# lets learn

# basics of gnu / linux

## Major Topics Covered

- Linux File System Architecture
- Basics of Linux
- The Desktop XWindows
- Linux Command line Interface
- Redirection etc.

GNU/Linux

Operating System

**Note**: This document will be always be in work in progress, thus few topics will feel like not fully developed.

Mayank Johri, Meenakshi Johri, **Ver**: 0.0.1 

# Table of Contents

# An Introduction

**Operating System** provides a means of communication between computer and user. It is also responsible for the management and coordination of activities and the sharing of the resources of the computer that acts as a host for commuting applications on the machine.

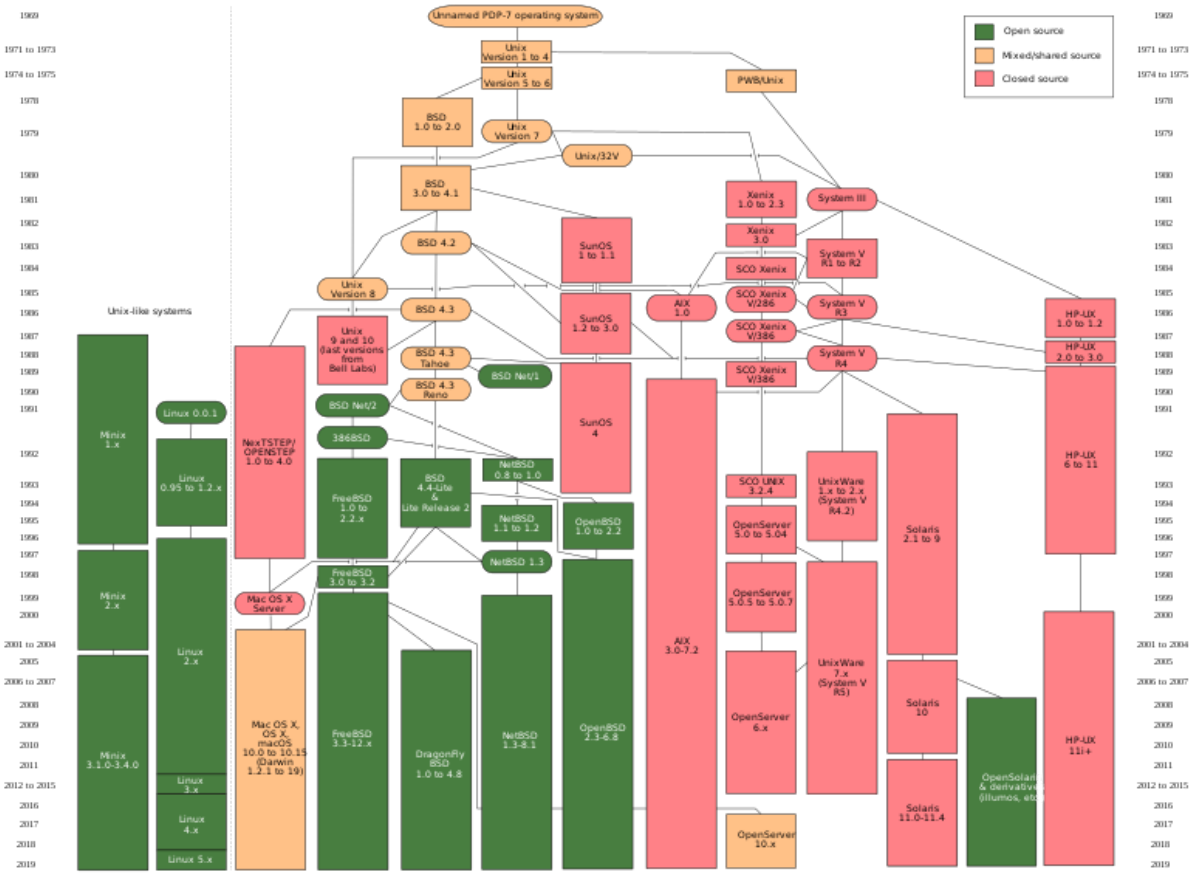
Main tasks for an operating system are as follows

1. Manage Hardware such as processors, harddisk, printers, mouse, keyboards etc
2. Host Applications
3. Manage user data

## Brief History of Unix

UNIX is a computer operating system, originally developed in 1969 by AT&T employees at BELL Labs. Now a days, it is used to for any OS which conforms to Unix standards (POSIX), which means that the core OS words as same as the original Unix operating system.

Below is the diagram showing the progress of key Unix and Unix like OS'es.




**(image from [https://en.wikipedia.org/wiki/Unix#/media/File:Unix_history-simple.svg](https://en.wikipedia.org/wiki/Unix#/media/File:Unix_history-simple.svg))**

During the late 1970s and early 1980s, Unix was highly popular in academic circles due to which many commercial startups started using it. Some of the popular ones are `Solaris`, `HP-UX` and `AIX`. Most of them were **certified Unix**.

The term "traditional Unix" is used to describe a Unix or an OS which follows the characteristics of either **Version 7 Unix** or **UNIX System V**. Most of the derivatives of Unix are called Unix-like or “Unix System-like”.

UNIX trademark is owned by *The Open Group*. UNIX technically refers to a computer operating system that conforms to **The Single UNIX Specification** which defines the names of, interfaces to, and behaviors of all mandatory UNIX operating system functions. This specification is a superset of an earlier series of specifications, such as the `P1003`, or **POSIX** specifications, developed by the IEEE (Institute of Electrical and Electronic Engineers).

As Unix and Unix derivatives have to get certification from "The Open Group" to be called Unix, few developers were not happy with the Unix ecosystem and what is represent. This led to the development of Linux and BSD.

# Linux

**Linux** kernel was developed by Linus Torvalds in 1991 at the University of Helsinki, with the help of UNIX programmers from across the world connected by Internet. It began as a hobby project heavily inspired by **Andy Tanenbaum’s** Minix, which is a small UNIX-like Operating System, used by students to learn basics of operating systems.

The intention in creating Linux was that the its kernel will not incorporate proprietary code and will contain nothing but freely distributable code.

Today Linux OS is refered as Unix-like computer OS based on the Linux kernel and contain Linux Kernel, GNU utilities and user applications. Thus many personals also refer it as **GNU/Linux**.

# Linux History

The history of Linux begin with this simple message posted by **Linus Torvalds** to the comp.os.minix newsgroup on August 25, 1991:

```
*From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)*

*Newsgroups: comp.os.minix*
*Subject: What would you like to see most in minix?*
*Summary: small poll for my new operating system*
*Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>*
*Date: 25 Aug 91 20:57:08 GMT*
*Organization: University of Helsinki*

*Hello everybody out there using minix -*
```

```
*I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing since
april, and is starting to get ready.I'd like any feedback on things people
like/dislike in minix, as my OS resembles it somewhat (same physical layout of
the file-system(due to practical reasons) among other things). I've currently
ported bash(1.08) and gcc(1.40),and things seem to work.This implies that I'll
get something practical within a few months, andI'd like to know what features
most people would want. Any suggestions are welcome, but I won't promise I'll
implement them :-)*

*Linus (torvalds@kruuna.helsinki.fi)*
*PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is
NOT protable (uses 386 task switching etc), and it probably never will support
anything other than AT-harddisks, as that's all I have :-(.*
```

*Please note that Minux was developed by Andrew Tanenbaum and had license restrictions at that time which Linus was not comfortable with and to overcome those restriction he wrote Linux.*

# Linux Philosophy

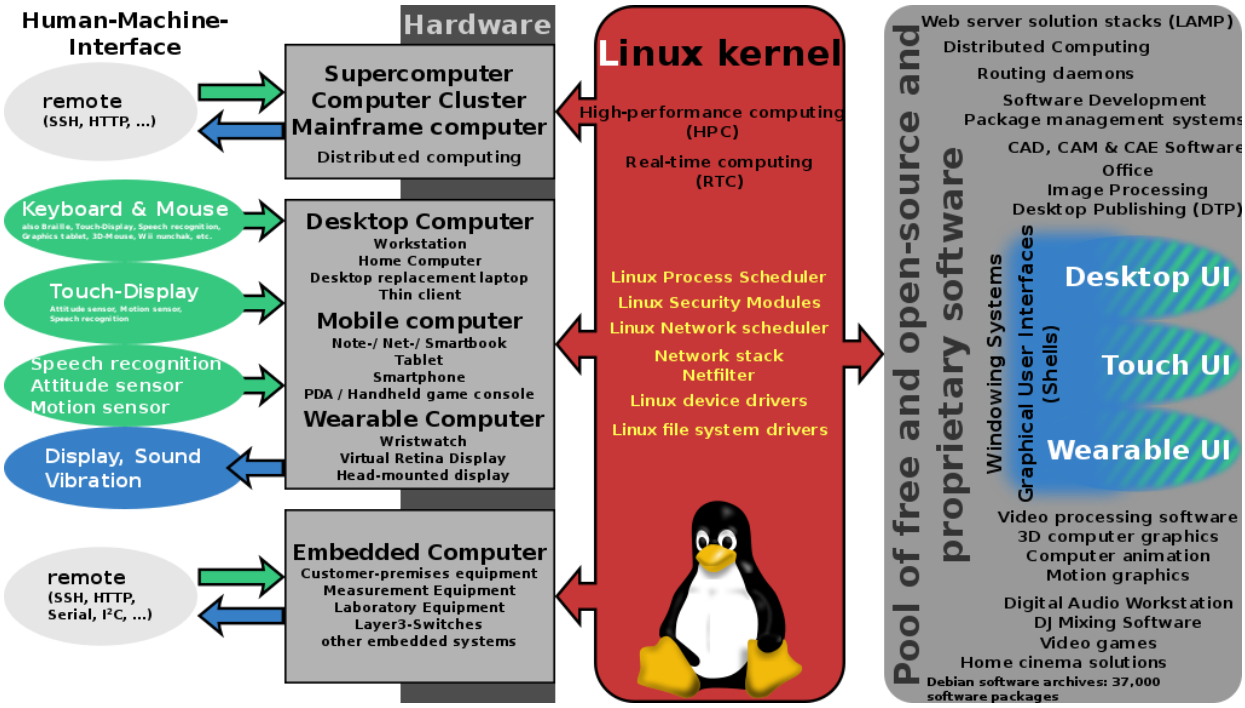
Linux and Unix both have similar philosophies regarding the design and programming of the OS.

- Simplicity
- Focus
- Portability
- Reusable components
- Filters
- Open File Formats
- Flexibility
- Input Output redirection

# Design

Linux is a Unix-like operating system and tries to be POSIX compliant. The Linux kernel takes care of process control, networking, device management, peripheral and file system access. Also the device drivers are integrated directly with the kernel.

Separate projects that interface with the kernel provide much of the system's higher-level functionality. The GNU userland is an important part of most Linux-based systems, providing the most common implementation of the C library, a popular shell, and many of the common Unix tools which carry out many basic operating system tasks. The graphical user interface (or GUI) used by most Linux systems is based on the X Window System.




**(Image from: [https://en.wikipedia.org/wiki/Linux_distribution#/media/File:Linux_kernel_ubiquity.svg](https://en.wikipedia.org/wiki/Linux_distribution#/media/File:Linux_kernel_ubiquity.svg))**

## Monolithic Kernel

A monolithic kernel is a kernel architecture where the entire operating system is working in the kernel space and alone as supervisor mode. In difference with other architectures, the monolithic kernel defines alone a high-level virtual interface over computer hardware, with a set of primitives or system calls to implement all operating system services such as process management, concurrency, and memory management itself and one or more device drivers as modules.

## GNU

Its name is a recursive acronym for “GNU's not Unix!” This name was chosen because GNU's design is Unix-like, but differs from Unix by being free software and containing no Unix code. Development of GNU was initiated by Richard Stallman and was the original focus of the Free Software Foundation (FSF).

GNU is developed by the GNU Project, and programs released under the auspices of the project are called GNU packages or GNU programs. The system's basic components include

- GNU Compiler Collection (GCC)
- GNU Binary Utilities (binutils)
- Bash shell
- GNU C library (glibc) and
- GNU Core Utilities (coreutils)

**GNU Hurd** (official GNU kernel), is under active development since 1990 and currently not all GNU components work with it. And arrival of Linux kernel also contributed in less need for its development. It is also not officially adopted by the GNU project.

Some third-party software are also included in GNU, namely the X.Org release of the X Window System and the TeX typesetting system. GNU applications and utilities are widely used by other *nix operating systems such as BSD variants, Solaris and Mac OS X.

## FSF

Free Software Foundation of India

The Free Software Foundation, founded in 1985, is dedicated to promoting computer users' right to use, study, copy, modify, and redistribute computer programs. The FSF promotes the development and use of free (as in freedom) software -- particularly the GNU operating system and its GNU/Linux variants -- and free documentation for free software. The FSF also helps to spread awareness of the ethical and political issues of freedom in the use of software, and its Web sites, located at fsf.org and gnu.org, are an important source of information about GNU/Linux. Donations to support the FSF's work can be made at http://donate.fsf.org. Its headquarters are in Boston, MA, USA.

## Linux Distributions

A Linux distribution (also called GNU/Linux distribution by some vendors and users) is a member of the family of Unix-like software distributions built on top of the Linux kernel. They consist of a large collection of software applications such as games, word processors, spreadsheets, internet based applications, media players, database applications ,etc. It will consist of the Linux kernel, a set of libraries and utilities from the GNU project, with graphics support from the X Window System. Distributions optimized for size may not contain X, and tend to use more compact alternatives to the GNU utilities such as Busybox, uClibc or dietlibc. There are currently over six hundred Linux distributions. Over three hundred of those are in active development, constantly being revised and improved.

## Major Linux Distributions

- RedHat
- Ubuntu
- Debian
- SUSE Linux
- Slackware
- PCLinux OS
- Mint Linux OS
- Gentoo

We can club the distributions based on multiple factors such as package type, intended target, intended usage etc.

