TP n°1: Introduction to Linux

1. Objectives

The purpose of this practical work is to familiarize you with the Unix environment and provide you with the essential tools for the rest of the practical work.

2. Linux environment

Linux is a family of free and open-source operating systems based on the Linux kernel. Operating systems based on Linux are known as Linux distributions or distros. Examples include Debian, Ubuntu, Fedora, CentOS, Gentoo, Arch Linux, and many others.

The Linux system is a multi-user system, so to access it the user must enter a name and password.

3. Communication with the machine in interactive mode

In this first part of the practical exercises, we use the command line in interactive mode, i.e. each command is followed by an immediate return, and in order to understand each other when we talk about the Shell, we present the following elements:

- Command interpreter: the UNIX command interpreter, also known as *Shell* (reads the name of the command and all that follows serves as arguments to the command except for certain special characters called metacharacters & < > * ? | which should be interpreted).

- **Terminal**: A terminal is an input and output environment that presents a text-only window running a Shell.

- **Command:** A Linux command is a program or utility that runs on the command line. Example (ls, cat, ...).

- Shell: the Shell is a program that acts as a text-mode interface between the kernel and the user. It is a text-mode interface with the keyboard as input and the screen as output.

- **Prompt:** commands are entered from the prompt (or command prompt), which can take different forms; in general, the prompt displays the user's name (user), an arrobase (@), the name of the machine (localhost), then the current directory (pwd), and finally a hash sign (#) for the super user (root), the plus sign (>) or the percent sign (%) for a normal user.

- **User**: Linux is a multi-user system, which means that several people can interact with the system at the same time. There are many ways of creating new users in Linux. Each user is identified by a login and a password. The system administrator is responsible for managing the various users on the system.

- **Login**: this is associated with a unique number (the UID: User IDentifier) and is used to identify each user. It contains no spaces or special characters.

- **Absolute path and relative path**: The location of each resource (file or directory) in the file system is called its ***path***. In a Linux path, the separator between two directories is the / character. There are two types of paths:

- An absolute path identifies a resource by starting at the root of the tree, with the / character. An absolute path does not depend on the current directory and is therefore valid everywhere.
- A relative path identifies a resource from the current directory. It therefore depends on the current directory and is not valid everywhere.

4. Linux directory structure

The Linux directory structure is defined in the Linux FHS (Filesystem Hierarchy Standard). Directories are referenced and accessed using the deeper sequential names of directories linked by the slash '/'.

The following is a description of the main Linux top-level directories and their roles:

- **Root (/)**: the root of a file system is the base folder which contains all the other folders, the entire tree structure. In Linux, the root is the / folder. There is nothing above it, no folder contains the root, and all files, folders and subfolders are contained in /.

- **/boot:** It includes the static kernel and bootloader configuration and executable files needed to start a Linux computer.

- **/bin:** This directory includes user executable files.

- **/dev:** It includes the device file for all hardware devices connected to the system. These aren't device drivers; instead, they are files that indicate all devices on the system and provide access to these devices.

- **/etc:** It includes the local system configuration files for the host system.

- **/lib:** It includes shared library files that are needed to start the system.

- **/home:** The home directory storage is available for user files. All users have a subdirectory inside /home. /home stores user directories. When you create a new user, a /home subfolder is automatically created (unless you specify otherwise). This is where the user can store their personal data.

- **/root:** It's the home directory for a root user (/root is the system administrator's home directory). Keep in mind that it's not the '/' (root) file system.

- **/tmp:** It is a temporary directory used by the OS and several programs for storing temporary files. Also, users may temporarily store files here. Remember that files may be removed without prior notice at any time in this directory.

- **/sbin:** These are system binary files. They are executables utilized for system administration.

- **/usr:** They are read-only and shareable files, including executable libraries and binaries, man files, and several documentation types.

- **/var:** Here, variable data files are saved. It can contain things such as MySQL, log files, other database files, email inboxes, web server data files, and much more.

5. Opening and closing a session under Linux (login and logout)

Before attempting to connect to your terminal, you should first check with your network administrator that you are indeed a user and that you have been assigned a password. We will see how to open and close a session and change your password.

Open a session: login

login is your identifier. Enter your username.

In the password field, you enter your password. In most cases, what you type will not appear.

Note: the password field is case-sensitive, i.e. it make difference between uppercase and lowercase letters.

Once you have logged in, the prompt appears and you are now connected to the machine.

5.1 Change password: password

To change your password, we use the command ***passwd***. For that, we need to enter your old password, the new one and then confirm it.

5.2 Close a session: logout

Never forget to close your session - someone could use it without your knowledge and delete or modify your data.

