Linux

2022-2023

Junia-ISEN - AP4

Contents

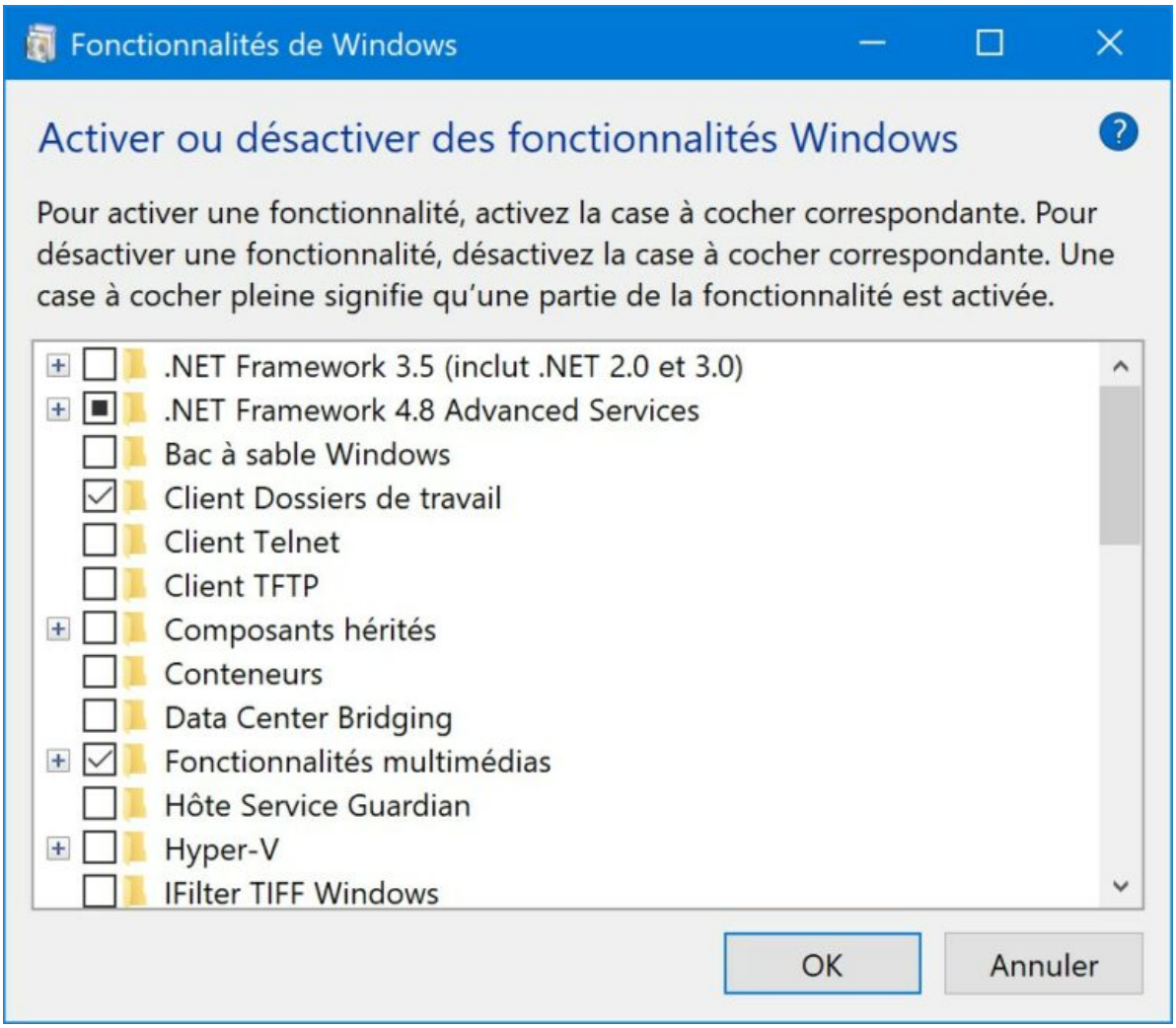

Figure 1:

Introduction

1 Pre-requisites

1.1 Windows

If you don't have a Linux OS (shame on you!), you can activate the Windows Subsystem Linux (WSL).

> **Info.** Ouvrez la fenêtre Fonctionnalités de Windows :
> - via la commande Exécuter : optionalfeatures.
> - via les Paramètres → Applications et fonctionnalités → Fonctionnalités facultatives → Plus de fonctionnalités Windows.

Info.

- Cochez la case Sous-système Windows pour Linux puis faites OK.
- Définir WSL2 comme système par défaut
- Dans powershell, exécutez `wsl --set-default-version 2`.
- Installer une distribution Linux
- Soit via le Microsoft Store
- Via powershell `wsl --install ubuntu`

1.2 Linux

Open a terminal and you should be set already!

