COMPUTER SCIENCE

Assignment 1.

Introduction to the Linux Operating System.

GIEyAI
Fall Semester, 2024-2025
v2.8(02.09.24)

INTRODUCTION TO LINUX.

The Unix OS has been under development since the 70s. It started out in AT&T but, because of its success, many variants appeared early on (Solaris, AIX, HP-UX). Some of them were developed by computing companies (IBM, Sun, HP), whereas some others were the result of work by a university (BSD).

As of today, the most widespread variant is arguably Linux, a free software project started by Linus Torvalds. The fact that Linux is free to distribute has encouraged a multitude of programmers to collaborate in developing the system, coding as well as debugging, elaborating documentation, and so on.

Given this interest in this OS, and because of the easy access to source code, there are nowadays many groups which offer a customized version of Linux. These versions are commonly known as distributions, and consist of the Linux kernel, which is the part that handles the computer and manages the hardware resources that the computer has, and a hand-picked selection of applications to broaden the basic functionality of the kernel and allow tasks that range from generation of documentation (text processors, spreadsheets, databases) to program development (compilers, software libraries, debuggers), as well as management of multimedia content (audio and video players, editors, photography suites), etc. Many of these distributions are made freely available by the groups which assemble them, but there are also others, more professional in nature, which require some form of payment. The one selected for the lab work in this course is Ubuntu, one of the free ones, very easy to obtain, install, and manage. It is a very popular distribution, which makes the task of finding the related documentation very easy to the student.

Basic Linux management.

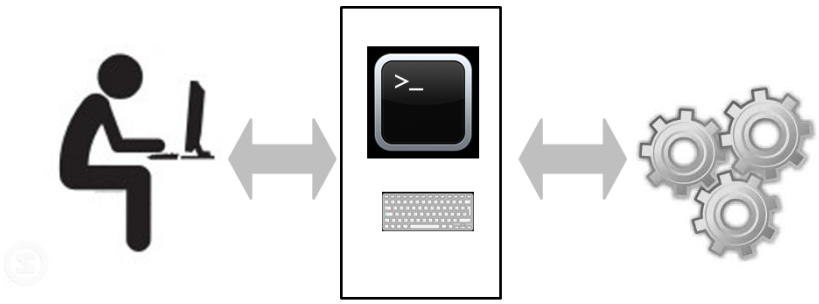
Following the popular trend in computer interfaces, Linux has a graphical one for most of the tasks that one needs to do. The graphical interface revolves around the desktop which, aesthetical considerations aside, is very similar to that of almost any other OS. There is a means to access the applications, the menu bar (usually) on the top part of the desktop, and from this bar it is possible to launch programs such as the file manager, web browsers, system config apps, the terminal, etc.

These applications let one manage the computer with a graphical interface, which makes the task very comfortable, but in this assignment, in order to make the student see in a more direct manner what is being done, the basic management of the system is to be performed by means of commands. To do this, we are going to use the most basic command interface of all: the terminal.

The terminal.

The terminal is a text-mode basic interface. It is bidirectional, which means that it outputs information (on the screen) and also reads information from the user (from the keyboard). Associated with the act of launching a terminal is the launch of a command interpreter, which is the application that the user is really interacting with. The figure below illustrates this.

It all works like this: the user inputs a command by way of the keyboard, which is the part of the terminal which controls the input. This command is handed over to the command interpreter, which has it run (it creates another process to execute the command), and outputs the results to the text window (which is the part of the terminal which handles the output), for the user to read.



Commands

If the computer is to be managed by means of the terminal and the command interpreter, the user must know these commands. There are several sources of information where one can look but, possibly the most interesting one is the manual. All Linux systems have a manual included with them, in the form of the `man` command. Just like any other command, `man` can be run from the terminal. One just has to type it. For instance:

```
>man ls
```

(Just after the command, the `Enter` key must be hit). The `man` command calls the application which shows the manual page requested. In the case of the example, the one for comand `ls`. The following table shows a collection of the most frequently used commands, and a brief description of what they do.

