The Linux Journal

1st Edition

The Linux Journal

Author
Suresh Mishra

About the book

This book is written by Suresh Mishra targeting all noob linux users out there. This book covers all the necessary things that are required to a noob linux user. With well structure and simple langauge, this book comes up with a pin to point guide towards command line.

The main motive of this book is to make people more friendly towards Linux and help them to understand it easily. Once you read this book you’ll become comfortable with the Linux ecosystem.

About the author

Suresh Mishra is a passionate programmer of the new age world, who loves to explore different fields of Technology. Linux is one of his favourite subject which he loves to explore a lot. Apart from this, he had a keen interest in Python, Java, Flutter Development and Cybersec. His main goal behind writting this book is to enlight the world with Linux.

What is Linux?

Linux is a **kernel**, which is a main core of functional system and this kernel is a **UNIX** like kernel and **Linus Torvalds** created this kernel. This kernel is purely made from scratch with the help of **C** and **Assembly** language.

What is GNU and GNU/Linux?

When Linus Torvalds created only kernel, during that time **Richard Stallman** launched a project called **GNU project** which is actually a goal for creating Operating System that are open source and free.

The GNU is a group of tools, libraries, input devices,etc that are required to build an Operating System. GNU stands for “**GNU’S NOT UNIX**”. The main thing in GNU is, these all GNU based Operating Systems use Linux as it’s kernel. So it is said to be as **GNU/Linux**.

GNU/Linux Distribution

The distribution of GNU/Linux is something like this:

- **Parent Operating System:** Debian, Slackware,RedHat and Arch

Parent Operating Systems are purely made using the **GNU tools and libraries and Linux Kernel**.

- Operating Systems derived from the parent Operating Systems:
  - Ubuntu, Kali Linux, etc are derived from **Debian**.
  - Open SUSE, etc are derived from **Slackware**.
  - Cent OS, Fedora, etc are derived from **RedHat**.
  - Manjaro, Black Arch, Arch Labs, etc are derived from **Arch**.

As it is so hard task to write and develop and complete Operating System, developers and engineers choose a parent

Operating System and derive a new Operating System from it. For example, Zorin OS, Pop OS, Linux Mint, etc are derived from Ubuntu.

Desktop Environment

Desktop Environment is nothing but the UI (User Interface) for an Operating System. This provides a **Graphical User Interface** to an Operating System.

There are a tons of Desktop Environment out in the market, some of them are **Gnome, KDE, XFCE, Mate,etc**.

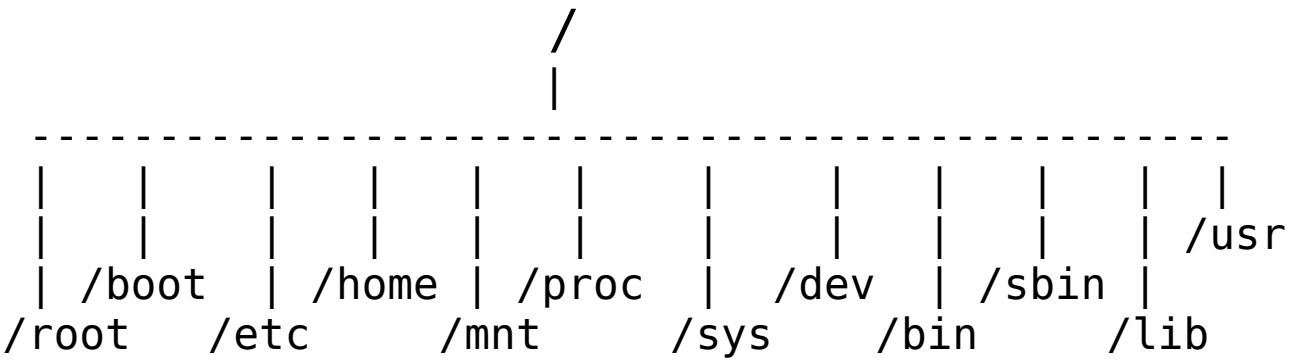
The Shell

The shell is basically a program that takes your commands from the keyboard and sends them to the Operating System to perform.

