UNIX

R1.04 TP5

N. Delomez – J.F Kamp - X. Roirand

Prénom Nom : Lucien Carré Date : 10/12/24
Groupe : 1D2

REMARQUES : Comme dans le TP précédent, vous devez fournir une copie d'écran de la commande et de l'effet de la commande (quand il y en a un) pour chaque question dans ce document.

Ce TP5 a un énoncé en anglais, vous pouvez y répondre en anglais ou en francais. Tout ce qui sera sur fond bleu signifiera qu'il faut répondre à une question. Le reste est de l'information ou des élements qui vont vous permettre de répondre aux questions suivantes.

TP 05 : The command line in Linux

Introduction: This is the second of a two-part introduction to the GNU/Linux operating system installed on the machines of the Technoloy University Institute department. It gives you some basic knowledge that you will use throughout your courses. All notions discussed here will be considered as acquired and mastered for all future courses regardless of the module.
This third TP invites you to (re)discover the command line (shell).
Connecting to your session gives you access to your work environment. You will need to open a shell (Terminal) to enter the instructions of this TP.

4 Process Management

We call "process management" the manipulation of programs running from the terminal. There are also graphical interfaces, such as in Windows or in Mac OS, that can do what we will do in this section.

A running program is called a **process**. Processes are identified on the system by a number called "PID" (process identifier).

Under Unix on the command line, a process can be executed in two different ways :

- In the background : While the process is running in the background, you can launch other processes in the shell. The background process can print output to the terminal, but cannot receive input from the keyboard.

- Foreground : This is the normal mode of operation. The foreground process receives input from the keyboard, and the shell waits until the process has finished before letting you start other command.

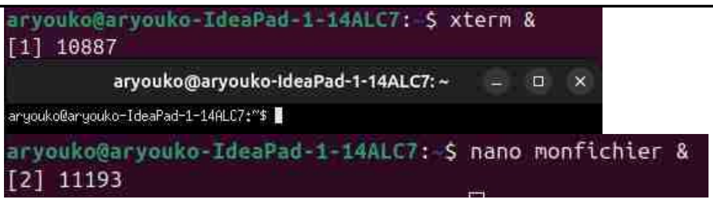
We will see that it is possible to switch a process from one state to another in the terminal, and to kill processes. We will also see that it is possible to list running processes in a terminal thanks to the **"jobs"** command.

4.1 Process in the background

Ensure that you have xterm command installed on your laptop before running the above commands. In case you can not have xterm executable installed on your laptop, you can replace it with any executable like kconsole or gnome-terminal.

Putting an "ampersand"(&) at the end of a command makes a process launch in the background. Run the following code :

```
xterm  &
nano   monfichier &
```



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ xterm &
[1] 10887
```

aryouko@aryouko-IdeaPad-1-14ALC7: ~

```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ 
```

```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ nano monfichier &
[2] 11193
```

Q1) Why the current shell is not displaying the editor with the monfichier edited inside ?

Je ne peux pas éditer le fichier car il est exécuter en arrière plan je ne peux donc pas voir l'exécution de nano.

Q2) What are the two numbers displayed on the screen just after your nano command ? Explain what they are representing ?

Le premier numéro signifie le numéro du job que le shell lui a attribué.

Le second numéro signifie le PID du processus

Q3) Are the nano and xterm processes still running ?

Oui mais on ne le voit pas

Example of a possible output, after running xterm& :

```
[1]   1360
```

and after running nano & :

