# Welcome

# Boot Process in LINUX

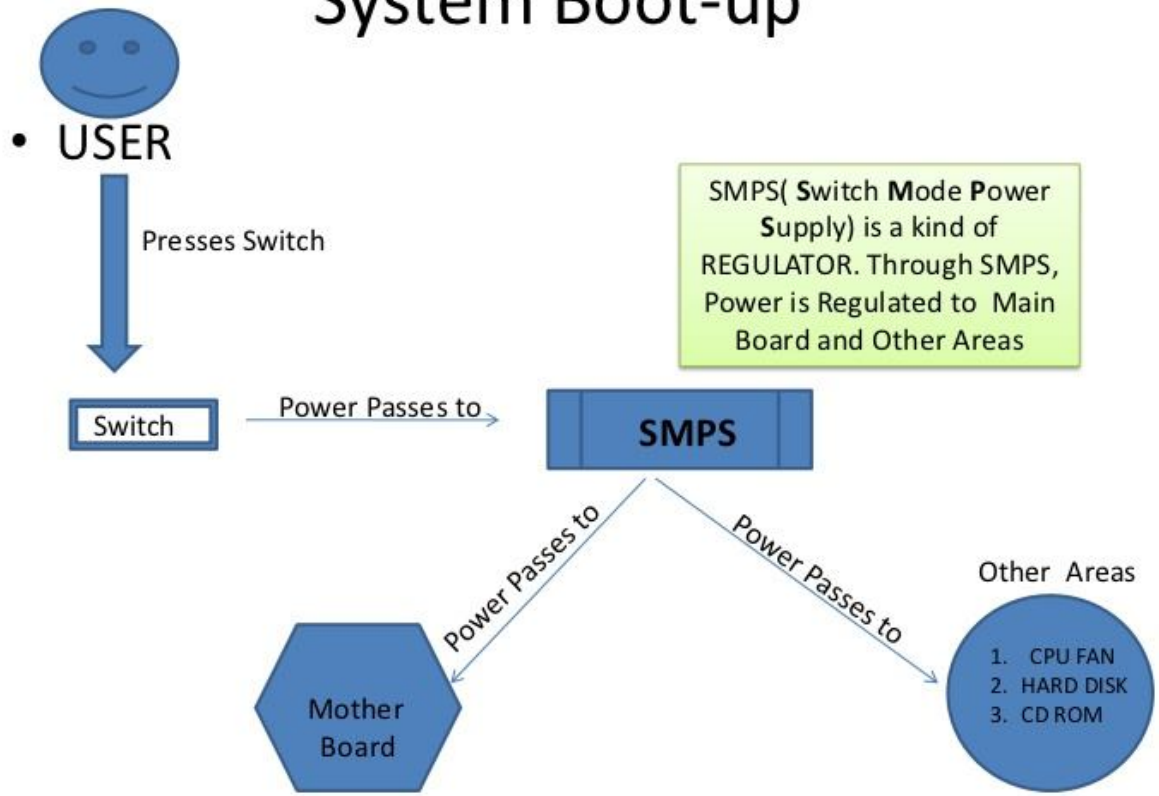
Courtesy : **Prof. Sadiq Bashir**

**Main Objectives are :**

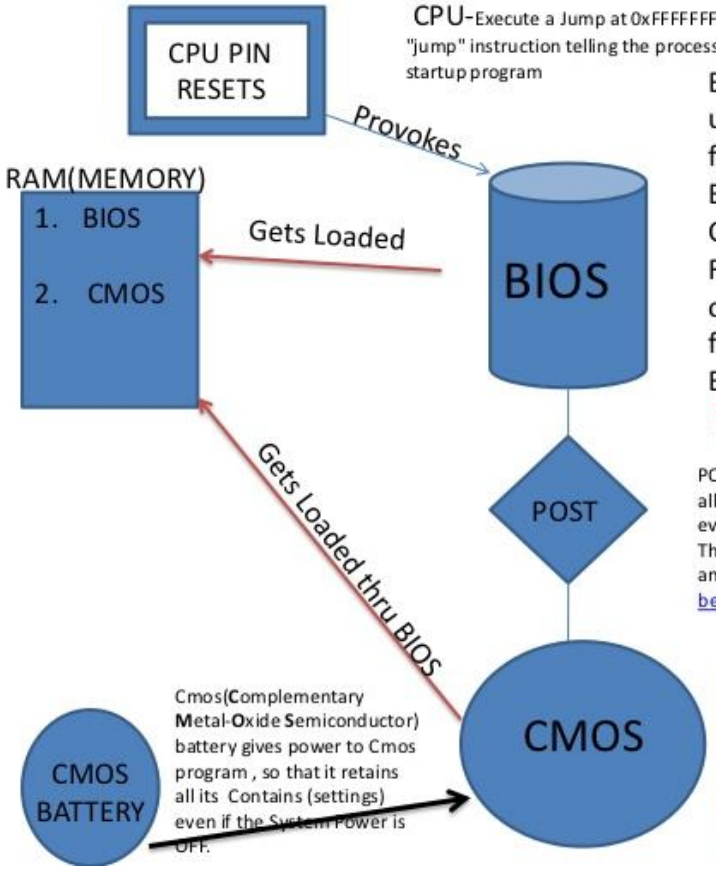
1. You can change the Behavior of System , Look-n-feel as you desire (within certain limits) .

2. You can also Troubleshoot the problems arising at the time of booting.

# System Boot-up



## IN MOTHER BOARD (A.K.A MAIN BOARD)




CPU-Execute a Jump at 0xFFFFFFF0 ( 1st instruction after power on ) "jump" instruction telling the processor where to go to find the real BIOS startup program

BIOS(**B**asic **I**nput/**O**utput **S**ystem) is built using Assembly Language. Its main function is to Load Kernel into Memory. BIOS is a Program , located at a ROM Chip in Motherboard . It is also known as ROM Program . It Keeps the information of all the Hardware and is responsible for major tasks in Boot Process.
BIOS Gets Loaded in MEMORY.
(Boot Strapping).

POST(power-on self test ) is a Program that provides Status of all the peripherals connected to the System and checks if every device is functioning properly or
The BIOS performs the power-on self test (POST). If there are any fatal errors, the boot process stops. POST beep codes can be found in this area of the Troubleshooting Expert

CMOS is a Program , it checks the boot devices priority and also identifies the 1st sector (0 Cylinder , 0 Tracks) of H/D ,512 bytes in Size.

## At the **First** Sector of HD (0 Cylinder = 0 Tracks = 1st Sector)

RAM(MEMORY)

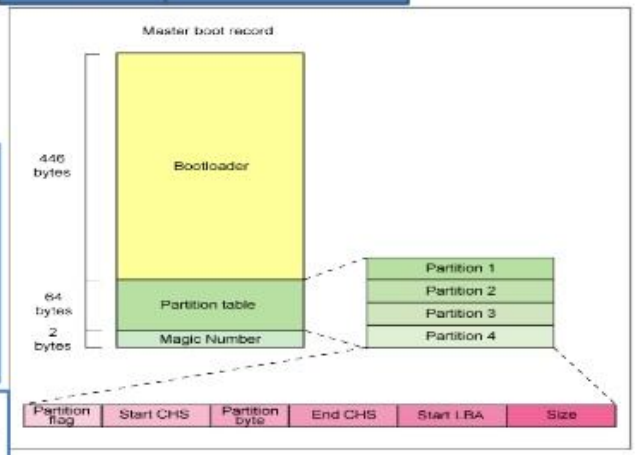
1.BIOS
2.CMOS
3.MBR

**HARD DISK**

MBR

1st Sector

Gets Loaded (thru BIOS)




Stage 1 boot loader
The primary boot loader that resides in the MBR is a 512-byte image containing both program code and a small partition table (see Figure 2). The first 446 bytes are the primary boot loader, which contains both executable code and error message text. The next sixty-four bytes are the partition table, which contains a record for each of four partitions (sixteen bytes each). The MBR ends with two bytes that are defined as the magic number (0xAA55). The magic number serves as a validation check of the MBR.

