# 

Sami Taktak

sami.taktak@cnam.fr

September 2023

During the laboratory sessions we will use the GNU/Linux operating system. Start by login under OpenSuse GNU/Linux.

### Part 1: Introduction

# a) File System

A file system is a part of an operating system which deals with files management. It allow to structure information in units called file, and store them on a storage unit.

A file consists of a tuple (''name'', ''data''). name is the file name which allow a user to reference its content, data. data are any information a user want to deal with on a computer: pictures, text documents, videos, ...

A file system is organized as a tree where files are located inside directories.

A directory (or Folder) is a file that contains a list of files. A directory could contain other directories, allowing to create a hierarchy of directories.

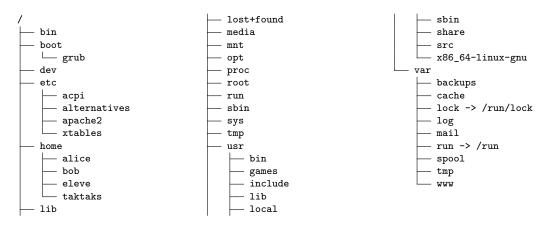


Figure 1: A typical Linux Root File System

Under UNIX, there exists a unique root tree to access all files located on any storage unit. A storage unit could be a hard drive (HDD, SSD, NVME), a USB key, a flash card, ... The root tree is designated as '/'.

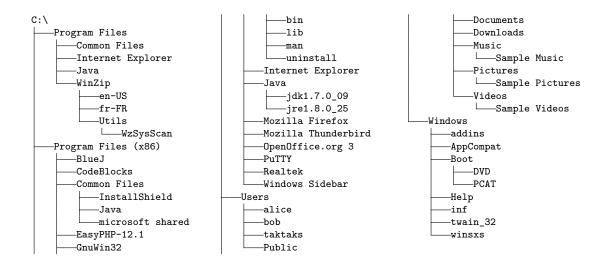


Figure 2: A typical Windows C Root File System

Under MS Windows, there exist one root tree per storage unit. Each root tree is referenced by a letter, and each root tree could be accessed as the a uppercase letter followed by a ':': C:, D:, ...

During these laboratory sessions we will consider UNIX file systems.

On UNIX, the file system is organised as a unique tree structure containing the files. This is the logical view presented to the user. Each storage unit contain it proper file system, but only a consistent view of all available file systems is shown to the user as a unique tree structure.

#### b) Files

In a file systems, for the user point of view, files are organized as a unique tree, where leaves are the files and nodes are the directories.

There exist 3 kinds of files:

- ordinary filess: they contains data useful to run the computer (system data, application data, ...) and to the user (user documents, user pictures, ...)
- directory files or folders: a directory (or a folder) is a file which contains references to other files
- special files: symbolic links, device files, communication pipe, ...

A file name might consist of any sequence of characters, only the character '/' is forbidden within a file name. From the file system point of view, the is no notion of *extension* within a file name, and the dot character could be used as any other character. For the convenience of users (and some applications), we usually use the dot character to specify extension as a indication to the user: *picture.jpeg*, *video.mkv*, *archive.tar.xz*, . . .

**Remarque.** The dot character has a particular meaning when used as first character of a file name: a file with a file name starting with '.' is a hidden file. Hidden files are not shown by default when the content of a directory is listed.

The syntax of a file name is not very strict, but some characters must be avoided for simplicity:

• characters which have a special meaning within a shell:

```
\ > < | $ ? & [ ] * ! " ' ( ) @ ~ _
```

• characters not convenient to use: other special characters and the one not directly accessible on the keyboard (keep portability in mind)

### c) Special Repositories

As the file system is organized as a tree, it possesses a file system root directory denoted by '/' and usually simply called root. That directory contains all the files and directories accessible by the operating system.

A directory might contain sub-directories and/or files. A file is referenced by its name and its position within the tree:

/directory/sub-directory/sub-sub-directory/file-name

/directory/sub-directory/sub-sub-directory/my\_file is the *full path* to the file. /directory/sub-directory/sub-sub-directory/ is the path to the directory containing the file my\_file.

Remarque. More than one file can be called my\_file as long as they belong to different directories

\verb+/directory/sub-directory/sub-sub-directory/my\_file+
\verb+/directory/sub-directory/my\_file+

Specific file names:

- The current directory is designated by .
- The parent directory of the current directory is designated by ...