Command	**Description**
`man`	shows manual pages
`ls`	lists the contents of a directory
`cd`	changes the working directory
`pwd`	displays the working directory
`mkdir`	creates a new directory
`cp`	copies files or directories
`rm`	deletes files or directories
`rmdir`	deletes a directory
`mv`	moves files or directories
`cat`	displays the contents of a file
`chmod`	changes the permissions for a file or directory
`chown`	changes the owner of a file or directory

In a moment we will see these commands with some more detail but, first, it is important to deal with the file system a little.

The file system

A file system is a way of organizing the information available in the devices of a system. Linux accepts a great variety of file systems, from its own to those of other OSs (like Windows). The low-level characteristics of a file system are beyond the scope of these assignment (or of this course, for that matter), so we won't deal with them here. Here, we will content ourselves with a superficial overview.

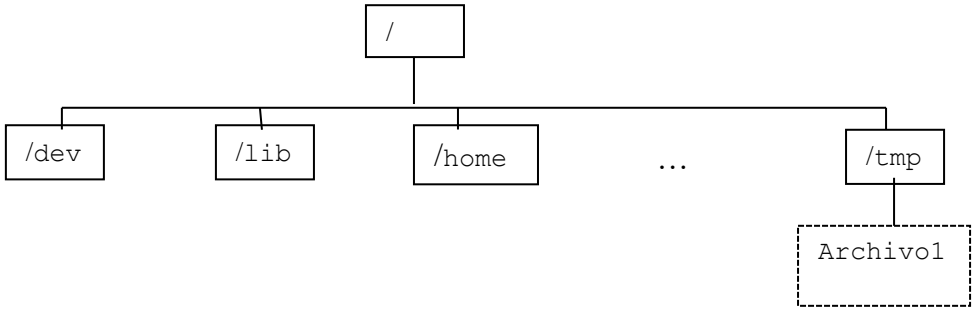
A file system is organized in the form of a tree. There is a point, the root directory, where the tree (the file system) starts. The root directory has a number of directories and files. Each directory, in turn, can have some other directories and some other files, and so on recursively.

The file system is a simple and useful way of organizing the content of the storage devices in a system. Each one (mostly the disks) is organized in some file system and Linux, when it starts, creates an additional one, which forms an initial structure, which the other ones are aggregated to (*mounted* to) as the boot procedure progresses. Once the boot sequence is over, Linux has all the content of all the devices present in the system arranged in a single logical (virtual) file system, made up of all the file systems of all the devices in the system.

As for which devices are present: a system usually has at least one hard disk drive, which is possibly divided into partitions. Each configuration is, in principle, possible, from having a single disk with a single partition, to several disks with several partitions each. Each partition is internally organized as an independent file system. Out of all the partitions present in the system, one of them is the main one, because it has the operating system files. This specific partition or, more precisely, its file system, is placed in (the technical word is *mounted to*) the root directory. The root directory is said to be the mount point of the main partition. Each file system of each partition or each disk which is added to the Linux logical file system has a mount point, which is a directory in the Linux logical file system.

When a terminal is run, an *active directory* is defined, which is the directory that commands are run from. In other words, the current directory. This is usually the user's home directory (i.e., folder). We use the term *path* to define the location of a directory or file with respect to a reference position. If this reference position is the root directory of the file system, then the path is said to be the *absolute* (or *full*) *path*.

When a command needs a file or directory to work on, this can be specified either with full path or with a relative path. In the figure below, file `File1` is located in directory `tmp`, which in turn is located in the root directory. If the active (or current) directory is `tmp`, to refer to `File1` there are two options. One can notice that the current active directory is `tmp`, and just refer to `File1` as `File1`. Or one can use the full path. In this case, `File1` is `/tmp/File1`. If the current directory were `home`, instead, then the path relative to the current directory would be `../tmp/File1`, whereas the full path would remain the same. In a path, the symbol "`..`" represents the directory immediately above (in this case, it would refer to the root directory, which is where `home` is located).