2 Linux distributions and forks

Linux is a very open operating system. It is based on an open source kernel (which acts as the baseline to all input and output) and then you can add and attached all kinds of packages that will interact with the kernel. Based on that, anyone can create a distribution. Though, those distributions are based on **main** (or **base**) distributions that are actively developed. To name a few : Debian, Ubuntu (which is already a fork of Debian), Fedora, RedHat, Arch... The list goes on and on and on.

Almost one thousand Linux distributions exist. Because of the huge availability of software, distributions have taken a wide variety of forms, including those suitable for use on desktops, servers, laptops, netbooks, mobile phones and tablets, as well as in minimal environments typically for use in embedded systems. For instance, Android is based on a modified Linux kernel. You can find Linux on bank ATMs, airport flight screens or even sometimes when you order your burgers at your favorite fast-food restaurants.

In this course, we will use Linux as a desktop environment heavily use the terminal. Here is a list of the most popular, modern, distributions available :

Rank	Project
1	MX Linux
2	Mint
3	Manjaro
4	Debian
5	openSUSE
6	Ubuntu
7	Fedora
8	deepin
9	Solus
10	Pop!_OS
11	Zorin
12	Arch
13	KDE neon
14	elementary
15	EndeavourOS
16	Devuan
17	ArcoLinux
18	Lite
19	Kubuntu
20	Peppermint
21	Void
22	antiX
23	Slackware
24	Q4OS
25	Ubuntu MATE
26	Lubuntu
27	Mageia
28	PCLinuxOS
29	Garuda
30	Artix
31	Xubuntu
32	SparkyLinux
33	Puppy
34	Gentoo
35	Antergos
36	Linuxfx
37	Feren
38	Bodhi
39	Parrot
40	OpenBSD

Figure 2:

3 Labs

3.1 Shells

There are several popular shells or command interpreters for the Unix / Linux systems.

`sh`: Bourne Shell. The oldest and installed on all Unix-based OS, but poor in functionality compared to other shells.

`bash`: Bourne Again Shell. An improved shell, available by default on Linux and Mac OS X

`ksh`: Korn Shell. A powerful shell quite present on the proprietary Unix, but also available in free version, compatible with bash (Korn's shell includes Bourne's shell)

`csh`: C Shell. A shell using a syntax close to the C language

`tcsh`: Tenex C Shell. An improved C Shell

`zsh`: Z Shell. Pretty new shell with the best ideas of bash, ksh and tcsh

To exit a terminal type `exit`. It should be straightforward when you are using a graphical interface (you can click on the cross to exit the window) but when you are in remote instance, `exit` is the way to go.

Try this command `echo 'Hello world!'`. It should print `Hello world!` in the terminal. It is the equivalent of the `print('Hello world!')` in the terminal.

To edit files in the terminal, you can use `nano` or `vim`. `nano` is more simple in its approach than `vim` which requires knowledge on keybindings but has much more advanced features. If they are not present, please refer to the next section to install them.

3.2 Install packages

How you install packages depend on the distributions you are using

Debian and Ubuntu : `apt-get`

Fedora : `dnf`

CentOS and RedHat : `yum`

Arch and Manjaro : `pacman`

Do not underestimate the use of the help command `man`.

Question 3.1.

- On your distribution, use the package manager to install the `python` and `lua` languages. If not present, install as well `nano` and `vim` as terminal editors.

3.3 System identification

Use the `man` command to get help on commands described below (man command)

`date` : Show the date and time

`uname` : Get system identification

`who` : View the list of connected users

`whoami` : View the user of the current session

Question 3.2.
- Display the date as jj-mm-aaaa format (use + format option)
- Display the hour as hh:mm:ss format
- Identify the operating system
- Identify the operating system version
- Identify the hardware platform
- Identify the identity of the current user logged

3.4 Process management

`command` : Launch a command

`command &` : Launch a command on background

`[ctrl z]` followed by `bg`: Put a command on background

`ps` : View the list of user processes

`kill` : Stop a process

`top` : View the list of all processes

Question 3.3.
- Display all processes with details using `ps`
- Launch less command to browse a file
- Put the process corresponding to the `less` command to background
- Find the PID (process ID) of this process
- Stop the `less` process
- Display all processes with `top` and sort the processes by CPU usage then MEM usage

3.5 Folder structure

`/` : The root directory, represented by a forward slash (/), stores all the directories in Linux

`/boot` : The boot directory contains important files needed by the boot loader. The initial ram file system or initramfs is also stored here along with the kernel.

/bin : The /bin directory contains system commands and other executable programs. It is usually a link to /usr/bin. The binaries in this directory are available system-wide.

/etc : The /etc directory contains vital system configuration files such as startup scripts, networking files, user account-related files, etc. You have to edit configuration files in the /etc directory to make any system-wide changes.

