# Linux

Cloudlytics

## Linux Facts

- About **8,50,000** Linux running Android phones are activated every single day
- Nearly **7,00,000** TV’s are sold every day which runs on Linux

## Linux terminology

- Kernel

  It is the brain of OS

  It controls the hardware and makes hardware interact with the applications

## Linux terminology

- **Filesystem**

  It is a method of storing and organising files

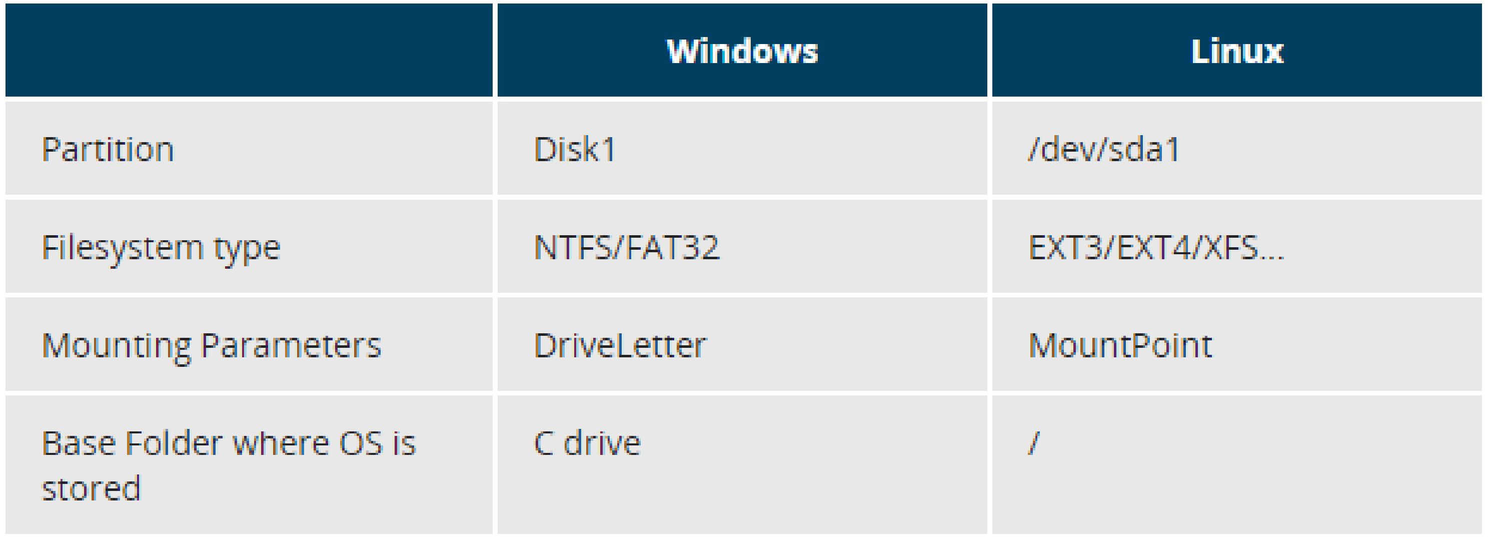
- **Different Types of Filesystems Supported by Linux:**
  - Conventional disk filesystems: `ext2`, `ext3`, `ext4`, `XFS`, `Btrfs`, `JFS`, `NTFS`, etc.
  - Flash storage filesystems: `ubifs`, `JFFS2`, `YAFFS`, etc.
  - Database filesystems
  - Special purpose filesystems: `procfs`, `sysfs`, `tmpfs`, `debugfs`, etc.

## Linux terminology

- **Partitions and Filesystems**

A **partition** is a logical part of the disk, whereas a **filesystem** is a method of storing/finding files on a hard disk (usually in a partition).

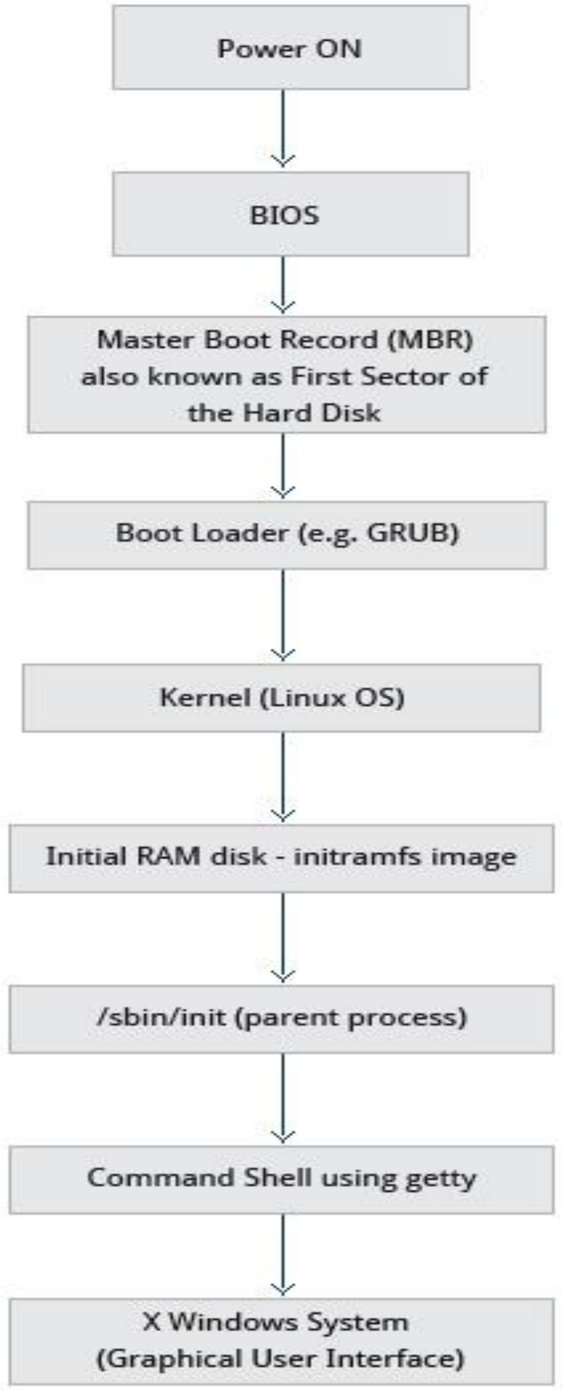
A comparison between filesystems in Windows and Linux



| | Windows | Linux |
|---|---|---|
| Partition | Disk1 | /dev/sda1 |
| Filesystem type | NTFS/FAT32 | EXT3/EXT4/XFS... |
| Mounting Parameters | DriveLetter | MountPoint |
| Base Folder where OS is stored | C drive | / |

## The Boot Process

The Linux **boot process** is the procedure for initializing the system. It consists of everything that happens from when the computer power is first switched on until the user interface is fully operational.