```
[2]   1366
```

4.2 Switching from one state to another

When a process is started in "foreground", you cannot interact with the shell that started it. To "get your control back" you have to switch the process to the "background". The combination of keys "CTRL + Z" **suspends** a foreground process and hands control to the shell. Then the "bg" command (for "background") resumes the process, but in the background.

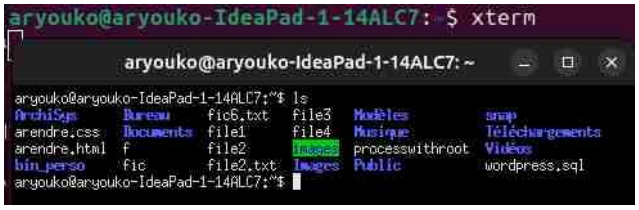
Note : You'll find that the "Control" key has different notations. Two of the those are «CTRL» and «ˆ».

Conversely, the last process put in the background (for example with ending the command by "&"), can be switched to the foreground with the command "fg".

Preform and answer the following questions :

— In a terminal launch a new terminal "xterm" (be careful to regulary start the xterm, not in backgound)

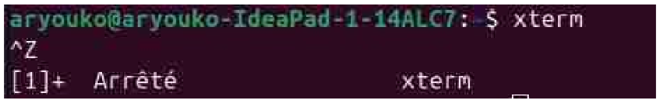
— run the command “ls” in the new terminal and check that it works fine



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ xterm
```

```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ ls
ArchiSys      Bureau     fic6.txt   file3   Modèles          snap
arendre.css   Documents  file1      file4   Musique          Téléchargements
arendre.html  f          file2      images  processwithroot  Vidéos
bin_perso     fic        file2.txt  Images  Public           wordpress.sql
aryouko@aryouko-IdeaPad-1-14ALC7:~$
```

— Q4) put the new terminal to sleep

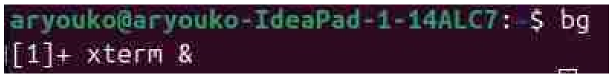


```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ xterm
^Z
[1]+  Arrêté                 xterm
```

— Q5) to run the ls command in the new terminal, what's going on ? Why ?

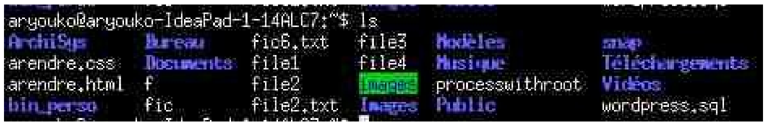
Il est impossible de faire des entrées avec le clavier dans le nouveau terminal car le processus est arrêté on ne peut donc plus y “accéder”.

— Q6) switch the terminal in the background



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ bg
[1]+ xterm &
```

— Q7) relaunch "ls" what's going on ? Why ?



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ ls
ArchiSys      Bureau     fic6.txt   file3   Modèles          snap
arendre.css   Documents  file1      file4   Musique          Téléchargements
arendre.html  f          file2      images  processwithroot  Vidéos
bin_perso     fic        file2.txt  Images  Public           wordpress.sql
```

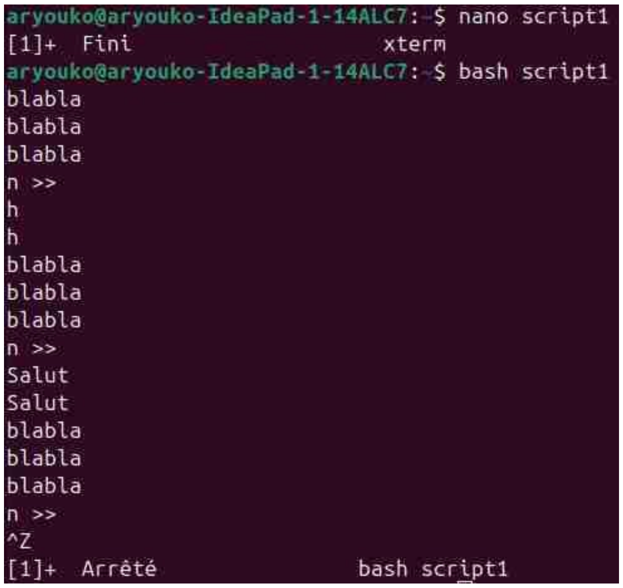
J'ai pu relancer le ls car il n'est plus arrêté il a redémarré mais en arrière plan cette fois. Petite information, tout ce qu'on écrit lors de l'arrêt réapparait dès qu'on réécrit lorsqu'on relance.

4.2.1 Application and process states

Create a new file with the name "script1" and the following code in it using your favorite editor

```
while true; do
for i in {1 .. 5}; do
    echo blabla
    sleep 1
done
echo n ">> "
read a
echo $a
done
```

Run the following command "bash script1", when the line in terminal displays « >> » enter a character and confirm with "Enter".

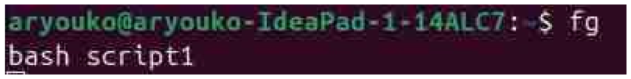


```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ nano script1
[1]+  Fini                    xterm
aryouko@aryouko-IdeaPad-1-14ALC7:~$ bash script1
blabla
blabla
blabla
n >>
h
h
blabla
blabla
blabla
n >>
Salut
Salut
blabla
blabla
blabla
n >>
^Z
[1]+  Arrêté                  bash script1
```

Q8) After several executions, put the script to sleep. What is happening ?