/lib or /usr/lib. A library is a collection of pre-compiled code that executable binaries can use.

/opt : The /opt directory contains optional software packages to facilitate better compatibility of certain applications.

/tmp : Temporary folder. This folder is cleaned at each reboot.

/srv : Usually used for web or network services like network folders, ftp, etc.

/home : The /home directory stores an individual user's home directory. Each user has its own home directory.

/proc : The /proc directory is a pseudo-filesystem containing information about processes and kernel parameters. It is populated with data during boot-up and is cleaned when you shut down your Linux machine.

/usr : The /usr directory contains most of the files, libraries, programs, and system utilities.

/var : The /var directory is the storage space for system-generated variable files, and it includes logs, caches, and spool files.

/var/log : Logs are stored in this specific folder.

Go to the root directory using `cd /` and list all the directory on a list using `ls -al`. What do you see? What are the differences with the list above?

3.5.1 Paths

Absolute path: /dir/sub_dir_1/sub_dir_2/file

Relative path form working directory /dir: sub_dir_1/sub_dir_2/file (notice there is no / at the beginning)

3.6 Wilcards

/ system's root directory

. working directory

.. directory one level up

~ or $HOME login directory

~otheruser otheruser's home directory, if otheruser permission access is granted

`*` all files

3.6.1 Directory management

`pwd` : View the name of the working directory

`cd` : Change working directory

`ls` : View the content of a directory

`mkdir` : Create a directory

`rmdir` : Delete a directory

Question 3.4.

- Go to the home directory
- Find the name of the working directory
- Create a directory named `lab1`
- Display the content of the working directory
- Display the content of `lab1` directory
- Change the working directory to `lab1`
- Display the content of the working directory
- Create a directory named `lab2`
- Display the detailed content of the working directory
- Display all hidden entries of the working directory
- Display the content of the parent directory of working directory
- Remove the `lab1` directory

3.6.2 File management

`touch` : Create an empty file and change file date/time

`mv` : Rename or move a file

`cp` : Copy a file into another file

`rm` : Delete a file

Question 3.5.

- Create an empty file `newfile`
- Rename the `newfile` file to `file1`
- Copy `file1` file to `file2`
- Create a directory named ”lab1” in the working directory
- Move the `file1` file to `lab1` directory
- Delete `file1`, `file2` files and `lab1` directory

3.6.3 File content

`cat` : Display the content of a file or merge a list of files and display the result

`more` : Display the content of a file page per page

`less` : Display the content of a file page per page

`head` : Display the beginning of a file

`tail` : Display the end of a file

`wc` : Count the number of words, lines or characters

`diff` : Find differences between two files

Question 3.6.

- Dump the ‘journalctl‘ content into a new file with the command ‘journalctl ¿ journalctl.txt‘
- Display the content of ‘journalctl.txt‘ file
- Display the content of ‘journalctl.txt‘ file page by page
- Display the content of ‘journalctl.txt‘ file page by page of 10 lines
- Display the content of ‘journalctl.txt‘ file page by page of 10 lines, skipping the first 20 lines
- Display the beginning of ‘journalctl.txt‘ file
- Display the end of ‘journalctl.txt‘ file
- Count the number of lines of ‘journalctl.txt‘ file

3.7 Inputs, outputs

At each process created (command or program in execution) are associated with one input and two standard outputs:

The standard input : the keyboard

Standard output : the terminal screen

The standard error output : the terminal screen

The shell allows to modify the default inputs/outputs of a process.

Info.

- Redirection of standard input (`command < input_file_name`)
- Redirection of standard input, keybord is used as input and the `key_word` is used to stop the input process (`command << key_word`)
- Redirection of standard output (`command > output_file_name`) or (`command >> output_file_name`)
- Redirection of error (`command 2> error_file_name`) or (`command 2>> error_file_name`)
- Redirection of standard output and error (`command > output_file_name 2> error_file_name`) or (`command >> file_name 2> error_file_name`) or (`command > file_name 2>> error_file_name`) or (`command >> file_name 2>> error_file_name`)
- Redirection of standard output and error (`command > common_file_name 2>&1`) or (`command >> common_file_name 2>&1`)

3.7.1 Chaining commands

Sequential chaining It's possible to launch several commands (processes) sequentially with the operator ; (`command_1; command_2;...`)

Pipelines It's possible to launch several commands (processes) sequentially and redirect the output of the previous process to the input of the next process with the operator | (`command_1 | command_2 |...`)

Success and error It's possible to launch several commands (processes) sequentially with a condition of success and error.

`command_2` is launch if the `command_1` finish with success (`command_1 && command_2`)

`command_2` is launch if the `command_1` finish with error (`command_1 || command_2`)

Question 3.7 .