## Package

Package is a mechanism using with Linux maintain its application and default configurations. The major package formats are

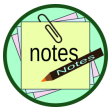
- deb (Debian package management)
- RPM (RedHat Package Management)
- PKG (used by Arch Linux etc)
- TXZ (used by Slackware)

## Derivatives

Many teams/users customise (by changing various aspects of the OS such as packages, desktop customisation, ) their own version of OS using existing Linux Operating Systems and release it. These are called derivatives of existing OS'es. Most notable derivatives are Ubuntu -> Debian, CentOS -> RedHat.

For detailed please visit https://en.wikipedia.org/wiki/Linux_distribution and 100 most popular open source distribution list can be viewed at www.distrowatch.com along with others as well

# Linux File System Architecture




> On UNIX/Linux system, everything is a file; if something is not a file, then it is a process

The above statement is mostly true, as Linux have special files which acts little more than normal files such as named pipes & sockets and to keep everything simple, it is an acceptable generalization. Also, directory is also a special file, since a directory is just a file containing names of other files. Programs, texts, images, database and so forth, are all files. Similarly, I/O devices, & generally all devices, are considered to be files. In Linux, files are organised in tree like structure. It has one root (`/`) and all the files/folder reside under it.

## Filesystems

Filesystem is used by an operating system to keep track of files. Or in other word, it provides mechanism for files to be organised on the disk. It is also used to denote a partition or the type of the filesystem.

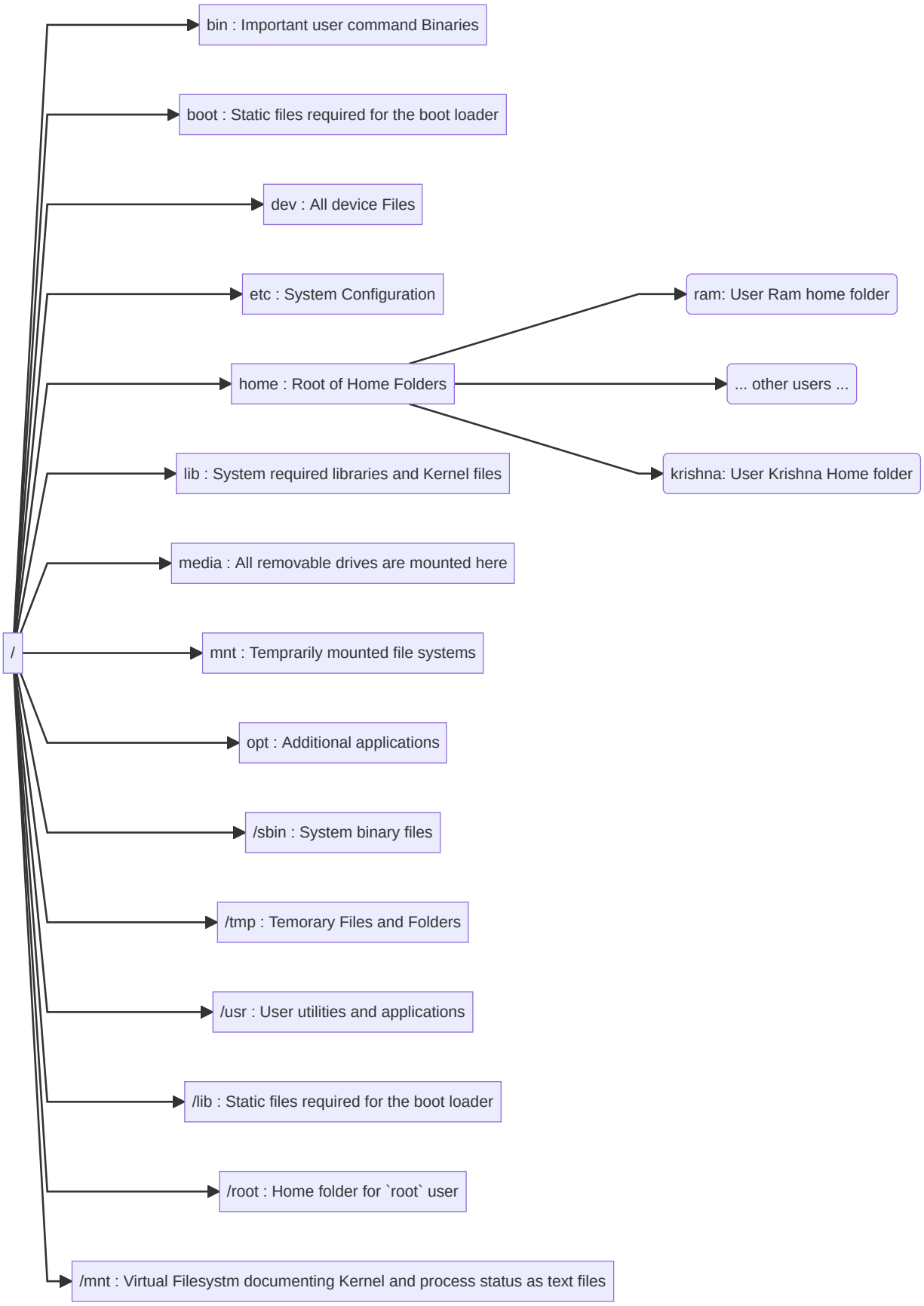
The difference between a disk or partition and the filesystem it contains is important. A few programs (including, reasonably enough, programs that create filesystems) operate directly on the raw sectors of a disk or partition; if there is an existing file system there it will be destroyed or seriously corrupted. Most programs operate on a filesystem, and therefore won't work on a partition that doesn't contain one (or that contains one of the wrong type).

Before a partition or disk can be used as a filesystem, it needs to be initialised, and the bookkeeping data structures need to be written to the disk. This process is called *making a filesystem*.

The central concepts are *superblock*, *inode*, *data block*, *directory block*, and *indirection block*. The superblock contains information about the filesystem as a whole, such as its size (the exact information here depends on the filesystem). An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are indirect blocks; the name indicates that in order to find the data block, one has to find its number in the indirect block first.

Linux chooses to have a single hierarchical directory structure. Everything starts from the root directory, represented by /, and then expands into sub-directories instead of having so-called 'drives'. In the Windows environment, one may put one's files almost anywhere: on C drive, D drive, E drive etc. Such a file system is called a hierarchical structure and is managed by the programs themselves (program directories), not by the operating system. On the other hand, Linux sorts directories descending from the root directory / according to their importance to the boot process.

In Linux, programs put their documentation into /usr/share/doc/[program-name], man(ual) pages into /usr/share/man/man[1-9] and info pages into /usr/share/info. They are merged into and with the system hierarchy.



## /bin

`/bin` contains commands that may be used by both the system administrator and by users, but which are required when no other filesystems are mounted (e.g. in single user mode). It may also contain commands which are used indirectly by scripts

## /boot

This directory contains everything required for the boot process except configuration files not needed at boot time and the map installer. Thus /boot stores data that is used before the kernel begins executing user-mode programs. This may include saved master boot sectors and sector map files.

## /dev

The `/dev` directory is the location of special or device files.

## /etc

The `/etc` hierarchy contains configuration files. A "configuration file" is a local file used to control the operation of a program; it must be static and cannot be an executable binary

## /etc/opt

Host-specific configuration files for add-on application software packages must be installed within the directory `/etc/opt/<subdir>`, where `<subdir>` is the name of the subtree in `/opt` where the static data from that package is stored.

## /etc/X11

*/etc/X11* is the location for all X11 host-specific configuration. This directory is necessary to allow local control if */usr* is mounted read only.

## /home

`/home` is a fairly standard concept, but it is clearly a site-specific filesystem. The setup will differ from host to host. Therefore, no program should rely on this location.

## /lib

The `/lib` directory contains those shared library images needed to boot the system and run the commands in the root filesystem, ie. by binaries in `/bin` and `/sbin`.

## /media

This directory contains subdirectories which are used as mount points for removeable media such as floppy disks, cdroms and zip disks.

## /mnt

This directory is provided so that the system administrator may temporarily mount a filesystem as needed. The content of this directory is a local issue and should not affect the manner in which any program is run.

This directory must not be used by installation programs: a suitable temporary directory not in use by the system must be used instead.

## /opt

`/opt` is reserved for the installation of add-on application software packages. A package to be installed in `/opt` must locate its static files in a separate `/opt/<package>` or `/opt/<provider>` directory tree, where `<package>` is a name that describes the software package and `<provider>` is the provider's LANANA registered name.

## /root

The root account's home directory may be determined by developer or local preference, but this is the recommended default location.

## /sbin

Utilities used for system administration (and other root-only commands) are stored in `/sbin`, `/usr/sbin`, and `/usr/local/sbin`. `/sbin` contains binaries essential for booting, restoring, recovering, and/or repairing the system in addition to the binaries in `/bin`. Programs executed after `/usr` is known to be mounted (when there are no problems) are generally placed into `/usr/sbin`. Locally-installed system administration programs should be placed into `/usr/local/sbin`.

## /tmp

The `/tmp` directory must be made available for programs that require temporary files.

Programs must not assume that any files or directories in `/tmp` are preserved between invocations of the program.

## /usr

`/usr` is the second major section of the filesystem. `/usr` is shareable, read-only data. That means that `/usr` should be shareable between various FHS-compliant hosts and must not be written to. Any information that is host-specific or varies with time is stored elsewhere.

Large software packages must not use a direct subdirectory under the `/usr` hierarchy.

## /var

`/var` contains variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files. Some portions of `/var` are not shareable between different systems. For instance, `/var/log`, `/var/lock`, and `/var/run`. Other portions may be shared, notably `/var/mail`, `/var/cache/man`, `/var/cache/fonts`, and `/var/spool/news`. `/var` is specified here in

order to make it possible to mount `/usr` read-only. Everything that once went into `/usr` that is written to during system operation (as opposed to installation and software maintenance) must be in `/var`. If `/var` cannot be made a separate partition, it is often preferable to move `/var` out of the root partition and into the `/usr` partition. (This is sometimes done to reduce the size of the root partition or when space runs low in the root partition.) However, `/var` must not be linked to `/usr` because this makes separation of `/usr` and `/var` more difficult and is likely to create a naming conflict. Instead, link `/var` to `/usr/var`.

Applications must generally not add directories to the top level of `/var`. Such directories should only be added if they have some system-wide implication, and in consultation with the FHS mailing list.

# Linux Basics

## Linux/Unix Philosophy

- *Do one thing and do it well* - Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.
- *Everything is file* - Ease of use and security is offered by treating hardware as a file.
- *Small is beautiful*
- *Store data and configuration in flat text files* - Text file is a universal interface. Easy to create, backup and move files to another system
- *Use shell scripts to increase leverage and portability* - Use shell script to automate common tasks across various UNIX / Linux installations
- *Chain programs together to complete complex task* - Use shell pipes and filters to chain small utilities that perform one task at time
- *Choose portability over efficiency*
- *Keep it Simple, Stupid (KISS)*

## Login

Users can login to Linux machines either locally or remotely. Linux provides various ways to connect and following are few of them.

- Console
  - **Local**: login console
  - **Remote**: `ssh`, `telnet`, `rsh`, etc
- GUI
  - Local: KDM, GDM, XDM, SDDM
  - Remote console:
    - XDMCP: Remmina
    - RDP: FreeRDP, Remmina, Apache Guacamole, XRDP
    - VNC: TigerVNC, RealVNC, UltraVNC, Remmina, Apache Guacamoleetc

### `su`

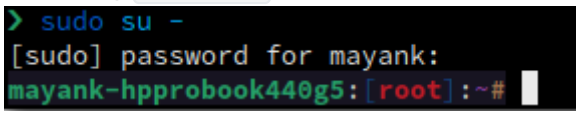
`su` command allows user to login as another user. By default, it allows user to login with root privileges.

## root & `sudo`

On Linux user will **NEVER** login as `root`. Whenever any user want to do the admin task they either login using `su` or `sudo` which every is enabled for him and once the task is completed he/she will logout of the session.

The main benefit of using `sudo` is that it only works for one command and once the command is completed user is again back looses the privilege. If user wants to run more than one command then user can run **`sudo su -`** or **`su -`** command to get root access on the shell where the command is executed and his command prompt will also change as shown in the below screenshot.


or if using `sudo su -`



# The desktop or X Window System

The window system is not integral part of a Linux system and the base of the most graphical user interface  of any UNIX, UNIX Like or Linux systems is X Window system version 11, also known as X Window, X11 or simple X. A complete X-based environment includes a number of other components that are based on X: toolkits provide widgets such as buttons and menus that users can interact with, and a window manager draws window decorations and manages policies for manipulating windows, input focus and much more.

There are two user facing component in Linux GUI desktop

- Display manager/Login Manager
- Desktop environment

Linux provides wide range of both from which to choose from depending on your requirement.

# Display Manager / Login Manager

Linux supports multiple display managers are few  of them are listed below.

- **Entrance** — Enlightenment display manager.
- **GDM** — GNOME display manager.
- **LightDM** — Cross-desktop display manager, can use various front-ends written in any toolkit.
- **LXDM** — LXDE display manager. Can be used independent of the LXDE desktop environment.
- **SDDM** — QML-based display manager and successor to KDM; recommended for Plasma and LXQt.
- **XDM** — X display manager with support for XDMCP, host chooser.

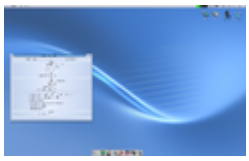
## Desktop Environment / Window Manager

A **window manager** is system software that controls the placement and appearance of windows within a windowing system in a graphical user interface. Most window managers are designed to help provide a desktop environment. They work in conjunction with the underlying graphical system which provides required functionality such as support for graphics hardware, pointing devices, and a keyboard, and are often written and created using a widget toolkit

Few examples of Window Managers for X are afterstep, metacity, blackbox, enlightenment, IceWM, Fluxbox, Openbox, ROX Desktop and Window Maker.  Extensive list can be viewed at https://www.gilesorr.com/wm/table.html

Major Desktop Environments are KDE/Plasma, GNOME, Cinnamon, Mate, Xfce, LXDE, and Aqua.