Le script se met en arrêt (pause) et on ne peut plus continuer

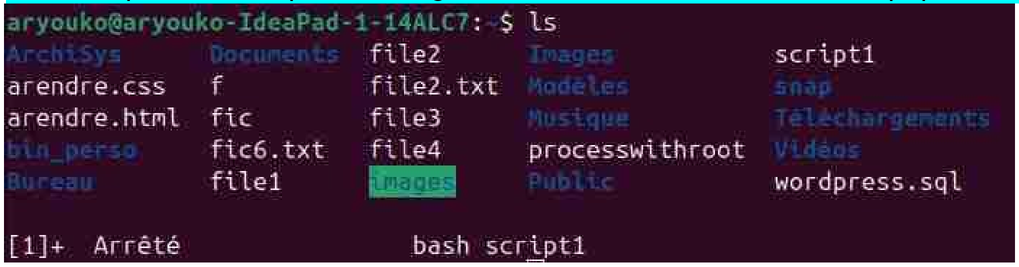
Q9) Wake up the process in the foreground, what's going on ?



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ fg
bash script1
```

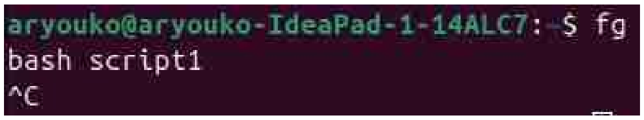
La commande de base se réaffiche et le script attend un n comme avant sans le redemander

Q10) Toggle the process in the background. What happens if you type "ls" once the process has been suspended and put in background ? Re-take control of the script process and stop it.



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ ls
ArchiSys      Documents   file2      Images          script1
arendre.css   f           file2.txt  Modèles         snap
arendre.html  fic         file3      Musique         Téléchargements
bin_perso     fic6.txt    file4      processwithroot Vidéos
Bureau        file1       images     Public          wordpress.sql

[1]+  Arrêté                  bash script1
```

ls affiche les fichiers et dossiers mais aussi le script qui est en arrière plan avec sa commande



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ fg
bash script1
^C
```

4.3 Kill a process in foreground

To kill a process running in the foreground, enter the sequence "CTRL + C" in the terminal running the process.

4.4 Jobs

Using bash’s help page in the "JOB CONTROL" section, run the following commands and keyboard sequences and analyze the behavior of the processes (you will of course need two terminals, one for the manpage, the other for exercise).

Note : Remember to reposition yourself in the terminal after launching applications.

Note: If you don’t have firefox installed on your computer, used any other navigator instead. If you have firefox, you may need to change the command to start it, and use:

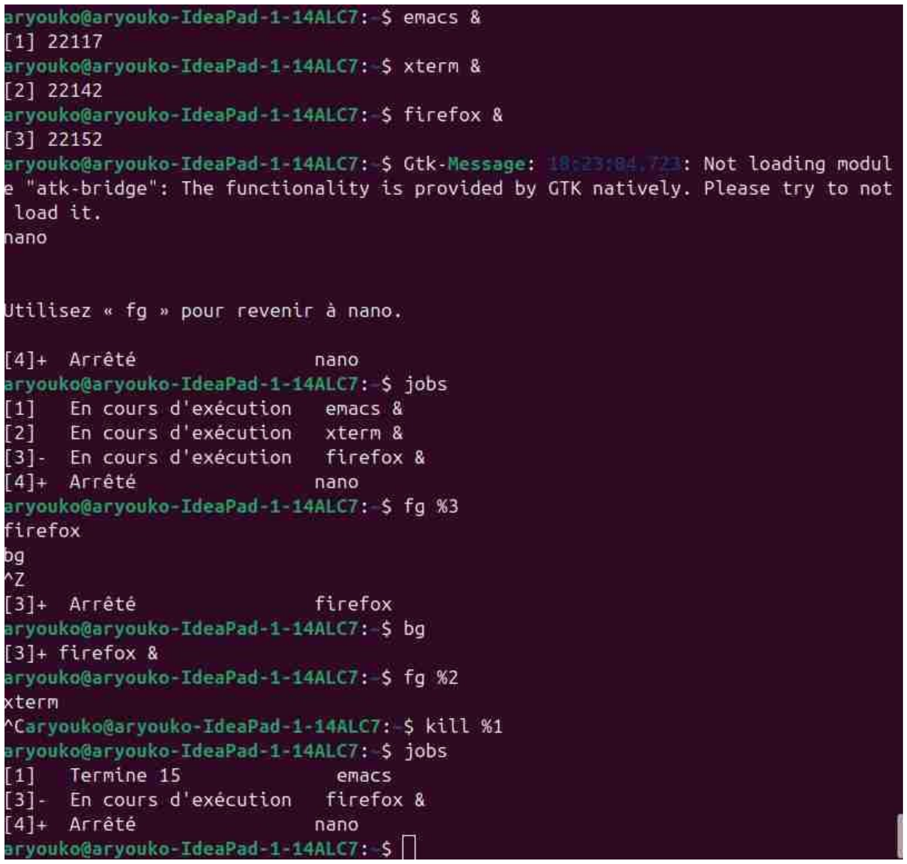
/snap/firefox/2154/usr/lib/firefox/firefox

If not you may encouter an immediat exit. To find the proper command, start a firefox instance using the desktop or toolbar icon, then run “ps auwx | grep firefox” and find the proper command in the last part of the line

Notice that nano handles Ctrl+Z so you have to do a Ctrl+T+Z to suspend it.

