

Unit 3.1. Basic GNU/Linux

Héctor D. Menéndez

Endless Science
endsci.net

Index

- 1 Introduction
- 2 GNU/Linux as an Operative System
- 3 Commands
- 4 Exercises

Index

- 1 Introduction
- 2 GNU/Linux as an Operative System
- 3 Commands
- 4 Exercises

Introduction

What is GNU/Linux?

GNU/Linux is, in simplest terms, an operating system. It is the software on a computer that enables the applications and the computer operator to access the devices and perform their desired functions.

The operating system (OS) runs the applications' instructions through the computer's processor. The processor performs the instructed task, then sends the results back to the application via the operating system.

Where is Linux?

Linux, which began its existence as a server OS (using GNU), has become useful as a desktop OS (also using GNU), and it can also be used on several devices. Some examples are:

- Android.
- TiVo Digital Video Recorder.
- PS4.
- Sony Bravia TV.
- Yamaha Motif Keyboard.

Index

- 1 Introduction
- 2 GNU/Linux as an Operative System**
- 3 Commands
- 4 Exercises

GNU/Linux Architecture

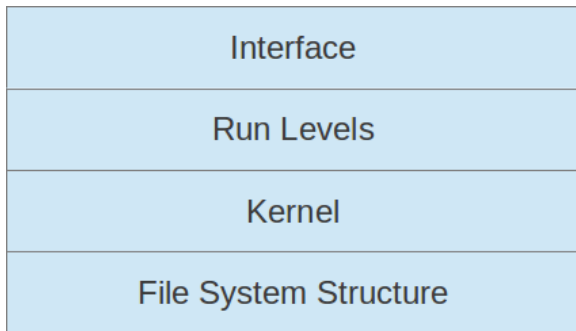
Files Structure: This part contains the directories and files. They are used by the kernel, applications and users.

Kernel: It manages input/output requests from software and translates them into data processing instructions for the central processing unit and other electronic components of a computer.

Run Levels: OS mode operation.

Interface: Used by the users to interact with the computer. It allows the user to execute task through programs. There are text and graphical user interfaces.

GNU/Linux Architecture



Filesystem Structure

File: It is a set of bits uploaded in a device. It is identified by a name and the folder description which contains it.

Folder: It is a virtual container where the files (or other folders) are grouped according to their information, goal, or any user criteria.

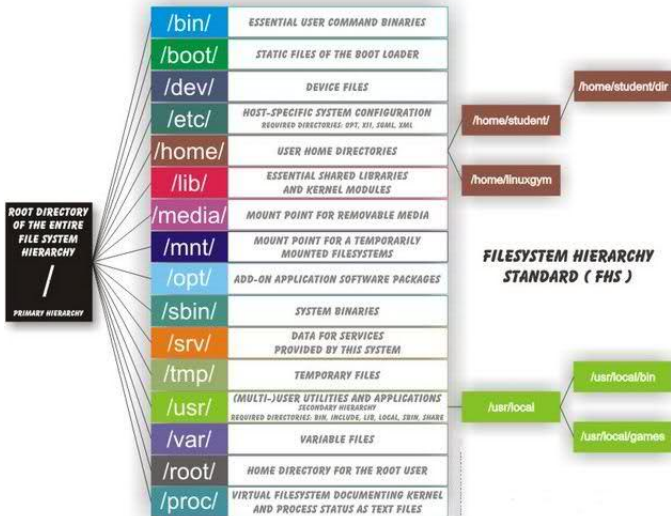
Filesystem Structure

Root Folder “/”: Root of the directory tree used by the OS.

Standard folders (directory tree):

- `/bin`: essential user command binaries.
- `/etc`: specific system configuration.
- `/var`: variable files.
- `/home`: user home directories.
- `/usr`: user utilities and applications.
- `/dev`: device files.
- `/mnt` and `/media`: mount points.
- `/lib`: essential shared libraries and kernel modules.