### d) Useful Commands to Interact with the File System

#### • Working with directories

command	$\operatorname{argument}(s)$	Description
cd	dirname	call / change the current working directory
ls	dirname	list the content of dirname
mkdir	dirname	make a new directory called dirname
rmdir	dirname	remove the directory dirname
pwd	(none)	$\mathbf{p}$ rint current/ $\mathbf{w}$ orking $\mathbf{d}$ irectory

### Examples.

• Get the name of the current working directory:

```
$ pwd
/home/bob
```

• List the content of the current working directory:

\$ ls			
Archives	Documents	public_html	
bin	Downloads	tmp	
hello.txt	mbox	test.txt	
Bash-Lab-01.pdf	Music	Videos	
Desktop	Pictures		

• list the content of a specified directory:

\$ ls /usr/bin			
7z	ab	adb2mhc	
7za	aclocal	addftinfo	
a2p	aconnect	addpart	
a2ping	acpi	add-patch	
a2x	$acpi\_listen$	adhocfilelist	
a5booklet	acyclic	aj	
a5toa4	adb	aj5	

• Create a new directory:

```
$ mkdir Bash-Lab
```

• remove a directory (the directory should be empty):

```
$ rmdir tmp
```

# • working with files

command	$\operatorname{argument}(s)$	Description
cat	file names	Display the content of the files as a concatenation, concatenate files
less	filename	display the content of a text file one page at a time, its a pager
ср	src_name dest_name	copy a file
ср	-r src_dir dest_dir	copy a directory and its content (recursive copy)
mv	src_name dest_name	move a file, a directory
rm	$file\_name$	remove a file

# ${\bf Examples.}$

• Display the content of the file hello.txt

```
$ cat hello.txt
Hello World !
```

• Copy a file:

```
$ 1s
hello.txt
$ cp hello.txt hello2.txt
$ 1s
hello2.txt hello.txt
```

• Copy a directory:

```
$ 1s dir1
hello2.txt hello.txt
$ cp -r dir1 dir2
$ 1s
dir1/ dir2/
$ 1s dir2
hello.txt hello2.txt
```

• Rename a file (move a file to a new name/path):

```
$ mv dir2 dir_hello
$ ls
dir1/ dir_hello/
```

• Remove the file hello2.txt within the directory dir1:

```
$ rm dir1/hello2.txt
$ ls dir1
hello.txt
```

#### Exercises

Open a terminal and execute the command corresponding to the following questions:

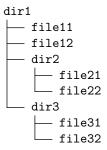
#### • Exercise 1

Execute the commands shown in the previous examples. Compare the results with the ones provided in the previous section.

### • Exercise 2

- 1. Move into the root directory
- 2. List the content of the root directory
- 3. Move into the directory /etc/;
- 4. Open the file protocols located in /etc/ with the tool more then less;
- 5. What difference do you see between the commands more and less?
- 6. How could we open the file /etc/protocols without moving first into the directory /etc/?
- 7. Go back into your home directory
- 8. Display the name of the current working directory
- 9. Execute the command "cd ." then display the name of the current working directory
- 10. Execute the command "cd .." then display the name of the current working directory. Execute the command "cd ." again then display the name of the current working directory. What happen?

- 11. List the content of the current working directory. What difference do you see between the execution of the commands "ls" and "ls ."? What do you conclude about the directories . and ..?
- 12. Go back into your home directory and create a file named test with a text editor (nano, vi, emacs, kwrite, ...);
- 13. Create the following structure:



14. Move the directory dir3 in dir2.

# Part 2: Help with manual pages

All the commands showed here have many options. Command usage and help on available options are described in manual pages.

Help could be obtained in 3 different manners, depending of the command:

• inline help can be usually obtained with option --help or -h. *Exemple* 1s

```
$ ls --help
```

• to open manual pages, man command can be used. Exemple: display 1s manual page

```
$ man ls
```

Don't forget man manual page: "man man".

• using the info pages with command info.

Exemple: display the info page of command 1s

```
$ info ls
```

#### **Exercises**

- 1. Find how to use ls display information on a whole sub-tree in long format in one command line;
- 2. Find how to use rm to remove the directory dir1 and all its content in one command line.

# Part 3: File access management

As any multi-user operating system, there is under GNU/Linux a file access mangement system. Each file (and directory) belong to a dedicated user and a dedicated group of users. The specific access rights are granted for the user owning the file, the users group owning the file, and for the other users.

Those right access restriction to not apply to the super user root.

For each file, UNIX systems distinguish three king of users:

- The user owner of the file
- The users members of the group owning the file
- The others users.

For each file and each kind of user, 3 access modes are defined:

- read access "r";
- write access "w";
- execution access "x".

To display access right of a file, use the command 1s with option "-1":

```
$ ls -l test
-rw-r--r-- 1 taktaks ensinf 34 5 mars 16:40 test
```

The output of ls gives the following information:

- "- rw- r-- give the file type and its access rights:
  - the first character specify the file type: "-" for a regular file. "d" for a directory;
  - the 3 following blocks of 3 characters define access right to the file for its owner (rw-), its groups (r--) and other systems users (r--). In each groups, the first characters specify the read access mode "r", the second one, the write access "w", and the third one the execution access "x". If, instead, a dash '-' is present the corresponding access is forbidden;

"rw- r-- r--" specify the file test is accessible in read and write mode for its owners, in read only mode for its group members and any other users;

- "1" is the number of hard link to the file;
- "taktaks" is the name of the file's owner;
- "ensinf" is the name of the file's group;
- "34" is the file's size:
- "5 mars 16:40" is the last modification time;
- "essai" is the filename.

