Coursework 1: Basic GNU/Linux (Practising Shell)

Héctor Menéndez[1]

AIDA Research Group
Computer Science Department
Universidad Autónoma de Madrid

September 27, 2013

[1]based on the original slides of the subject

Index

Index

Introduction

- GNU/Linux is, in simplest terms, an operating system. It is the software on a computer that enables applications and the computer operator to access the devices on the computer to perform desired functions.
- The operating system (OS) relays instructions from an application to, for instance, 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), can also be used on several devices.
- Some examples are:
  - Android.
  - TiVo Digital Video Recorder.
  - Sony Bravia TV.
  - Yamaha Motif Keyboard.

How is GNU/Linux run?

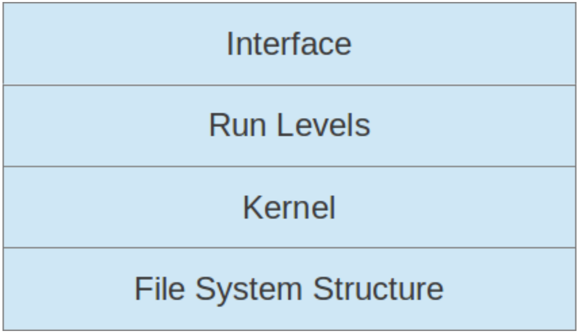
- Switch on the computer. The bootloader will appear.
- Choose Linux (Ubuntu).
- Once the OS is loaded, you will have to introduce your user login information.
- Next, you will have to introduce your password.

Index

GNU/Linux Architecture

- **Files Structure**: This is the part which 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**: mode of OS 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



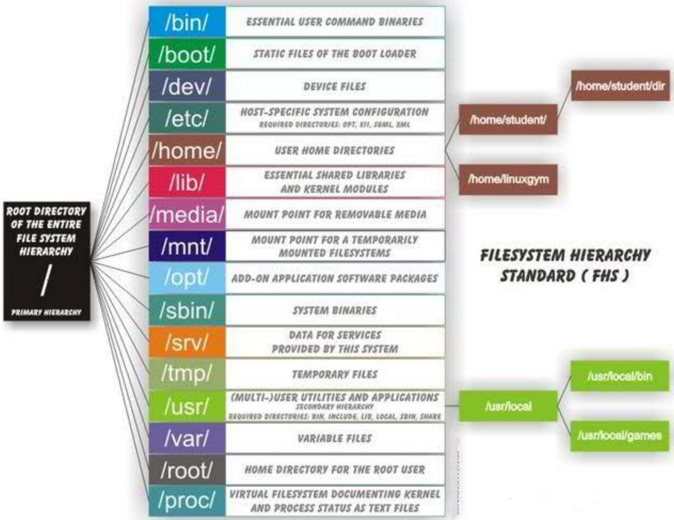
Filesystem Structure

- **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` y `/media`: mount points.
  - `/lib`: essential shared libraries and kernel modules.

Filesystem Structure



/bin/	ESSENTIAL USER COMMAND BINARIES
/boot/	STATIC FILES OF THE BOOT LOADER
/dev/	DEVICE FILES
/etc/	HOST-SPECIFIC SYSTEM CONFIGURATION REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/	USER HOME DIRECTORIES
/lib/	ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/	MOUNT POINT FOR REMOVABLE MEDIA
/mnt/	MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/	ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/	SYSTEM BINARIES
/srv/	DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/	TEMPORARY FILES
/usr/	(MULTI-)USER UTILITIES AND APPLICATIONS SECONDARY HIERARCHY REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/	VARIABLE FILES
/root/	HOME DIRECTORY FOR THE ROOT USER
/proc/	VIRTUAL FILESYSTEM DOCUMENTING KERNEL AND PROCESS STATUS AS TEXT FILES