Filesystem Structure



The Kernel (Linux)

Multi-user: Several users can use the system at the same time.

Multi-task: users can run several applications at the same time.

Multi-platform: it is able to use different hardware architectures: Pentium III, Pentium 4, Pentium Dual Core, AMD, etc.

Multi-processor: it use several processors.

Process Memory Restriction: to protect the memory associated to each program.

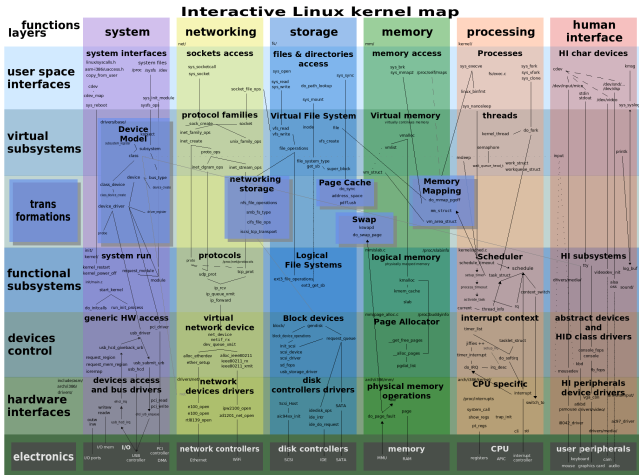
Executable load demand: the system only loads the parts of a program which are currently used.

The Kernel: Processes

Process: it is an instance of a computer program that is being executed. It contains the program code, the associated resources and its current activity (or state).

Thread o Subprocess: it is the smallest sequence of program instructions that can be managed independently by an operating system scheduler.

The Kernel Structure



Run Levels

Run Level	Name	Description
0	<i>Halt</i>	Shuts down all services when the system will not be rebooted.
1	<i>Single User</i>	Used for system maintenance. No Networking capabilities.
2	<i>MultiUser</i> <i>No Network Support</i>	Used for maintenance and system testing.
3	<i>MultiUser</i> <i>Network Support</i>	Non-Graphical Text Mode operations for server systems.
4	-	Custom Mode, used by SysAdmin
5	<i>Graphical</i> <i>X11</i>	Graphical login with same usability of Run Level 3.
6	<i>Reboot</i>	Shuts down all services when the system is being rebooted.

Run Levels: Users and Groups

User: is an agent, either a human agent (end-user) or software agent, who uses a computer or network service. Every user has an user account (login) and a password. The types of user are:

- **Normal users**: They have access to certain applications and data.
- **Root**: It has access and permission (privileges) to perform any operation on the system.
- **Daemon**: It is a program that runs in the background, waiting for certain events to occur and offering services. In GNU / Linux daemons have an associated user.

Run Levels: Users and Groups

Group: Users can be assembled into a “ group”, and, likewise, may choose to join an existing group to use the access privileges granted to that group.

Run Levels: Permissions

Linux lets you to restrict access to elements of the directory tree system (files or directories) with different criteria.

You can set permissions indicating the different ways users are allowed to use files.

There are three types of permits:

- 1 Read (r):** allows to read the file contents.
- 2 Write (w):** allows to write the file contents.
- 3 Execution (x):** allows to execute the program file.

Run Levels: Permissions

These permissions also refer to directories and, in this case, the meaning is:

- 1 Read (r):** it enables to list the content of a directory.
- 2 Write (w):** it enables to add or remove contents in a directory.
- 3 Execution (x):** similar to read permission. it also provides access to the content of the files included in the directory.

Run Levels: Permissions

Permissions can be applied at three levels:

- 1 user (u):** which refers to the person who owns the items (files or directories), which is usually who created it.
- 2 group (g):** which refers to the user group established by the system administrator, it includes the owner of the item.
- 3 other (o):** it refers to any other user of the system which is not included in the previous two sets.

