM IN
RHEL6: #mkfs.ext4 /dev/hda4
RHEL7: #mkfs.xfs /dev/hda3

HOW TO REPAIR A FILE SYSTEM IN
RHEL6: #fsck -y /dev/hda3
RHEL7: #xfs_repair /dev/hda3

COMMAND TO MANAGE NETWORK IN RHEL6 AND RHEL7
RHEL6: #setup
RHEL7: #nmtui

HOSTNAME CONFIGURATION FILE
RHEL6: /etc/sysconfig/network
RHEL7: /etc/hostname

DEFAULT ISO IMAGE MOUNT PATH
RHEL6: /media
RHEL7: /run/media/root

FILE SYSTEM CHECK
RHEL6: e2fsck
RHEL7: xfs_repair

RESIZE A FILE SYSTEM
RHEL6: #resize2fs -p /dev/vg00/lv1
RHEL7: #xfs_growfs /dev/vg00/lv1

TUNE A FILE SYSTEM
RHEL6: tune2fs
RHEL7: xfs_admin

IPTABLES AND FIREWALL
RHEL6: iptables
RHEL7: firewalld

IPtables
To see firewall status in RHEL7
#firewall-cmd --state
To see Firewall status in RHEL6
#service iptables status
To stop firewall in RHEL7
#systemctl stop firewalld.service
To stop firewall in RHEL6
#service iptables stop

COMMUNICATION BETWEEN TCP AND UDP IN BACK END

RHEL6: netcat
RHEL7: ncat

INTERFACE NAME

RHEL6: eth0
RHEL7: ens198(N)

COMBINING NIC

RHEL6: Network Bonding
RHEL7: Team Driver

NSF Server Version

RHEL6: NFSv2
RHEL7: NFSV4

DATABASE USED

RHEL6: Mysql
RHEL7: mariaDB
RHEL7 also support Mysql

MANAGING SERVICES

RHEL6:

#service sshd restart
#chkconfig sshd on

RHEL7:

#systemctl restart sshd
#systemctl enable shhd

File System.

RHEL6 default file system is ext4
xfs is RHEL7 default file system.

Kernel Version

RHEL6 default kernel version is 2.6 while RHEL7 is 3.10

UID Allocation

In RHEL6 default UID assigned to users would start from 500 while in RHEL7 it's starting from 1000.
But this can be changed if required by editing **/etc/login.defs** file.

Maximum supported File Size.

In RHEL6 maximum file size of an individual file can be up to 16TB while in RHEL7 it can be up to 500TB which is very large in comparison to RHEL6.

Maximum Supported File System Size.

In RHEL6 maximum file system size=16TB (for 64bit Machine) and 8TB (for 32 bit machine). While in RHEL7 maximum file system size is 500TB.
Also keep in mind that RHEL6 does not support XFS on 32-bit machines.

Change in file system structure.

In rhel6 /bin,/sbin,/lib and /lib64 are usually under /
In rhel7, now /bin,/sbin,/lib and /lib64 are nested under /usr.
The /tmp directory can now be used as a temporary file storage system (tmpfs)

Space Required to Installing RHEL7?

Now if you want to install RHEL7 in your machine, RedHat recommends minimum 5 GB of disk space to install this release of RHEL series for all supported architectures.

.Hostname lookup and setup

In rhel5 and rhel6 versions, we can edit file /etc/sysconfig/network to set hostname but in rhel7 we can directly change the hostname using below commands.

hostnamectl
nmtui
nmcli

Example:

in RHEL6 #hostname
in RHEL7 #hostnamectl status and #hostname

Few More notable changes in RHEL 7.
Netstat and **ifconfig** commands also disappeared from RHEL7 but it can be used by installing net-tools.
The move from sysvinit to systemd is one of most important change that has been made and which is a matter of concerned.
Command **tail -n** is replaced by **journalctl -n**
Command **tail -f** is replaced by **journalctl -f**
For displaying kernel messages instead of **dmesg** now in RHEL7 we use **journalctl -k**