### Window Manager Gallery

- 

Ambient

- 

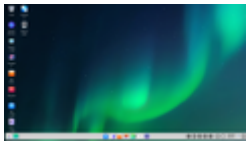
Budgie

- 

CDE

- 

Cinnamon

- 

Deepin DE

- 

EDE

- 

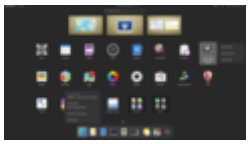
Elokab

- 

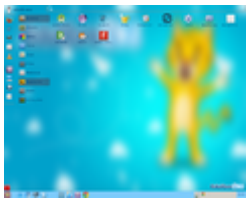
Enlightenment

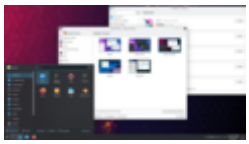
- 

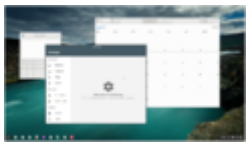
Étoilé

- 

  GNOME Shell
- 

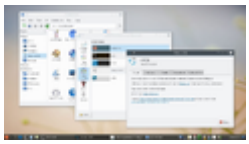
  GNUstep
- 

  Innova
- 

  KDE Plasma 5
- 

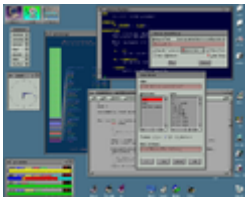
  Liri Shell
- 


  Lumina
- 


  LXDE
- 

  LXQt
- 

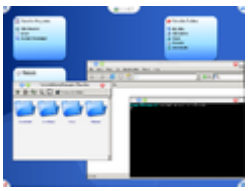
MATE

- 

MaXX

- 

Maynard

- 

Mezzo

- 

Moksha

- 

Pantheon

- 

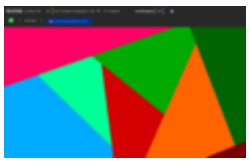
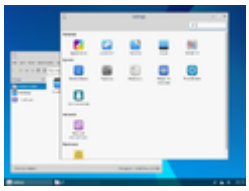
Project Looking Glass

- 

Razor-qt

- 

ROX Desktop

- 
  Sugar
- 
  theShell
- 
  Trinity
- 
  Unity
- 
  vera
- 
  Xfce

## Shells

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (% on our systems).

| Shell | Name | URL | License |
|---|---|---|---|
| BASH | (Bourne-Again SHell) | https://www.gnu.org/software/bash/ | GPL-3.0-or-later |
| CSH | (C SHell) | https://en.wikipedia.org/wiki/C_shell | BSD license |

| Shell | Name | URL | License |
|---|---|---|---|
| KSH | (Korn SHell) | http://www.kornshell.com/ | Eclipse Public License |
| TCSH | TCSH | https://github.com/tcsh-org/tcsh, http://www.tcsh.org/ | BSD-3-Clause |
| Fish | Fish-shell | http://fishshell.com/ | GPL-2.0-only |

Table: List of Most of the Shells

| Shell | Usual environment | Introduced | License | Unicode support |
|---|---|---|---|---|
| Thompson shell | UNIX | 1971 | N/A | No |
| Bourne shell 1977 version | 7th Ed. UNIX | 1977 | Proprietary | No |
| Bourne shell latest | Various UNIX | 1977 | CDDL | Yes |
| POSIX shell[6] | POSIX | 1992 | N/A | Yes |
| bash (v4) | POSIX | 1989 | GPL | Yes |
| csh | POSIX | 1978 | BSD | No |
| tcsh | POSIX | 1983 | BSD | Yes |
| Scsh | POSIX | 1994 | BSD-style | - |
| ksh (ksh93t+) | POSIX | 1983 | Common Public License | Yes |
| pdksh | POSIX | 1989 | Public Domain | No |
| zsh | POSIX | 1990 | MIT-style | Yes |
| ash | POSIX | 1989 | BSD-style | Partial (for BusyBox) |
| CCP | CP/M, MP/M | 1976 (1974) | Freeware (originally proprietary) | No |
| COMMAND.COM | DOS | 1980 | MS-EULA | - |
| OS/2 CMD.EXE | OS/2, eComStation, ArcaOS | 1987 | IBM-EULA | No |

| Shell | Usual environment | Introduced | License | Unicode support |
|---|---|---|---|---|
| Windows CMD.EXE | Win32 | 1993 | MS-EULA | Partial (CHCP 65001 for UTF-8 |
| VMS DCL | OpenVMS | 1977 | Proprietary | Yes |
| PowerShell | .NET/ .NET Framework | 2006 | MIT-style | Yes |
| rc | Plan 9/ POSIX | 1989 | MIT License | Yes |
| BeanShell | Java | 2005 | LGPL | Yes |
| fish | POSIX | 2005 | GPL | Yes |
| Ion | Redox / Linux | 2015 | MIT | Yes |

# Linux Command Line Interface

Linux provides many commands which we can execute from Linux Shells and in this section we will learn about few of them.

They can be executed from consoles directly or from within shell scripts. One of the primary principles is that commands should be small, designed to do only one task and do it well.

Complex commands can be build using then using shell scripts or by directing the output of one to another Several commands connected to one another in such a fashion is called a **pipeline**.

## Basic Linux commands

### Size

#### df

Lets me know the size of all the mounted partitions

```
$ df
df: /run/user/1000/doc: Operation not permitted
Filesystem           1K-blocks     Used Available Use% Mounted on
udev                   7975792        0   7975792   0% /dev
tmpfs                  1606708     3152   1603556   1% /run
/dev/mapper/root.fsm 481563184 65529028 391498604  15% /
tmpfs                     5120        4      5116   1% /run/lock
tmpfs                  3213400   518664   2694736  17% /dev/shm
/dev/nvme0n1p2          996780   136036    791932  15% /boot
/dev/nvme0n1p1          258095      282    257813   1% /boot/efi
cgroup                      12        0        12   0% /sys/fs/cgroup
tmpfs                  1606704        0   1606704   0% /run/user/118
tmpfs                  1606704      116   1606588   1% /run/user/1000
```

#### du

Lets me know the size of the file/folder

```
$ du -m final/
3       final/
```

```
$ du  final/
2196    final/
```

# List Files and Directories

## `ls`

`ls` command allows us to view the listing of a directory

### Syntax

```
❯ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      with -l, scale sizes by SIZE when printing them;
                               e.g., '--block-size=M'; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'always' (default
                               if omitted), 'auto', or 'never'; more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
  -F, --classify             append indicator (one of */=>@|) to entries
      --file-type            likewise, except do not append '*'
      --format=WORD          across -x, commas -m, horizontal -x, long -l,
                               single-column -1, verbose -l, vertical -C
      --full-time            like -l --time-style=full-iso
  -g                         like -l, but do not list owner
      --group-directories-first
                             group directories before files;
                               can be augmented with a --sort option, but any
                               use of --sort=none (-U) disables grouping
  -G, --no-group             in a long listing, don't print group names
  -h, --human-readable       with -l and -s, print sizes like 1K 234M 2G etc.
      --si                   likewise, but use powers of 1000 not 1024
  -H, --dereference-command-line
                             follow symbolic links listed on the command line
      --dereference-command-line-symlink-to-dir
                             follow each command line symbolic link
                               that points to a directory
      --hide=PATTERN         do not list implied entries matching shell PATTERN
                               (overridden by -a or -A)
      --hyperlink[=WHEN]     hyperlink file names; WHEN can be 'always'
```

```
                               (default if omitted), 'auto', or 'never'
      --indicator-style=WORD  append indicator with style WORD to entry names:
                               none (default), slash (-p),
                               file-type (--file-type), classify (-F)
  -i, --inode                 print the index number of each file
  -I, --ignore=PATTERN        do not list implied entries matching shell PATTERN
  -k, --kibibytes             default to 1024-byte blocks for disk usage;
                               used only with -s and per directory totals
  -l                          use a long listing format
  -L, --dereference           when showing file information for a symbolic
                               link, show information for the file the link
                               references rather than for the link itself
  -m                          fill width with a comma separated list of entries
  -n, --numeric-uid-gid       like -l, but list numeric user and group IDs
  -N, --literal               print entry names without quoting
  -o                          like -l, but do not list group information
  -p, --indicator-style=slash
                              append / indicator to directories
  -q, --hide-control-chars    print ? instead of nongraphic characters
      --show-control-chars    show nongraphic characters as-is (the default,
                               unless program is 'ls' and output is a terminal)
  -Q, --quote-name            enclose entry names in double quotes
      --quoting-style=WORD    use quoting style WORD for entry names:
                               literal, locale, shell, shell-always,
                               shell-escape, shell-escape-always, c, escape
                               (overrides QUOTING_STYLE environment variable)
  -r, --reverse               reverse order while sorting
  -R, --recursive             list subdirectories recursively
  -s, --size                  print the allocated size of each file, in blocks
  -S                          sort by file size, largest first
      --sort=WORD             sort by WORD instead of name: none (-U), size (-
S),
                               time (-t), version (-v), extension (-X)
      --time=WORD             change the default of using modification times;
                               access time (-u): atime, access, use;
                               change time (-c): ctime, status;
                               birth time: birth, creation;
                              with -l, WORD determines which time to show;
                              with --sort=time, sort by WORD (newest first)
      --time-style=TIME_STYLE  time/date format with -l; see TIME_STYLE below
  -t                          sort by time, newest first; see --time
  -T, --tabsize=COLS          assume tab stops at each COLS instead of 8
  -u                          with -lt: sort by, and show, access time;
                               with -l: show access time and sort by name;
                               otherwise: sort by access time, newest first
  -U                          do not sort; list entries in directory order
  -v                          natural sort of (version) numbers within text
  -w, --width=COLS            set output width to COLS.  0 means no limit
  -x                          list entries by lines instead of by columns
  -X                          sort alphabetically by entry extension
  -Z, --context               print any security context of each file
  -1                          list one file per line.  Avoid '\n' with -q or -b
```

```
      --help     display this help and exit
      --version  output version information and exit

The SIZE argument is an integer and optional unit (example: 10K is 10*1024).
Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000).
Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

The TIME_STYLE argument can be full-iso, long-iso, iso, locale, or +FORMAT.
FORMAT is interpreted like in date(1).  If FORMAT is FORMAT1<newline>FORMAT2,
then FORMAT1 applies to non-recent files and FORMAT2 to recent files.
TIME_STYLE prefixed with 'posix-' takes effect only outside the POSIX locale.
Also the TIME_STYLE environment variable sets the default style to use.

Using color to distinguish file types is disabled both by default and
with --color=never.  With --color=auto, ls emits color codes only when
standard output is connected to a terminal.  The LS_COLORS environment
variable can change the settings.  Use the dircolors command to set it.

Exit status:
 0  if OK,
 1  if minor problems (e.g., cannot access subdirectory),
 2  if serious trouble (e.g., cannot access command-line argument).
```

**Examples**

- To list all files in the current directory, type the following:

```
ls -a
```

- To display detailed information

```
ls -l
```

- To display detailed information about a directory

```
ls -l -d ./folder
```

- List files in the user's home directory

```
ls -l ~
```

- List only directories

```
ls -d .
```

- List files with subdirectories recursively

```
ls -R /etc
```

# Copy Files and Directories

`cp` command can be used to copy files and directories

## Syntax

```
cp [additional_option] <source_file(s)> <target_file>
```

## Options

```
Usage: cp [OPTION]... [-T] SOURCE DEST
  or:  cp [OPTION]... SOURCE... DIRECTORY
  or:  cp [OPTION]... -t DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.
  -a, --archive                same as -dR --preserve=all
      --attributes-only        don't copy the file data, just the attributes
      --backup[=CONTROL]       make a backup of each existing destination file
  -b                           like --backup but does not accept an argument
      --copy-contents          copy contents of special files when recursive
  -d                           same as --no-dereference --preserve=links
  -f, --force                  if an existing destination file cannot be
                                 opened, remove it and try again (this option
                                 is ignored when the -n option is also used)
  -i, --interactive            prompt before overwrite (overrides a previous -n
                                  option)
  -H                           follow command-line symbolic links in SOURCE
  -l, --link                   hard link files instead of copying
  -L, --dereference            always follow symbolic links in SOURCE
  -n, --no-clobber             do not overwrite an existing file (overrides
                                 a previous -i option)
  -P, --no-dereference         never follow symbolic links in SOURCE
  -p                           same as --preserve=mode,ownership,timestamps
      --preserve[=ATTR_LIST]   preserve the specified attributes (default:
                                 mode,ownership,timestamps), if possible
                                 additional attributes: context, links, xattr,
                                 all
      --no-preserve=ATTR_LIST  don't preserve the specified attributes
      --parents                use full source file name under DIRECTORY
  -R, -r, --recursive          copy directories recursively
      --reflink[=WHEN]         control clone/CoW copies. See below
      --remove-destination     remove each existing destination file before
                                 attempting to open it (contrast with --force)
      --sparse=WHEN            control creation of sparse files. See below
      --strip-trailing-slashes  remove any trailing slashes from each SOURCE
                                 argument
  -s, --symbolic-link          make symbolic links instead of copying
  -S, --suffix=SUFFIX          override the usual backup suffix
  -t, --target-directory=DIRECTORY  copy all SOURCE arguments into DIRECTORY
  -T, --no-target-directory    treat DEST as a normal file
```

```
  -u, --update                copy only when the SOURCE file is newer
                                than the destination file or when the
                                destination file is missing
  -v, --verbose               explain what is being done
  -x, --one-file-system       stay on this file system
  -Z                          set SELinux security context of destination
                                file to default type
      --context[=CTX]         like -Z, or if CTX is specified then set the
                                SELinux or SMACK security context to CTX
      --help     display this help and exit
      --version  output version information and exit

By default, sparse SOURCE files are detected by a crude heuristic and the
corresponding DEST file is made sparse as well.  That is the behavior
selected by --sparse=auto.  Specify --sparse=always to create a sparse DEST
file whenever the SOURCE file contains a long enough sequence of zero bytes.
Use --sparse=never to inhibit creation of sparse files.

When --reflink[=always] is specified, perform a lightweight copy, where the
data blocks are copied only when modified.  If this is not possible the copy
fails, or if --reflink=auto is specified, fall back to a standard copy.
Use --reflink=never to ensure a standard copy is performed.

The backup suffix is '~', unless set with --suffix or SIMPLE_BACKUP_SUFFIX.
The version control method may be selected via the --backup option or through
the VERSION_CONTROL environment variable.  Here are the values:

  none, off       never make backups (even if --backup is given)
  numbered, t     make numbered backups
  existing, nil   numbered if numbered backups exist, simple otherwise
  simple, never   always make simple backups

As a special case, cp makes a backup of SOURCE when the force and backup
options are given and SOURCE and DEST are the same name for an existing,
regular file.
```

### Examples

- **Copy file with another name**

```
cp file.txt file_new.txt
```

  This is specially useful if we are taking a backup of existing file.