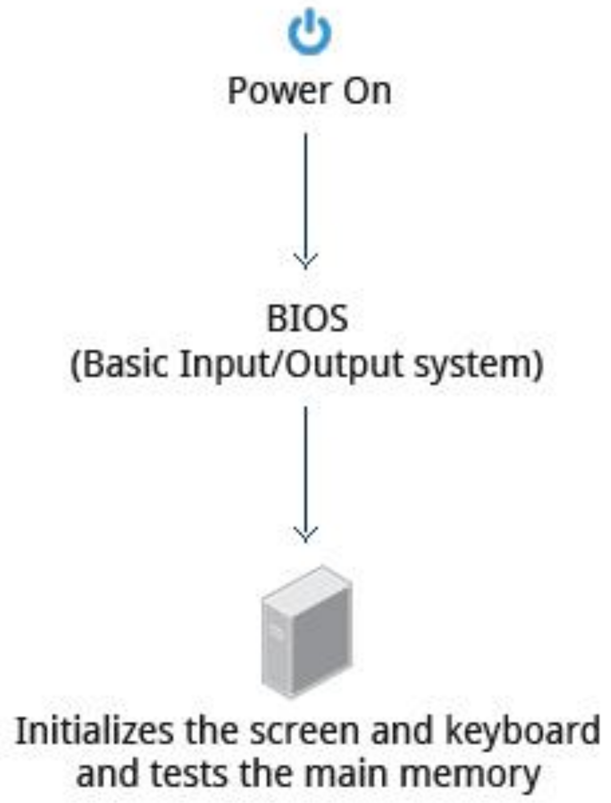

Power ON
BIOS
Master Boot Record (MBR) also known as First Sector of the Hard Disk
Boot Loader (e.g. GRUB)
Kernel (Linux OS)
Initial RAM disk - initramfs image
/sbin/init (parent process)
Command Shell using getty
X Windows System (Graphical User Interface)

## BIOS – The first step

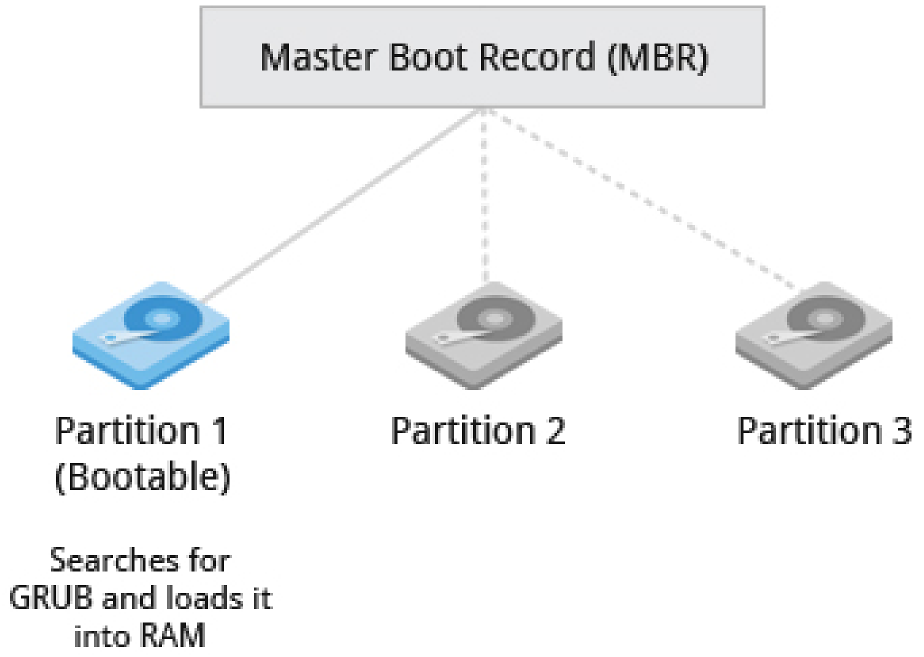
Starting a Linux system involves a number of steps.

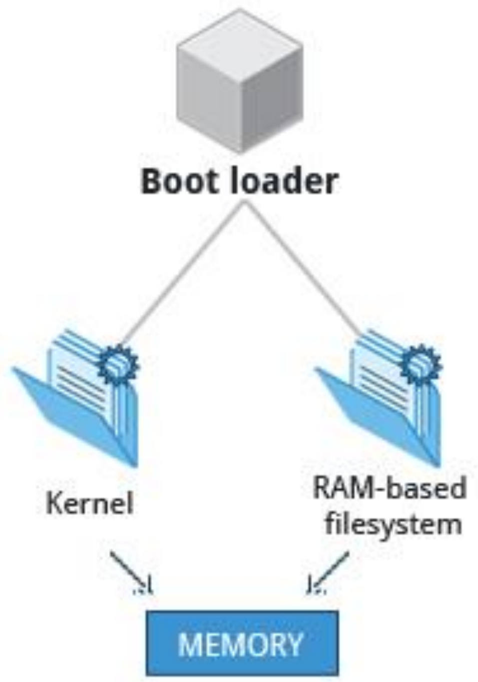
When the computer is powered on, the **Basic Input/output System** (**BIOS**) initializes the hardware, including the screen and keyboard, and tests the main memory.

This process is also called **POST** (**Power On Self Test**).