- Display all processes (`ps -ax`) with details page by page of 10 lines
- Output the list of all processes with details to a file called `process_list`
- Count the number of lines recorded in the `process_list` file
- Output the count of the number of lines recorded in the `process_list` file to a new file called `number_of_lines`
- Display the last line of the list of all processes with details
- Output the last line of the list of all processes with details to the `process_list` file
- Output the first and the last line of the list of all processes with details to the `process_list` file

3.8 Find and filters

3.8.1 System wise

`find` : search for files in a directory hierarchy

`fzf` : a command-line fuzzy finder

`sed` : stream editor for filtering and transforming text

Question 3.8 .

- Find files in your home directory
- Find directories in your home folder, use `$HOME` and `~` as wildcards for your home folder. What are the differences?
- List executables only in `/usr/`
- List directories only (without recursivity) in `/usr/lib`
- Sort the last result by name
- Pipe the result in `fzf` and select a directory, what is happening?
- Find all files in your home folder and filter to display only the `.txt` files
- Pipe the result in `fzf`
- Use a second pipe to open the select `.txt` file in a text editor, you need to use the command `xargs nano -`
- Type the command `fzf` in the current directory, what is happening?

Info . `fzf` is a **f**uzzy **f**inder, it is not a search tool. It can search/filter within a list. The command that `fzf` is running behind depends on the command passed to the variable `FZF_DEFAULT_COMMAND`

Question 3.9 .

- Print the command stored in the variable to your terminal using `echo $FZF_DEFAULT_COMMAND`, what do you see?
- Change the variable to find directories in your home folder (use `export FZF_DEFAULT_COMMAND=<yourcommand>`), then run `fzf` again
- Change the variable to find all files in the current working directory
- Go to `/usr/lib` and run `fzf`
- Run `journalctl | tail -n 10` to display the last 10 lines of the linux journal.?
- Change `$1` by `$2` then `$3` and so on... What do you see?
- Using `df` you can see all filesystems mounted on your system. Try to isolate the name of filesystem which is mounted as root (mounted on `/`)

The command `awk` is a filter information. We will use it to isolate information and compare or condition them to write small functions. Many UNIX utilities generates rows and columns of information. AWK is an excellent tool for processing these rows and columns, and is easier

to use AWK than most conventional programming languages. The syntax is `awk '{ action }'`, where action is what you want to do.

Question 3.10 .

- Let's start `cd $HOME/.config` and with `ls -al`. What do you see?
- Each column can be isolated using `$X` where `X` is the column number starting from `1`.
- Using pipe and `ls -al`, isolate the file name, the owner and group owner of the file.

3.8.2 In files

awk You can as well use `awk`, in files.

Question 3.11 . Output the content of `/etc/passwd` in the terminal and isolate the username. (Hint: you need to change the Field Separator (FS))

In order to quickly filter letters, strings, numbers... It is usually faster to use `grep`.

Question 3.12 .

- With the `journalctl.txt` file or the command `journalctl`, find the kernel version of your distribution. Find the occurence `Linux version`
- Using `grep`, filter all the text files (.txt) file in your home folder
- What are the differences with fzf?
- Using `grep`, display all the executables that start with an `a` in `/usr/bin`
- Display all the executables that do NOT contain the letter `e` in `/usr/bin`
- Display all the hidden directories in your home folder

sed `sed` is a powerful text stream editor. It can do insertion, deletion, search and replace (substitution).

Question 3.13 .
Create a text file with this text inside :