```
emacs &
xterm &
firefox &
nano
Ctrl T+Z
jobs
fg %3
Ctrl +Z
bg
fg %2
Ctrl +C
kill %1
jobs
```

/snap/firefox/5361/usr/lib/firefox/firefox



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ emacs &
[1] 22117
aryouko@aryouko-IdeaPad-1-14ALC7:~$ xterm &
[2] 22142
aryouko@aryouko-IdeaPad-1-14ALC7:~$ firefox &
[3] 22152
aryouko@aryouko-IdeaPad-1-14ALC7:~$ Gtk-Message: 18:23:04.723: Not loading modul
e "atk-bridge": The functionality is provided by GTK natively. Please try to not
 load it.
nano


Utilisez « fg » pour revenir à nano.

[4]+  Arrêté                nano
aryouko@aryouko-IdeaPad-1-14ALC7:~$ jobs
[1]   En cours d'exécution   emacs &
[2]   En cours d'exécution   xterm &
[3]-  En cours d'exécution   firefox &
[4]+  Arrêté                nano
aryouko@aryouko-IdeaPad-1-14ALC7:~$ fg %3
firefox
bg
^Z
[3]+  Arrêté                firefox
aryouko@aryouko-IdeaPad-1-14ALC7:~$ bg
[3]+ firefox &
aryouko@aryouko-IdeaPad-1-14ALC7:~$ fg %2
xterm
^Caryouko@aryouko-IdeaPad-1-14ALC7:~$ kill %1
aryouko@aryouko-IdeaPad-1-14ALC7:~$ jobs
[1]   Termine 15             emacs
[3]-  En cours d'exécution   firefox &
[4]+  Arrêté                nano
aryouko@aryouko-IdeaPad-1-14ALC7:~$ 
```

Q11) What process remains after the execution of these commands ?

Nano reste après l'exécution des commandes en statut arrêté et emacs en statut terminé et firefox en cours

Q12) Type the command "monnavigateur" and in a terminal explain all the result lines

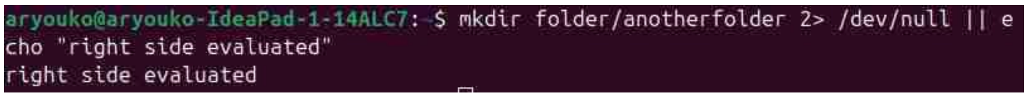
aryouko@aryouko-IdeaPad-1-14ALC7:~$ ps aux | grep firefox
aryouko   11596 0.0 1.2 2406352 73236 ? Sl   déc.09   0:01 /snap/firefox/5361/usr/lib/firefox/firefox
on voit l'utilisateur qui l'a exécuté avec aryouko, son pid avec 11596, son statut avec Sl pour sleeping et son chemin d'accès avec /snap/firefox/5361/usr/lib/firefox/firefox

4.5 Chaining processes

When a program finish executing, it returns a value related to how the process stopped. This value is usually interpreted as an indication of whether the command succeeded. It can be used to chain commands. To this end we will use the operators "&&" for "AND" and "||" for "OR".