File access mode can be modified with the command "chmod":

#### \$ chmod g+w test

add write access to all group members.

chmod command has the following syntax:

chmod [who]op[permission] file\_name

#### Where:

- who: is a combination of the letters u (user), g (group), o (others) or a (all (equivalent to ugo);
- op: + add a access mode, remove a access mode, and = set access mode for the specified users;
- permission: is a combination of letters r (read), w (write), x (execution)

#### **Exercises**

- 1. Change access right to directory dir1 so all group members have the write access;
- 2. Move into the directory dir2 and remove execution right on directory dir3 for all users. Move into the directory dir3. What happen? Fix it!
- 3. Change access right of directory dir3 so only the owner has read, write and execution access, its group has read and execution access only, and others have no access to it.

### Part 4: Useful Shortcuts

To ease the usage of the command line interface (CLI), many shortcuts are available:

Touches	Description
Up	Recall previous command in the history of run commands
Down	Call next command in the history of run commands
Shift - Page Up	Display previous page of terminal output
Shift - Page Down	Display next page of terminal output
Tab	allow for completion — auto-fill filenames, commands, argu-
	ments,
Ctrl - a	Move the cursor at the begging of the line
Ctrl - e	Move the cursor at the end of the line
Ctrl - u	Clear the current line / clear the line from the cursor position
	up to the begging of the line
Ctrl - w	Delete the work before the cursor
Ctrl - t	Permute (transpose) the character at the cursor position with
	the preceding one
Ctrl - r	Retrieve a command for the command history

Remarque. "Crlt - a" means the key Control should be maintained pressed while key a is stroked. "Ctrl - a" could also be represented as "C - a", or "^a".

#### Exercises

- 1. Test keys Page Up and Page Down in combination with Shift within a terminal;
- 2. Test key Tab effect on a command followed by a file name.
- 3. Test the other shortcuts.

# Part 5: Expansion de noms de Fichiers

The shell allow for automatic file name expansion. This allow to reference a file without specifying its complete name but also to specify group of files sharing a common name property. Let's consider a directory containing the following files:

```
$ ls
prog1 prog1.c prog2 prog2.c prog2 prog2.c prog3 prog3.c proga proga.c
```

How could we list all the file ending with ".c"?

```
$ ls *.c
prog1.c prog2.c prog3.c proga.c
```

We can also list only files containing exactly on digit in its name.

Motif	Description
*	Specify zero or more characters
?	Specify exactly one caracter
[]	Specify one character from a set of characters explicitly given
[]	Specify one character from a set of characters given as an interval (for
	example, lower case letters: [a-z])

To find all filenames beginning by prog and containing the digit 2 or the letter a:

```
$ ls prog[2a]*
prog2 prog2.c proga proga.c
```

### Exercises

- 1. Give the command allowing to show files starting with prog and containing a digit;
- 2. Give the command allowing to show files starting with prog and containing a uppercase or lowercase letter.

# Part 6: Filename and special characters

Here is the content of an other directory:

```
$ ls
prog* prog_1 prog_1.c prog_2 prog_2.c prog_2 prog2.c prog_3 prog_3.c prog_a
prog_a.c prog*.c
```

**Remarque.** Note the white space in the filenames. Spaces (\_) are explicitly shown here but they are not on a terminal.

#### **Exercises**

- 1. List files "prog 1" and proc 1.c";
- 2. What is the output of "ls prog 1\*"?
- 3. List all files containing the character '\*';

### **Protection Against Expansion**

To disable expansion, we need to escape special characters using '\'.

```
$ ls prog\*
prog*
$ ls prog\_1
prog_1
```

An other way is to enclose the sting in simple quote:

```
$ ls 'prog_1'
prog_1
```

### Exercises

In one commande:

- 1. List files "prog 1" and "proc 1.c";
- 2. List files starting with "prog" and containing a white space;
- 3. List files containing the characters \*;

# Bibliographie

- "A practical Guide to Linux Commands, Editors, and Shell Programming, A, 4th edition", Mark G. Sobell and Matthew Helmke, Addison-Wesley Professional ISBN-13: 9780134774602
- "Advanced Programming in the UNIX Environment, 3rd edition", W Richard Stevens and Stephen A. Rago Addison-Wesley Professional ISBN-13: 9780321637734
- Shell Scripting Tutorial: https://www.tutorialspoint.com/unix/shell\_scripting.htm