To close a screen session (terminal) you can use the ***logout*** or ***exit*** commands or use the shortcut **^*D***. However, to close your session completely, simply use one of the following two commands: ***gnome-session-quit*** (if you are in your session) or ***sudo pkill -KILL -u name-of-the-session-of-user.***

For example, if the name of your session is *kamel* you use the following command: ***sudo pkill -KILL -u kamel.***

6. Terminal interface under Linux

Every time you open a terminal it displays :

Name_of_session@name_of_host:~$

In the following we present some commands that provide information about users.

logname : to find out the user's name, simply run the following command: *$ logname*

hostname : to find out the name of the host computer, simply run the following command: *$ hostname*

whoami : to display the username currently logged into the operating system.

who : displays information about all users currently on the local system. The following information is displayed: login name, tty, date and time of login.

w: the *w command* is a utility in Linux that displays information about the users currently logged into the system and their processes.

poweroff, *reboot*, and *halt : these commands* may be used to power off, reboot, or halt the machine. All three commands take the same options.

shutdown: the *shutdown command in Linux* is used to *shutdown* the system *in* a safe way. You can *shutdown* the machine immediately, or schedule a *shutdown.*

pwd: to know the current directory (where you are), run the following command: *$ pwd*

where the ~ (tilde) directory is equivalent to /home/username

7. Basic commands under Linux

A Linux command is written in the following form:

Command <options> arguments

In Linux, there are three ways of requesting help:

- Consulting the manual pages in /usr/share/man, /usr/share/doc/,...
- The use of commands that consult manual pages efficiently and very quickly such as *man* whose syntax is :

  man <command name>

  Examples: *man find*, or *man whatis*
- If you want to know what a command does, use the ***whatis*** command followed by the command using the following syntax:

  whatis <name-of-command>
- The ***apropos*** command is used to list manuals whose description or name includes the words passed as arguments. For example, if you need a text editor, enter the command ***apropos*** followed by the word "text" and it will return the text editors for your operating system.

  apropos texte or text

7.1 Location commands

a) pwd (print working directory)

Displays the current directory.

b) cd (change directory)

The `cd` command is used to navigate to a different directory or change the current working directory.

Examples

- *$ cd* : returns you to the /home/user directory (same as $ cd ~).
- *$ cd -* : returns you to the previous directory.
- *$ cd ..* : To navigate to the previous (parent) directory.
- *$ cd /* : can be traced back to the root (to move to the root).
- *$ cd /usr/share/doc/* : to move to the /usr/share/doc/ directory.

7.2 Viewing commands

a) ls *:* lists the contents of a directory. We can use several options at the same time, as follows:

$ls option1 option2 .....

Examples

$ ls -l –a or $ ls -la

$ ls -l --all

Some options of the ls command :

`ls -l`: This command is used to list directories and files together with additional information like permissions, owner, size, and when it was last modified.

`ls -a`: This command lists the contents of a directory together with the hidden file. Hidden files in Linux start with the dot (.) character.

`ls -h`: makes the output readable, where file sizes will be displayed in Kilobytes, Megabytes or Gigabytes.

`ls -lt`: lists files and directories in order of file modification date.

Examples of use :

- *`ls -a`* ***:*** Displays all files and folders in the current directory.
- *`ls /etc/`* ***:*** Displays the contents of the /etc/ directory.

b) cat : The cat command is one of the most useful commands. Its name is inspired by the word concatenate. This command lets you create, merge or print files in the standard output screen or to another file, and much more besides. Its syntax is :

cat [OPTION] [FICHIER]

Some options of the cat command :

`cat -n` : to display the contents of a file with line numbers.

`cat -v` : to display all non-printable characters.

Example

`cat -n /etc/passwd` : Displays the passwd file, numbering the lines from 1.

c) more : displays a file page by page without modification**.**

Some options of the more command :

`more -s` : groups consecutive blank lines into a single line.

`more -f` : causes lines to be counted logically (rather than on-screen), i.e. long lines are not cut.

Example

```
more -sf /etc/passwd
```

This command displays the passwd file page by page, concatenating empty lines without cutting long lines.

d) less : less is a command that displays a text file page by page (without modifying it). Its function is similar to the more command, but it also allows you to go back or search for a string. Its syntax is:

```
less [OPTIONS] filename
```

The less command can also be used for several files at once. Simply type them in, separating them with a space as follows:

```
less fichier1 fichier2 fichier3
```

less first displays the first file.

- To display the next file, press : then n for Next.

- To return to the previous file, press : then p for Previous.

Some options for use :

`-e or -E`: Exits automatically the second time the end of the file is reached, or the first time with -E.

`-F`: Scroll forward, and always try to read even when the end of the file has been reached.

`-r or -R`: Allows special characters.

`-x`: Sets the tab size.

`-~`: Does not fill empty lines with ~.