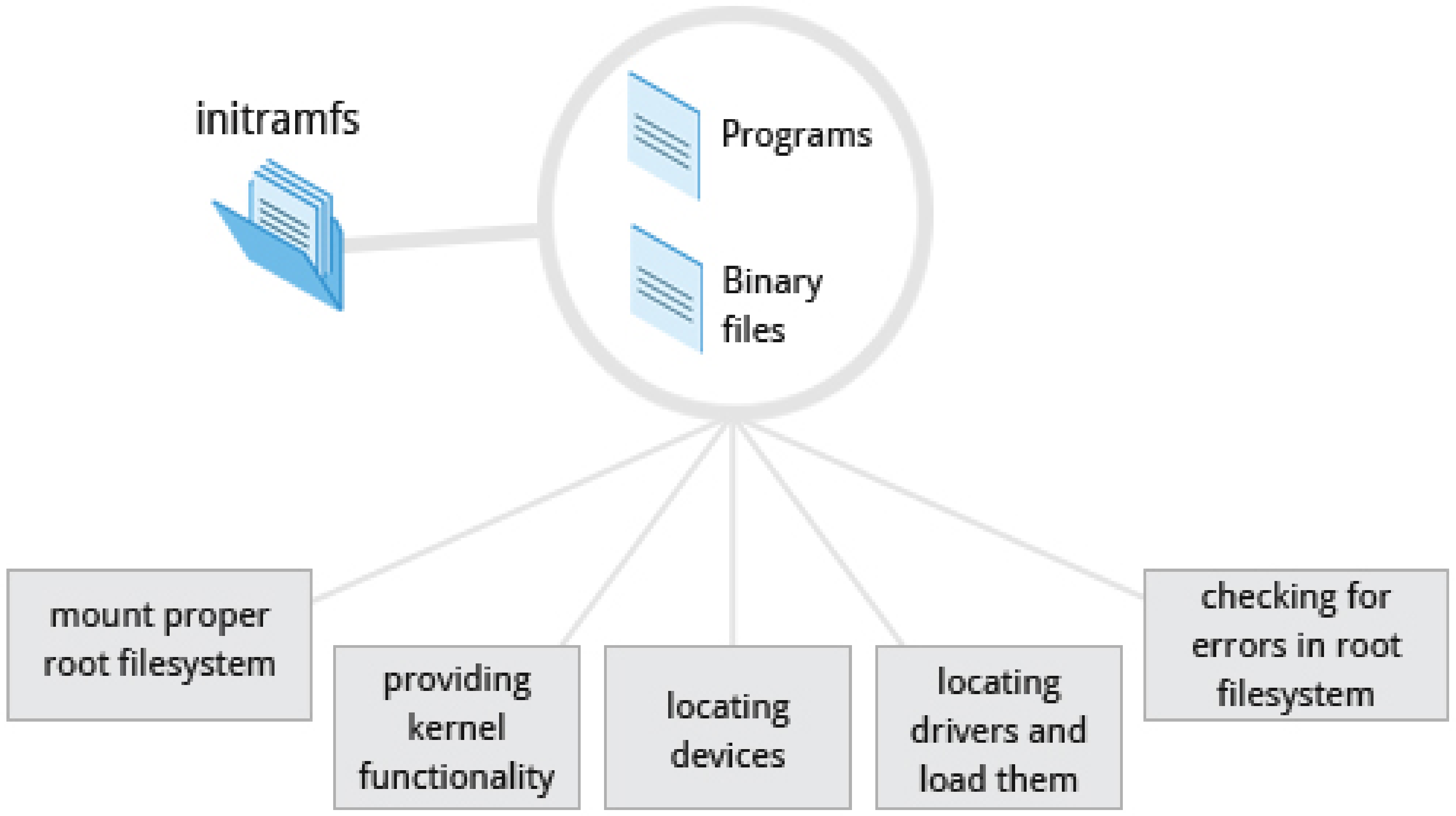

Power On
BIOS
(Basic Input/Output system)
Initializes the screen and keyboard
and tests the main memory

# Master Boot Records (MBR) and Boot Loader

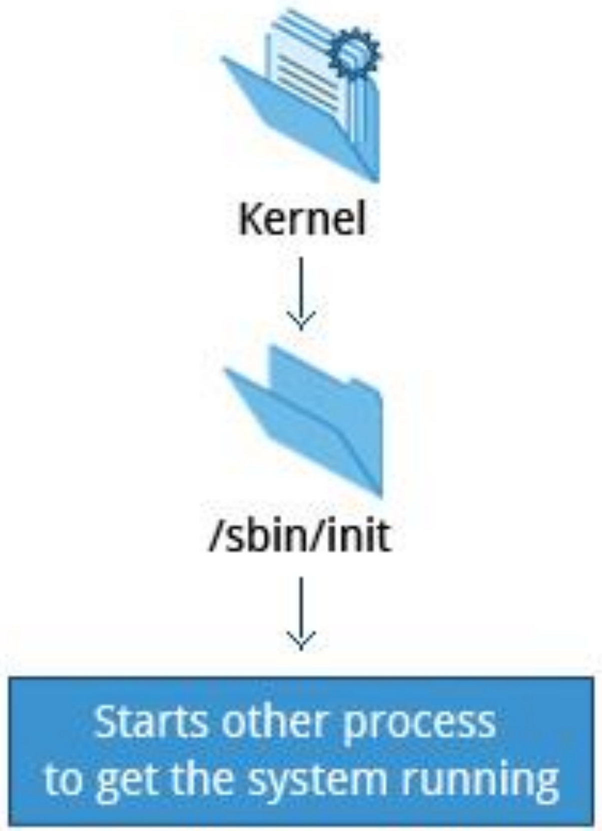


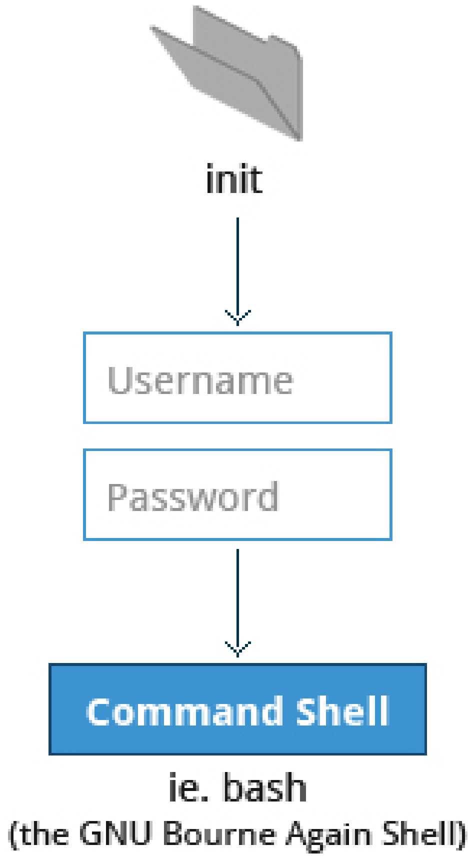


## The Initial RAM Disk

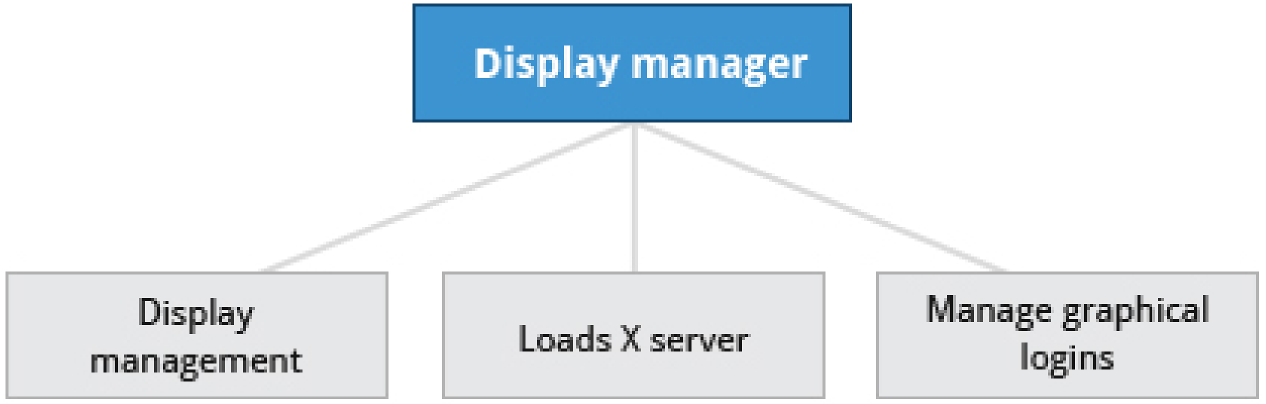

initramfs
Programs
Binary files
mount proper root filesystem
providing kernel functionality
locating devices
locating drivers and load them
checking for errors in root filesystem