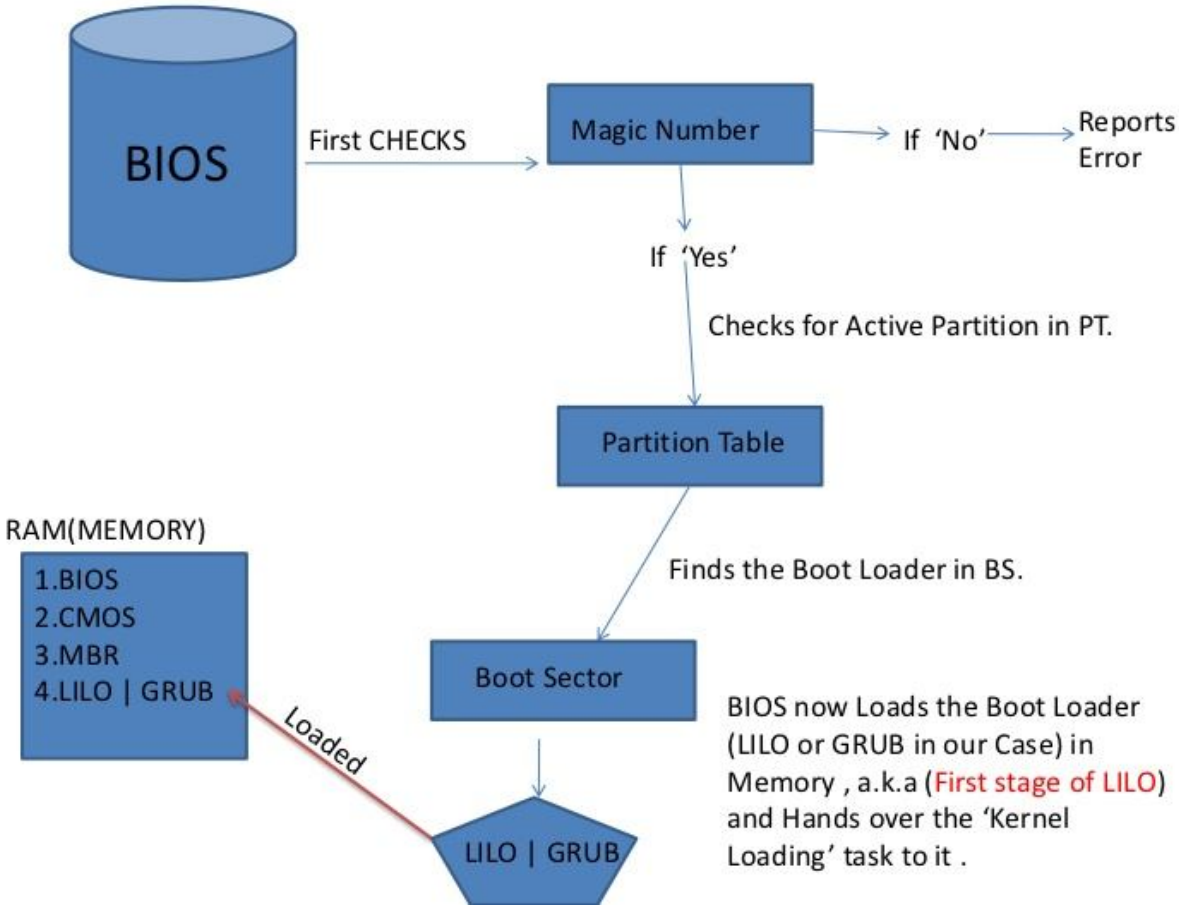
The job of the primary boot loader is to find and load the secondary boot loader (stage 2). It does this by looking through the partition table for an active partition. When it finds an active partition, it scans the remaining partitions in the table to ensure that they're all inactive. When this is verified, the active partition's boot record is read from the device into RAM and executed.

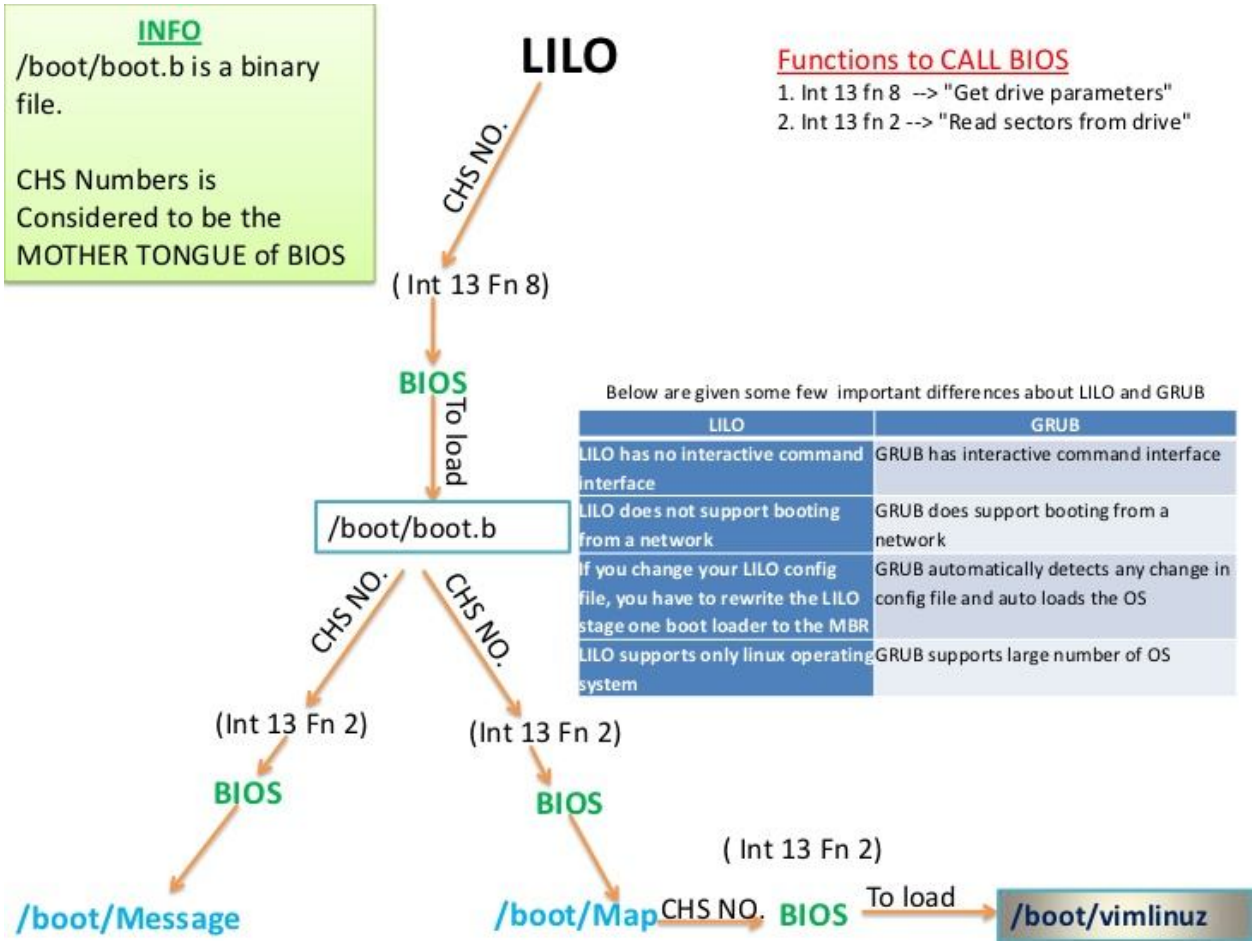
Stage 2 boot loader
The secondary, or second-stage, boot loader could be more aptly called the kernel loader. The task at this stage is to load the Linux kernel and optional initial RAM disk.