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os. Learn unix. Unix is
    powerful.
```

- We can find and substitute `unix` by `linux` by doing the command : `sed 's/unix/linux/ NOMDUFICHIER.txt`. The `/` is the delimiter for the strings. What do you see?
- Now to do it globally, we need to add `g` at the very end. It should be `sed 's/unix/linux/g' NOMDUFICHIER.txt`. Are there any `unix` left?
- Try to delete all 'unix' occurences in the file.

3.9 Users and groups

3.9.1 User definition

User's privileges are defined by a UID (User ID) and a GID (Group ID). Both are set in the `/etc/passwd` file. Groups are defined in the `/etc/group` file.

3.9.2 UID

The UID `0` has a special role: it is always the root account. The UIDs `1` through `99` are traditionally reserved for special system users.

Some Linux distributions begin UIDs for non-privileged users at `100`. Others, such as Red Hat, begin them at `500`, and still others, such Debian, start them at `1000`. The UID `65534` is commonly reserved for nobody, a user with no system privileges, as opposed to an ordinary. Also, it can be convenient to reserve a block of UIDs for local users, such as `1000` through `9999`, and another block for remote users (i.e., users elsewhere on the network), such as `10000` to `65534`. The important thing is to decide on a scheme and adhere to it.

3.9.3 GID

Unlike user 0 (the root user), group 0 does not have any special privilege at the kernel level. Traditionally, group 0 had special privileges on many unix variants — either the right to use su to become root (after typing the root password), or the right to become root without typing a password. Basically, the users in group 0 were the system administrators. When group 0 has special privileges, it is called wheel

3.9.4 umask and file permissions

On Linux, you need permissions to be able to read, write or execute files or directories.

The file permissions works as follows:

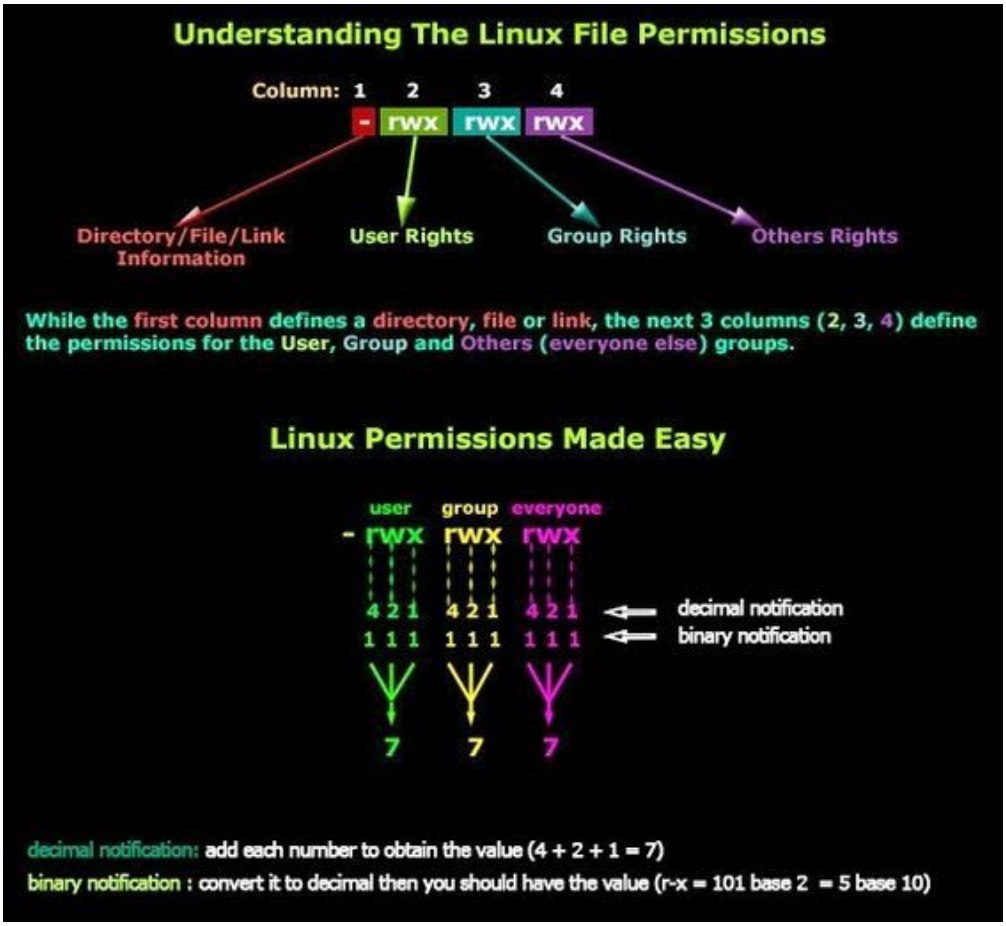

Figure 3: Linux file permission system

The umask is how you will **restrict**, by default, on newly created files. Think of working the opposite way of the file permissions describe in the image above.

Here is an example:

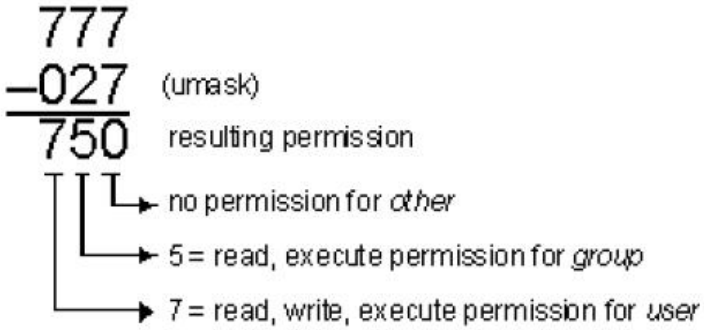

Figure 4: `umask` property to remove the permissions to a device or a file

You start with the full permission on read, write and execute for **owner**, **group** and **guest** then you substract what you want to restrict. Here we do not want **guest** to be able to use the files, and we restrict write permission for **group**

3.9.5 User management

`id` : View the user of the current session

`useradd` or `adduser` : Add a user

`groupadd` or `addgroup` : Add a group

`passwd` : Create or change password

3.9.6 Directory and file properties

`chmod` : Change the protections of a file `chown` : Change the owner of a file

Question 3.14 .

- Identify the user ID and group ID numbers
- Login as root user (`sudo -s`)
- Change working directory to the folder where you created files earlier
- Change the permissions as root on a file to give access to only the group `0` (Hint: modify the owner of the file)
- Login as regular user and try to open or read the file
- Identify the UID and GID of a library file in `/usr/lib`
- Print the content of `/etc/passwd` in the terminal
- In a new folder in your home folder, create a new file named `file_new` and give the following permissions: read and write for your user, read and execute for your group user and execute for guests

Question 3.15 .

- Grant read access to created files to all members of your group
- Grant read access to the working directory to all members your group
- Grant read permissions on the working directory to everyone on the system

3.10 Network

3.10.1 IP address

Your ip address is the label to identify your machine in a network. You usually have a local network (with a local ip address) and a DHCP, which serves locally ip addresses. Nowadays, your internet box is serving as a DHCP. Your ip address usually ressembles to `192.168.X.X` where `X` is a number between 0 and 255. Though, certain numbers are restricted, like 0 (identify the network) or 254 (usually router address or gateaway), 255 serves as broadcast address. In practice, we usually use a range from 1 to 253. In order to get an ip address, your machine needs an interface. This interface is either the ethernet connexion or Wi-Fi cards. Respectively, they are named `eth0` and `wlp0`.

Question 3.16 .

- `ifconfig -a` or `ip a` gives you the name of the interfaces on your machine
- Identify the interfaces you have
- Identify the ip address that is attributed to the active interface
- Deactivate your active interface and reactivate it, look at the ip address. Is it the same?
- Change the ip address of the active interface using `ifconfig`
- Verify that the ip address your entered is the correct one
- Using `ip` identify the ip address and gateway

3.10.2 Download files from internet

Question 3.17 .

- Do `curl google.com` and `wget google.com`, what are the outputs and differences? Try to use what you see and `man wget` and `man curl` to explain.
- You can as well use `git clone` to copy a git repository. Create a new folder and do `git clone https://github.com/torvalds/linux.git`. What do you see?

3.10.3 SSH

The Secure Shell Protocol (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the `ssh-keygen` program.

Question 3.18 .

- Connect to the host of your partner (command `ssh username@host`). In order to achieve this create a new user with a password (choose whatever you want for username and password).
- Next step is to create an SSH key.

To set up an SSH key, type in the following command:

```
ssh-keygen -t rsa
```

Follow the instructions prompt to you. The next step is to copy the key to the host you want to SSH in. Run the command:

```
ssh-copy-id -i ~/.ssh/id_rsa username@host
```

Question 3.19 .

- To increase security to your system, you could disable password authentication. To do so you need to change two settings in `/etc/ssh/sshd_config`. `PasswordAuthentication` and `ChallengeResponseAuthentication` should be set to `no`
- Now, try to connect again to the host, what is happening?
- Rename the file `~/.ssh/id_rsa` to `~/.ssh/id_rsa.bak` and try again to connect to the host. Explain what is happening.
- Try to securely copy a file to the other computer using the command `scp`

3.11 Scripting

Shell scripting is the set of commands to be executed such that the shell can execute them. It is said to be the combination of long and repeatable command sequences into one script so that it can be executed as and when required. The main idea behind creating a shell script is to lessen the load of the end-user.

A script file is just a text file with a shebang at the very beginning. A shebang is started with `#!` following by the binary of the shell you want to use. Here we will use `bash`, then the complete shebang is `#!/bin/bash`

You can set variables by doing for instance `test='this is a test string'` and to call the value of the variable you can use `$test`. Here let's print the string `echo $test`. It should display `this is a test string`. You can do the same with integers, floats, paths... (See below at functions to have more details)

Once your script is finished, you need to make it executable. To do so, in the terminal, you can type `chmod +x myScript.sh`, if the name of your script is `myScript.sh`. Then to execute your script you have to type `./myScript.sh` directly in your terminal.

3.11.1 Script parameters

`$#` : This parameter represents the total number of arguments passed to the script.

`$0` : This parameter represents the script name.

`$n` : This parameter represents the positional arguments corresponding to a script when a script is invoked such `$1`, `$2`...

`$*` : This parameter describes the positional parameters to be distinct by space. For example, if there are two arguments passed to the script, this parameter will describe them as `$1`, `$2`.

`$$` : This parameter represents the process ID of a shell in which the execution is taking place.

`$!` : This parameter represents the process number of the background that was executed last.

`$@` : This parameter is similar to the parameter `$*`.

`$?` : This parameter represents exit status of the last command that was executed. Here 0 represents success and 1 represents failure.

`$_` : This parameter represents the command which is being executed previously.

`$-` : This parameter will print the current options flags where the set command can be used to modify the options flags.