- **Copy File to Another Directory**

```
cp file.txt ./another_dir
```

- **Copy File to Another Directory and rename it**

```
cp file.txt ./another_dir/file_new.txt
```

- Copy Multiple Files from One Directory

```
cp file1.txt file2.txt file3.txt ./another_dir
```

  This will copy `file1.txt`, `file2.txt` and `file3.txt` to `another_dir` folder.

- **copy an entire folder and its subfolders and files**

```
cp -R src_folder dst_folder
```

  This command will copy entire `src_folder` to `dst_folder`

- Copy file(s) to current location

```
cp Test/test.txt .
```

  This command will copy `test.txt` file to current folder.

## `rsync`

`rsync` is used mostly for backup of files and folder to backup location and keeping them upto date.

`rsync` is a fast and extraordinarily versatile file copying tool. It can copy locally, to/from another host over any remote shell, or to/from a remote rsync daemon. It offers a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied. It is famous for its delta-transfer algorithm, which reduces the amount of data sent over the network by sending only the differences between the source files and the existing files in the destination. Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.
Rsync finds files that need to be transferred using a "quick check" algorithm (by default) that looks for files that have changed in size or in last-modified time. Any changes in the other preserved attributes (as requested by options) are made on the destination file directly when the quick check indicates that the file's data does not need to be updated.
Some of the additional features of rsync are:

- support for copying links, devices, owners, groups, and permissions
- exclude and exclude-from options similar to GNU tar
- a CVS exclude mode for ignoring the same files that CVS would ignore
- can use any transparent remote shell, including ssh or rsh
- does not require super-user privileges
- pipelining of file transfers to minimize latency costs

**Examples**:

- Copying files to remote destination

```
rsync -ar <source_folder> <destination_user>@<destination_host>:<path>
```

We will cover more about remote copying in seperate section

## Move Files and Directories

### `mv`

Moves the file(s) and folders from source to destination.

```
❯ mv --help
Usage: mv [OPTION]... [-T] SOURCE DEST
  or:  mv [OPTION]... SOURCE... DIRECTORY
  or:  mv [OPTION]... -t DIRECTORY SOURCE...
Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.
      --backup[=CONTROL]       make a backup of each existing destination file
  -b                           like --backup but does not accept an argument
  -f, --force                  do not prompt before overwriting
  -i, --interactive            prompt before overwrite
  -n, --no-clobber             do not overwrite an existing file
If you specify more than one of -i, -f, -n, only the final one takes effect.
      --strip-trailing-slashes  remove any trailing slashes from each SOURCE
                                  argument
  -S, --suffix=SUFFIX          override the usual backup suffix
  -t, --target-directory=DIRECTORY  move all SOURCE arguments into DIRECTORY
  -T, --no-target-directory    treat DEST as a normal file
  -u, --update                 move only when the SOURCE file is newer
                                  than the destination file or when the
                                  destination file is missing
  -v, --verbose                explain what is being done
  -Z, --context                set SELinux security context of destination
                                  file to default type
      --help     display this help and exit
      --version  output version information and exit

The backup suffix is '~', unless set with --suffix or SIMPLE_BACKUP_SUFFIX.
The version control method may be selected via the --backup option or through
the VERSION_CONTROL environment variable.  Here are the values:

  none, off       never make backups (even if --backup is given)
  numbered, t     make numbered backups
  existing, nil   numbered if numbered backups exist, simple otherwise
  simple, never   always make simple backups
```

## Rules:

- When multiple files or directories are given as a `SOURCE`, the `DEST` must be a directory. In this case, the `SOURCE` files are moved to the target directory.
- If you specify a single file as `SOURCE`, and the `DEST` target is an existing directory, then the file is moved to the specified directory.

- If you specify a single file as `SOURCE`, and a single file as `DEST` target then you're renaming the file .
- When the `SOURCE` is a directory and `DEST` doesn't exist, `SOURCE` will be renamed to `DEST`. Otherwise if `DEST` exist, it be moved inside the `DEST` directory.

## Examples

- Moving one file to a directory

```
mv file1 ./dest_folder
```

- rename a file

```
mv file1 file2
```

- moving a directory to another directory

```
mv src_folder dst_folder
```

- Moving Multiple Files and Directories

```
mv src1 src2 src3 dst_folder
```

- (-b) Move and backup existing files

```
mv -b src1 dst_folder
```

  If the destination file exists you can create a backup of it using the `-b` option

- move hidden files/folder

```
mv a/{.*,*} ~/det_folder
```

- (-f) Force move for minor protection

```
mv -f src dst
```

- (-n) move files but do not override existing files

```
mv -n src1 dst
```

```
~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ mv -n test.txt Test/^C

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ cat Test/test.txt

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ cat test.txt
This is new file.
```

- (`-i`) Move and prompt before overriding existing files

```
mv -i src dst
```

- (-u) move file only when it is newer than the destination file
- (-v) verbose details of what all it is doing

## Removing files and Directories

Linux provide two commands to achieve it. `rm` and `unlink`

### `rm`

**Examples**

- Remove or delete a file

```
rm dummy.log
```

- Remove multiple files at once

```
rm file1 file2 file3
```

- Remove the files interactively

```
rm -i file1 file2 file3
```

- Remove an empty directory

```
rm -d empty_folder/
```

- (-r)  Remove a directory recursively

```
rm -r folder
```

- **(-f) Remove files forcefully**

```
rm -f file.txt
```

- (-I) Prompt once before deleting more than three files or recursive delete

```
 rm -I folder/*
```

- Using Regular expression

```
# will only work on bash and not in fish
rm test_00{1..3}.log

rm  *.txt
rm  *.???
```

- Remove a file which starts with hyphen symbol (-)

```
rm \~test.txt
```

## rmdir - Delete Empty directory, ie directory without files.

**Syntax:**

```
rmdir [options...] [directories ...]
```

**Exmaple:**

- Remove an empty directory

```
mkdir empty_folder
```

- (-p) Remove folder recursively

```
~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ mkdir -p dummy/test/test2/test3

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ ls
dummy  GNU3.txt  test  Test

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ cp GNU3.txt dummy/test/

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ rmdir -p dummy/test/test2/test3/
rmdir: failed to remove directory 'dummy/test': Directory not empty

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
❯ ls -R dummy/
```

```
dummy/:
test

dummy/test:
GNU3.txt
```

If we try to remove folder recursively which cotains file, then it will delete child folders till it encounters first file and then it will raise an exception and quit as shown in above example. Please use `rm -rf <folder_name>` instead. But use it with care.

## find

We can also use `find` command to delete files

```
find ~/dummy/ -type f -delete
```

## difference between `rm` and `unlink`

POSIX specifies that the `unlink` utility calls the C library `unlink` function and nothing else. It takes no option. If you pass a valid path name to something which isn't a directory, and if you have write permissions to the directory where that object lives, then `unlink` will remove it.

`rm` is a traditional Unix command which has a bit of other functionality, and isn't quite a superset of `unlink` (see below).

Firstly, `rm` performs safety checks. If you try to `rm` an object to which you don't have write permissions (which are irrelevant to your ability to remove it: the containing directory's permissions are!) `rm` nevertheless refuses unless `-f` is specified. `rm` normally complains if the file doesn't exist, as does `unlink`; however with `-f`, `rm` does not complain. This is often exploited in Makefiles (`clean: @rm -f $(OBJS) ...`) so `make clean` doesn't fail when there is nothing to remove.

Secondly, `rm` has the `-i` option for interactively confirming the delete.

Thirdly, `rm` has `-r` for recursively removing a directory, which is something that `unlink` isn't required to do, since the C library function doesn't do that.

**The `unlink` utility isn't exactly a stripped-down `rm`.** It performs a subset of what `rm` does, but it has semantics which is a combination of `rm` **with** `-f` and rm **without** `-f`.

**Suppose you want to just remove a regular file regardless of what its own permissions are. Furthermore, suppose you want the command to fail if the file doesn't exist, or any other reason.** Neither `rm file` nor `rm -f file` meets the requirements. `rm file` will refuse if the file isn't writable. But `rm -f file` will neglect to complain if the file is missing. `unlink file` does the job.

`unlink` was probably introduced because `rm` is too clever: sometimes you just want the pure Unix `unlink` semantics: *"please make this directory entry go away if directory permissions allow"*.

At least in GNU system, `unlink` can never delete the name of a directory.

## Make Directory

**Syntax:**

```
mkdir [options...] [directories ...]
```

## Exmaple:

- Create an empty directory

  ```
  mkdir empty_folder
  ```

- Create directory and any needed parent directories

  ```
  mkdir -p test/test1/test2/test3/test4
  ```

- Create Verbosely folder

  ```
  mkdir -v test1 test3 test2
  ```

- Make directory with permission

  ```
  mkdir -m a=rwx test_folder
  ```

- make multiple directories

  ```
   mkdir -p test4/test5/{t1,t2,t3}
  ```

# Display commands

## `clear` - Clear screen

Cleans the screen

## `cat` - View/concatenate/redirect the file

**cat** command allows us to create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files.

**Syntax**

```
cat [OPTION] [FILE]...
```

**Examples**

- Display Contents of File

```
cat test.txt
```

  Displays the content of the `test.txt` file on the console

- View Contents of Multiple Files

```
cat test_001.txt test_002.txt
```

- Using `cat` as editor

```
cat > test.txt
```

- Copy a file

```
cat test.txt > test1.txt
```

  This will create `test1.txt` file with the same content as `test.txt`

- Display Line Numbers with file content

```
cat -n test.txt
```

- Display `$` at EOF & New Line

```
cat -e test.txt
```

  **Output**

```
❯ cat -e test.txt
H$
H$
F$
F$
H$
F$
F$
F$
F$
F$
This is new file.$
```

- (-T) Display Tab Separated Lines

```
cat -T test1.txt
```

## `less` - Display one page at a time

Less command is a Linux utility that can be used to read the contents of a text file one page(one screen) at a time. It has faster access because if file is large it doesn't access the complete file, but accesses it page by page.

**Syntax:**

```
less <options> filename
```

**Most commonly Options**

| Option | task |
|---|---|
| -E | Automatically exit the first time it reaches end of file |
| -f | forces non-regular file to open |
| -F | causes less to exit if entire file can be displayed on first screen |
| -g | highlight the string which was found by last search command |
| -G | suppresses all highlighting of strings found by search commands |
| -i | cause searches to ignore case |
| -n | suppresses line numbers |
| -p pattern | it tells less to start at the first occurrence of pattern in the file |
| -s | causes consecutive blank lines to be squeezed into a single blank line |

**Keystrokes for Navigation**

| Key | Command |
|---|---|
| Space bar | Next Page |
| d | Next half Page |
| b | Previous Page |
| u | Previous half Page |
| v | Edit Content |
| j or ↵ Enter | Next Line |
| k | Previous Line |
| Home | Top of file |
| End | End of file |

| Key | Command |
|---|---|
| F | Follow Mode (for logs). Interrupt to abort. |
| g or < | First Line |
| G or > | Last Line |
| `⟨n⟩` G | Line `⟨n⟩` |
| / `⟨text⟩` | Forward Search for `⟨text⟩`. Text is interpreted as a regex. |
| ? `⟨text⟩` | Backward Search like / |
| n | Next Search Match |
| N | Previous Search Match |
| Escu | Turn off Match Highlighting (see `-g` command line option) |
| - `⟨c⟩` | Toggle option `⟨c⟩`, e.g., -i toggles option to match case in searches |
| m `⟨c⟩` | Set Mark `⟨c⟩` |
| ' `⟨c⟩` | Go to Mark `⟨c⟩` |
| = or Ctrl+G | File information |
| :n | Next file |
| :p | Previous file |
| h | Help. This is presented with `less`, q to quit. |
| q | Quit |

**Examples**

- Display a long file.

```
less GNU3.txt
```

- Display commands output

```
ls -lR /home | less
```

- (`-N`) Display Line numbers
- (`-X`) Leave Content of Screen
- (+F) Watch for content change
- (`-i`) Search case-insensitively
- (`-p`) start at first occurrence of pattern

```
cat gpl-3.0.txt  | less -p "terminal interaction"
```

It will start from the line where `terminal interaction` is present.

- (`-m`) Show more detailed prompt, including file position.
- (`-x3`) Set tabstops
- (`-S`) Disable line wrapping

### `head` - Display the beginning of a text file or piped data

head is a program on Unix and Unix-like operating systems used to display the beginning of a text file or piped data.

**Syntax:**

```
head [options] <file_name>
```

**Examples**

- Display first 10 lines

```
head GNU3.txt
```

- (`-n`) Display `n` lines

```
head -n 20 GNU3.txt
```

- Show multiple files

```
head -n 15 *.md
```

### `tail` - display the tail end of a text file or piped data

Similar to `head` except views the end of the file.

## Search commands

### grep, egrep, fgrep - print lines that match pattern

grep searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. Typically PATTERNS should be quoted when grep is used in a shell command.