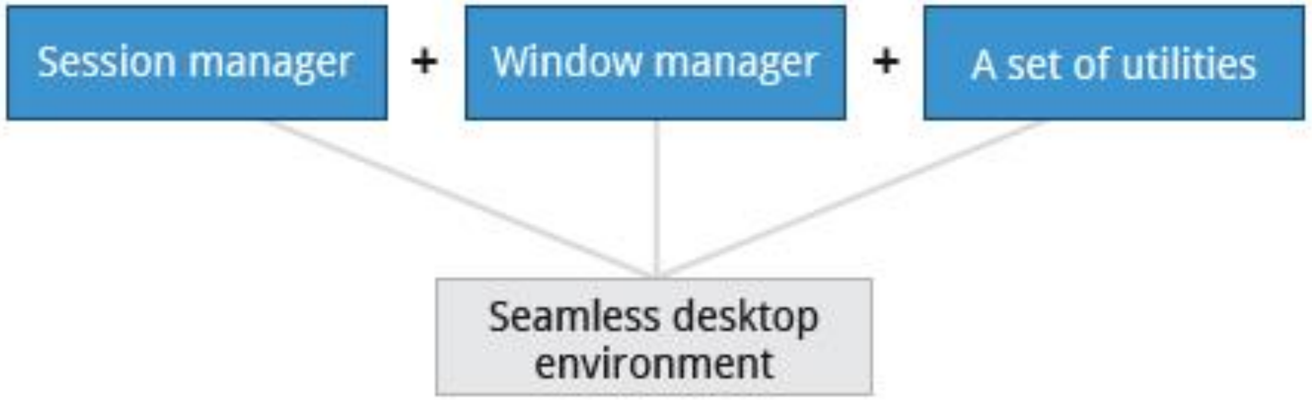
# /sbin/init and Services





# X-Window System

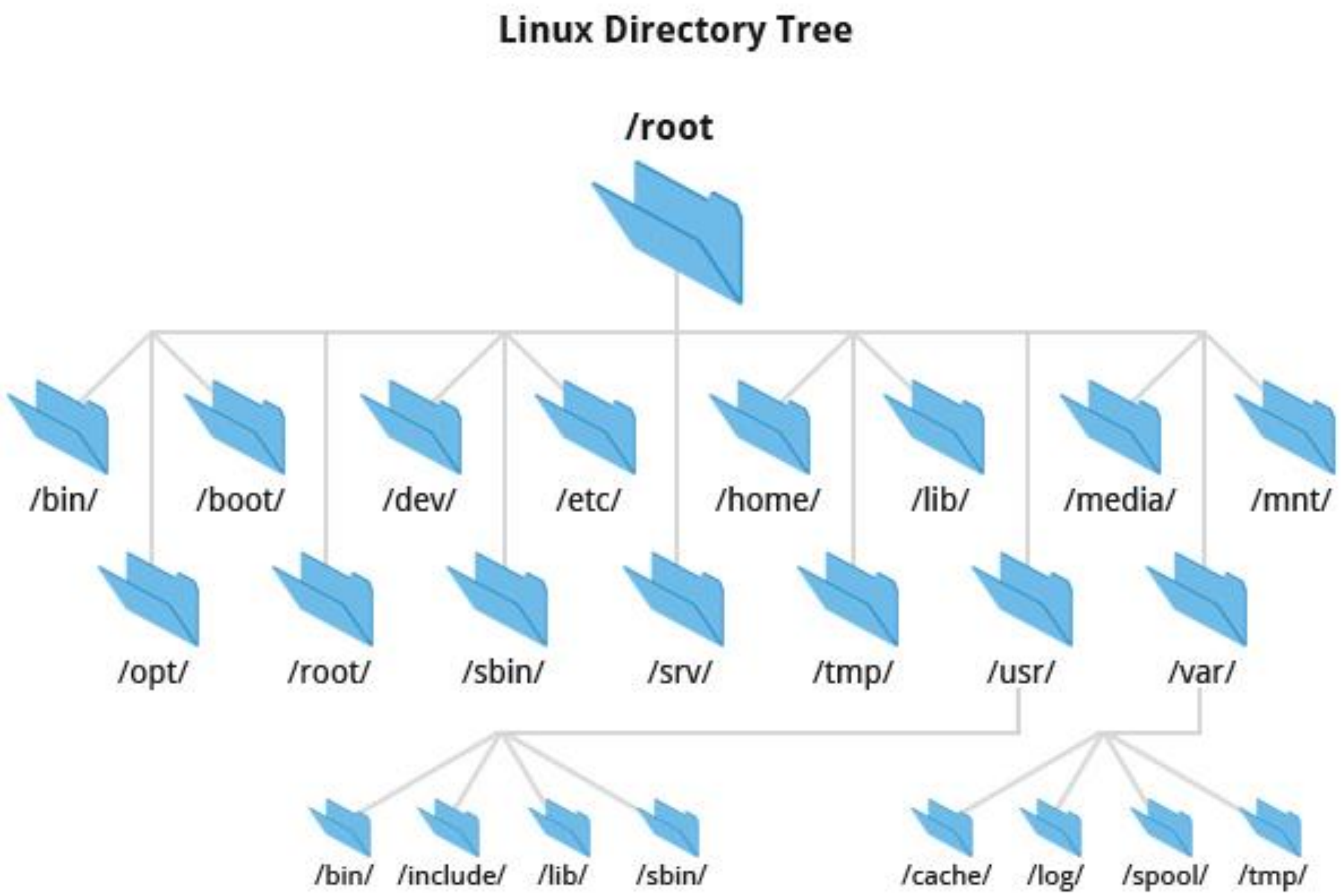




# Linux file systems

Understanding Root

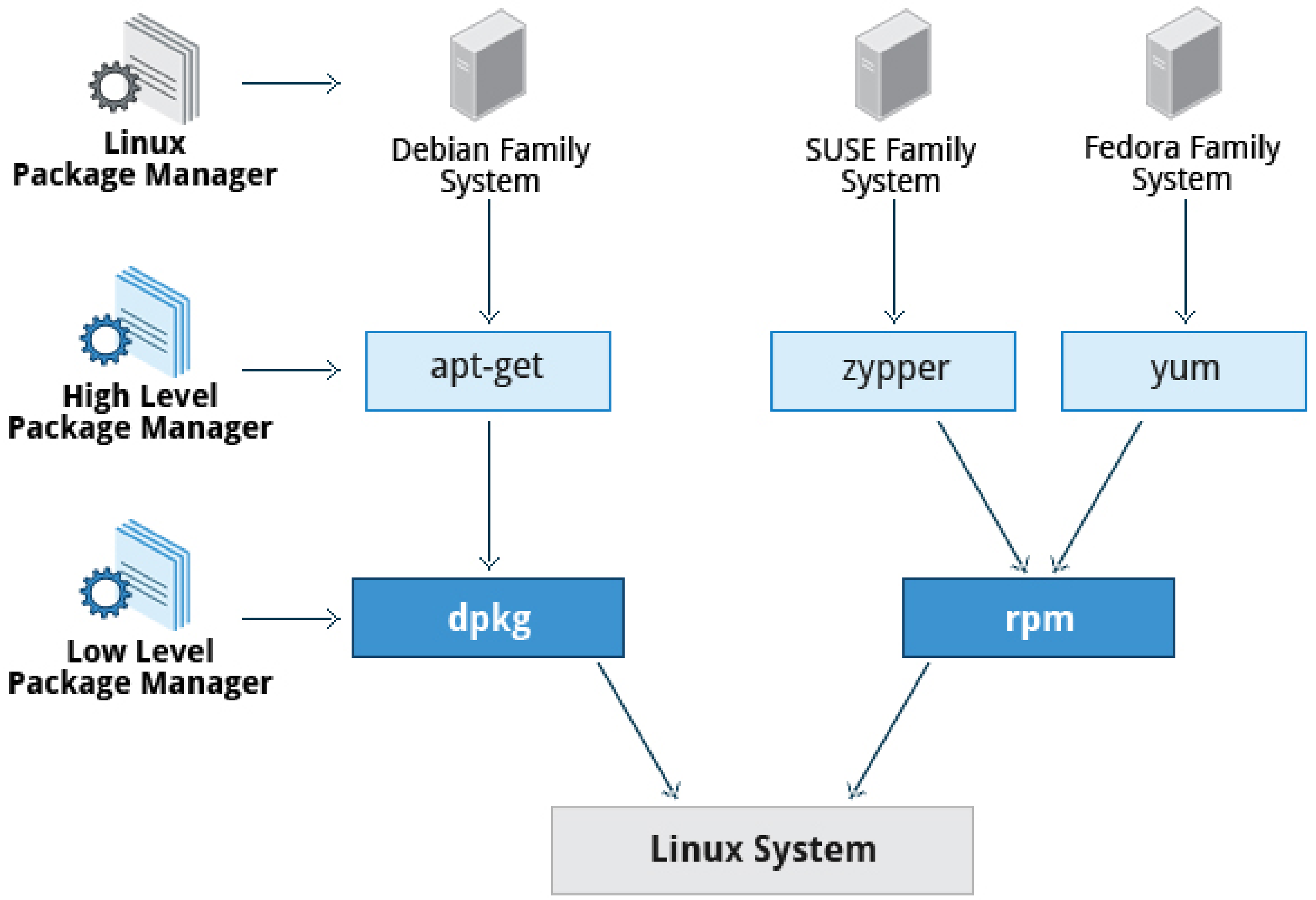
- / : root as hierarchy
- root : root as user
- /root : root directory


Linux Directory Tree
/root
/bin/
/boot/
/dev/
/etc/
/home/
/lib/
/media/
/mnt/
/opt/
/root/
/sbin/
/srv/
/tmp/
/usr/
/var/
/bin/
/include/
/lib/
/sbin/
/cache/
/log/
/spool/
/tmp/

- `/bin` directory contains executable binaries
- The `/dev` directory contains **device nodes**
- The `/var` directory contains files that are expected to change in size and content as the system is running (**var**stands for **variable**) such as the entries in the following directories:
  - System log files: `/var/log`
  - Packages and database files: `/var/lib`
  - Print queues: `/var/spool`
  - Temp files: `/var/tmp`
- The `/etc` directory is the home for system configuration files.
- The `/boot` directory contains the few essential files needed to boot the system.

| Directory name | Usage |
|---|---|
| `/opt` | Optional application software packages. |
| `/sys` | Virtual pseudo-filesystem giving information about the system and the hardware. Can be used to alter system parameters and for debugging purposes. |
| `/srv` | Site-specific data served up by the system. Seldom used. |
| `/tmp` | Temporary files; on some distributions erased across a reboot and/or may actually be a ramdisk in memory. |