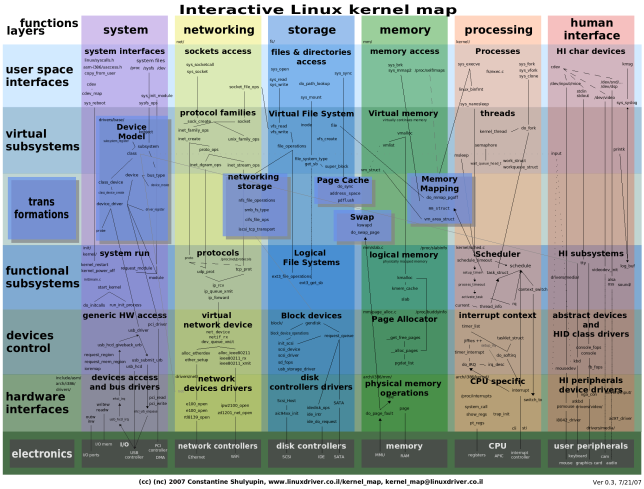
The Kernel (Linux)

- **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 program parts which are currently being used are the only load.

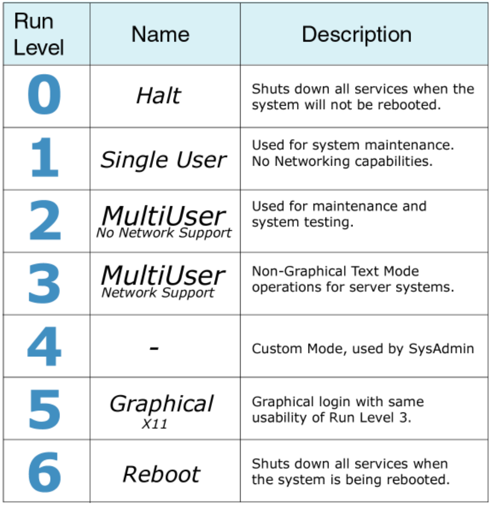
The Kernel: Process

- **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 programmed instructions that can be managed independently by an operating system scheduler.

The Kernel Structure



Run Levels



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

Run Levels: Users y groups

- <u>**User:**</u> is an agent, either a human agent (end-user) or software agent, who uses a computer or network service. Every user account has an user nickname 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 occur and offering services. In GNU / Linux deamons have an associated user.

Run Levels: Users y 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 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 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 can also refer to directories and, in this case, the meaning is:
  1. **Read (r)**: it enables to list the contents of a directory.
  2. **Write (w)**: it enables to add or remove contents of 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 refer to the user group established by the system administrator which integrates the user owner of the item.
  3. **other (o)**: it is any other user of the system which is not included in the previous two sets.

Run Levels: Acceso 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).

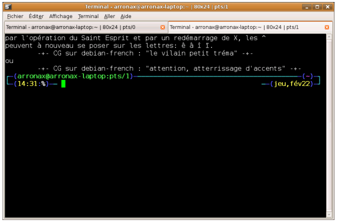
Types of Interfaces

- **text interfaces (shell or command line)**: Fast and very efficient, very used to perform repetitive tasks automatically. Use commands that 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.

Interfaces: Terminal


Terminal - arronax@arronax-laptop:~ | 80x24 | pts/1
Fichier Éditer Affichage Terminal Aller Aide
Terminal - arronax@arronax-laptop:~ | 80x24 | pts/0
Terminal - arronax@arronax-laptop:~ | 80x24 | pts/1

```
par l'opération du Saint Esprit et par un redémarrage de X, les ^
peuvent à nouveau se poser sur les lettres: ê â î ï.
        -+- CG sur debian-french : "le vilain petit tréma" -+-
ou
        -+- CG sur debian-french : "attention, atterrissage d'accents" -+-
┌─(arronax@arronax-laptop:pts/1)──────────────────────────────────────(~)─┐
└─(14:31:%)──                                                ──(jeu,fév22)─┘
```

- It is a shell inside the graphical environment.
- You can open as many terminals as needed.
- They are independent and each runs a shell.
- Besides, there is the System Console. It's a full screen terminal, normally accessed with <Ctrl><Alt><Fx> (<F1>...<F7>).