# Let us Understand MBR in Detail

Master Boot Record (MBR) is a Common Program in OS , whenever you boot your System with the Bootable CD , The Installer (Anaconda in Linux) writes the MBR at the First Sector of your H/D . As shown in above diagram, MBR is divided into 3 main parts .

1. **Boot Sector (446 Bytes) :** Boot Sector is a Area in MBR , which contains the information of 'Boot Loader' like LILO & GRUB (of Linux **(POWERFUL)**) and NTLDR (of Windows **(Less Powerful than LILO & GRUB)**).
   Boot Loader will be responsible for Loading the 'Kernel' (Vmlinuz in our case) , after BIOS assigns it the TASK to do so.
2. **Partition Table (64 Bytes) :** Partition Table is again a sub-divided part of MBR . It has 4 programs of 16 Bytes Each (4 X 16 b = 64 Bytes) . Each Program is Responsible for each Partition in H/D . Hence you cannot create more than 4 partitions in a H/D. Out of this 4 partition , you can create one partition as 'Extended' and create Sub partitions or Logical Partitions in it.
3. **Magic Number (2 Bytes) :** Magic Number basically shows the status of other two divisons of MBR. If Boot Sector and Partition Table are written Sucessfully , Magic Number will be 'Yes' , Otherwise it will be 'No'.


BIOS
First CHECKS
Magic Number
If 'No'
Reports Error
If 'Yes'
Checks for Active Partition in PT.
Partition Table
Finds the Boot Loader in BS.
RAM(MEMORY)
1.BIOS
2.CMOS
3.MBR
4.LILO | GRUB
Loaded
Boot Sector
LILO | GRUB
BIOS now Loads the Boot Loader (LILO or GRUB in our Case) in Memory , a.k.a (First stage of LILO) and Hands over the 'Kernel Loading' task to it .


INFO
/boot/boot.b is a binary file.
CHS Numbers is Considered to be the MOTHER TONGUE of BIOS
LILO
CHS NO.
( Int 13 Fn 8)
BIOS
To load
/boot/boot.b
CHS NO.
CHS NO.
(Int 13 Fn 2)
(Int 13 Fn 2)
BIOS
BIOS
/boot/Message
/boot/Map
( Int 13 Fn 2)
CHS NO.
BIOS
To load
/boot/vimlinuz
Functions to CALL BIOS
1. Int 13 fn 8 --> "Get drive parameters"
2. Int 13 fn 2 --> "Read sectors from drive"
Below are given some few important differences about LILO and GRUB
LILO
GRUB
LILO has no interactive command interface
GRUB has interactive command interface
LILO does not support booting from a network
GRUB does support booting from a network
If you change your LILO config file, you have to rewrite the LILO stage one boot loader to the MBR
GRUB automatically detects any change in config file and auto loads the OS
LILO supports only linux operating system
GRUB supports large number of OS

Installer (Anaconda), provides the Cylindrical Head Sector (CHS) number of /boot/boot.b , /boot/Message & /boot/Map file to LILO. LILO can Load all these file with the help of CHS Number , however it doesn't understand CHS number , So it Calls BIOS to help it Load all these files . LILO uses a function (Int 13 Fn 8) for /boot/boot.b & function (Int 13 Fn 2) for other files , to Call BIOS.

After Function (Int 13 Fn 8) is executed ,BIOS Loads boot.b file into Memory , it is known as Second stage of LILO . And When Function (Int 13 Fn 2) is executed one by one ,BIOS Loads both Message and Map file into Memory.

RAM(MEMORY)

1. BIOS
2. CMOS
3. MBR
4. LILO || GRUB
5. Boot.b
6. Message
7. Map

Important files in /boot directory

1 . Boot.b
2. Message
3. Map
4. Vmlinuz
5. Initrd.img

**1. /Message has CHS Number of 'Splash Screen (also known as Kernel Listings)' and it calls BIOS to Load this , for the same reason ,as it doesn't understand CHS Number.**

**2. /Map has CHS Number of 'vmlinuz' and it calls BIOS to Load this , as it doesn't understand CHS Number.**

/Message

CHS No.

( Int 13 Fn 2)

BIOS

Splash Screen


! Hope this is Self Explanatory !

RAM(MEMORY)

1. BIOS
2. CMOS
3. MBR
4. LILO || GRUB
5. Boot.b
6. Message
7. Map
8. Vmlinuz

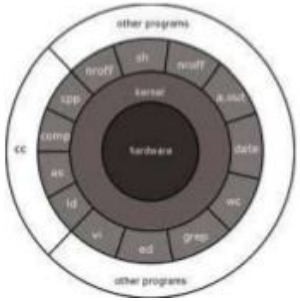
/Map

CHS No.

( Int 13 Fn 2)

BIOS

Vmlinuz (Kernel)



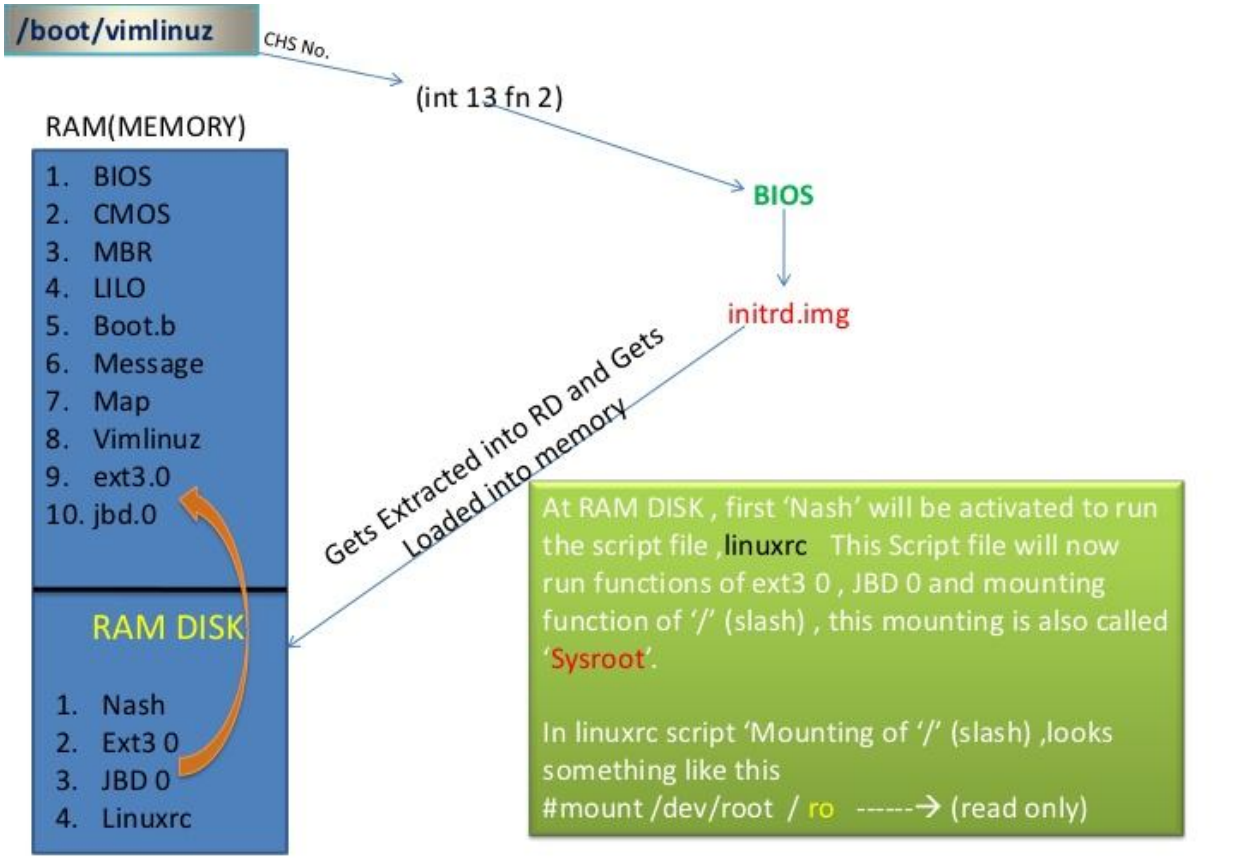
The kernel is the central part of an **operating system**, that directly controls the **computer hardware**. Usually, the kernel is the first of the user-installed **software** on a computer, booting directly after the **BIOS**. Kernel is in bzip format.
Kernel has a CHS Number of initrd.img (INITIAL RAM DISK) , it Calls BIOS by (Int 13 fn 2) function and it loads initrd.img into Memory . Initrd.img is located at /boot and is in gzip format. Initrd.img has 4 main files :