| `/usr` | Multi-user applications, utilities and data. |

# Package Management Systems on Linux



## Command Line Operations

Most input lines entered at the shell prompt have three basic elements:

- Command
- Options
- Arguments

The **command** is the name of the program you are executing. It may be followed by one or more **options** that modify what the command may do. **Arguments** specify on what the command will operate on.

## Locating applications

- $which

  Eg : $which diff

- $whereis

  Eg : $whereis diff

- $locate

  Eg : $locate diff

## Viewing files

- $cat
- $tac
- $tail
- $head
- $touch
- $file <name>

- $mkdir – create a directory

- $rmdir – remove directory*
- $mv – rename a file

- $rm
- $rm -f
- $rm -i
- $rm -rf

## User operations

- $who : list of users currently logged in

  $who –a : detailed info

- $whoami :  current user

- $adduser : adding a user
- $deluser : deleting a user
- $usermod : modify user account
- $addgroup : add a group
- $id <username> : gives details about user

- $adduser <un> --ingroup <gn> :
- $addgroup <un> <gn> : adding existing user to a group
- $usermod <un> -g <gn> : modiy group of user
- $usermod <un> --gid <gid> : modify gid of user

## File ownership

- **Chown** : Used to change user ownership of a file or directory

- **Chgrp** : Used to change group ownership

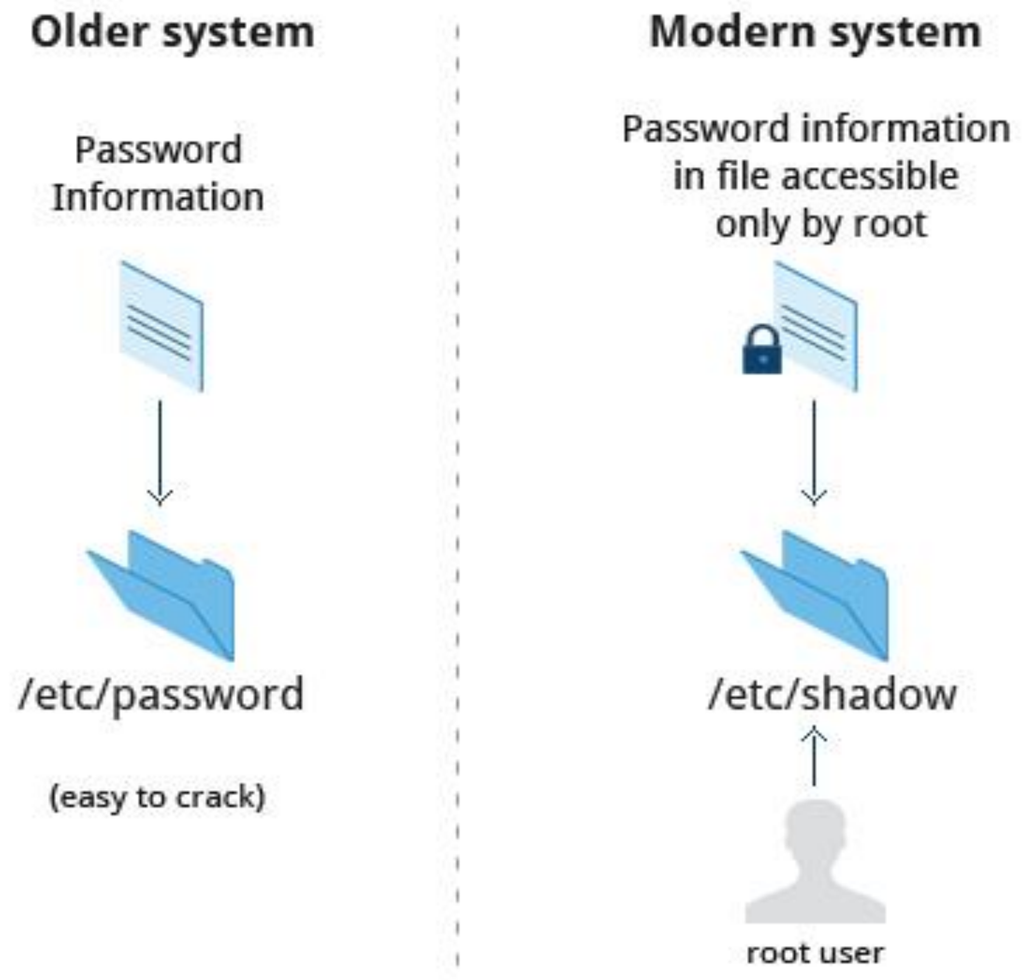
  Eg : $chgrp <username/groupname> <filename>