**Syntax:**

```
grep [OPTION...] PATTERNS [FILE...]
grep [OPTION...] -e PATTERNS ... [FILE...]
grep [OPTION...] -f PATTERN_FILE ... [FILE...]
```

**Options**

| Options | Description |
|---|---|
| -c | This prints only a count of the lines that match a pattern |
| -h | Display the matched lines, but do not display the filenames. |
| -i | Ignores, case for matching |
| -l | Displays list of a filenames only. |
| -n | Display the matched lines and their line numbers. |
| -v | This prints out all the lines that do not matches the pattern |
| -e exp | Specifies expression with this option. Can use multiple times. |
| -f file | Takes patterns from file, one per line. |
| -E | Treats pattern as an extended regular expression (ERE) |
| -w | Match whole word |
| -o | Print only the matched parts of a matching line, with each such part on a separate output line. |
| -A n | Prints searched line and `n` lines after the result. |
| -B n | Prints searched line and n line before the result. |
| -C n | Prints searched line and n lines after before the result. |

**Examples**

- (`i`) **Case insensitive search**

```
$ grep -i "Programmer" gpl-3.0.txt

```

**Output:**

```
You should also get your employer (if you work as a programmer) or school,
```

```
[redacted]~/code/mj/ebooks/lllinux$ grep -i "kin" gpl-3.0.txt
software and other kinds of works.
these rights or asking you to surrender the rights.  Therefore, you have
  "Copyright" also means copyright-like laws that apply to other kinds of
in a fashion requiring copyright permission, other than the making of an
distribution (with or without modification), making available to the
  To "convey" a work means any kind of propagation that enables other
for making modifications to it.  "Object code" means any non-source
is widely used among developers working in that language.
not control copyright.  Those thus making or running the covered works
and control, on terms that prohibit them from making any copies of
unpacking, reading or copying.
License by making exceptions from one or more of its conditions.
any patent claim is infringed by making, using, selling, offering for
by this License, of making, using, or selling its contributor version,
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
may consider it more useful to permit linking proprietary applications with
```

- (`-c`) Displaying the count of number of matches

```
grep -c "Programmer" gnu3.txt
```

```
$ grep -ic "PRogrammer" gpl-3.0.txt
```

**Output:**

```
1
```

```
$ grep -c "kin" gpl-3.0.txt
```

**Output:**
15

- (-r) Search recursively

```
grep -r "rolling" /etc/
```

- (`-I`) List the file content matching the pattern

```
grep -l "rolling" /etc
```

```
grep -l "rolling" /etc/os-release
/etc/os-release
```

```
$ grep -irl "PRogrammer"
```

**Output**:

```
01_Lets Learn Linux.md
gpl-3.0.txt
code/Section-1/diff_chapter/gnu3.txt
code/Section-1/diff_chapter/gnu3_update.txt
code/Section-1/tar_example/hidden/combined.txt
```

```
code/Section-1/tar_example/hidden/gnu3.txt
code/Section-1/tar_example/hidden/GNU3.txt
code/Section-1/tar_example/hidden/dummy/test/GNU3.txt
code/Section-1/tar_example/hidden/Test/gnu3.txt
code/Section-1/tar_example/hidden/Test/GNU3.txt
code/Section-1/tar_example/hidden/test/gnu3.txt
code/Section-1/tar_example/hidden/test/GNU3.txt
code/Section-1/tar_example/hidden.tar
LICENSE
Lets Learn Linux.md.backup
```

- (`-w`) Listing lines with word

```
 grep -w "GNU" gnu3.txt
```

- (`-v`) Listing lines which don't have word

- (`-o`) Listing word with matched pattern

```
 grep -o "GNU" gnu3.txt
```

- (`-n`) Listing lines with line number

```
   $ grep -ion "pro" gpl-3.0.txt
```

**Output**:

```
16:pro
20:pro
27:pro
29:pro
34:pro
40:pro
```

- **RegEx**: lines that start with a string

```
grep "^lic" gnu3.txt
```

- (`-B`) Display lines before and after the match

```
$ grep -iwn -B 2  "PRogrammer" gpl-3.0.txt

```

**Output**:

```
662-might be different; for a GUI interface, you would use an "about box".
663-
664:  You should also get your employer (if you work as a programmer) or
school,
```

- (`--color`) Colouring the selection

```
grep --color "gnu" gnu3.txt
```

we can also change the colour from default Red to another colour using the environment variable `GREP_COLOR` as shown in below example

```
GREP_COLOR='1;35' grep --color=always  "gnu" gnu3.txt
```

**Table**: Standard Attributes

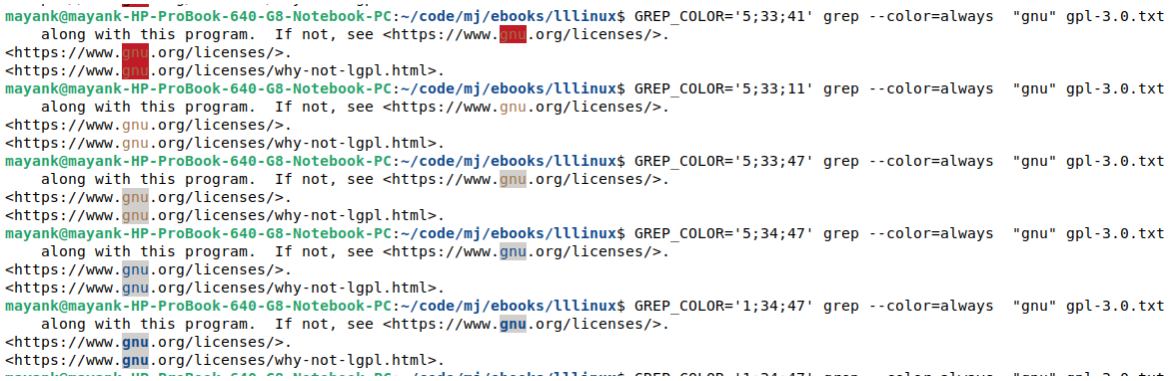
| Value | Meaning |
|---|---|
| 0 | Reset all attributes |
| 1 | Bright |
| 2 | Dim |
| 4 | Underscore |
| 5 | Blink |
| 7 | Reverse |
| 8 | Hidden |

| Color | Background Colours | Foreground Colours |
|---|---|---|
| Black | 40 | 30 |
| Red | 41 | 31 |
| Green | 42 | 32 |
| Yellow | 43 | 33 |
| Blue | 44 | 34 |
| Magenta | 45 | 35 |
| Cyan | 46 | 36 |
| White | 47 | 37 |

```
GREP_COLOR='5;36;41' grep --color=always  "gnu" gnu3.txt
```

This command will blink & highlight gnu with background as `red` and text color as `Cryan`.

We can also set them in `.bashrc` to make them default setting.




- Mixing and Matching options

```
grep -C 4 -B 5 -A 2 --color 'gnu'  *
```

## `which`

Find the full path of the executable file

```
$ which ls
```

```

/usr/bin/ls
```

To get all instances, run the `which` with `-a` option

```
$ which -a ls
```

```

/usr/bin/ls
/bin/ls
```

## `cd` - Change Directory

### Examples

- Change to another directory

```
cd /home
```

- Change to Home folder

```
cd
```

or

```
cd ~
```

- Change to parent folder

```
cd ..
```

- Change back to previous working folder

```
cd -
```

- Change to directory with space in name

```
cd My\ Folder
```

- Change to actual location of symlink

```
cd -P symlink
```

## `chmod` - Change mode

| Octal Notation | Permission | Symbolic Representation |
|---|---|---|
| 0 | No Permission | --- |
| 1 | Execute Permission Only | --x |
| 2 | Write Permission Only | -w- |
| 3 | Write and Execute Permissions (1+2)=3 | -wx |
| 4 | Read Permission Only | r-- |
| 5 | Read and Execute Permissions (1+4)=5 | r-x |
| 6 | Read and Write Permissions (2+4)=6 | rw- |
| 7 | Read, Write and Execute Permissions, Means Full Permissions (1+2+4)=7 | rwx |

| Read | Write | Execute | Int Value |
|---|---|---|---|
| 0 | 0 | 0 | 0 |

| Read | Write | Execute | Int Value |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

- chmod a-w first.txt

  This means all users have no write access to the file first.txt.

- `chmod u + x script.sh`

  The owner of script.sh can execute the file

## `chown` - Change Owner

**Examples**

- Change the owner of file/folder

```
chown demo gnu3.txt
```

- (`-c`) Check when the ownership has changed

```
chown -c gnu3.txt
```

- To Change group ownership

```
chown :usergroup gnu3.txt
```

- change the owner as well as group

```
chown demo:demo gnu3.txt
```

- change the owner from particular ownership only

```
chown --from=demo gnu3.txt
```

- change group from a particular group

```
chown --from=:demo gnu3.txt
```

- copy ownership of one file to another

```
chown --reference=GNU3.txt gnu3.txt
```

- (-R) Change ownership recursively

```
chown -R folder
```

## `wc` - Word Count

- **wc -l** : Prints the number of lines in a file.
- **wc -w** : prints the number of words in a file.
- **wc -c** : Displays the count of bytes in a file.
- **wc -m** : prints the count of characters from a file.
- **wc -L** : prints only the length of the longest line in a file.

```
~/code/mj/ebooks/lllinux$ wc gpl-3.0.txt
  674  5644 35149 gpl-3.0.txt
→  ~/code/mj/ebooks/lllinux$ wc -l  gpl-3.0.txt
674 gpl-3.0.txt
→  ~/code/mj/ebooks/lllinux$ wc -w  gpl-3.0.txt
5644 gpl-3.0.txt
→  ~/code/mj/ebooks/lllinux$ wc -c  gpl-3.0.txt
35149 gpl-3.0.txt
→  ~/code/mj/ebooks/lllinux$ wc -m  gpl-3.0.txt
35149 gpl-3.0.txt
→  ~/code/mj/ebooks/lllinux$ wc -L  gpl-3.0.txt
78 gpl-3.0.txt
```

## `pwd` - Present working directory

```
→ $ pwd

```

**Output**:

```
/home/mayank/code/mj/ebooks/v1/lllinux
```

## `touch` - Create Empty files

## `find` - find a file

The `find` is used searching and listing of files/folders based on conditions specified for matching in argument.

**Syntax**

```
find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]
```

**Examples**

- Find Files Using Name in Current Directory

  ```
  find . -name gnu.txt
  ```

- Find files in different folder

  ```
  find /etc -name os-release
  ```

  **Output**:

  ```
  find: ‘/etc/ssl/private’: Permission denied
  /etc/os-release
  find: ‘/etc/polkit-1/localauthority’: Permission denied
  find: ‘/etc/cups/ssl’: Permission denied
  ```

- (`-iname`) Find files in different folder Ignore case

  ```
  find /etc -iname OS-release
  ```

- (`-type`) Find Directories Using Name

  ```
  find ~ -type d -name code
  ```

- Find all `py` Files in the Directory

  ```
  find ~/code -type f -name "*.py"
  ```

- Find Files With abc Permissions

  search file with permission 777

  ```
  find . -type f -perm 0777 -print
  ```

  Search folder with permission 740

  ```
  find . -type d -perm 740 -print
  ```

- Find Files Without `abc` Permissions

```
find . -type f ! -perm 0777 -print
```

- (`-prem`) Find Read-Only Files

```
find ~ -perm /u=r
```

- Find Executable Files
- Find files with `abc` permission and changing it to `xyz`

```
find . -type f -perm 0777 -print -exec chmod 644 {} \;
```

- Find all Empty Files

```
find /tmp -type f -empty
```

- Find all Empty Directories

```
find /tmp -type d -empty
```

- Find Last 50 Days Modified Files

```
find / -mtime 50
```

- Find Changed Files in Last 1 Hour

```
find . -cmin -60
```

- Find Last 50 Days Accessed Files

```
find / -atime 50
```

- Find Size between 50MB – 100MB

```
find / -size +50M -size -100M
```

## `diff` - Diff finder

**Syntax**

```
diff <options> file1 file2
```

**Examples**

- Compare two files

```
diff GNU3.txt gnu3.txt
```

- (`-c`) Diff command output in context format

```
diff -c GNU3.txt gnu3.txt
```

- (`-u`) unified format

```
diff -u GNU3.txt gnu3.txt
```

- (`-i`) Ignore Case Sensitive

```
diff -i GNU3.txt gnu3.txt
```

- (`-b`)Ignore White Space

```
diff -i GNU3.txt gnu3.txt
```

- (-qr) Compare folders

```
❯ diff -qr  . ~/hidden/
Only in /home/mayank/hidden/: 1
Only in /home/mayank/hidden/: combined.txt
Only in /home/mayank/hidden/: dummy
Files ./gnu3.txt and /home/mayank/hidden/gnu3.txt differ
Only in /home/mayank/hidden/: GNU3.txt
Only in .: gnu3_update.txt
Only in /home/mayank/hidden/: helloworld.py
Only in /home/mayank/hidden/: highlight.css
Only in /home/mayank/hidden/: host.txt
Only in /home/mayank/hidden/: list.txt
Only in /home/mayank/hidden/: My Files
Only in /home/mayank/hidden/: pipe
Only in /home/mayank/hidden/: q
Only in /home/mayank/hidden/: redirect
Only in /home/mayank/hidden/: sort
Only in /home/mayank/hidden/: test
Only in /home/mayank/hidden/: Test
Only in .: test21.txt
Only in /home/mayank/hidden/: test2.txt
Only in /home/mayank/hidden/: test.txt
```

## `tar` - One ring to bind them all

Compress and expands

**Syntax**

```
tar [options] [archive-file] [file or directory to be archived]
```

**Options:**
-c : Creates Archive
-x : Extract the archive
-f : creates archive with given filename
-t : displays or lists files in archived file
-u : archives and adds to an existing archive file
-v : Displays Verbose Information
-A : Concatenates the archive files
-z : zip, tells tar command that creates tar file using gzip
-j : filter archive tar file using tbzip
-W : Verify a archive file
-r : update or add file or directory in already existed .tar file

**Examples**

- (`-cvf`) Creating an uncompressed tar Archive using option

```
tar cvf code.tar *.py
```

- (`-xvf`) Extracting files

```
 tar xvf code.tar
```

- (`-z`) gzip compression

```
tar czvf code.tar *.py
```

- (`-z`) Extracting a gzip tar Archive

```
tar xzvf code.tar *.py
```

- (`-j`) Creating compressed tar archive

```
tar cjvf code.tbz *.py
```

- (`-C`) Untar single tar file or specified directory

```
tar xjvf code.tbz *.py -C NewFolder
```

- (`-r`) Update existing tar

```
tar rjvf code.tbz *.py
```

- (`-tvf`) Viewing the Archive using option

```
tar -tvf code.tbz
```

## Process Control Commands

When you start a job in shell, shell itself will pause, and give control of the terminal to the program just started. Sometimes, you want to continue using the command-line, and have the job run in the background.