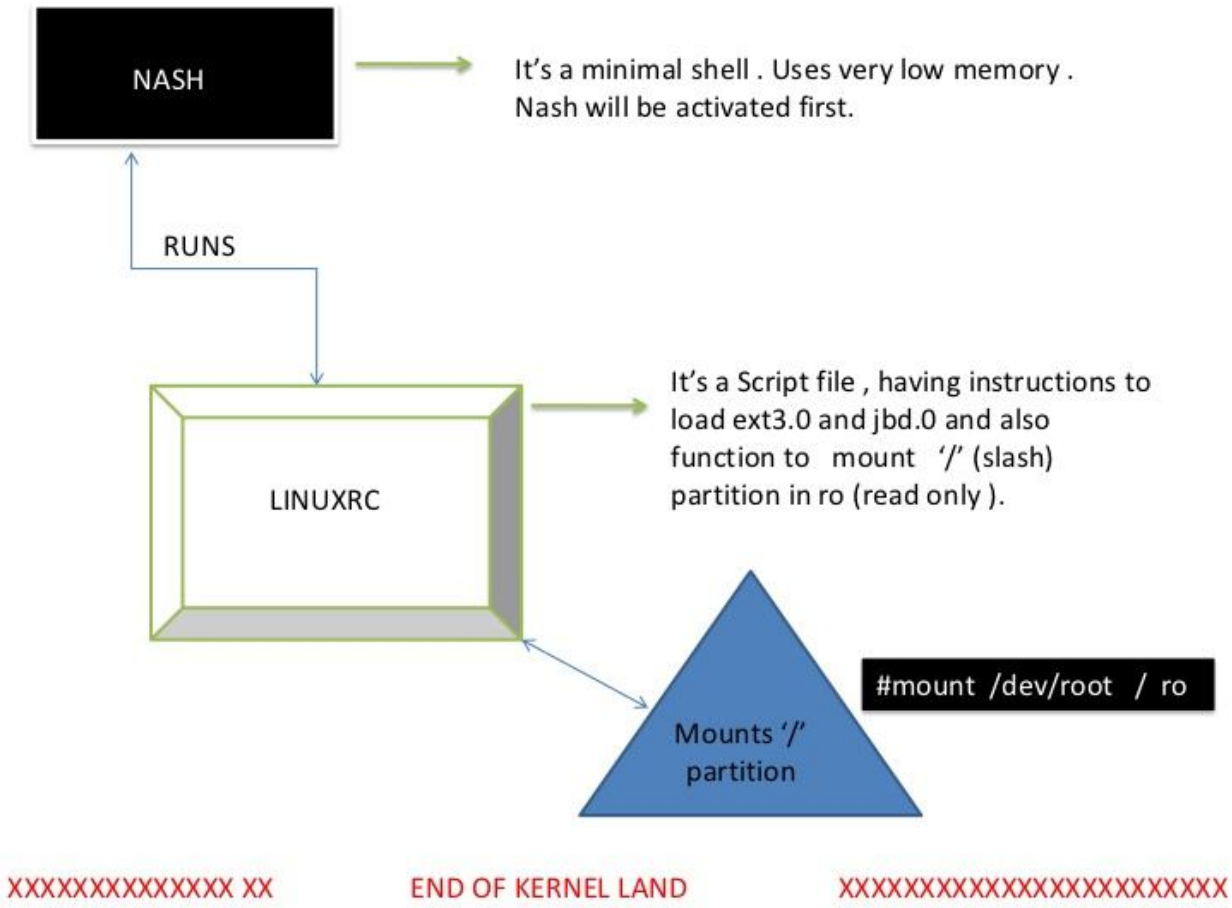
1. nash - Nash is a Minimal shell , it takes low memory . It is used to run linuxrc script.
2. ext3 0 - kernel needs a driver file called 'ext3 0' for filesystems without which , you you cannot mount any partition.
3. JBD 0 – It's a H/D disk driver file.
4. Linuxrc – It's a Script file , having functions of ext 3 0 and JBD 0 and also function to mount '/' (slash) partition.

Initrd.img is extracted / decompressed and its Contents (above 4 files) are Loaded into 'RAM DISK'

WHAT IS RAM DISK ?

A RAM disk is a portion of RAM which is being used as if it were a disk drive. RAM disks have fixed sizes, and act like regular disk partitions. Access time is much faster for a RAM disk than for a real, physical disk. However, any data stored on a RAM disk is lost when the system is shut down or powered off. RAM disks can be a great place to store temporary data.

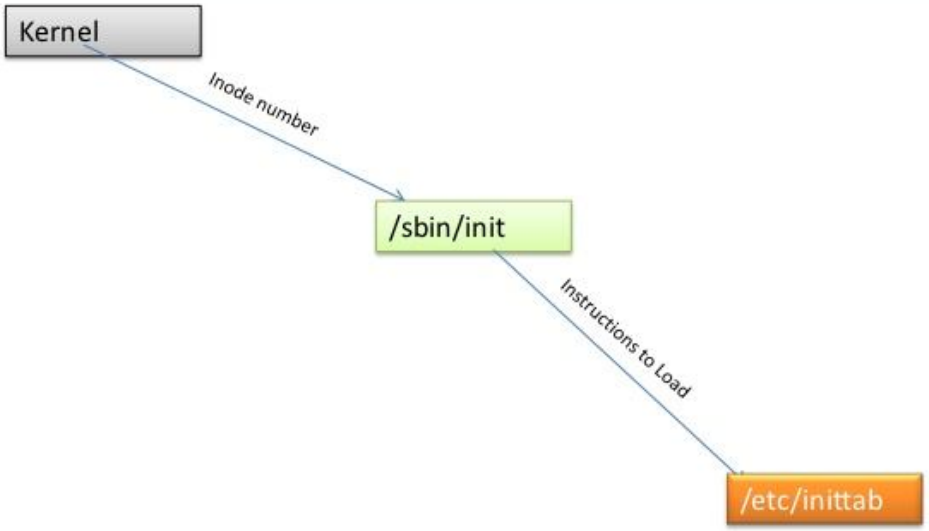

/boot/vimlinuz
CHS No.
(int 13 fn 2)
RAM(MEMORY)
1. BIOS
2. CMOS
3. MBR
4. LILO
5. Boot.b
6. Message
7. Map
8. Vimlinuz
9. ext3.0
10. jbd.0
BIOS
initrd.img
Gets Extracted into RD and Gets Loaded into memory
RAM DISK
1. Nash
2. Ext3 0
3. JBD 0
4. Linuxrc
At RAM DISK , first 'Nash' will be activated to run the script file ,linuxrc This Script file will now run functions of ext3 0 , JBD 0 and mounting function of '/' (slash) , this mounting is also called 'Sysroot'.
In linuxrc script 'Mounting of '/' (slash) ,looks something like this
#mount /dev/root / ro ------→ (read only)
XXXXXXXXXXXXX XX
END OF KERNEL LAND
XXXXXXXXXXXXXXXXXXXXXXX