## File Permission Modes and chmod

- Files have three kinds of permissions: read (r), write (w), execute (x). `Rwx.` These permissions affect three groups of owners: user/owner (u), group (g), and others (o).

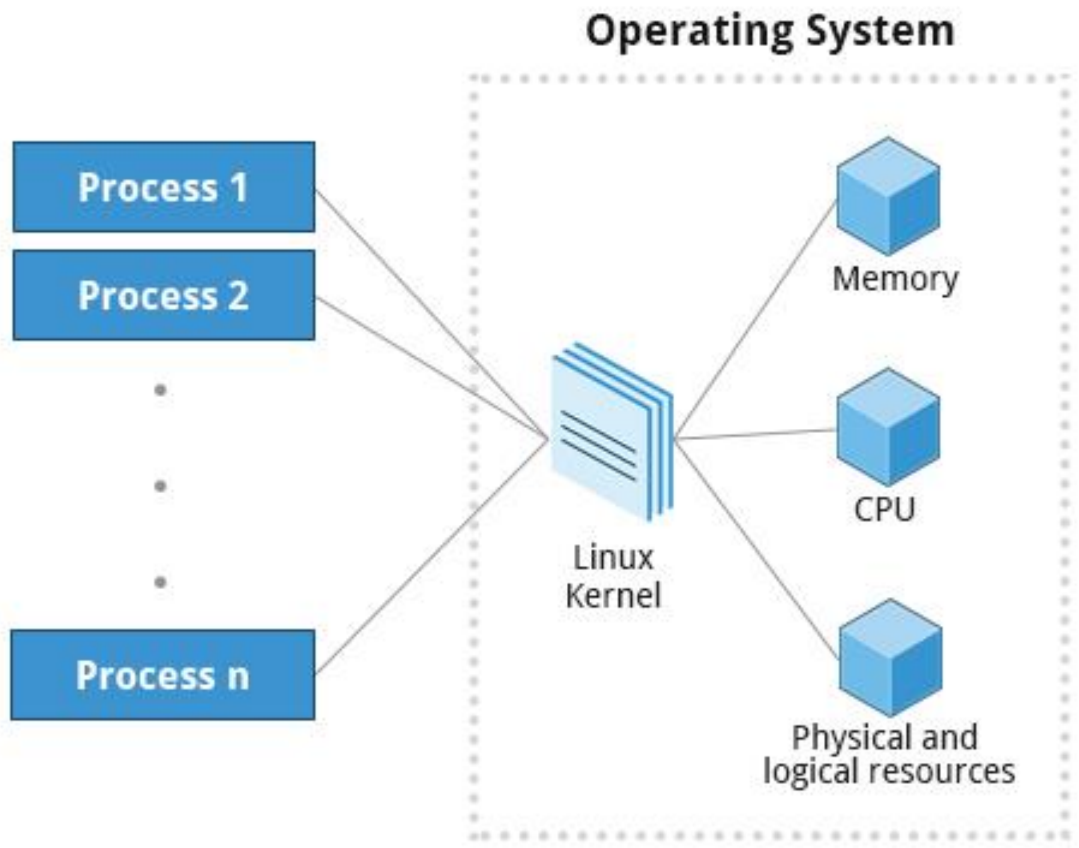
- Numbers for permissions 4 2 1

## How passwords are stored



## Process

- **A process** is simply an instance of one or more related **tasks (threads)** executing on your computer.



## Commands related to process

- ps
- Ps –u <username>
- Ps -ef
- Ps –eLf
- Pstree

## Monitoring process

- top
- Htop
- Commands to alter top

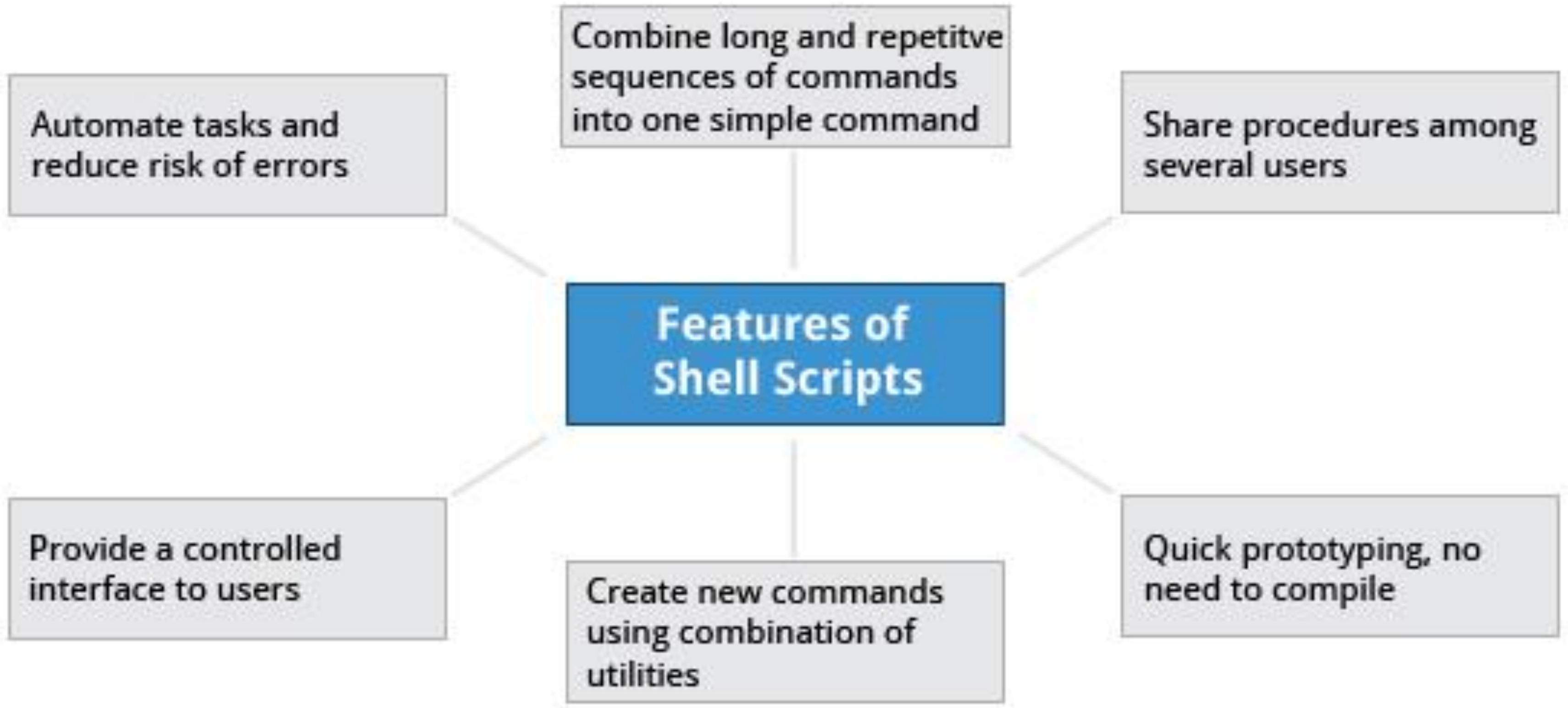
| Command | Output |
|---|---|
| `t` | Display or hide summary information (rows 2 and 3) |
| `m` | Display or hide memory information (rows 4 and 5) |
| `A` | Sort the process list by top resource consumers |
| `r` | Renice (change the priority of) a specific processes |
| `k` | Kill a specific process |
| `f` | Enter the top configuration screen |
| `o` | Interactively select a new sort order in the process list |