"&&" will run the command on its right side only if the left side "succeeded"' and "||" will run the command on its right side only if the left side "failed". For example command

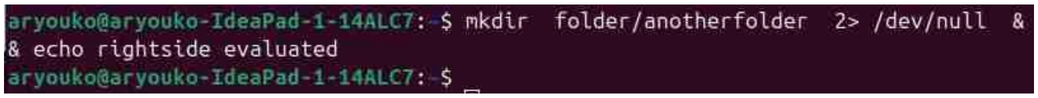
```
mkdir  folder/anotherfolder  2> /dev/null  || echo rightside evaluate
```



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ mkdir folder/anotherfolder 2> /dev/null || e
cho "right side evaluated"
right side evaluated
```

will print "right side evaluated", while the following command :

```
mkdir  folder/anotherfolder  2> /dev/null  && echo rightside evaluated
```



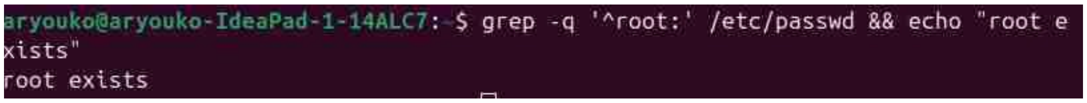
```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ mkdir  folder/anotherfolder  2> /dev/null  &
& echo rightside evaluated
aryouko@aryouko-IdeaPad-1-14ALC7:~$
```

will not print anything.

4.5.1 First process chain

Q13) With the "grep" command, test the existence of the "root" account on your system and display the message : "root exists" if the account is found.

Note : The file "/etc/passwd" contains the list of existing accounts on the system.

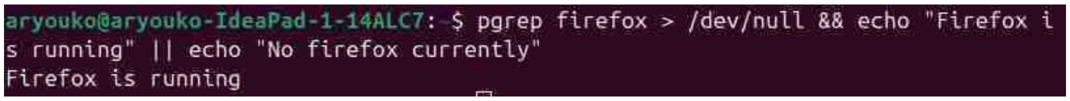


```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ grep -q '^root:' /etc/passwd && echo "root e
xists"
root exists
```

4.5.2 More advanced chain

Q14) Using the operators and what has been seen previously, write a command displaying **ONLY** the text "Firefox is running" on the standard output if the firefox program is running on your machine and "No firefox currently" otherwise.

Note: you can choose another navigator if firefox is not installed on your computer.



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ pgrep firefox > /dev/null && echo "Firefox i
s running" || echo "No firefox currently"
Firefox is running
```

5 The GNU/Linux operating system

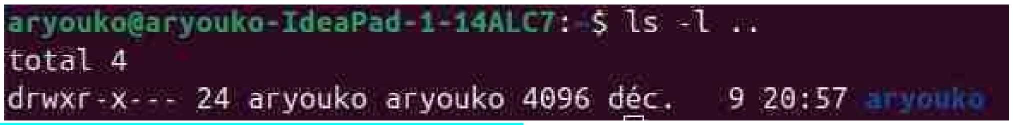
5.1 Overview

— The root of the Linux operating system is marked "/".
— On Linux everything is file.
— A user is a person or program registered on the system (with an account).
— Groups are used to classify users based on criteria defined by the system administrator.
— Users can belong to one or more groups.
— Users have privileges based on their membership groups.
— For the system, a user is a number called "UID" (User identifier) associated with a "login".
— For the system, a group is also a number.
— The user has a primary or default group whose number is "GID" (Group Identifier).
— The rights, or permissions, that can be assigned to a file are : no rights noted "-", "read" denoted by "r", "writing" denoted by "w" and execution rights or right to traverse the folder "x".
— Rights can be granted to users, groups or others (people not belonging to the two previous sets).
— Rights are displayed as triplets "rwx" for each set to which they can be granted (in the end we get permissions from «   », no rights, and « rwxrwxrwx » everyone can read write and access the file).
— The "-l" option of "ls" shows the rights of to the files.