Run Levels: Accesso a Red

Linux allows to connect to the Internet when the Network access level is activated.

The kernel network module is loaded and allows the configuration.

The system allows you to set an IP, router and DNS server settings that normally are configured by the user.

It also has interfaces for network protocols at all levels (sockets).

Types of Interfaces

Text interfaces (shell or command line): Fast and very efficient, very useful for performing repetitive tasks automatically. It uses commands to communicate with the system. Examples: bash, ash, csh, Zsh, ksh, tcsh.

Graphical interfaces: For inexperienced users. More simple to operate but with limited functionalities. Used: icons, menus, windows, etc.. Examples: KDE, Gnome and Xfce.

Interfaces: System access

They are used to provide access to the user.

This interface requires the user to log in before using the system.

Index

- 1 Introduction
- 2 GNU/Linux as an Operative System
- 3 Commands**
- 4 Exercises

Commands in Linux

Command: A command is an instruction or order that the user provides to the system, from the command line (shell) or a system call. Can be internal (contained in the own interpreter) or external (contained in an executable file).

Command execution in Linux

The terminal displays a text string (configurable), called system prompt.

Indicates that the shell is waiting for user commands, usually includes character '\$', '>', '# '.

To execute a command, write it and push <enter>.

The command structure is:

- An order (word known to the system),
- Some parameters might be added to the command. Some arguments are used to change the basic behaviour of the order, while others are necessary parameters for the execution.

Command execution in Linux

Structure:

```
$>Order Param1 Param2 ... ParamN <Enter>
```

Example:

```
$>ls -l /etc
```

- Displays the contents of the /etc directory.
- Displays information associated with files and directories in long format (-l).

Always separate the commands and parameters with spaces.

Command execution in Linux

Some Shells such as Bash, provide facilities to write commands, like “auto” words when pressed key < TAB > or using special characters (such as ‘*’ and ‘?’) to replace part of the commands.

Commands Classification

- **Help:** go to 31.
- **System files/directories:** go to 33.
- **File Manipulation:** go to 35.
- **Search and Sort:** go to 38.
- **Administration:** go to 40.
- **Permissions:** go to 42.
- **Network:** go to 43.
- **Shell Programming:** go to 44.
- **System Status:** go to 45.
- **Compress:** go to 47.

Help Commands: `man`

`man` (manual): it provides help about a command.

```
$>man ls  
$>man man
```

To move through the manual, use `< AvPag >` and `< RePag >` and `< q >` to exit.

You can perform manual searches using the symbol `/` followed by the text or pattern you want to search.

Filesystem commands

`mkdir` (make dir): creates a directory.

```
$>mkdir dir1
```

`rmdir` (remove dir): removes a directory.

```
$>rmdir dir1
```

`cd` (change directory): change the current directory. The parent directory is considered as “..”.

```
$>cd ..
```

```
$>cd /home
```

```
$>cd directory
```


Filesystem Commands

`pwd` (print working directory): shows the whole path for the current directory.

```
$>pwd
```

`ls` (list): list the directory content.

```
$>ls
```

```
$>ls -la
```

`touch`: creates an empty file.

```
$>touch file1.txt
```

Filesystem Commands

`cp` (copy): copy files or directories.

```
$>cp file1.txt dir1  
$>cp file1.txt file2.txt  
$>cp -r dir1
```

`mv` (move): move or rename files or directories.

```
$>mv dir1 dir2  
$>mv file1.txt file3.txt  
$>mv file1.txt dir2
```

`rm` (remove): remove files or directories.

```
$>rm file1.txt  
$>rm -rf dir2
```

File Manipulation Commands

emacs: text editor.

```
$>emacs file1.txt
```

vi: text editor.

```
$>vi file1.txt
```

vim text editor (similar to vi).

```
$>vim file1.txt
```

nano: simple text editor.

```
$>nano file1.txt
```

File Manipulation Commands

`cat` (concatenate): shows the content of one or several files.

```
$>cat file1.txt
```

```
$>cat file1.txt file2.txt
```