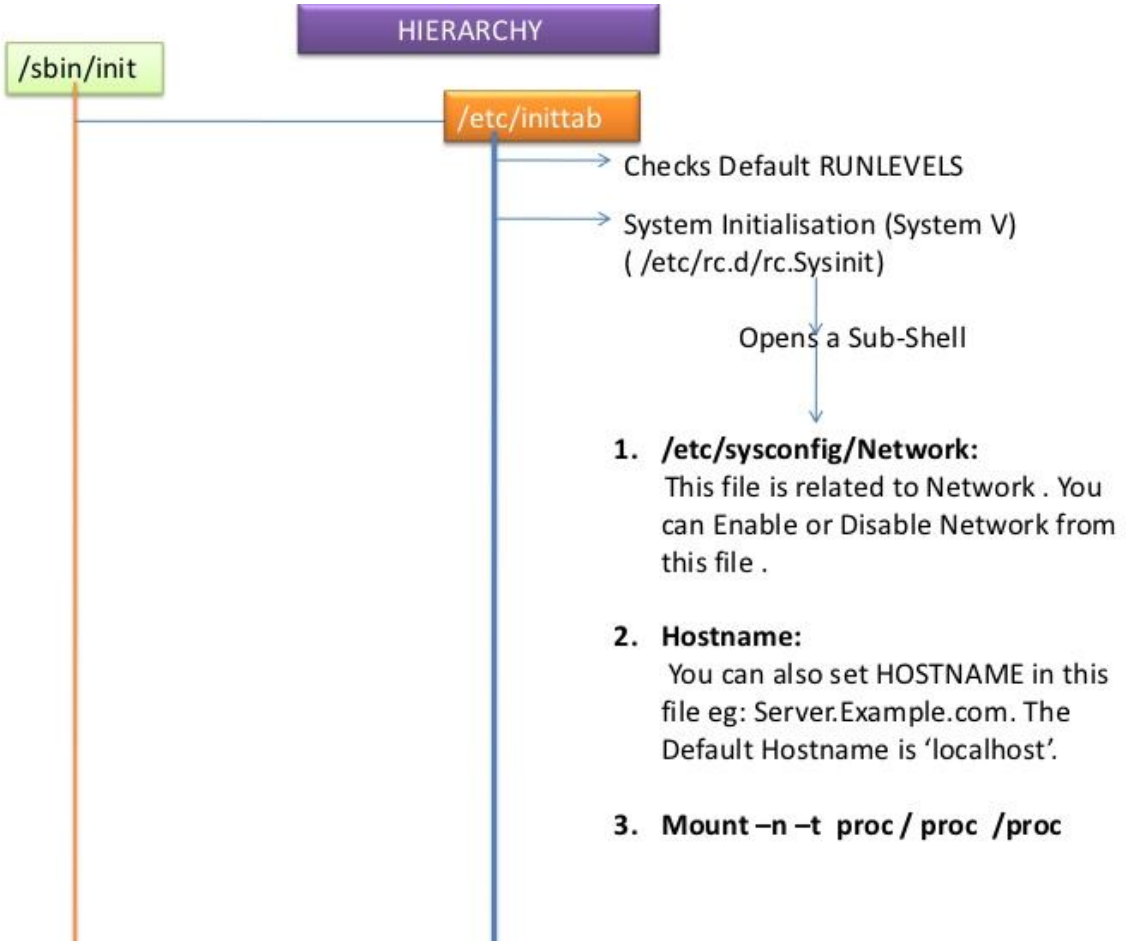

NASH
It's a minimal shell . Uses very low memory .
Nash will be activated first.
RUNS
LINUXRC
It's a Script file , having instructions to
load ext3.0 and jbd.0 and also
function to mount '/' (slash)
partition in ro (read only ).
#mount /dev/root / ro
Mounts '/'
partition
XXXXXXXXXXXXX XX
END OF KERNEL LAND
XXXXXXXXXXXXXXXXXXXXXXX

********** START OF USERLAND **********

As the '/' (slash) Partition is now mounted , Kernel now opens the First file with the 'Inode Number' . The File it opens is located at '/sbin/init' . Init is the very first process to start and hence it has the pid (process Id) Number as 1.

Init file is a Binary file (executable) and it has been given an Instructions to Load a file called /etc/inittab.




HIERARCHY
/sbin/init
/etc/inittab
Checks Default RUNLEVELS
System Initialisation (System V)
( /etc/rc.d/rc.Sysinit)
Opens a Sub-Shell
1. /etc/sysconfig/Network:
This file is related to Network . You can Enable or Disable Network from this file .
2. Hostname:
You can also set HOSTNAME in this file eg: Server.Example.com. The Default Hostname is 'localhost'.
3. Mount –n –t proc / proc /proc

4. **/etc/init.d/functions:**{global umask
global PATH, defines 17 shell functions
{ success,failure,passed,warning echo_success echo_failure
echo_passed, echo_warning ,killproc, pidofproc,pidfileofproc
action,checkpid,confirm, status,strstr,daemon }

5. **/etc/redhat-release:**
   You can change the Release name with this file .

6. **Press 'i' to enter interactive setup:**
   You can enter into 'Interactive' mode to customise your booting .

7. **/etc/sysconfig/clock:**
   This file contains UTC Time Zone.This file updates the file at /etc/localtime.

8. **/sbin/start_udev:**
   Udev stands for Universal Device . It is a very critical file.Udev initialises 'dev' files under /dev directory. You can also change Owner and root of device driver files

9. **/etc/sysconfig/init :**
   You can the set this file's 'Graphical' variable to 'Yes' or 'No' . This basically gives booting information in Graphics or console .

10. **/etc/sysctl.conf:**
    This file is for KERNEL tuning.

11. **/etc/sysconfig/keyboard:**
    You can Understand 'Control keys' with this file.

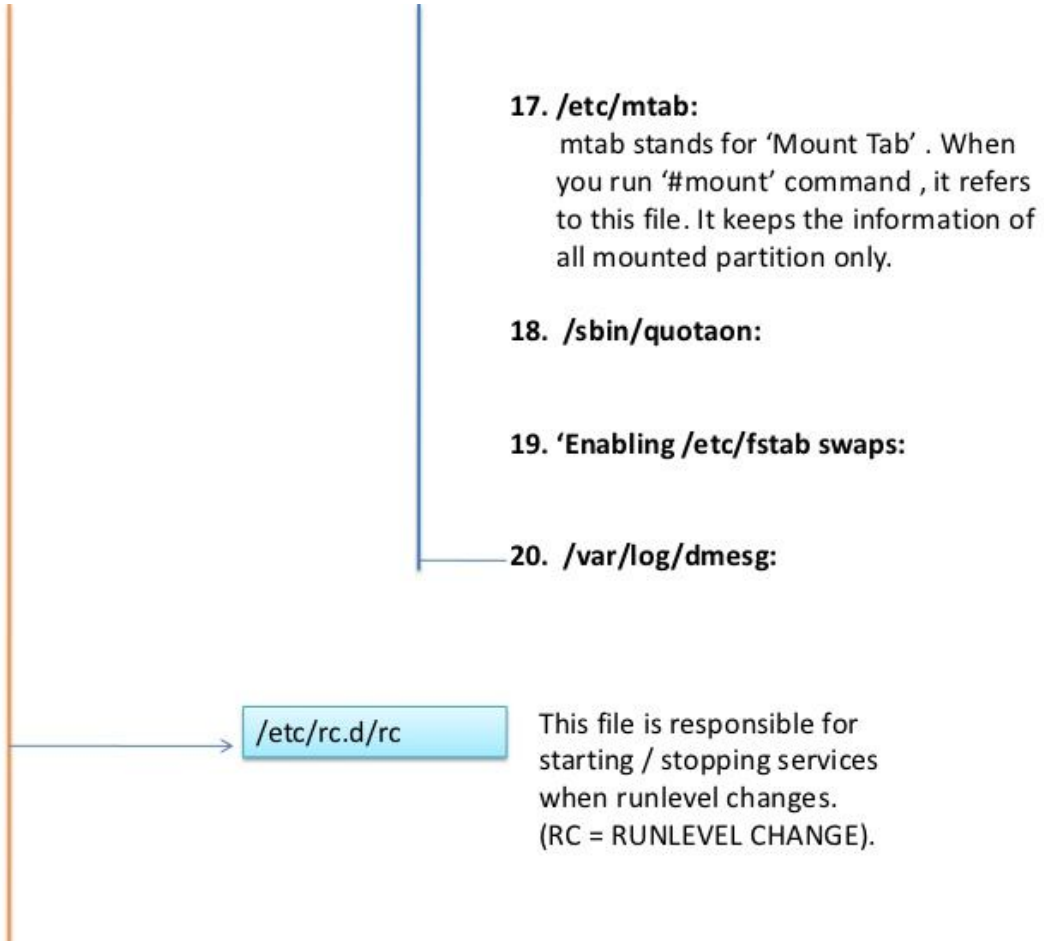
12. **/fastboot:**
    This file , if created in '/' (slash) partition , ensures that the 'fsck' (File system check) operation is skipped. This file will even skip the 'fsck' operation even if its mentioned in '/etc/fstab' file.