```
1    cat program.sh
2    echo "The File Name is: $0"
3    echo "The First argument is: $1"
```

```
4     echo "The Second argument is: $2"
```

```
1     sh program.sh ab cd
2     The File Name: program.sh
3     The First argument is: ab
4     The Second argument is: cd
```

3.12 Functions

Bash functions are like usual functions in C or PYTHON. You can call them numerous times to do the same task. The syntax for declaring a bash function is straightforward.

```
1     function_name () {
2       commands
3     }
```

Or you can specify the reserved word 'function':

```
1     function function_name {
2       commands
3     }
```

As an exemple, a simple function printing `hello, world` is:

```
1     hello_world () {
2        echo 'hello, world'
3     }
```

You can as well use variables, like in any other programmatic languages:

```
1     var1='A'
2     var2='B'
3
4     my_function () {
5     local var1='C'
6       var2='D'
7       echo "Inside function: var1: $var1, var2: $var2"
8     }
9
10    echo "Before executing function: var1: $var1, var2: $var2"
11
12    my_function
13
14    echo "After executing function: var1: $var1, var2: $var2"
```

Question 3.20 . Execute this simple program and explain what `local` is doing (put this in a shell script and execute the script).

To return values or strings, you can use the following method:

```
my_function () {
  local func_result="some result"
  echo "$func_result"
}

func_result="$(my_function)"
echo $func_result
```

Here the local variable `func_result` is created, and you can print to STDOUT the result using `echo`. Then you can pass the result of the function to a global variable using `$(my_function)`.

Lastly, to have functions asking for an input, you can use the following:

```
greeting () {
  echo "Hello $1"
}

greeting "Joe"
```

Question 3.21 .

- Here calling the function `greeting` with `"Joe"` in front of it you pass the string `"Joe"` to the first positional argument.
- Create a shell script that takes as input a text file and ask for a word (could be a simple string or a number, you need to use two positional arguments `$1` and `$2`) or a sentence to be found in the text file. Please think of potential errors (check if the given file is a text file, word or sentence and if it is not empty...)

3.13 Conditions and loops

Conditions are useful to control your script behaviour. It can bypass unnecessary parts if a condition is met for instance. In bash script, you can find the traditionnal `if`:

```
  if [[ conditional_statement ]]

then

  command

```

```
fi
```

As for loops, you can use the `for`:

```
#!/bin/bash
for i in {1..5}
do
   echo "Welcome $i times"
done
```

By combining both of them, we can do interesting things:

```
#!/bin/bash
FILES="$@"
for f in $FILES
do
        # if .bak backup file exists, read next file
    if [ -f ${f}.bak ]
    then
        echo "Skipping $f file..."
        continue # read next file and skip the cp command
    fi
        # we are here means no backup file exists, just use cp command to
            copy file
    /bin/cp $f $f.bak
done
```

In the `if` condition, it checks a .bak file exists for each file, if not, it creates one.

Question 3.22 .

- You can do as well the same if a folder exists or not.
- Create a small script to check if in the folder `/home/downloads/` you have a folder named `book`, and then checks if an `.pdf` file exists. If not, download the file at https://cloud.capiod.fr/s/WsMNqWnRAdSwnWk/download/Book-LinuxCommandLineandShellScriptingBible.pdf
- The input `$@` is waiting for a folder, if you do not enter a folder, the script won't work. How to modify the script to make it more robust?

4 Web servers

Info. First install `apache2` on your distribution. To be sure that it is correctly installed, do `apache2 --version`.

If there is something written on your terminal, apache should be running. In order to sure,

you can run the following command `hostname -I` and use your web browser to connect to `http://IPADDRESS` (replace `IPADDRESS` by the address given by `hostname -I`). You should see the apache welcome page.

Question 4.1. Create a folder in `/var/www/` to put your html (or other) content. Use `sudo` to create the folder. Don't forget that you need to change the permissions of this folder so that apache can read and write (use `chown` and `chmod`).

Now that the folder is created and configured, we need to put what we want to display.

Question 4.2. Create a HTML page called `index.html` in the created folder and put the following HTML code in it:

```
<html>
<head>
<title>This module is an introduction to Linux</title>
</head>
<body>
<h1>You are running an apache2 server on your machine, congratulations!</
    h1>
</body>
</html>
```

Now, we should tell apache to look at our newly created folder and read what's in it. In order to do that, we will create a configuration file called `my-site.conf` and we will place it in `/etc/apache2/sites-available/`

We will put inside the following content:

```
<VirtualHost *:80>
ServerAdmin your@email.address
ServerName XXX
ServerAlias XXX
DocumentRoot <FOLDER WHERE YOU PUT YOUR HTML CONTENT>
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Question 4.3 .