`ln` (link): creates links.

```
$>ln -s /dir1 link
```

`more`: shows the file content with pauses.

```
$>more file1.txt
```

File Manipulation Commands

`less`: shows a file content like `man`.

```
$>less file1.txt
```

`tail`: shows the last lines of a file.

```
$>tail /var/log/Xorg.2.log
```

```
$>tail -40 /var/log/Xorg.2.log | less
```

`head`: shows the first lines of a file.

```
$>head file1.txt
```

```
$>head -40 /var/log/Xorg.2.log | less
```

Search and Sort Commands

`grep`: searches in a file content.

```
$>cat file1.txt | grep Hello
```

`sort`: sorts the file content.

```
$>sort file1.txt
```

Administration Commands

`sudo`: allows users in `/etc/sudoers`, execute commands with privileges of other users (root included).

```
$>sudo apt-get install wine  
$>sudo user1 nano fich1
```

`su` (switch user): allows to changed the user session. If the user name is not specified, it is changed to root session.

```
$>su  
$>su user1
```

Administration Commands

mount: mount a device (such as USB, CD-ROM).

```
$>mount /dev/hdb2 /mnt/home -t vfat
```

```
$>mount /dev/cdrom /mnt/cdrom
```

umount: unmount a device.

```
$>umount /dev/hda2
```

```
$>umount /mnt/cdrom
```

uname (unix name): gets information about Linux version, kernel, etc.

```
$>uname
```

```
$>uname -a
```


`adduser`: adds a user to the system.

```
$>adduser usr1
```

`userdel`: removes a user from the system.

```
$>userdel usr1
```

`usermod`: modifies a user in the system.

```
$>usermod -s /bin/bash usr1
```

`passwd`: change the user password.

```
$>passwd
```

```
$>passwd usr1
```

Permission Commands

`chmod` (change mode): change files and directories permission.

```
$>chmod +x file1.txt
```

```
$>chmod -r file1.txt
```

```
$>chmod +rw dir1 -R
```

`chown` (change owner): change the owner of a file or directory.

```
$>chown root:root file1.txt
```

```
$>chown -R usr1:usr1 dir1
```

Network commands

`ifconfig` or `ip` (interface config): shows and modifies the Network configuration.

```
$>ifconfig
```

`ping`: checks if a remote system is working.

```
$>ping www.isp1.com
```

`ssh`: remote access to a computer using a secure shell.

```
$>ssh maquina1.isp1.com
```

`ftp`: FTP file transfer protocol.

```
$>ftp ftp.isp1.com
```

`mail`: sends and read e-mails.

```
$>mail usr1@isp1.com < mess1.txt
```

Shell Programming Commands

echo shows a message.

```
$>echo "hello_world"
```

sleep: waits.

```
$>sleep 5
```

exit: closes an open session.

```
$>exit
```

System State Commands

`df` (disk free): shows free disk space.

```
$>df
```

```
$>df -h
```

`du` (disk use): shows disk used.

```
$>du *
```

`who`: shows the users log in the system.

```
$>who
```

```
$>whoami
```

`ps` (process): shows the process running.

```
$>ps
```

```
$>ps aux
```

`kill`: kills a process.

```
$>kill pid1
```

System State Commands

`jobs`: shows running jobs.

```
$>jobs
```

`fg` (foreground): moves a job to foreground.

```
$>fg 1
```

`bg` (background): moves a job to background.

```
$>bg 1
```

`env` (environment): shows the environment variables.

```
$>env
```

Compress Commands

tar (Tape ARchiver): file package.

```
$>tar cvf file1.tar dir1
$>tar xvf file1.tar
$>tar zcvf file2.tgz dir2
$>tar zxvf file2.tgz
```

gzip: compress and uncompress files.

```
$>gzip file1.tar
$>gzip -d file1.tar.gz
```

zip/unzip: compress and uncompress files or directories.

```
$>zip file1.zip dir1/
$>unzip file1.zip
```