Example

```
less -Er~ /etc/passwd
```

This command displays the passwd file page by page, taking special characters into account and not filling empty lines with ~.

7.3 Handling commands

a) mkdir: mkdir is a command used to create directories. *mkdir* is an abbreviation of "make directory". This command is also known as *md* (make directory) on other operating systems. its syntax is as follows

```
mkdir  [options]  name_of_directory
```

Some mkdir command options :

`mkdir -p`: used to create the missing parent directories with *mkdir*.

It sometimes happens that you want to create several successive directories (included in each other); for example, you want to create /folder1/folder2/. The mkdir command allows you to do just that, using the *-p* or *-parents* option.

`mkdir -m`: allows you to specify the permissions assigned to the directory when it is created.

Examples

```
mkdir TP1
```

This command creates the *TP1* directory

```
mkdir -m 700 TP1
```

This command creates a directory with the name *TP1* and the permissions 700.

b) touch *:* the touch command is mainly used to create empty files, and to modify the timestamp of files or folders. There are three types of file timestamp information:

- Access time (atime): the last time a file was read or accessed.
- Modification time (mtime): the last time the contents of a file were modified.
- Modified time (ctime): the last time a file's metadata was modified (for example, permissions).

The syntax of the touch command is :

touch [options] [name_of_file]

Some options for use :

`-a`: changes the access time of a file to the current time,

`-m`: allows you to change the modification time of a file to the current time.

`-t`: It is also possible to set the access and modification time of a file to a particular date by using the *-t* option followed by the date-time using the following syntax:

```
touch -t CCYYMMDDhhmm.ss   name_of_file.txt
```

- CC: first two digits of the year
- YY: the last two digits of the year
- MM: the month of the year [01-12]
- DD: day of the month [01-31]
- hh: the hour of the day [00-23]
- mm: the minute of the hour [00-59]
- ss: the second of the minute [00-59].

Examples

```
$touch test.txt
```

touch changes the access and modification time of the *test.txt* file if it exists, otherwise it will be created anew.

```
$touch -t 202210021150.30 filename.txt
```

touch changes the access and modification time of file *filename.txt* to 02/10/2022 at 11:50 and 30 seconds.

c) mv (move)*:* is used to move a file, directory or any tree structure on Linux. It is used when you want to move a file or directory from one location to another, and can also be used to rename a file. Its syntax is :

```
mv [OPTIONS] Source Destination
```

Some options for the *mv* command :

`-f`: Do not request anything before overwriting (-force).

`-i`: Display an interaction before overwriting (-interactive).

`-u`: Move only when the source file is newer than the destination file or when the destination file is missing (-update).

Examples

- `$mv file1.txt  Folder1/`
  Moves file1.txt to the Folder1 directory
- `$mv Folder1/file.txt TP1/`
  Moves file.txt from the Folder1 directory to the TP1 directory
- `$mv Rep Rep2`
  Renames the Rep directory to Rep2
- `$mv file1.txt file2.txt`
  Renames file file1.txt to file file2.txt

d) cp *:* This command is used to copy files or a group of files or directories. It creates an exact image of a file on a disk with a different file name or the same name. Its syntax is :

```
cp [OPTIONS] SOURCE... DESTINATION
```

The source can contain one or more files or directories as arguments and the destination argument can be a single file or directory.

- When the source and destination arguments are both files, the cp command copies the first file into the second. If the file does not exist, the command creates it.
- When the source has several files or directories as arguments, the destination argument must be a directory. In this situation, the source files and directories are moved to the destination directory.
- When the source and destination arguments are both directories, the cp command copies the first directory into the second.

Some options for the cp command :

`-a`: retain as much of the structure and attributes of the original files as possible in the copy.

`-i`: display a message each time a file is to be overwritten.

`-f`: this option forces the copy even if the destination folder is not available for writing.

`-r`: recursive tree copying.

`-u`: this option does not copy files that have an identical or more recent timestamp modification in the destination folder. It is an update of a copy.

`-v`: display the name of each file before copying it.

Examples :

- *`cp file1.txt TP1/`*
  Copies the file file1.txt into the TP1/ directory
- *`cp -r TP1/ TP2/`*
  Copies the TP1 directory into the TP2/ directory.

e) rmdir *: It is used to remove empty directories from the Linux file system. The* rmdir *command deletes each directory specified on the command line only if these directories are empty. If the directory specified contains directories or files, these cannot be deleted by the* rmdir *command. Its syntax is :*

rmdir [-p] [-v | -verbose] [-ignore-fail-on-non-empty] directories ...

Some options :

`-p`: with this option, each directory argument is treated as a path name from which all components will be removed, if they are already empty, starting with the last component.

"`$rmdir -p a/b/c`" is similar to "`$rmdir a/b/c a/b a`".