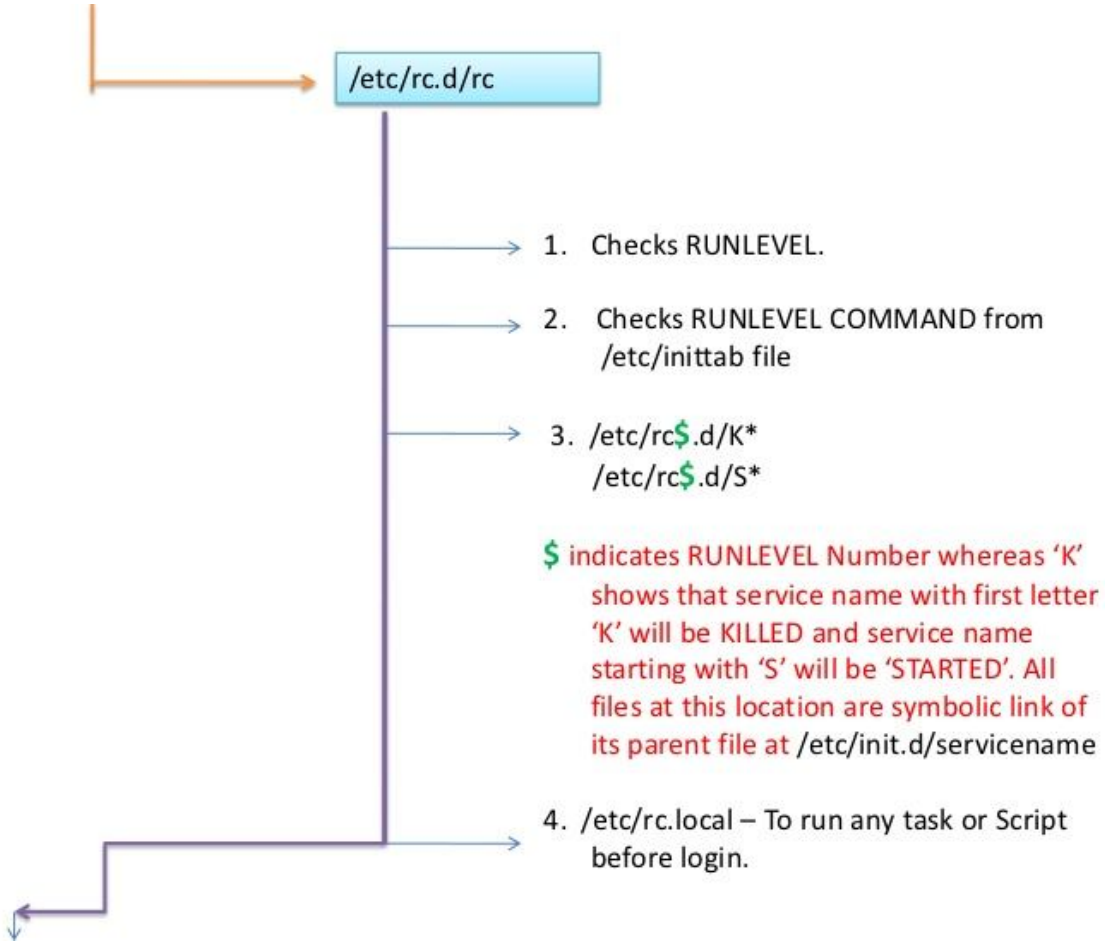
13. **/forcefsck :**
    This file will make sure , if created at '/' Slash Partition, the 'fsck' operation is performed . Even if its not mentioned in '/etc/fstab' file.

14. **/etc/sysconfig/readonly-root:**
    You can set the Entire Filesystems to Readonly = 'Yes' or 'No'

15. **/etc/rwtab:**
    This can be considered as an exception to '/etc/sysconfig/readonly-root' which means if you make the entire filesystem to read only and wanted to give exceptions to certain files/dir then you can make use of this file to give 'Read-Write' permission to them.

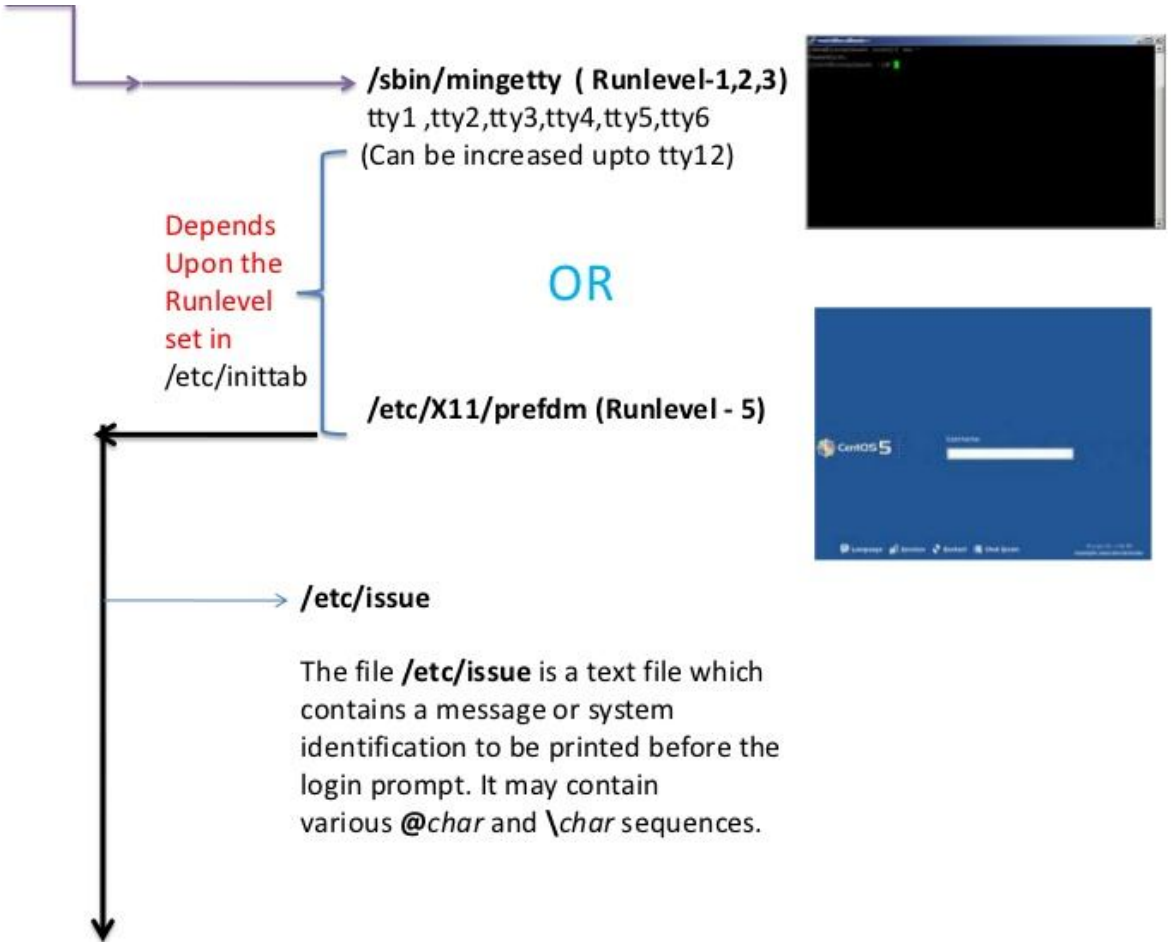
16. **/etc/fstab:** This file keeps the information of all mounted and umounted partitions.(Now hear / Is remounted with rw as it was in ro)