Index

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.
    - 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 `/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 $< \text{TAB} >$ or using special characters (such as ‘$*$ ’and‘ ?’) to replace part of the commands.

Commands Classification

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.
- Can realizase manual searches using the symbol '/ 'followed by the text or pattern you want to search.

Filesystem commands

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

```
$>ls
$>ls -la
```

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

```
$>cat file1
$>cat file1 file2
```

- cp (copy): copy files or directories.

```
$>cp fch1 /tmp
$>cp fch1 fch2
$>cp -r dir1 /tmp
```

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

```
$>mv dir1 dir2
$>mv fch1 fch2
$>mv fch1 dir1
```

Filesystem Commands

- rm (remove): remove files or directories.

```
$>rm -i fch1
$>rm -rf dir1
```

- mkdir (make dir): creates a directory.

```
$>mkdir dir1
```

- rmdir (remove dir): removes a directory.

```
$>rmdir dir1
```

- ln (link): creates links.

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

Filesystem Commands

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

```
$>chmod +x fch1
$>chmod -r fch1
$>chmod +rw dir1 -R
```

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

```
$>chown root:root fch1
$>chown -R usr1:gusr1 dir1
```

- `more`: shows the file content with pauses.

```
$>more fch1
```

Filesystem Commands

- less: shows a file content like man.

```
$>less fch1
```

- tail: shows the last lines of a file.

```
$>tail -f /var/log/mail.log
$>tail -100 /var/log/mail.log | more
```

- head: shows the first lines of a file.

```
$>head fichero
$>head -100 /var/log/maillog | more
```

Network commands

- `ifconfig` (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
```

Search and Sort Commands

- grep: searchs in a file content.

```
$>cat fch1 | grep cadena
```

- sort: sorts the file content.

```
$>sort file1
```

Edition Commands

- vi: text editor.

```
$>vi fch.txt
```

- vim[2] text editor (similar than vi).

```
$>vim fch.txt
```

- nano: simple text editor.

```
$>nano fch.txt
```

[2] http://www.glump.net/files/2012/08/vi-vim-cheat-sheet-and-tutorial.pdf

Shell Programming Commands

- `exit`: closes an open season.

```
$>exit
```

- `echo` shows a message.

```
$>echo "hello␣world"
```

- `sleep`: waits.

```
$>sleep 5
```

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
```

- `passwd`: change the user password.

```
$>passwd
$>passwd usr1
```

Administration Commands

- `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`: modify a user in the system.

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

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
  ```

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
$>w
$>who am i
```

- ps (process): shows the process running.

```
$>ps
$>ps -A
```

- kill: kills a process.

```
$>kill pid1
```

System State Commands

- `job`: shows running jobs.

```
$>job
```

- `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 fch1.tar dir1
$>tar xvf fch1.tar
$>tar zcvf fch2.tgz dir2
$>tar zxvf fic2.tgz
```

- `gzip`: compress and uncompress files.

```
$>gzip fch1.tar
$>gzip -d fch1.tar.gz
```

Index

Exercise 1: Basic Commands

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

1. Check the content of august2013 with `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 temp folder, create a subfolder named “Months” inside the Calendar folder.
4. In temp folder, move the file august2013 to folder Months.
5. In temp folder, with a single instruction, move the file september2013 to 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 temp folder, list all the content of the folder Months.
2. In temp folder, show the content of 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 folder Months. What response do you get?
7. Make a diagram in which you represent the tree structure with the files and folders created in this exercise, and also the content of temp folder.

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). You can use the sequence of buttons <Ctrl><Alt><Fx> and execute the command `who` with one or more sessions and explain the results.
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 do you should call “df” to see the information in MB and GB instead of in blocks of 1024?
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 seen in this assessment.

1. Check whether a partner machine 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 ssh command to an IP that does not exist?