Examples :

`$rmdir TP1`

Removes the *TP1* directory

f) rm : the *rm* command is used to remove files and directories on the Linux command line. Its syntax is :

```
rm [OPTIONS]... FILE/DIRECTORY...
```

Some options for the rm command :

-f: Ignore non-existent files and arguments.

-r: Remove directories and their contents recursively.

Examples

- `$rm file1.txt`
  Removes the file *file1.txt*
- `$rm -rf /home/user/TP1`
  Removes the */home/user/TP1* directory and all its files without asking for confirmation.

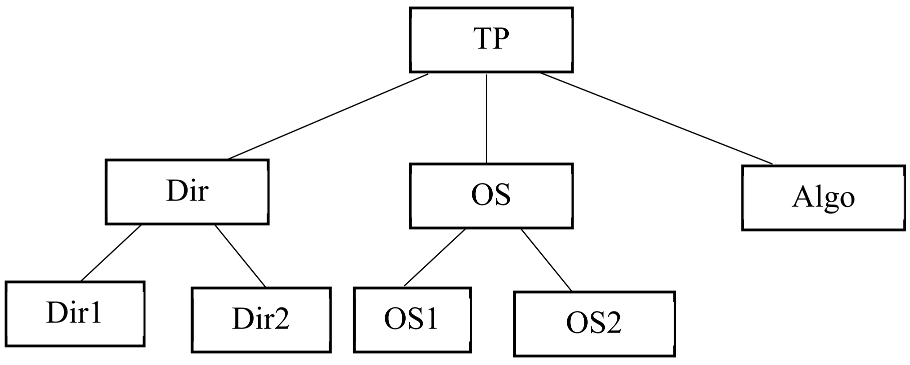
<u>Exercises</u>

<u>Exercise n° 1</u>

1) Run the ***pwd*** command and interpret the result?
2) Which command do you use to move into the root directory (/)?
3) Move into the directory (***/tmp***).
4) From the current directory (***/tmp***) give the path to your working directory in two different ways:
   a) Using an absolute path
   b) Using a relative path
5) Access to your working directory using either the absolute or relative path.
6) Create a directory called ***Linux*** in your working directory, then move to this directory.
7) Create a directory called ***Redhat***, then move to this directory.
8) What should the ***pwd*** command display?
9) What is the effect of executing the ***cd ..*** command?
10) Create a directory called ***Mandrake***?
11) Run the ***cd ..*** command again? in which directory would you move?
12) Illustrate with a diagram the tree structure created after executing the previous commands?

<u>Exercise n° 2</u>

Create the following tree structure in your working directory.



I) Indicate the path in each case:

1) the path to the ***OS1*** directory from your work directory.
2) the path to the ***OS1*** directory from the ***TP*** directory.
3) the path to the ***Dir1*** directory from the ***OS*** directory.
4) the path to the ***Dir1*** directory from the ***Dir*** directory.
5) the path to the ***OS1*** directory from the ***Dir2*** directory.
6) the path to the ***Algo*** directory from the ***Dir2*** directory.

II) Run the following commands :

1) Run the (ls) command with and without the (-l) option in your working directory. What is the difference between the two displays?
2) Display the list of hidden files.
3) Move to the ***/usr/include*** directory and display all the files in this directory with their attributes.
4) Run *ls -l *.h.* What is the difference between this display and the display in the previous question? What is the role of the * metacharacter?
5) Run ls -l z*.h and what is the difference between this display and the one in question 4)?
6) Run ls -l *e.h. What is the difference between this display and that of 4)?
7) Run ls -l ???.h. What is the difference between this display and the one in question 4) and what is the role of the metacharacter "?" ?
8) Discover the following options for the ls command: -ld , -lt
9) Go back to your working directory.
10) Move to the ***OS*** directory and create two new empty files ***file1*** and ***file2***.
11) Copy the file ***file1*** into the ***OS1*** directory.
12) Move the file ***file2*** into the ***OS2*** directory.
13) Remove the file ***file1*** from the ***OS*** directory.
14) In your working directory, run the command: ***mkdir -p Rep1/Rep2/Rep3***. What is the result of this command?
15) Move to the ***OS1*** directory, copy the file ***file1*** into the ***Rep1***, ***Rep2*** and ***Rep3*** directories and change its name (***doc1***).
16) Copy the ***Rep3*** directory into the ***OS1*** directory.
17) Move the ***Rep2*** directory into the ***OS2*** directory.
18) Rename the ***Rep1*** directory to ***Folder1***.
19) Remove the ***Rep2*** directory from the ***OS2*** directory using the ***rmdir*** command. What do you notice?

III) Illustrate with a diagram the tree structure created after executing the previous commands?