To create a background job, append an `&` (ampersand) to your command. This will tell shell to run the job in the background. Background jobs are very useful when running programs that have a graphical user interface.

### `jobs` -list processes

jobs prints a list of the currently running jobs and their status

**Syntax**

```
jobs [OPTIONS] [ PID | %JOBID ]
```

**Options**

- `-c` or `--command` prints the command name for each process in jobs.
- `-g` or `--group` only prints the group ID of each job.
- `-l` or `--last` prints only the last job to be started.
- `-p` or `--pid` prints the process ID for each process in all jobs.

**Examples**

In order to use `jobs` command we need to execute few jobs in the console similar to shown below

```
vim &
gnome-system-monitor &
gnome-calculator &
```

It will start three commands: one console and how GUI based programs

- To display the process ID or Job ID:

```
jobs -p %2
```

In the above it will display the `pid` of the second job.

- pids only

```
jobs -p
```

  It will display the `pids` of all the jobs.

## bg - put job to background

**Syntax:**

```
bg [PID...]
```

`bg` sends jobs to the background, resuming them if they are stopped.

A background job is executed simultaneously with shell, and does not have access to the keyboard. If no job is specified, the last job to be used is put in the background. If `PID` is specified, the jobs containing the specified process IDs are put in the background.

**Examples:**

```
bg %1
```

Will send the process 1 to background

```
bg 123 456 789
```

will send the processes with pids 123, 456, 789 to background

```
bg %1
```

will background job 1.

## fg - bring job to foreground

`fg` brings the specified job to the foreground, resuming it if it is stopped. While a foreground job is executed, fish is suspended. If no job is specified, the last job to be used is put in the foreground. If `PID` is specified, the job containing a process with the specified process ID is put in the foreground.

```
fg [PID]
```

**Example**

```
fg 113234
```

```
fg
```

This command will put the last job in the foreground.

```
fg %3
```

will put job 3 into the foreground.

## `disown` - remove a process from the list of jobs

**Syntax:**

```
disown [ PID ... ]
```

`disown` removes the specified job from the list of jobs. The job itself **continues to exist**, but shell does not keep track of it any longer.

Jobs in the list of jobs are sent a hang-up signal when shell terminates, which usually causes the job to terminate; `disown` allows these processes to continue regardless (very useful in `ssh` session).

If no process is specified, the most recently-used job is removed (like `bg` and `fg`). If one or more PIDs are specified, jobs with the specified process IDs are removed from the job list. Invalid jobs are ignored and a warning is printed.

If a job is stopped, it is sent a signal to continue running, and a warning is printed. It is not possible to use the `bg` builtin to continue a job once it has been disowned.

`disown` returns 0 if all specified jobs were disowned successfully, and 1 if any problems were encountered.

## `nohup` - Don't hang up on me

One persistent friend who will not leave you even if you hang up on him ;). As it ignore the `HUP` signal sent by the system and continues to run even when the user has logged out and as long as it remains in background only restart or `kill` will kill it.

```
nohup vim &
```

`nohup` is often used in combination with the nice command to run processes on a lower priority.

```
$ nohup nice emacs &
```

## `ps` - Snapshot of the current ~~running~~ processes

**Examples:**

- Display processes for the current shell.

  ```
  ps
  ```

  **Output:**

```
❯ ps
  PID TTY          TIME CMD
 7634 pts/1    00:00:06 fish
29629 pts/1    00:00:00 ps
```

```
❯ vim m &
❯ ps
  PID TTY          TIME CMD
 7634 pts/1    00:00:06 fish
29661 pts/1    00:00:00 vim
29708 pts/1    00:00:00 ps
```

- List all processes in BSD format

```
ps au
```

**Output:**

```
❯ ps au
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      2303  2.0  0.6 611108 95948 tty7     Ssl+ Aug19  20:14 /usr/lib/Xorg :
root      2586  0.0  0.0   5480  1640 tty2     Ss+  Aug19   0:00 /sbin/agetty tt
root      2594  0.0  0.0   5480  1716 tty6     Ss+  Aug19   0:00 /sbin/agetty tt
root      2596  0.0  0.0   5480  1856 tty1     Ss+  Aug19   0:00 /sbin/agetty -J
root      2607  0.0  0.0   5480  1820 tty3     Ss+  Aug19   0:00 /sbin/agetty tt
root      2608  0.0  0.0   5480  1812 tty5     Ss+  Aug19   0:00 /sbin/agetty tt
root      2614  0.0  0.0   5480  1632 tty4     Ss+  Aug19   0:00 /sbin/agetty tt
mayank    7463  0.0  0.0 169708  9100 pts/0    Ss+  Aug19   0:00 fish
mayank    7634  0.0  0.0 238512 13204 pts/1    Ssl  Aug19   0:06 fish
mayank   25399  0.0  0.0 161724  9196 pts/2    Ss+  07:41   0:00 fish
mayank   29661  0.1  0.0  15976 11072 pts/1    T    08:34   0:00 vim m
mayank   29753  0.0  0.0   7252  2700 pts/1    R+   08:36   0:00 ps au
```

- List all running processes in BSD format

```
❯ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2952  1724 ?        S    Aug19   0:00 /sbin/init
. . . . . .
. . . . . .
mayank   29661  0.0  0.0  15976 11072 pts/1    T    08:34   0:00 vim m
root     29742  0.0  0.0      0     0 ?        I<   08:35   0:00
[kworker/u9:0]
mayank   29791  0.0  0.0   7252  2784 pts/1    R+   08:36   0:00 ps aux
```

- (`-f`) full-format listing

```
❯ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Aug19 ?        00:00:00 /sbin/init
root         2     0  0 Aug19 ?        00:00:00 [kthreadd]
root         3     2  0 Aug19 ?        00:00:00 [rcu_gp]
root         4     2  0 Aug19 ?        00:00:00 [rcu_par_gp]
. . . . . .
. . . . . .
```

- Display User Running Processes:  all processes owned by you

```
❯ ps -x
  PID TTY      STAT   TIME COMMAND
 2679 ?        Sl     0:03 /usr/bin/gnome-keyring-daemon --daemonize --login
 2682 ?        Ssl    0:03 cinnamon-session --session cinnamon
 2689 ?        S      0:00 /usr/bin/dbus-launch --sh-syntax --exit-with-
session
. . . . . .
. . . . . .
```

- Display User Running Processes:  all processes owned by real user ID (**RUID**) or name

```
❯ ps -fU mayank
UID        PID  PPID  C STIME TTY          TIME CMD
mayank    2679     1  0 Aug19 ?        00:00:02 /usr/bin/gnome-keyring-daemon
--daemonize --login
mayank    2682  2669  0 Aug19 ?        00:00:03 cinnamon-session --session
cinnamon
mayank    2689     1  0 Aug19 ?        00:00:00 /usr/bin/dbus-launch --sh-
syntax --exit-with-session
mayank    2690     1  0 Aug19 ?        00:00:07 /usr/bin/dbus-daemon --syslog
--fork --print-pid 5 --print-address 7 --session
. . . . . .
. . . . . .
```

- (`-u`) Display User Running Processes:  all processes owned by effective user **ID** (**EUID**) or name

```
❯ ps -fu mayank
UID        PID  PPID  C STIME TTY          TIME CMD
mayank    2679     1  0 Aug19 ?        00:00:02 /usr/bin/gnome-keyring-daemon
--daemonize --login
mayank    2682  2669  0 Aug19 ?        00:00:03 cinnamon-session --session
cinnamon
mayank    2689     1  0 Aug19 ?        00:00:00 /usr/bin/dbus-launch --sh-
syntax --exit-with-session
. . . . . .
. . . . . .
```

- Display Processes by **PID**

```
❯ ps -fp 1
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Aug19 ?        00:00:00 /sbin/init
```

- Display process by (**PPID**) Parent Process ID

```
❯ ps -f --ppid 2682
UID        PID  PPID  C STIME TTY          TIME CMD
mayank    2730  2682  0 Aug19 ?        00:00:00 csd-mouse
mayank    2731  2682  0 Aug19 ?        00:00:00 csd-a11y-keyboard
mayank    2733  2682  0 Aug19 ?        00:00:00 csd-screensaver-proxy
mayank    2734  2682  0 Aug19 ?        00:00:00 csd-xsettings
mayank    2735  2682  0 Aug19 ?        00:00:00 csd-sound
mayank    2736  2682  0 Aug19 ?        00:00:00 csd-wacom
mayank    2747  2682  0 Aug19 ?        00:00:10 csd-background
```

- Display process in tree structure

```
❯ ps -e --forest
  PID TTY          TIME CMD
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00  \_ rcu_gp
 2248 ?        00:00:00 supervise-daemo
 2249 ?        00:00:00  \_ lightdm
 2303 tty7     00:20:47      \_ Xorg
 2669 ?        00:00:00      \_ lightdm
 2682 ?        00:00:04          \_ cinnamon-sessio
 2730 ?        00:00:00              \_ csd-mouse
 2731 ?        00:00:00              \_ csd-a11y-keyboa
 2768 ?        00:00:00              \_ csd-print-notif
 2901 ?        00:00:09              \_ cinnamon-launch
 2904 ?        00:20:51              |   \_ cinnamon
 6732 ?        00:32:22              |       \_ firefox-bin
 6885 ?        00:03:57              |       |   \_ Web Content
 6912 ?        00:00:08              |       |   \_ Privileged Cont
 6955 ?        00:02:56              |       |   \_ WebExtensions
22296 ?        00:00:00              |       |   \_ RDD Process
```

```
28338 ?        00:00:12              |       |   \_ Web Content
28392 ?        00:00:33              |       |   \_ Web Content
 6783 ?        00:00:07              |       \_ nemo
26391 ?        00:00:12              |       \_ VirtualBox
 2930 ?        00:00:00             \_ blueberry-obex-
 2936 ?        00:00:13             \_ nm-applet
 2939 ?        00:00:00             \_ cinnamon-killer
 2948 ?        00:00:00             \_ polkit-gnome-au
 3091 ?        00:00:04             \_ nemo-desktop
 3115 ?        00:11:10                 \_ chrome
 3121 ?        00:00:00                     \_ cat
 3122 ?        00:00:00                     \_ cat
 3132 ?        00:00:00                     \_ chrome
 3161 ?        00:12:01                     |   \_ chrome
 3133 ?        00:00:00                     \_ chrome
 3134 ?        00:00:00                     |   \_ nacl_helper
 3137 ?        00:00:01                     |   \_ chrome
 3187 ?        00:00:04                     |       \_ chrome
 3255 ?        00:01:28                     |       \_ chrome
28980 ?        00:00:00                     |       \_ chrome
 3168 ?        00:04:07                     \_ chrome
 5357 ?        00:00:18                     \_ chrome
 2701 ?        00:00:00 gvfsd-fuse
 7444 ?        00:00:31 gnome-terminal-
 7463 pts/0    00:00:00  \_ fish
 7634 pts/1    00:00:07  \_ fish
29661 pts/1    00:00:00  |   \_ vim
30836 pts/1    00:00:00  |   \_ ps
25399 pts/2    00:00:00  \_ fish
 7909 ?        00:02:02 Typora
 7913 ?        00:00:00  \_ Typora
 7947 ?        00:02:54  |   \_ Typora
 7914 ?        00:00:00  \_ Typora
 7916 ?        00:00:00  |   \_ Typora
 7965 ?        00:00:00  \_ Typora
 7976 ?        00:16:49  \_ Typora
26427 ?        00:00:06 VBoxXPCOMIPCD
26433 ?        00:00:18 VBoxSVC
26464 ?        00:02:05  \_ VirtualBoxVM
```

## `kill` - The silent killer

If you have an unresponsive program, you can terminate it manually by using the **kill** command. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.

**Syntax:**

```
kill [signal option] PID
```

There is a total of sixty-four signals that you can use, but people usually only use two signals:

- **SIGTERM (15)** — requests a program to stop running and gives it some time to save all of its progress. If you don't specify the signal when entering the kill command, this signal will be used.
- **SIGKILL (9)** — forces programs to stop immediately. Unsaved progress will be lost.

**Table**: Other common Linux and UNIX signal names and numbers

| Number | Name | Description | Trapped | Used for |
|---|---|---|---|---|
| 0 | SIGNULL (NULL) | Null | | Check access to `PID` |
| 1 | SIGHUP (HUP) | Hangup | Yes | Terminate |
| 2 | SIGINT (INT) | Interrupt | Yes | Terminate; |
| 3 | SIGQUIT (QUIT) | Quit | Yes | Terminate with core dump |
| **9** | **SIGKILL (KILL)** | Kill | No | Forced termination |
| **15** | **SIGTERM (TERM)** | Terminate | Yes | Terminate |
| **24** | **SIGSTOP (STOP)** | Stop | No | Pause the process.<br>This is **default** if signal not provided to kill command. |
| 25 | SIGTSTP (STP) | Terminal | Yes | Stop/pause the process |
| 26 | SIGCONT (CONT) | Continue | | Run a stopped process |

Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to **kill**. If you don't know the PID, simply run the command **ps ux**.