USE of SSHPASS in server-client configuration
https://www.tecmint.com/sshpass-non-interactive-ssh-login-shell-script-ssh-password/
https://www.cyberciti.biz/faq/noninteractive-shell-script-ssh-password-provider/

A port in Linux is nothing but a logical connection place. The TCP/IP use port for communication across the LAN, WAN, and Internet. Typically, a server program such as Apache (httpd) listens on TCP port 80 or 443. A client program such as a web browser connects to TCP port 80 to request web page. You can find preassigned port numbers in /etc/services files with the help of cat command or grep command/egrep command linux command. For example:

```
$ cat /etc/services
$ grep -w 80/tcp /etc/services
$ grep -w 443/tcp /etc/services
$ egrep -w '(80|22|443)/tcp' /etc/services
```

How to use lsof command to display open port on Linux

```
$ sudo lsof -i :portNumber(Here we need to put number like 22 for open ssh) to check whether it is used or not.
$ sudo lsof -i tcp:portNumber
$ sudo lsof -i udp:portNumber
$ sudo lsof -i :22
$ sudo lsof -i :22 | grep LISTEN

[root@sterlite ~]# lsof -i :22 |grep LISTEN
```

Using netstat command

```
$ sudo netstat -tulpn
$ sudo netstat -tulpn | more
$ sudo netstat -tulpn | grep ':port'
$ sudo netstat -tulpn | grep ':22'
```

You can combine multiple operations to be done on permission like this next example. It will make sure owner has read/write/execute, also add write permission for group and remove execution for everyone else:
`chmod u=rwx,g+w,o-x /path/to/file`

This last one will use rFile as a reference to set permission on file. When completed, the permission of file will be exactly as they are for rFile.

```
chmod --reference=/path/to/rFile /path/to/file
chmod u=rwx,o=r test1.txt
chmod u=rwx,o-x data1/
chmod --reference=data1 data2/
```

Access control list

The command "setfacl" refers to Set File Access Control Lists and "getfacl" refers to Get File Access Control List. Each file and directory in a Linux filesystem is created with a specific set of file permissions for its access. Each user can have different set of file access permissions. The permissions can be set using the setfacl utility. In order to know the access permissions of a file or directory we use getfacl. The getfacl command displays the access permissions of files and directories with file name, owner, group and the ACL's(Access Control List). When we create a directory it is created with a default set of access permissions and by using getfacl we will be able to see the access rights.

What are the default access permissions for a newly created directory?

To know this, first open a terminal and open the folder in which you want to create a subfolder. Next type "mkdir <folder-name>" and press the ENTER key. This will create a folder with default access permissions. To know the access permissions, type

getfacl <folder-name>

Now you will see the output of getfacl as something like the following:

```
# file: file-name
# owner:
# group:
user::rwx
user:x:---
user:y:r--
group::r--
mask::rw-
other::---
```

Thus from the output of getfacl we will be able to see the access permissions of a file. In the above example, when we type getfacl <file-name>

the output will be shown as in the above format. It displays the owner of the file, the group which has access to it and also its various users and their access rights. In the above case the users are x and y, where the user 'x' is having no permission on this file and therefore it is shown with --- symbol indicating no read/write/execute permissions for the user x. Now considering the other user 'y', it is having the permission r-- which means read-only rights. The default umask is set to rw- (read/write permissions).

How to copy the ACL of one folder to other?

Consider an example of copying the ACL of the directory named "x" to "y". For this, firstly we should know the ACL of the directory named "x". To obtain this type the command

```
getfacl x
```

This will display the ACL of the directory named "x" in the above mentioned format:

```
# file: x
# owner:
# group:
user::rwx
user:x:---
user:y:r--
group::r--
mask::rw-
other::---
```

To copy the ACL of one directory to the other we use the setfacl command. That is

```
setfacl --setfile =- y
```

As mentioned we want to copy the ACL of "x" to "y", for this we have to type the command

```
getfacl x | setfacl -R –setfile = -y
```

Here "getfacl x" will get the ACL of the directory named "x" and this output is given to the setfacl command using pipe. Thus getfacl will give the ACL of the directory "x" and

```
setfacl -R –setfile = -y
```

will set that ACL to the directory named "y". "-R" is used to set this ACL recursively.