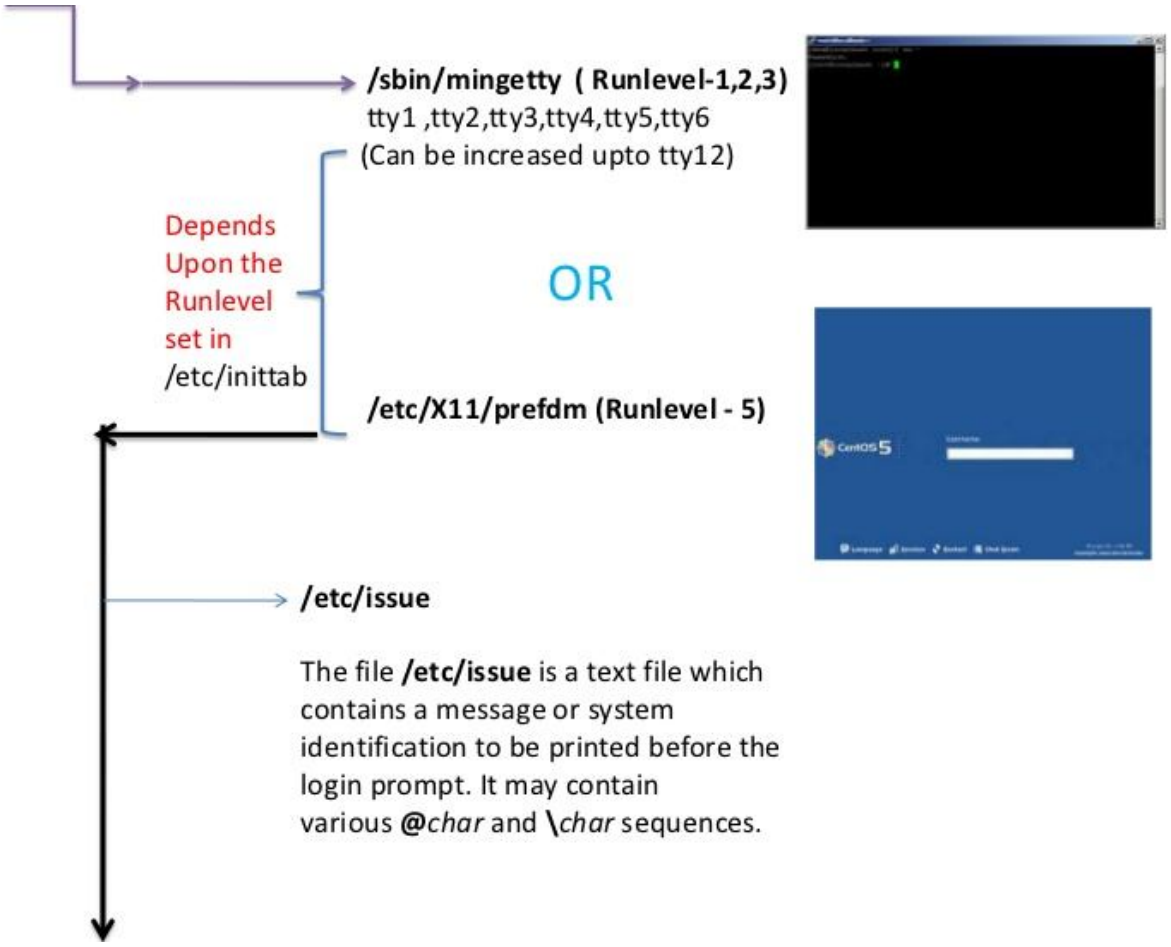

17. /etc/mtab:
mtab stands for 'Mount Tab' . When you run '#mount' command , it refers to this file. It keeps the information of all mounted partition only.
18. /sbin/quotaon:
19. 'Enabling /etc/fstab swaps:
20. /var/log/dmesg:
/etc/rc.d/rc
This file is responsible for starting / stopping services when runlevel changes. (RC = RUNLEVEL CHANGE).

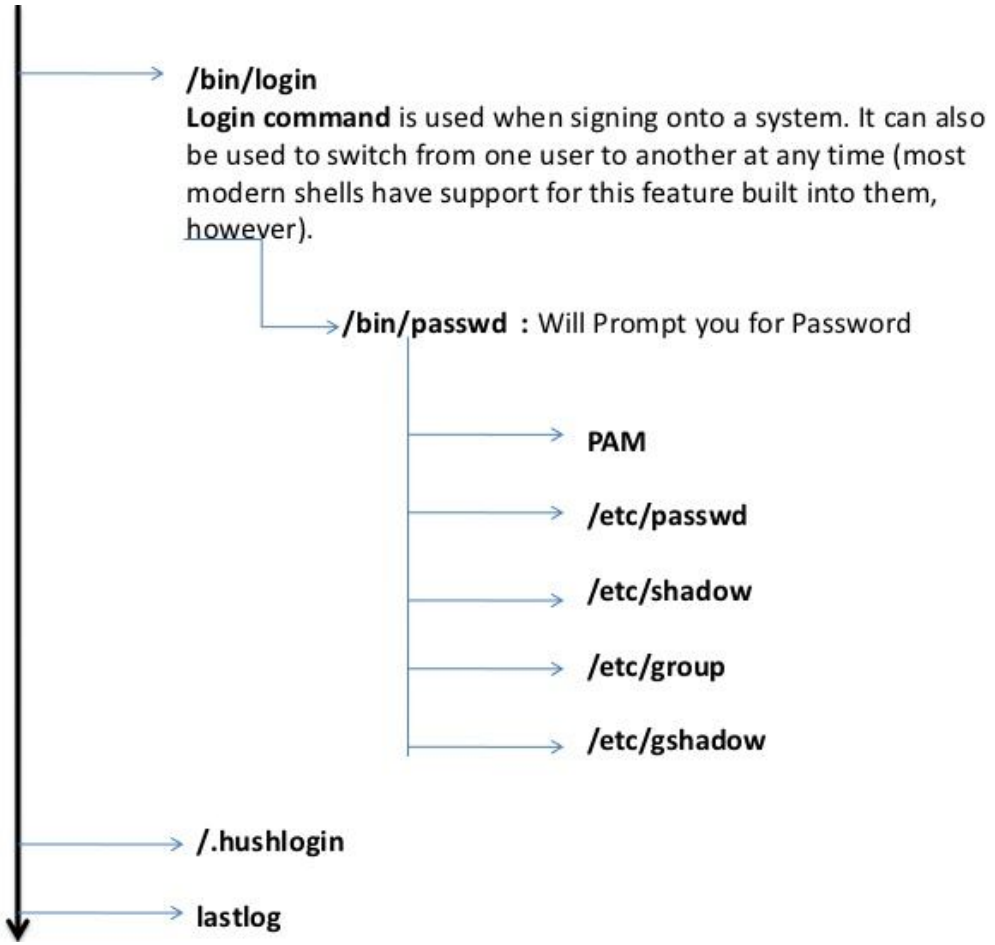

/etc/rc.d/rc
1. Checks RUNLEVEL.
2. Checks RUNLEVEL COMMAND from /etc/inittab file
3. /etc/rc$.d/K*
/etc/rc$.d/S*
$ indicates RUNLEVEL Number whereas 'K' shows that service name with first letter 'K' will be KILLED and service name starting with 'S' will be 'STARTED'. All files at this location are symbolic link of its parent file at /etc/init.d/servicename
4. /etc/rc.local – To run any task or Script before login.