Index

- 1 Introduction
- 2 GNU/Linux as an Operative System
- 3 Commands
- 4 Exercises**

Exercise 1: Basic Commands

Use the commands `mkdir`, `ls`, `cal` y `cat`, and the special symbols (`* y ?`) and `>` to run the following task:

- 1 Create a subfolder “Calendar” in the folder `temp`.
- 2 Check with `ls` that the folder has been created.
- 3 Create three files `april2013`, `august2013` and `september2013` that contain the calendar of April, August and September for 2013. The instruction to generate a calendar of the month n of year a is:

```
$>cal n a
```

This instruction shows the calendar in the screen. If you want to save the output in a file you must add `>` and the name of the output file. For example, if you want to save the calendar in a file called `myCal`: `cal n a > myCal`.

Exercise 1: Basic Commands

Use the commands `mkdir`, `ls`, `cal` y `cat`, and the special symbols (`*` y `?`) and `>` to run the following task:

- 1 Check the content of `august2013` with the `cat` instruction.
- 2 Use the command `ls a*` to list all files in folder `temp` that begins with the character 'a'. Use the command `ls *2013` to list all files that finish with 2013.
- 3 Create a file named `august2010` that contains the calendar of August 2010. Use the command `ls *201?` to show all files that finish in 201 followed by any character.
- 4 List all files that start with 'a' and finish with '201' followed by any character.

Exercise 2: Basic Commands

Execute the following task using `cal`, `cat`, `ls`, `mkdir`, `mv`, `pwd` y `rm`:

- 1 Create a file named “2013” containing the calendar for year 2013.
- 2 Move this file to the Calendar subfolder.
- 3 In the temp folder, create a subfolder named “Months” inside the Calendar folder.
- 4 In the temp folder, move the file `august2013` to the folder Months.
- 5 In the temp folder, with a single instruction, move the file `september2013` to the folder Months and rename it as `09.2013`.

Exercise 2: Basic Commands

Execute the following task using `cal`, `cat`, `ls`, `mkdir`, `mv`, `pwd` y `rm`:

- 1 In the temp folder, list all the content of the folder Months.
- 2 In the temp folder, show the content of the file 09.2013.
- 3 Change the current folder to Months.
- 4 Show the name of the current folder in the screen.
- 5 Change the current folder to Calendar.
- 6 Try to delete the folder Months. What response do you get?

Exercise 3: Basic Commands

Use the commands `df`, `du`, `ps`, `top`, `uname` y `who` to run the following task:

- 1 Try to open a new session by using a new console or terminal (tty).
- 2 Execute the command `who` with the following parameters “-q”, “-m” and “-H”. Explain the differences (if you need some help you can use `man who`).
- 3 Open different sessions and execute `ps`, `ps x` and `ps aux` and explain the differences.

Exercise 3: Basic Commands

Use the commands `df`, `du`, `ps`, `top`, `uname` y `who` to run the following task:

- 1 How should you call “`df`” to see the information in MB and GB instead of in blocks of 1024 bits?
- 2 Discover how much space the following folder and file take:
`/usr/bin` folder and any file located in the user folder (`/home`).
- 3 What Linux version and what type of machine are you using in the computer?
- 4 Try to discover the processes that are taking the most percentage of CPU usage.

Exercise 4: Basic Commands

Answer the following questions using the communication commands.

- 1 Check whether Google is available using the command `ping`
- 2 Connect to a remote host using the command `ssh` and execute the command `who` and `ps`. Discover the name of the terminal (“pts/n”) you are connected and try to know which processes are executed by the different users.
- 3 What happens if you try to connect using the `ssh` command to an IP that does not exist?