**Examples**

- Listing all kill signal names

```
kill -l
```

**Output:**

```
$:❯ kill -l
HUP INT QUIT ILL TRAP ABRT IOT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM
TERM STKFLT CHLD CLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF
WINCH IO POLL PWR SYS RT<N> RTMIN+<N> RTMAX-<N>
```

### `killall` - Kills them all

### `uname` - Linux system Details

```
$:❯ uname
Linux

~/hidden on ☁  maya.john0005@gmail.com(us-east1)
$:❯ uname -a
Linux mayank-hpprobook440g5 5.13.10-artix1-1 #1 SMP PREEMPT Fri, 13 Aug 2021
06:13:17 +0000 x86_64 GNU/Linux
```

## lsmod

To see what modules are currently loaded on your system, use the lsmod command:

```
# lsmod
```

### lsusb

```
$ lsusb
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 005: ID 06cb:00f0 Synaptics, Inc.
Bus 003 Device 019: ID 0b0e:0300 GN Netcom Jabra EVOLVE 20 MS
Bus 003 Device 003: ID 30c9:0046 Luxvisions Innotech Limited HP HD Camera
Bus 003 Device 007: ID 8087:0026 Intel Corp. AX201 Bluetooth
Bus 003 Device 022: ID 046d:c534 Logitech, Inc. Unifying Receiver
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

```

### `history` - I know what you did on command prompt

`ctrl-r search`

### `hostname` - Prints the `hostname`

### `poweroff` - Shutsdown the box

### `reboot` - Restarts the box

### `corn` and `anacorn`

### `env`

### Creating Links (`ln`)

#### Hardlink

A given inode can have any number of hard links, and the inode will persist on the file system until all the hard links disappear. When the last hard link disappears and no program is holding the file open, Linux will delete the file automatically. New hard links can be created using the ln command:

```
$ touch firstlink
$ ln firstlink secondlink
```

#### Symbolic links

In practice, symbolic links (or symlinks) are used more often than hard links. Symlinks are a special file type where the link refers to another file by name, rather than directly to the inode. Symlinks do not prevent a file from being deleted; if the target file disappears, then the symlink will just be unusable, or broken.

A symbolic link can be created by passing the -s option to ln.

```
$ ln -s secondlink thirdlink
```

## man page sections

The files that comprise manual pages are stored in **/usr/share/man** (or in **/usr/man** on some older systems). Inside that directory, you will find that the manual pages are organized into the following sections:

| **man1** | **User programs** |
|---|---|
| man2 | System calls |
| man3 | Library functions |
| man4 | Special files |
| man5 | File formats |
| man6 | Games |
| man7 | Miscellaneous |

# References

- https://unix.stackexchange.com/questions/151951/what-is-the-difference-between-rm-and-unlink
- https://askubuntu.com/questions/8653/how-to-keep-processes-running-after-ending-ssh-session
- https://en.wikipedia.org/wiki/Less_(Unix)
- https://en.wikipedia.org/wiki/Nohup
- https://en.wikipedia.org/wiki/Nice_(Unix)

# Redirection

It is a process of redirecting the steams to another file descriptors. Linux has three standard file descriptors

- `stdin`: Standard Input devices such as keyboard, mouse, files etc
- `stdout` : Standard Output file descriptors such as display screen/console, printer etc.
- `stderr`: Standard Error file descriptors such as display screen/console

Few common redirection relationship are as follows.

- redirect stdout to a file
- redirect stderr to a file
- redirect stdout to a stderr
- redirect stderr to a stdout
- redirect stderr and stdout to a file
- redirect stderr and stdout to stdout
- redirect stderr and stdout to stderr

Linux provides five operators for the redirection as discussed in following sections

## Output Redirection

The numbers are file descriptors and only the first three (starting with zero) have a standardized meaning:

```
0 - stdin
1 - stdout
2 - stderr
```

### Regular output `>` operator

This is the most commonly used redirection operators. It allows to redirect the data from left side to its right side as shown in the below example

```
❯ hostname > host.txt
❯ cat host.txt
maya-laptop-1
```

In the above example, the output of the `hostname` command will be redirected to file `host.txt`. Whose content we can view using `cat` command.

Note that redirecting to folder will result in the following error

```
❯ cat gnu3.txt > q
warning: An error occurred while redirecting file 'q'
open: Permission denied
```

## Examples:

- Output of first command to be written to another file

  ```
  ls -l > out.txt
  ```

- Saves the standard output (exclude error message) from first command to another file and it is same as first command

  ```
  ls -l /nonexistent  1> stdout.txt
  ```

- Saves the error message from first command to another file

  ```
  ls -l /nonexistent  2> errout.txt
  ```

We can redirect the data even to devices as shown in the below example

```
cat bhajan.mp3 > /dev/audio
```

Out native mp3 player ;).

One issue with `>` is that it will override the existing content. To show the effect, lets run the following command

```
❯ hostname > host.txt
❯ cat host.txt
maya-laptop-1
❯ uname -a > host.txt
❯ cat host.txt
Linux maya-laptop-1 5.13.10-artix1-1 #1 SMP PREEMPT Fri, 13 Aug 2021 06:13:17
+0000 x86_64 GNU/Linux
```

In the above example, we have executed two redirect commands first one `hostname > host.txt` will populate the content of `host.txt` file with the output of `hostname` command and the second command `uname -a > host.txt` will redirect the content of `uname -a` to the `host.txt`. You will observe that the previous content of `host.txt` file has been overwritten.

## Regular output append `>>` operator

Similar `>` it also redirects from left to right but while saving them it appends them as shown in the below example.

```
❯ hostname > host.txt
❯ cat host.txt
maya-laptop-1
❯ uname -a >> host.txt
❯ cat host.txt
mayank-hpprobook440g5
Linux maya-laptop-1 5.13.12-artix1-1 #1 SMP PREEMPT Thu, 19 Aug 2021 07:28:26
+0000 x86_64 GNU/Linux
```

In the above example, we executed `hostname > host.txt` and then executed `uname -a >> host.txt`, thus `host.txt` contains the output of both the commands

## Regular error `2>` operator

`stderr` can also be redirected to a file, so that we can log the error messages also. Linux provide `2>` operator for this reason and can be used as shown in the below example.

Lets run a command which will raise an error, say trying to list a folder which do not exist.

```
✦ ❯ ls -l /not_found
ls: cannot access '/not_found': No such file or directory
```

Now, lets try to capture it using `>` and view the content of `test.txt` file using `cat test.txt` command

```
✦ ❯ ls -l /not_found > test.txt
ls: cannot access '/not_found': No such file or directory

✦ ❯ cat test.txt
```

What we observed that `>` fails to capture the stderr output. Now lets try it with `2>` operator.

```
✦ ❯ ls -l /not_found 2> test.txt

✦ ❯ cat test.txt
ls: cannot access '/not_found': No such file or directory
```

Finally successful.

Lets see more examples

- Redirect stdout to one file and stderr to another file:

```
ls /etc/sudoers.d/ /etc/X11/ >  out.txt 2>error.txt
```

It will create `out.txt` and `error.txt` file which will contains `stdout` and `stderr` respectively.

- Many a times we need to have both `stderr` and `stdout` in the single file, we can use the following combination of operators `2>&1` to achieve it.

```
✦ ❯ ls /etc/sudoers.d/ /etc/X11/ > test.txt 2>&1
✦ ❯ cat test.txt
ls: cannot open directory '/etc/sudoers.d/': Permission denied
/etc/X11/:
xinit
xorg.conf.d
```

- Redirect both to a file (this isn't supported by all shells, bash and zsh support it, for example, but sh and ksh do not):

```
ls /etc/sudoers.d/ /etc/X11/  &> out.txt
```

## Input redirection `<`

Input redirection `<` allows us to provide arguments to the executing program as shown in the below example.

```
find ./  -printf "%f\n" > list.txt
```

`find ./  -printf "%f\n" > list.txt` command saves the name of all the files in the `list.txt` file in such a way that we have one filename per line.

Now lest sort them using `sort` command.

```
❯ find ./  -printf "%f\n" > list.txt

~/hidden via 🐍 v3.9.6 on ☁  maya.john0005@gmail.com(us-east1)
❯ sort <list.txt
./
2021-08-17_18-06Portable Network Graphic.png
20aero-india10.jpg
air
gnu3.txt
list.txt
q
tejas1.jpg
tejas_759.jpg
tejas-story,-facebook_647_070116034315.webp
test.txt~
.text
.text
.vimrc
.vimrc
```

another example can be reading file using `more` command

```
more < gnu3.txt
```

or, just first 10 lines using `head`

```
head < gnu3.txt
```

We can use both type of redirection's in the same command as shown below

```
sort < list.txt > sorted_files.txt
```

## Pipes

`Pipes` lets us redirect the output of one program to be used as input for the next, and all programs in a pipeline are run simultaneously.

```
ls -lR /etc | more
```

We are running two commands in the above example, `ls -lR /etc` and `more` the output of first is passed to the second which it process as its input.

**Examples:**

| Command | What it does |
| --- | --- |
| `ls -lt \| tail` | |
| `du -m \| sort -nr` | |
| `find . -type f -print \| wc -l` | |

## Filters

Filters take standard input and perform an operation upon it and send the results to standard output. In this way, they can be combined to process information in powerful ways. Here are some of the common programs that can act as filters:

| Program | What it does |
| --- | --- |
| `sort` | Sorts standard input then outputs the sorted result on standard output. |
| `uniq` | Given a sorted stream of data from standard input, it removes duplicate lines of data (i.e., it makes sure that every line is unique). |
| `grep` | Examines each line of data it receives from standard input and outputs every line that contains a specified pattern of characters. |
| `fmt` | Reads text from standard input, then outputs formatted text on standard output. |

| Program | What it does |
|---|---|
| `pr` | Takes text input from standard input and splits the data into pages with page breaks, headers and footers in preparation for printing. |
| `head` | Outputs the first few lines of its input. Useful for getting the header of a file. |
| `tail` | Outputs the last few lines of its input. Useful for things like getting the most recent entries from a log file. |
| `tr` | Translates characters. Can be used to perform tasks such as upper/lowercase conversions or changing line termination characters from one type to another (for example, converting DOS text files into Unix style text files). |
| `sed` | Stream editor. Can perform more sophisticated text translations than `tr`. |
| `awk` | An entire programming language designed for constructing filters. Extremely powerful. |

## Differences between `redirection` and `pipes`

- Redirection is (mostly) for files (you redirect streams to/from files).
- **Piping** is for processes: you pipe (redirect) streams from one process to another.
- Pipes have also the synchronization "side effect": they block one process (on reading) when the other has nothing to write (yet) or when reading process cannot read fast enough (when the pipe's buffer is full).
- pipelining applies to the output of process substitution, but not redirection

```
$ echo $'one\ntwo\nthree' | tee >(grep o) | cat > pipe_file
$ echo $'one\ntwo\nthree' | tee >(grep o) > redirect
$ one
two

$ cat pipe_file
one
two
three
one
two
$ cat redirect_file
one
two
three
```

- `|` pipes are not associated with an entry on disk, [therefore do not have an inode](#) number of disk filesystem (but do have inode in [pipefs](#) virtual filesystem in kernel-space), but redirections often involve  files, which do have disk entries and therefore have corresponding  inode.

- pipes are not `lseek()`'able so commands can't read some data and then rewind back, but when you redirect with `>` or `<` usually it's a file which is `lseek()` able object, so commands can navigate however they please.
- redirections are manipulations on file descriptors, which can be many; pipes have only two file descriptors - one for left command and one for right command
- redirection on standard streams and pipes are both buffered.
- pipes almost always involve forking and therefore pairs of processes are involved; redirections - not always, though in both cases resulting file descriptors are inherited by sub-processes.
- pipes always connect file descriptors (a pair), redirections - either use a pathname or file descriptors.
- pipes are Inter-Process Communication method, while redirections are just manipulations on open files or file-like objects
- both employ `dup2()` syscalls underneath the hood to provide copies of file descriptors, where actual flow of data occurs.
- redirections can be applied "globally" with `exec` built-in command ( see this and this ), so if you do `exec > output.txt` every command will write to `output.txt` from then on. `|` pipes are applied only for current command (which means either simple command or subshell like `seq 5 | (head -n1; head -n2)` or compound commands (but also please note that for such compound commands the amount of bytes that `read()` consumes will influence how much data is left on the sending end of the pipe for other commands inside the read end of the pipe ).
- When redirection is done on files, things like `echo "TEST" > file` and `echo "TEST" >> file` both use `open()` syscall on that file (see also) and get file descriptor from it to pass it to `dup2()`. Pipes `|` only use `pipe()` and `dup2()` syscall.
- Redirections involve file and directory permissions; anonymous pipes typically do not involve permissions (i.e. whether or not you can or cannot create a pipe), but named pipes ( made with `mkfifo` ) do involve typical file permissions and read-write-execute bits.
- As far as commands being executed, pipes and redirection are no more than file descriptors - file-like objects, to which they may write blindly, or manipulate them internally (which may produce unexpected behaviors; `apt` for instance, tends to not even write to stdout if it knows there's redirection).