How to inherit the ACL of parent directory to its child?

To copy the ACL of the parent directory to its child, use the following command

```
getfacl . | setfacl -R --setfile = -subdirectory_name
```

The "getfacl ." will get the ACL of the parent directory and setfacl will set that ACL to its sub-directories. Now for verification, type:

```
getfacl subdirectory_name
```

and also

```
getfacl directory_name
```

If both are same then the ACL of the sub-directory is same as the ACL of the parent.

What is Difference between L2 switch and a L3 switch?

A L2 switch does switching only. This means that it uses MAC addresses to switch the packets from a port to the destination port (and only the destination port). It therefore maintains a MAC address table so that it can remember which ports have which MAC address associated.

A L3 switch also does switching exactly like a L2 switch. The L3 means that it has an identity from the L3 layer. Practically this means that a L3 switch is capable of having IP addresses and doing routing. For intra-VLAN communication, it uses the MAC address table. For Inter-VLAN communication, it uses the IP routing table.

---

rsync command

Copy/Sync a Directory on Local Computer

The following command will transfer or sync all the files of from one directory to a different directory in the same machine. Here in this example, /root/rpmpkgs contains some rpm package files and you want that directory to be copied inside /tmp/backups/ folder.

```
[root@tecmint]# rsync -avzh /root/rpmpkgs /tmp/backups/

sending incremental file list

rpmpkgs/

rpmpkgs/httpd-2.2.3-82.el5.centos.i386.rpm

rpmpkgs/mod_ssl-2.2.3-82.el5.centos.i386.rpm
```

```
rpmpkgs/nagios-3.5.0.tar.gz

rpmpkgs/nagios-plugins-1.4.16.tar.gz

sent 4.99M bytes  received 92 bytes  3.33M bytes/sec

total size is 4.99M  speedup is 1.00
```

Copy/Sync Files and Directory to or From a Server

This command will sync a directory from a local machine to a remote machine. For example: There is a folder in your local computer "rpmpkgs" which contains some RPM packages and you want that local directory's content send to a remote server, you can use following command.

```
[root@tecmint]$ rsync -avz rpmpkgs/ root@192.168.0.101:/home/

root@192.168.0.101's password:

sending incremental file list

./

httpd-2.2.3-82.el5.centos.i386.rpm

mod_ssl-2.2.3-82.el5.centos.i386.rpm

nagios-3.5.0.tar.gz

nagios-plugins-1.4.16.tar.gz

sent 4993369 bytes  received 91 bytes  399476.80 bytes/sec

total size is 4991313  speedup is 1.00
```

Copy/Sync Files and Directory to or From a Server

This command will sync a directory from a local machine to a remote machine. For example: There is a folder in your local computer "rpmpkgs" which contains some RPM packages and you want that local directory's content send to a remote server, you can use following command.

```
[root@tecmint]$ rsync -avz rpmpkgs/ root@192.168.0.101:/home/

root@192.168.0.101's password:

sending incremental file list

./
```

```
httpd-2.2.3-82.el5.centos.i386.rpm

mod_ssl-2.2.3-82.el5.centos.i386.rpm

nagios-3.5.0.tar.gz

nagios-plugins-1.4.16.tar.gz

sent 4993369 bytes  received 91 bytes  399476.80 bytes/sec
total size is 4991313  speedup is 1.00
```

Copy/Sync a Remote Directory to a Local Machine

This command will help you sync a remote directory to a local directory. Here in this example, a directory /home/tarunika/rpmpkgs which is on a remote server is being copied in your local computer in /tmp/myrpms

```
[root@tecmint]# rsync -avzh root@192.168.0.100:/home/tarunika/rpmpkgs
/tmp/myrpms

root@192.168.0.100's password:

receiving incremental file list

created directory /tmp/myrpms

rpmpkgs/

rpmpkgs/httpd-2.2.3-82.el5.centos.i386.rpm

rpmpkgs/mod_ssl-2.2.3-82.el5.centos.i386.rpm

rpmpkgs/nagios-3.5.0.tar.gz

rpmpkgs/nagios-plugins-1.4.16.tar.gz

sent 91 bytes  received 4.99M bytes  322.16K bytes/sec

total size is 4.99M  speedup is 1.00
```

Automatically Delete source Files after successful Transfer

Now, suppose you have a main web server and a data backup server, you created a daily backup and synced it with your backup server, now you don’t want to keep that local copy of backup in your web server.