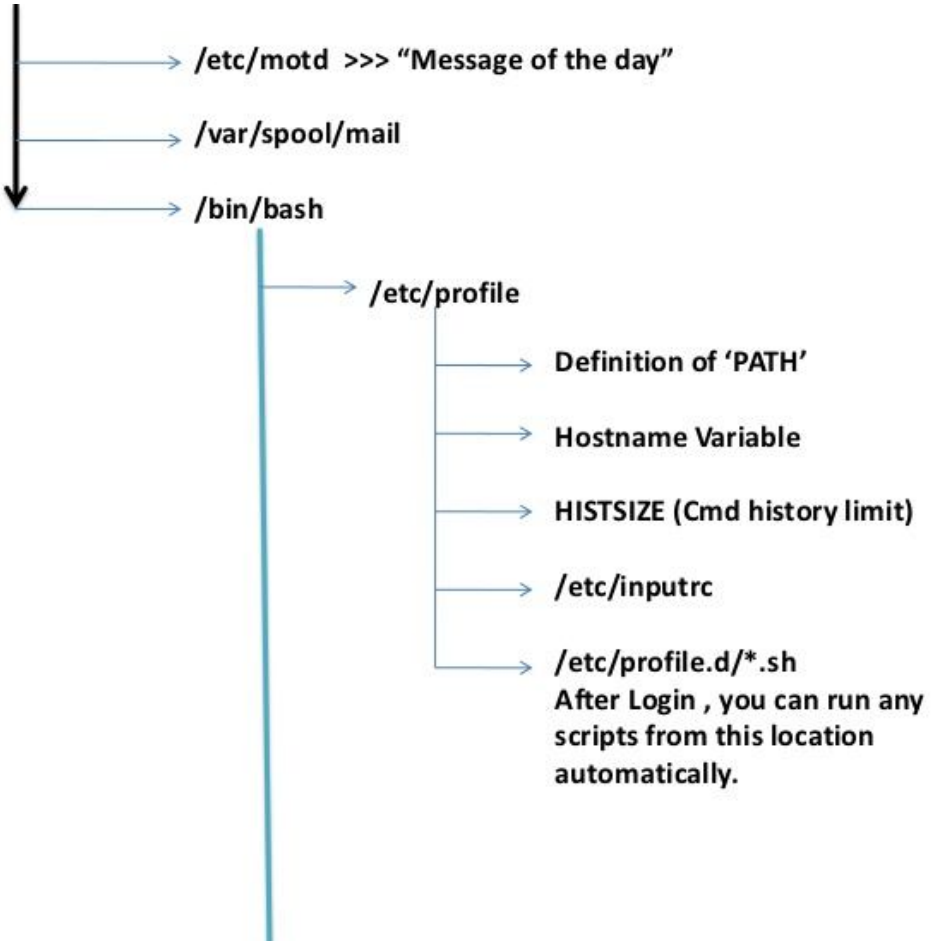
# Runlevels

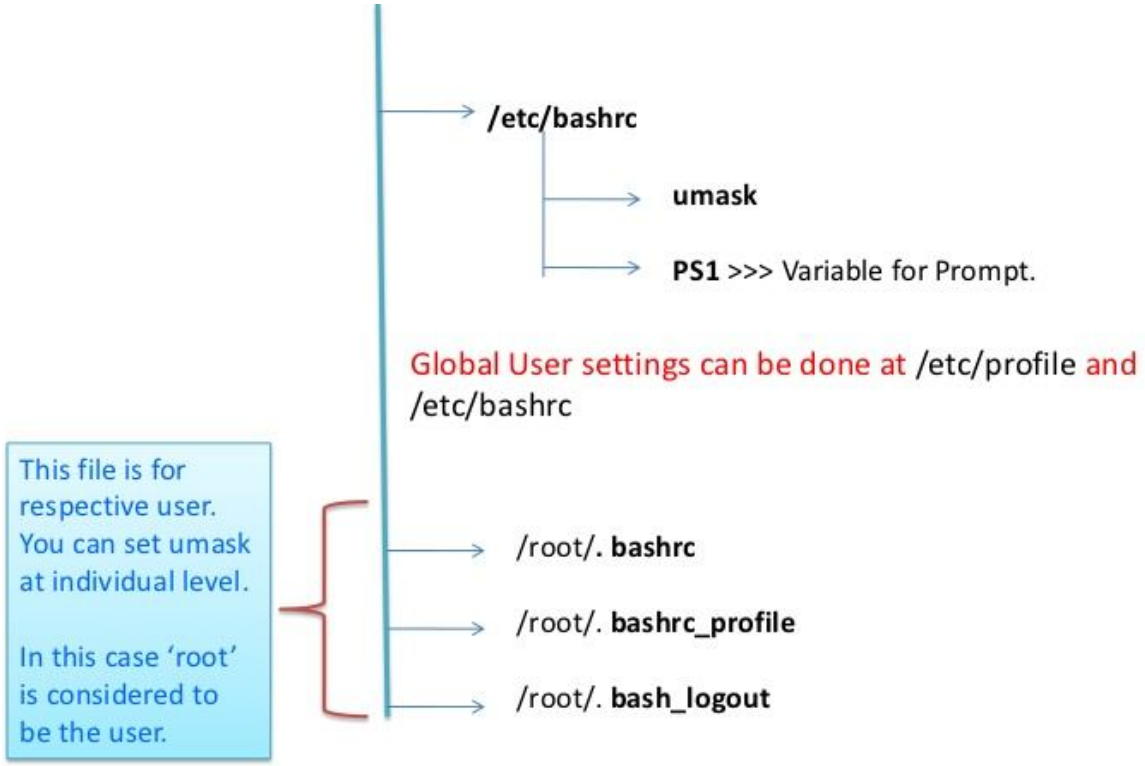
- A runlevel is a software configuration of the system which allows only a selected group of processes to exist
- The processes spawned by init for each of these runlevels are defined in the /etc/inittab file
- Init can be in one of seven runlevels: 0-6


/sbin/mingetty ( Runlevel-1,2,3)
tty1 ,tty2,tty3,tty4,tty5,tty6
(Can be increased upto tty12)
Depends Upon the Runlevel set in /etc/inittab
OR
/etc/X11/prefdm (Runlevel - 5)
/etc/issue
The file /etc/issue is a text file which contains a message or system identification to be printed before the login prompt. It may contain various @char and \char sequences.


/sbin/mingetty ( Runlevel-1,2,3)
tty1 ,tty2,tty3,tty4,tty5,tty6
(Can be increased upto tty12)
Depends Upon the Runlevel set in /etc/inittab
OR
/etc/X11/prefdm (Runlevel - 5)
/etc/issue
The file /etc/issue is a text file which contains a message or system identification to be printed before the login prompt. It may contain various @char and \char sequences.


/bin/login
Login command is used when signing onto a system. It can also be used to switch from one user to another at any time (most modern shells have support for this feature built into them, however).
/bin/passwd : Will Prompt you for Password
PAM
/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow
/.hushlogin
lastlog


/etc/motd >>> “Message of the day”
/var/spool/mail
/bin/bash
/etc/profile
Definition of ‘PATH’
Hostname Variable
HISTSIZE (Cmd history limit)
/etc/inputrc
/etc/profile.d/*.sh
After Login , you can run any scripts from this location automatically.


/etc/bashrc
umask
PS1 >>> Variable for Prompt.
Global User settings can be done at /etc/profile and /etc/bashrc
This file is for respective user. You can set umask at individual level.
In this case 'root' is considered to be the user.
/root/. bashrc
/root/. bashrc_profile
/root/. bash_logout

# END OF BOOT PROCESS

THANK YOU