- Replace the XXX by the name of your server (it can be whatever you want). Explain what could `Alias` stands for?
- Explain as well the `*:80` at the very top.

Everything should be set and ready to run our web server.

> **Question 4.4** . If you access your address via `http://IPADDRESS`, what do you see? Explain what could be the reason

We need to activate our configuration (and deactivate the default configuration). To do that run the command `sudo a2ensite my-site.conf` to activate our configuration. To deactivate the default one, you need the command `sudo a2dissite`.

> **Question 4.5** . Review the `/etc/apache2/sites-available` folder and find what is the default configuration and complete the commande given above.

Then, you need to restart the apache server using `systemd` with `sudo systemctl restart apache2`.

> **Question 4.6** . Run the command `sudo apache2ctl configtest` and explain what is the error.

To suppress the error, we need to set the ServerName but globally (remember, it is noted in the `VirtualHost`). Create a file `/etc/apache2/conf-available/servername.conf` and add to it `ServerName XXX`. With the same name as in the `VirtualHost`. Restart the apache server and access your site at `http://XXX`.

4.1 Test you knowledge

If you have time during the web server session, test your knowledge on how to install a web server by installing a blog server (You will have to install PHP first!). Download `htmly` at `https://github.com/danpros/htmly/archive/refs/tags/v2.8.2.zip` and unzip the content in a new folder in the root directory of your web server. You will need to create a new configuration file and a new `VirtualHost`. Once this is done, you will need to visit `https://www.example.com/install.php` to finalize the installation.

4.2 A small point on systemd

To start a systemd service `sudo systemctl start SERVICE`

To stop a systemd service `sudo systemctl stop SERVICE`

To start a systemd service at boot `sudo systemctl enable SERVICE`

To remove a systemd service at boot `sudo systemctl disable SERVICE`

To list services on your system `sudo systemctl list-units --type=service`

5 Home assignment

In most of home assignments, you will create scripts with a menu that asks for specific items. The menu and the scripts need to be robust. For example, if the script asks for a file, and you give it a number, it needs to output an error and ask again (Hint: you need to use a `while` loop. Once a job is done, the script goes back to the menu and the scripts asks again. Create an escape (for instance `quit` or `q` as an answer).

For web-server assignments, the job is to install and configure a simple web server based on apache. You do not need to do anything fancy. You will archive and transfer the web server with instructions to install it correctly.

For the Nextcloud web server, the task might be difficult, it is advised to do this assignment ONLY if you feel confident! The job is to setup the server and show that you could upload one file. If not, at least you could show the front setup page.

Assignments : Pick one (or two if you are adventurous!)

Question 5.1. Create a script that takes a text file as an input. You will create a menu that asks to cat the text file, grep a file or a sentence, or grep all words that starts with a selected letter

Question 5.2. Create a script that asks to update the package manager, searches for a software, installs it or not, cleans the cache of the package manager and removes packages with its depencies

Question 5.3. Create a script that adds a new user using simple functions (you cannot use useradd for instance)

Question 5.4. Create a script that rewrites the command `pgrep`, and create a menu that asks to kill, start or restart the process (you have to check if the process is running or not)

Question 5.5. Create a script that displays information about the internet config, changes the IP address, activates or deactivates the ethernet or wifi network (using `ip`)

Question 5.6. Create a script that finds either all files or folders in a specific folder, `cat` the content of the selected file or `cd` in the folder. Use `fzf` to filter the files or folders.

Question 5.7. Create a script you can use as a notepad. The script will take as input (first positional parameter `$1`) the name of the note you want. If the note already exists it will update and adds the sentence at the very end, if note it will create it. Then the script asks want you want to write down in the note (usually a sentence).

Question 5.8. Create a script that takes as input a message you want to send on a discord channel that you own. You will use as a back-end script, the following python script:

```
import requests
import sys

#Webhook of my channel. Click on edit channel --> Webhooks --> Creates webhook
mUrl = "https://discord.com/api/webhooks/WEBHOOKS"

data = {"content": sys.argv[1]}
response = requests.post(mUrl, json=data)

print(response.status_code)
print(response.content)
```

This requires to create a webhook, and replace the url in the `mUrl` variable above. The procedure is the following: https://support.discord.com/hc/en-us/articles/228383668-Intro-to-Webhooks

Question 5.9. Create a script that displays all the configuration files of web servers you have on your machine. The script will ask which file to edit. Once the file is edited, the apache web server will be restarted to take into account the modifications.

Question 5.10. Install and configure an apache web server and put a link to your report file (the file should be in the same folder as the web server index.html page).

Question 5.11. Install and configure a Nextcloud server. The procedure to install a Nextcloud web server on Ubuntu is the following: https://docs.nextcloud.com/server/latest/admin_manual/installation/source_installation.html