## When to use

use a pipe for command to command and use the redirect if outputting to or from a file

## References

- https://askubuntu.com/questions/172982/what-is-the-difference-between-redirection-and-pipe/172989

# Appendix

## References

- Bash Guide for Beginners: https://tldp.org/LDP/Bash-Beginners-Guide/html/
- Bash Guide: https://www.gnu.org/software/bash/manual/bash.pdf
- Debugging: https://unix.stackexchange.com/questions/155551/how-to-debug-a-bash-script
- https://en.wikipedia.org/wiki/Desktop_environment
- https://en.wikipedia.org/wiki/Comparison_of_command_shells
- https://www.funtoo.org/Linux_Fundamentals,_Part_1

## Citations: Images used

- Linux. (2023, April 3). In *Wikipedia*. https://en.wikipedia.org/wiki/Linux

## Cheat Sheets

### Linux commands: System

| uname | Displays Linux system information |
|---|---|
| `uname -r` | Displays kernel release information |
| `uptime` | Displays how long the system has been running including load average |
| `hostname` | Shows the system hostname |
| `hostname -i` | Shows the system hostname |
| `hostname -i` | Displays the IP address of the system |
| `last reboot` | Shows system reboot history |
| `date` | Displays current system date and time |
| `timedatectl` | Query and change the System clock |
| `cal` | Displays the current calendar month and day |
| `w` | Displays currently logged in users in the system |
| `whoami` | Displays who you are logged in as finger |
| `username` | Displays information about the user |

## Common Re-directions

| Redirection | Action |
|---|---|
| `> filename` | Redirect standard out to a new file |
| `>> filename` | Append standard out to an existing file |
| `1> filename` | Redirect standard out to a new file |
| `1>> filename` | Append standard out to an existing file |
| `2> filename` | Redirect standard error to a new file |
| `2>> filename` | Append standard error to an existing file |
| `&> filename` | Redirect standard out and standard error to a new file |
| `&>> filename` | Append standard out and standard error to an existing file |
| `> filename 2>&1` | Redirect standard out and standard error to a new file |
| `>> filename 2>&1` | Append standard out and standard error to an existing file |

## Common command list

| Command | Meaning |
|---|---|
| **a2ps** | Format files for printing on a PostScript printer. |
| **acroread** | PDF viewer. |
| **adduser** | Create a new user or update default new user information. |
| **alias** | Create a shell alias for a command. |
| **anacron** | Execute commands periodically, does not assume continuously running machine. |
| **apropos** | Search the whatis database for strings. |
| **apt-get** | APT package handling utility. |
| **aspell** | Spell checker. |
| **at, atq, atrm** | Queue, examine or delete jobs for later execution. |
| **aumix** | Adjust audio mixer. |
| **(g)awk** | Pattern scanning and processing language. |
| **bash** | Bourne Again SHell. |

| Command | Meaning |
| --- | --- |
| **batch** | Queue, examine or delete jobs for later execution. |
| **bg** | Run a job in the background. |
| **bitmap** | Bitmap editor and converter utilities for the X window System. |
| **bzip2** | A block-sorting file compressor. |
| **cat** | Concatenate files and print to standard output. |
| **cd** | Change directory. |
| **cdp/cdplay** | An interactive text-mode program for controlling and playing audio CD Roms under Linux. |
| **cdparanoia** | An audio CD reading utility which includes extra data verification features. |
| **cdrecord** | Record a CD-R. |
| **chattr** | Change file attributes. |
| **chgrp** | Change group ownership. |
| **chkconfig** | Update or query run level information for system services. |
| **chmod** | Change file access permissions. |
| **chown** | Change file owner and group. |
| **compress** | Compress files. |
| **cp** | Copy files and directories. |
| **crontab** | Maintain crontab files. |
| **csh** | Open a C shell. |
| **cut** | Remove sections from each line of file(s). |
| **date** | Print or set system date and time. |
| **dd** | Convert and copy a file (disk dump). |
| **df** | Report file system disk usage. |
| **dhcpcd** | DHCP client daemon. |
| **diff** | Find differences between two files. |
| **dig** | Send domain name query packets to name servers. |
| **dmesg** | Print or control the kernel ring buffer. |
| **du** | Estimate file space usage. |

| Command | Meaning |
| --- | --- |
| **echo** | Display a line of text. |
| **ediff** | Diff to English translator. |
| **egrep** | Extended grep. |
| **eject** | Unmount and eject removable media. |
| **emacs** | Start the Emacs editor. |
| **exec** | Invoke subprocess(es). |
| **exit** | Exit current shell. |
| **export** | Add function(s) to the shell environment. |
| **fax2ps** | Convert a TIFF facsimile to PostScript. |
| **fdformat** | Format floppy disk. |
| **fdisk** | Partition table manipulator for Linux. |
| **fetchmail** | Fetch mail from a POP, IMAP, ETRN or ODMR-capable server. |
| **fg** | Bring a job in the foreground. |
| **file** | Determine file type. |
| **find** | Find files. |
| **formail** | Mail (re)formatter. |
| **fortune** | Print a random, hopefully interesting adage. |
| **ftp** | Transfer files (unsafe unless anonymous account is used!)services. |
| **galeon** | Graphical web browser. |
| **gdm** | Gnome Display Manager. |
| **(min/a)getty** | Control console devices. |
| **gimp** | Image manipulation program. |
| **grep** | Print lines matching a pattern. |
| **grub** | The grub shell. |
| **gv** | A PostScript and PDF viewer. |
| **gzip** | Compress or expand files. |
| **halt** | Stop the system. |

| Command | Meaning |
| --- | --- |
| **head** | Output the first part of files. |
| **help** | Display help on a shell built-in command. |
| **host** | DNS lookup utility. |
| **httpd** | Apache hypertext transfer protocol server. |
| **id** | Print real and effective UIDs and GIDs. |
| **ifconfig** | Configure network interface or show configuration. |
| **info** | Read Info documents. |
| **init** | Process control initialization. |
| **iostat** | Display I/O statistics. |
| **ip** | Display/change network interface status. |
| **ipchains** | IP firewall administration. |
| **iptables** | IP packet filter administration. |
| **jar** | Java archive tool. |
| **jobs** | List backgrounded tasks. |
| **kdm** | Desktop manager for KDE. |
| **kill(all)** | Terminate process(es). |
| **ksh** | Open a Korn shell. |
| **ldapmodify** | Modify an LDAP entry. |
| **ldapsearch** | LDAP search tool. |
| **less** | **more** with features. |
| **lilo** | Linux boot loader. |
| **links** | Text mode WWW browser. |
| **ln** | Make links between files. |
| **loadkeys** | Load keyboard translation tables. |
| **locate** | Find files. |
| **logout** | Close current shell. |
| **lp** | Send requests to the LP print service. |

| Command | Meaning |
| --- | --- |
| **lpc** | Line printer control program. |
| **lpq** | Print spool queue examination program. |
| **lpr** | Offline print. |
| **lprm** | Remove print requests. |
| **ls** | List directory content. |
| **lynx** | Text mode WWW browser. |
| **mail** | Send and receive mail. |
| **man** | Read man pages. |
| **mcopy** | Copy MSDOS files to/from Unix. |
| **mdir** | Display an MSDOS directory. |
| **memusage** | Display memory usage. |
| **memusagestat** | Display memory usage statistics. |
| **mesg** | Control write access to your terminal. |
| **mformat** | Add an MSDOS file system to a low-level formatted floppy disk. |
| **mkbootdisk** | Creates a stand-alone boot floppy for the running system. |
| **mkdir** | Create directory. |
| **mkisofs** | Create a hybrid ISO9660 filesystem. |
| **more** | Filter for displaying text one screen at the time. |
| **mount** | Mount a file system or display information about mounted file systems. |
| **mozilla** | Web browser. |
| **mt** | Control magnetic tape drive operation. |
| **mtr** | Network diagnostic tool. |
| **mv** | Rename files. |
| **named** | Internet domain name server. |
| **ncftp** | Browser program for ftp services (insecure!). |
| **netstat** | Print network connections, routing tables, interface statistics, masquerade connections, and multi-cast memberships. |
| **nfsstat** | Print statistics about networked file systems. |

| Command | Meaning |
| --- | --- |
| **nice** | Run a program with modified scheduling priority. |
| **nmap** | Network exploration tool and security scanner. |
| **ntsysv** | Simple interface for configuring run levels. |
| **passwd** | Change password. |
| **pdf2ps** | Ghostscript PDF to PostScript translator. |
| **perl** | Practical Extraction and Report Language. |
| **pg** | Page through text output. |
| **ping** | Send echo request to a host. |
| **pr** | Convert text files for printing. |
| **printenv** | Print all or part of environment. |
| **procmail** | Autonomous mail processor. |
| **ps** | Report process status. |
| **pstree** | Display a tree of processes. |
| **pwd** | Print present working directory. |
| **quota** | Display disk usage and limits. |
| **rcp** | Remote copy (unsafe!) |
| **rdesktop** | Remote Desktop Protocol client. |
| **reboot** | Stop and restart the system. |
| **renice** | Alter priority of a running process. |
| **rlogin** | Remote login (telnet, insecure!). |
| **rm** | Remove a file. |
| **rmdir** | Remove a directory. |
| **rpm** | RPM Package Manager. |
| **rsh** | Remote shell (insecure!). |
| **scp** | Secure remote copy. |
| **screen** | Screen manager with VT100 emulation. |
| **set** | Display, set or change variable. |

| Command | Meaning |
|---|---|
| **setterm** | Set terminal attributes. |
| **sftp** | Secure (encrypted) ftp. |
| **sh** | Open a standard shell. |
| **shutdown** | Bring the system down. |
| **sleep** | Wait for a given period. |
| **slocate** | Security Enhanced version of the GNU Locate. |
| **slrnn** | text mode Usenet client. |
| **snort** | Network intrusion detection tool. |
| **sort** | Sort lines of text files. |
| **ssh** | Secure shell. |
| **ssh-keygen** | Authentication key generation. |
| **stty** | Change and print terminal line settings. |
| **su** | Switch user. |
| **tac** | Concatenate and print files in reverse. |
| **tail** | Output the last part of files. |
| **talk** | Talk to a user. |
| **tar** | Archiving utility. |
| **tcsh** | Open a Turbo C shell. |
| **telnet** | User interface to the TELNET protocol (insecure!). |
| **tex** | Text formatting and typesetting. |
| **time** | Time a simple command or give resource usage. |
| **tin** | News reading program. |
| **top** | Display top CPU processes. |
| **touch** | Change file timestamps. |
| **traceroute** | Print the route packets take to network host. |
| **tripwire** | A file integrity checker for UNIX systems. |
| **twm** | Tab Window Manager for the X Window System. |

| Command | Meaning |
|---|---|
| **ulimit** | Controll resources. |
| **umask** | Set user file creation mask. |
| **umount** | Unmount a file system. |
| **uncompress** | Decompress compressed files. |
| **uniq** | Remove duplicate lines from a sorted file. |
| **update** | Kernel daemon to flush dirty buffers back to disk. |
| **uptime** | Display system uptime and average load. |
| **userdel** | Delete a user account and related files. |
| **vi(m)** | Start the vi (improved) editor. |
| **vimtutor** | The Vim tutor. |
| **vmstat** | Report virtual memory statistics. |
| **w** | Show who is logged on and what they are doing. |
| **wall** | Send a message to everybody's terminal. |
| **wc** | Print the number of bytes, words and lines in files. |
| **which** | Shows the full path of (shell) commands. |
| **who** | Show who is logged on. |
| **whoami** | Print effective user ID. |
| **whois** | Query a whois or nicname database. |
| **write** | Send a message to another user. |
| **xauth** | X authority file utility. |
| **xcdroast** | Graphical front end to cdrecord. |
| **xclock** | Analog/digital clock for X. |
| **xconsole** | Monitor system console messages with X. |
| **xdm** | X Display Manager with support for XDMCP, host chooser. |
| **xdvi** | DVI viewer. |
| **xfs** | X font server. |
| **xhost** | Server access control program for X |

| Command | Meaning |
| --- | --- |
| **xinetd** | The extended Internet services daemon. |
| **xload** | System load average display for X. |
| **xlsfonts** | Server font list displayer for X. |
| **xmms** | Audio player for X. |
| **xpdf** | PDF viewer. |
| **xterm** | Terminal emulator for X. |
| **zcat** | Compress or expand files. |
| **zgrep** | Search possibly compressed files for a regular expression. |
| **zmore** | Filter for viewing compressed text. |

# Quests

## Basics of Linux

- What is Linux?
- How Linux and UNIX are different and similar?
- Explain briefly about its history?
- What are the basic components of Linux architecture
- Core of the Linux operating system is called?
- Discuss the process management system calls in Linux.
- What is the use of `LILO` or `GRUB`
- Explain the Linux directory structure.
- Explain permissioning in Linux
- Explain the differences between `cron` and `anacron`

## Basics of GNU Utility Commands

- What is BASH
- Which Linux commands can be used to view directory.
- When to use the `cd` command.
- What is the use of `cat` command
- Difference between `tail` and `head`
- Explain `PIPE`'s' in Linux
- What does following folders contain
    - `/etc`
    - `/var`
    - `/sbin`
    - `/usr`
    - `/opt`
- List /etc folder using linux command
- List /etc folder recursive using linux command
- List all the folders in `/var` using linux command
- List only files in `/var/log` folder using linux command
- Copy files from one folder to another using linux command
- Copy recursively one folder to another folder.
- Create folder `test1/test2/test3` using `mkdir` command

- Create folders `test1/test2/testa`, `test1/test2/testb` and `test1/test2/testc` using `mkdir` command
- View last 14 lines of a text file
- View first 14 lines of a text file
- View last 14 lines of a text file and search for a word in it.
- Find files with abc extension who have size between 2k to 5k
- Find all executable files in /usr folder
- Change ownership of a file to root and back
- Change file permission to 766 and then change it back to original
- Find all lines where "at" word is present in a file
- Name different commands that are available to check the disk usage.
- Explain ways to create shortcuts in Linux.
- How to check whether a link is a hard one or a soft link?
- How Hard link and Soft Link are different.
- Can we identify hard link file
- What happens to hark linked file, when original file is deleted.
- What happens to soft linked file, when original file is deleted.
- What are `sed` and awk commands used for?
- How to use pipe commands?
- Name the alternative command for echo.
- Create a shortcut for /tmp folder.
- What does `alias` command do. Explain in details
- Give the file's name where shell programmes are kept.
- What makes a hard link different from a soft link?
- what does `.` (dot) indicate at the beginning of the file name also explain its usecase?
- How do you work with processes and signals in a shell script?
- Write a command sequence to count the words in a given file.
- What command can you use to most effectively monitor a log file that is constantly updating?
- What are the four key elements of each Linux file system?
- What is a kernel
- How do you use regular expressions in a shell script?
- How do you use conditional statements (e.g., if, case) in a shell script?
- How do you handle errors and exceptions in a shell script?
- How do you work with files and directories in a shell script?
- What are some common pitfalls when writing shell scripts?
- How do you pass arguments to a shell script and how do you access them within the script?

- What does the. (dot) indicate at the beginning of a file name and how should it be listed?
- What is the alternative command available to `echo` and what does it do?
- Which command needs to be used to know how long the system has been running?
- What are the various commands that may be used to check the disc usage?
- Explain in brief about `awk` command with an example.
- What are the three different security provisions provided by UNIX for a file or data?
- Explain in brief about `sed` command with an example.
- What function does a pipe operator serve? How can you run several commands in one line?
- Differentiate between `grep` and `egrep`
- List 10 commonly used Linux commands