The “Terminal” is the GUI program that launch a shell for the user. Depending on the distribution, the shell prompt might change, but for the most part it should adhere the following format:

username@hostname: current_directory
darkwill@pc:/home/darkwill$

The Linux FileSystem



The root (/) of the filesystem is at the top of the tree, and the following are the most important subdiretories are:

- /root :- home directory of superuser
- /etc :- contains the Linux configuration files
- /home :- user’s home directory
- /mnt :- where other filesystem are mounted to the filesystem
- /media:- where CDs or USB devices are mounted to the filesystem
- /bin :- application binaries
- /lib :- where we will find libraries
- /boot :- kernel image
- /proc :- view of internal kernel data
- /dev :- special device files
- /sys :- kernel view of hardware

Everything in Linux is a file, as you journey deeper into Linux you'll understand this. Every file is organized in a **hierarchical directory tree**. The first directory is the root directory. The root directory has many folders and files which you can store more folders and files.

```
/
|--bin
|   |--file1
|   |--file2
|--etc
|   |--file3
|   `--directory1
|           |--file4
|           `--file5
|--home
|--var
```

This is how a directory tree look like

The loaction of these files and directories are referred to as **paths**. If

you had a folder named with ‘home’ with a folder named ‘myfiles’ and another folder in that folder named ‘Projects’, that path would look like:

/home/myfiles/Projects

To see where you are, you can use the pwd command, it just shows you which directory you are in.

Just remember we need to navigate our way using paths. There are two different ways to specify a path, with absolute and relative paths.

- **Absolute Path** is the path from the root directory. The root is the head honcho. The root directory is commonly shown as slash(‘/’). Every time the path starts with /, it means we are starting from the root directory. For example,

  /home/invisionchip/Desktop

- **Relative Path** is the path where you are currently in filesystem. If you

are in location /home/invisionchip /Documents and wanted to get to a directory inside Documents called projects, you don’t have to specify the whole path from root like /home/ invisionchip/Documents/projects, you can just go to /projects instead.

To change directory we have **cd** command:

```
$ cd /home/invisionchip/Documents
```

For easy navigation with absolute and relative path we have some shorcuts:

- cd. (current directory):- The directory you are currently in.
- cd.. (parent directory):- Takes to the directory above the current directory.
- cd~ (home directory):- Takes to the home directory.
- cd- (previous directory):- Takes to the previous directory.

To list directories and files in the current directory by default we have **ls** (list directory) command:

```
$ ls /home/invisionchip/Images
```

To create new empty files we have **touch** command:

```
$ touch file1
```

It is also used to change timestamps on existing files and directories.

cat (concatenate) command is used to read a file. It is not only displays file contents but it can combine multiple files and show the output of them:

```
$ cat file1 file2
```

It’s not great for viewing large file and it’s only meant for short content.

For copying files from a directory we have **cp** (copy) command:

```
$ cp file1 /home/invisionchip
```

Similarly, we have **mv** (move) command

for moving files from a directory:

```
$ mv file1 /home/invisionchip
```

It can also be used to rename files:

```
$ mv oldfile newfile
```

There is a command, **mkdir** which is used for making a new directory or folder:

```
$ mkdir folder_name
```

It can also be used to create subdirectories or subfolders at a time:

```
$ mkdir -p books/favourites
```

To remove any kind of file, we have **rm** (remove) command:

```
$ rm mytext.txt
```

For removing directories, we use **rmdir** (remove directory):

```
$ rmdir directory_name
```

If we’re looking for any specific directories or files, then we can use the **find** command to do so:

```
$ find /home -d -name Projects

$ find /home -name game.py
```

To get information about our diskspace usage, you can use the **df** (display filesystem) command:

```
$ df
```

To get the size of a directory and all it’s subdirectories, we can use the **du** (direct usage) command:

```
$ du
```

For getting information about the maount of free space available in our system, we have the **free** command:

```
$ free
```

All the above explained commands are the

file management commands, now let’s talk about other helpful command which are helpful in various ways.

If you want to view all the previously used commands then you have **history** command, which let’s you to see what commands you have used previously:

```
$ history
```

To display the processess using the most system resources at any given time we can use the **top** command:

```
$ top
```

For clearing the terminal screen, we use the **clear** command:

```
$ clear
```

To learn about all built-in commands we have a command called **help**:

```
$ help
```

If we’re stuck at anypoint and want a detail guide the we need to use the **man** command. It displays a manual page. Manual Pages are usually very detailed and it’s recommended that you read the man pages for any command you are unfamiliar with.

```
$ man
```

Similar to man command, we have another command called **info**, which provides more detailed or precies imformation about a specific command:

```
$ info
```

To get the current date and time we can use the **time** command:

```
$ time
```

If we’re using our machine for a long while and want to know about how long our system is running, then we have **uptime** command which provides imformation about how long the system

has been running in one line:

```
$ uptime
```

To display the calender of the current month, we hae **cal** command:

```
$ cal
```

To display the detailed informarion about the users who are logged in the system currently, we use **w** command:

```
$ w
```

To change the current user password, we have **passwd** command:

```
$ passwd
```

To exit from the terminal window, we use the **exit** command:

```
$ exit
```

The **username** command provides a wide range of basic information about the system.

```
$ username -a
```

To see the programs that are running on the machine, we use the **ps** (processes) command:

```
$ ps
```

Now let’s talk about various commands used for networking.