## Monitoring Resource Usage

- Install sysstat
- iostat
- sar
  - Enable sysstat (/etc/default/sysstat)
  - View cronjob for sysstat (/etc/cron.d/sysstat)
  - Restart sysstat
  - Use sar, sar –b, sar –r
- Vmstat
- free

# Bash shell scripting

**Shell** – It is a command line interpreter that interprets the commands & instructs OS to perform necessary tasks ($cat /etc/shells : for checking shells)


Combine long and repetitve sequences of commands into one simple command
Automate tasks and reduce risk of errors
Share procedures among several users
Features of Shell Scripts
Provide a controlled interface to users
Create new commands using combination of utilities
Quick prototyping, no need to compile

- Simple bash script

```
$ cat > exscript.sh
  #!/bin/bash
  echo "HELLO"
  echo "WORLD“
```

- Run it as : $ bash exscript.sh

- To make script executable change its permissions :

- Now run it as : $./abc.sh

## Interactive example

```
#!/bin/bash
    # Interactive reading of variables
    echo "ENTER YOUR NAME"
    read sname
    # Display of variable values
    echo $sname
```

## Basic Syntax and Special Characters

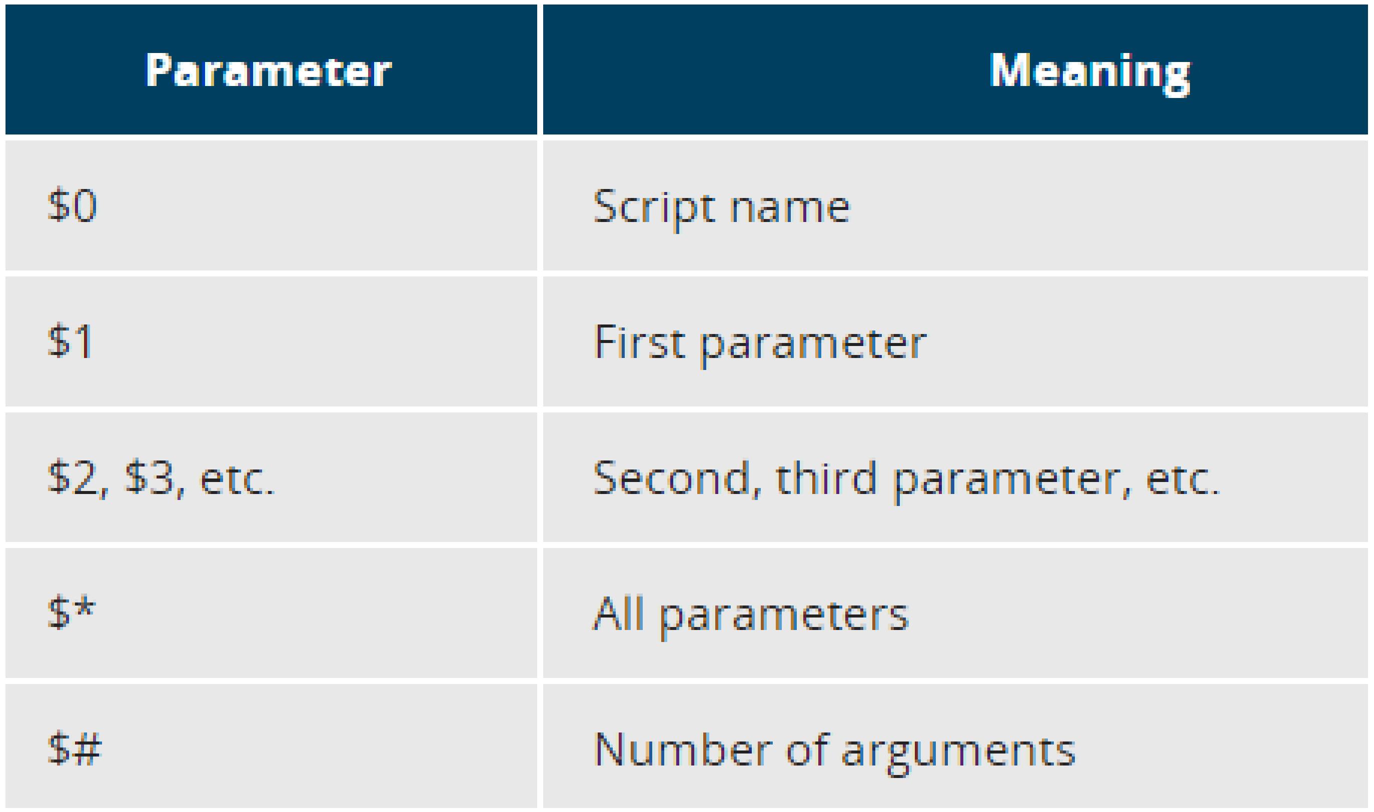
| Character | Description |
|---|---|
| `#` | Used to add a comment, **except** when used as `\#`, or as `#!` when starting a script |
| `\` | Used at the end of a line to indicate continuation on to the next line |
| `;` | Used to interpret what follows as a new command |
| `$` | Indicates what follows is a variable |

## Functions

Eg :

```
display () {
echo "This is a sample function"
}
```

## Script parameters



| Parameter | Meaning |
|---|---|
| $0 | Script name |
| $1 | First parameter |
| $2, $3, etc. | Second, third parameter, etc. |
| $* | All parameters |
| $# | Number of arguments |

- Eg:

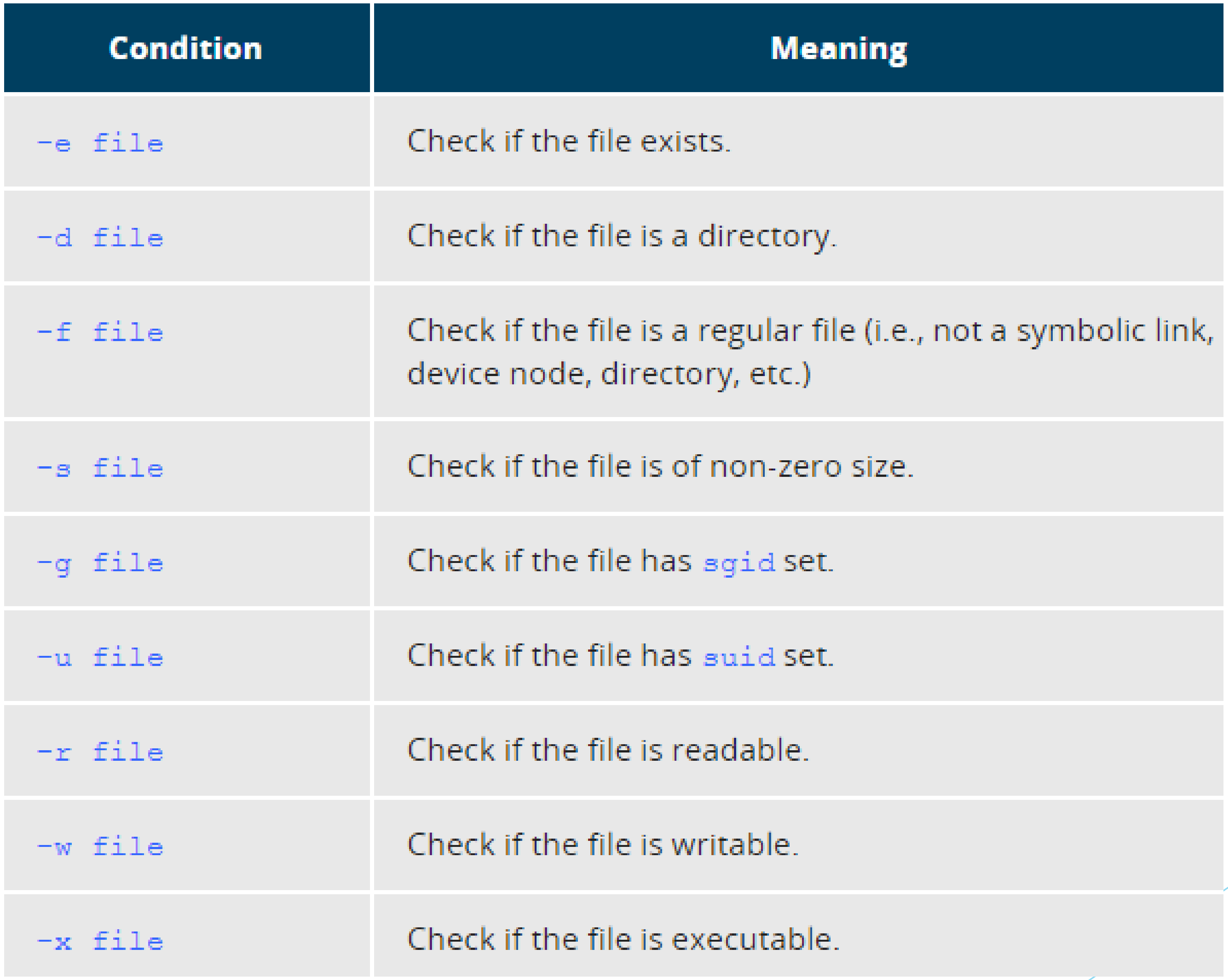
```
#!/bin/bash
echo “the name of script is $0”
echo “first parameter is $1”
echo “second parameter is $2”
```

## If statement

-
  ```
  if condition
  then
        statements
  else
        statements
  fi
  ```

- Eg:

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



| Condition | Meaning |
|---|---|
| `-e file` | Check if the file exists. |
| `-d file` | Check if the file is a directory. |
| `-f file` | Check if the file is a regular file (i.e., not a symbolic link, device node, directory, etc.) |
| `-s file` | Check if the file is of non-zero size. |
| `-g file` | Check if the file has `sgid` set. |
| `-u file` | Check if the file has `suid` set. |
| `-r file` | Check if the file is readable. |
| `-w file` | Check if the file is writable. |
| `-x file` | Check if the file is executable. |

| Operator | Meaning |
|---|---|
| -eq | Equal to |
| -ne | Not equal to |
| -gt | Greater than |
| -lt | Less than |
| -ge | Greater than or equal to |
| -le | Less than or equal to |

| Operator | Meaning |
|---|---|
| `[[ string1 > string2 ]]` | Compares the sorting order of string1 and string2. |
| `[[ string1 == string2 ]]` | Compares the characters in string1 with the characters in string2. |
| `myLen1=${#string1}` | Saves the length of string1 in the variable myLen1. |

| String Comparison | Description |
|---|---|
| Str1 = Str2 | Returns true if the strings are equal |
| Str1 != Str2 | Returns true if the strings are not equal |
| -n Str1 | Returns true if the string is not null |
| -z Str1 | Returns true if the string is null |

- String example

```
#!/bin/bash
echo “Enter the ipaddress”
read ip
if [ -n $ip ]; then
    ping –c 1 $ip
    if [ $? –eq 0 ]; then
        echo “Machine is giving a ping response”
    else
        echo “Machine is not giving a ping response”
    fi
else
echo “ip is empty”
fi
```

## Script Debuging

- **Debugging** helps you troubleshoot and resolve such errors, and is one of the most important tasks a system administrator performs.
- bash –x ./script_file
- It can debug only selected parts of a script (if desired) with:

```
set -x    # turns on debugging
...
set +x    # turns off debugging
```

## Cron

- Cron is a system daemon used to execute desired tasks (in the background) at designated times.
- A crontab file is a simple text file containing a list of commands meant to be run at specified times.
- To edit the crontab file enter : $crontab –e
- Each line has five time-and-date fields as : **minute (0-59), hour (0-23, 0 = midnight), day (1-31), month (1-12), weekday (0-6, 0 = Sunday)**
- Eg : 04 04 1 1 1 /somedir/somecommand

## Cron scripts

- #!/bin/bash

  echo “Cron ran successfully at : $(date)” >> /tmp/mybackup.log

- #!/bin/bash

  tar –czf /var/backup/mybup.tar.gz /home/quazi/

  echo “Backup performed successfully at : $(date)” >> /var/tmp/bup.log

| Keyboard Shortcut | Task |
|---|---|
| **CTRL-L** | Clears the screen |
| **CTRL-D** | Exits the current shell |
| **CTRL-Z** | Puts the current process into suspended background |
| **CTRL-C** | Kills the current process |
| **CTRL-H** | Works the same as backspace |
| **CTRL-A** | Goes to the beginning of the line |
| **CTRL-W** | Deletes the word before the cursor |
| **CTRL-U** | Deletes from beginning of line to cursor position |
| **CTRL-E** | Goes to the end of the line |
| **Tab** | Auto-completes files, directories, and binaries |

# Thank you

Cloudlytics