So, will you wait for transfer to complete and then delete those local backup file manually? Of Course NO. This automatic deletion can be done using '–remove-source-files' option.

```
[root@tecmint]# rsync --remove-source-files -zvh backup.tar /tmp/backups/

backup.tar

sent 14.71M bytes  received 31 bytes  4.20M bytes/sec

total size is 16.18M  speedup is 1.10

[root@tecmint]# ll backup.tar

ls: backup.tar: No such file or directory
```

Do a Dry Run with rsync

If you are a newbie and using rsync and don't know what exactly your command going will do,Rsync could really mess up the things in your destination folder and then doing an undo can be a tedious job.

Use of this option will not make any changes only do a dry run of the command and shows the output of the command, if the output shows exactly same you want to do then you can remove '–dry-run' option from your command and run on the terminal.

```
root@tecmint]# rsync --dry-run --remove-source-files -zvh backup.tar
/tmp/backups/

backup.tar

sent 35 bytes  received 15 bytes  100.00 bytes/sec

total size is 16.18M  speedup is 323584.00 (DRY RUN)
```

```
[root@sterlite 20190318_rawfiles]# for ((i=1 ;i<=100; i++));do
> cd ../20190318_rawfiles/;ll |wc -l
> sleep 2
> done
```

How to execute for loop.

Vim extended

Or you can force vim to discard your changes and edit the new file, using the force (!) character:

```
:edit! foo.txt
```

In vim we can execute external commands as well. For Unix, this should work. These commands are done from within vim:
:!mkdir Shashank

This command will make a directory with the name of shashank in the current location.

Choosing Colors

vim guesses the background color that you are using. If it is black (or another dark color) it will use light colors for text. If it is white (or another light color) it will use dark colors for text. If vim guessed wrong the text will be hard to read. To solve this, set the 'background' option. For a dark background:
:set background=dark

Editing A List Of Files

You can start vim to edit a sequence of files. For example:
vim one.c two.c three.c
This command starts vim and tells it that you will be editing three files. vim displays just the first file. After you have done your thing in this file, to edit the next file you use this command:
:next

If you have unsaved changes in the current file, you will get an error message and the ":next" will not work. This is the same problem as with ":edit" mentioned in the previous section. To abandon the changes:
:next!

But mostly you want to save the changes and move on to the next file. There is a special command for this:
:wnext

Backup Files

Usually vim does not produce a backup file. If you want to have one, all you need to do is execute the following command:
:set backup

The name of the backup file is the original file with a tilde ("~") added to the end. If your file is named data.txt, for example, the backup file name is data.txt~. If you do not like the fact that the backup files end with ~, you can change the extension:
:set backupext=.bak

Important vim command in production servers
To make vim keep the original file, set the 'patchmode' option. This specifies the extension used for the first backup of a changed file. Usually you would do this:
:set patchmode=.orig
When you now edit the file data.txt for the first time, make changes and write the file, vim will keep a copy of the unchanged file under the name **"data.txt.orig".**

On Unix this command should do the same thing:
view file

Saving As A New File Name

A clever way to start editing a new file is by using an existing file that contains most of what you need. For example, you start writing a new program to move a file. You know that you already have a program that copies a file, thus you start with:
:edit copy.c

You can delete the stuff you don't need. Now you need to save the file under a new name. The ":saveas" command can be used for this:

vim will write the file under the given name, and edit that file. Thus the next time you do ":write", it will write "move.c". "copy.c" remains unmodified.