The **ping** command is used to check if a remote system is running or up. It is used to detect whether a system is connected to the network or not.

```
$ ping url_of_website
```

Instead of using the url of website or domain, we can also use IP Address.

The **host** command is used to obtain netowrk address information about a remote system connected to our network. This information usually consists of system’s IP Address, domain name address and sometimes mail server also.

```
$ host domain_name
```

The **traceroute** command is used to track the sequence of computer networks. We can track to check the route through which we are connected to a host.

```
$ traceroute domain_name
```

The **netstat** command is used to check the status of ports whether they are open, closed, waiting and masquerade connections. It displays connection information, routing, table information, etc.

```
$ netstat
```

The **tracepath** is very similar function to that of traceroute. It takes complex options whereas traceroute can’t.

```
$ tracepath domain_name
```

The **dig** (Domain Information Grooper) command is used to solve DNS related queries.

```
$ dig domain_name
```

The **hostname** command is to see the hostname of a computer.

```
$ hostname
```

To display or modify the routing table, we use the **route** command:

```
$ route -n
```

For finding out DNS related query or testing and troubleshooting DNS server, we use the **nslookup** command:

```
$ nslookup doamin_name
```

The Superuser

The **superuser**, also known as **root user** is a special user account in Linux used for system administration. It is the most previleged user on the Linux system and it has access to all the commands and files. The superuser can do many things that an ordinary user can’t, such as installing new softwares, changing the ownership of files and managing other user accounts. It is not recommended to use root for ordinary tasks, such as browsing the web, writting text, etc.

The **sudo** command is used to get root previlleges.

Packages and Package Management

Packages in Linux refers to a compressed file archive containing all of the files that comes up with a particular application.

A **package manager** is a collection of software tools that automate the process of installing, upgarding, configuring, and removing software. It maintains a database of information about installed packages that enables the package manager to uninstall software, establish whether a new package’s dependencies have been met, and determine whether a package you’re traying to install has already been installed or not.

There are three versions of Package managers:

- RPM Package Manager
- Debian Package Manager
- yum Package Manager

RPM Package Manager

RPM stands for **Red-Hat Package Manager**. Red-Hat and all the Red-Hat based

distributions such as Cent OS, Fedora, OpenSUSE, etc use RPM package manager. It is used to build, install, verify, update, and uninstall softwares in these distributions.

Debian Package Manager

Debian packages are adopted by several Linux distributions, including Debian itself and all Debian based distributions such as Ubuntu, Linux Mint, Kali Linux, etc. These packages usually have the .deb extensions. To install, remove, or list Debian Packages the **dpkg** command is used.

yum Package Manager

yum stands for **Yellowdog Upadater, Modified**. It is a command-line package manager for RPM-based Linux distributions like Fedora, Cent OS, Red-Hat. It enables us to install a package and all its dependencies, delete a package, upgrade existing packages, search for packages,etc.

Advance Packaging Tool (APT)

Advance Packaging Tool (**APT**) is a package manager originallly designed for Debian as a frontend for the dpkg

utility. It is used to install or upgarde all necessary dependent applications so that .deb packages can be installed.

The APT suite comes up with couples of programs. The two most commonly used are:

- apt-cache
- apt-get

apt-cache command provides information about the Debian Package database.

apt-get command is used to install, upgrade, or remove software packages in Debian and Debian based Linux Distributions.

To obtain updated information about packages available from the installation sources we use the apt-get update command:

```
$ sudo apt-get update
```

To upgrade all installed packages, we use the apt-get upgrade command:

```
$ sudo apt-get upgrade
```

To install a package by its name we use the apt-get install command:

```
$ sudo apt-get install package_name
```

To remove a package by its name we use the apt-get remove command:

```
$ sudo apt-get remove package_name
```

To check the package database for consistency and broken package installations we use the apt-get check command:

```
$ sudo apt-get check
```

To remove unused package files we use the apt-get clean command:

```
$ sudo apt-get clean
```

Thank you reading this book, hope you gain some knowledge about Linux