Most frequently used commands.

The most frequently used commands in Linux are listed in the table above, along with a short description. `cd` changes the current directory, `ls` lists the content of a directory, `pwd` shows the full path of the current directory, `cp` copies files or directories to another directory in the file system. `mv` is similar to `cp`, only it removes the original files or directories: `cp` duplicates and `mv` moves. For creating a directory, one uses `mkdir`, and to eliminate it, `rmdir`. `rm` deletes files.

Each of these commands requires specific input parameters, which are what the command works on, and output parameters, which are what the command returns. Sometimes they can be omitted, since some commands have default parameters. For instance, `ls` lists the content of the current directory if no path is given. Alternately, one can specify another directory for the command to list its content, just by including the directory's path next to the command.

Further commands include `cat`, which outputs a file to the screen, and `chmod` and `chown`, which allow one to manage permissions and ownership of a file or a directory. A file or directory has three kinds of permission: to read, to write and to execute. In the case of a file, this means permission to access the content, to modify this content, and to execute (or run) the file. In the case of a directory, the meaning is similar, only adapted to the use of a directory: read means see which files or directories are inside it, write means add or remove files or directories, and execute means change into that directory (with a `cd` command, for instance). Each element in a file system (file or directory) has different sets of permissions for three types of user: the owner of the element, the group that the owner belongs to, and the rest of users in the system. The rationale of this is to have the possibility of, for instance, letting a user read and modify their own file, but having other users in their group being able only to read it, and not letting a generic system user (a user that is not the owner, and who does not belong to the owner's group) read it nor modify it. This information is shown onscreen by the `ls` command when it is used with the `-l` flag. The following example illustrates all this:

```
bash-3.2$ ls -l
drwxr-xr-x  6 nacho  staff   204 Nov  3  2008 images
-rw-r--r--  1 nacho  staff  2628 Nov  3  2008 index.html
```

Calling command `ls -l` causes a listing of the two elements present in the working directory to be output to screen: the `images` directory and file `index.html`. Both belong to user `nacho`, who belongs to group `staff`. The `images` directory (we can see that it is a directory because of the letter `d` at the beginning of its line) has read (`r`), write (`w`) and execute (`x`) permissions for user `nacho`. Members of `staff` can see its content (`r`) and move to it (`x`), but not modify its content. Lastly, a generic system user can see the contents of the directory and change to it, but not modify it, either. Permissions are next to the first letter (`d`), and come in three blocks of three letters (or a dash, if permission is denied) in the order `rwx`, i.e., read, write and execute.

File `index.html` (one can tell it is a file because the first character of its line is a dash, that is, it is not a directory) has read and write permissions for its owner, `nacho`, but not for executing (since one can tell from its name that it is not an executable file), and only read permissions both for users of `staff` and for any other user.

The `chmod` command lets the owner of a file or directory modify the file's or directory's permissions. In this way, a user can, for instance, allow other users to modify one of his files at some times, and not allow them to at other times, simply by removing the write permission. `chown` can change the owner of a file. However, for safety reasons, only the system administrator can execute this command.

EXERCISES.

Here follows a series of exercises. **A step previous to all of them** is to look at the manual pages for the commands described in the table above, so as to get a more thorough understanding of their working and their syntax. Once done that, one can proceed to the exercises.

1. Create a text file (for instance, with the gedit app). Save it to your home directory.

2. Launch a terminal. Move to your home directory. Check that the file is there. Find its size and permissions.

3. Modify those permissions so that it can only be modified by the owner (you). The rest of the users (all of them) will only have read access.

4. Create a directory in your home directory (a subdirectory). Copy into it the file that you created in the previous step.

5. Change (move) into the subdirectory that you created in the previous step (that is, make the subdirectory the active directory). Check that the file you just copied is there.

6. Delete the file.

7. Go back to your user directory (move up one level from the sub directory). Delete the subdirectory. It is only possible to delete a subdirectory if it is empty. It you didn't delete the file that you copied into it, you won't be able to delete the subdirectory.