— The "chmod" (change mode) command allows you to modify the file rights.
— The "chown" (change owner) command is used to modify the owner and/or group owning a file.
— On a corporate system, a single user can not elevate his privileges without authorization.
— The home directory of a user is noted "~login". If "login" is omitted, the current user is implied.
— "groups" allows to list the membership groups of a user and "id" specifies their number.
— "pwd" Locates you on the system (displays the current directory you are in).

5.2 Questions

Q15) What are the rights on your homedir ?



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ ls -l ..
total 4
drwxr-x--- 24 aryouko aryouko 4096 déc.   9 20:57 aryouko
```

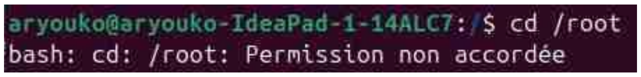
Q16) Which group does your homedir belong to ?

aryouko

Q17) Which group do you belong to ? and your neighbor ?

aryouko adm cdrom sudo dip plugdev users lpadmin
Je n'ai pas de voisin

Q18) Go to "/root". What is going on ? Why ?

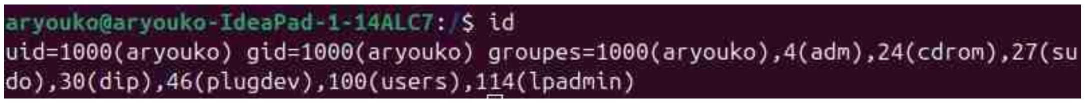


```
aryouko@aryouko-IdeaPad-1-14ALC7:/$ cd /root
bash: cd: /root: Permission non accordée
```

Je ne peux pas y accéder car je n'ai pas les permissions pour, seul le root peut.

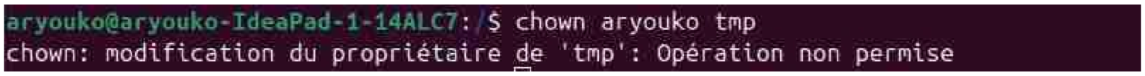
Q19) What is your gid ?

— "chmod" has two ways to represent these permissions with an octal number and with symbols. Give two commands, one with the octal notation, the other with alphabetical notation, to define the following rights on the directory "mydir" (starting rights "rwxr-xr-x") :
— rwx
— rwxr-x



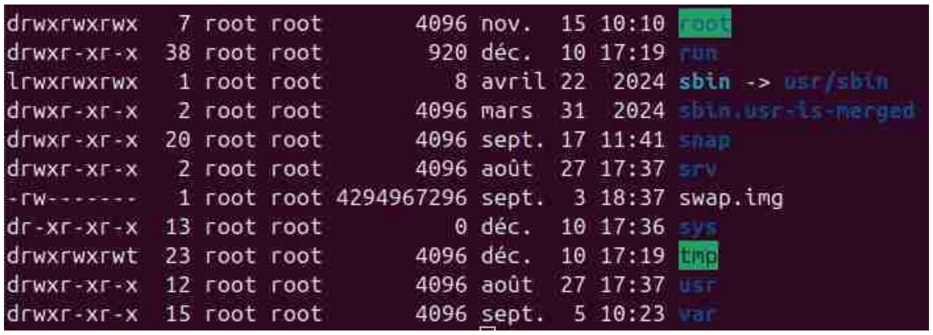
```
aryouko@aryouko-IdeaPad-1-14ALC7:/$ id
uid=1000(aryouko) gid=1000(aryouko) groupes=1000(aryouko),4(adm),24(cdrom),27(su
do),30(dip),46(plugdev),100(users),114(lpadmin)
```

Q20) Change the owner of the tmp directory. What is going on ? Explain.



```
aryouko@aryouko-IdeaPad-1-14ALC7:/$ chown aryouko tmp
chown: modification du propriétaire de 'tmp': Opération non permise
```

Ce n'est pas possible car là aussi seul le root peut.



```
drwxrwxrwx   7 root root        4096 nov.  15 10:10 root
drwxr-xr-x  38 root root         920 déc.  10 17:19 run
lrwxrwxrwx   1 root root           8 avril 22  2024 sbin -> usr/sbin
drwxr-xr-x   2 root root        4096 mars  31  2024 sbin.usr-is-merged
drwxr-xr-x  20 root root        4096 sept. 17 11:41 snap
drwxr-xr-x   2 root root        4096 août  27 17:37 srv
-rw-------   1 root root 4294967296 sept.  3 18:37 swap.img
dr-xr-xr-x  13 root root           0 déc.  10 17:36 sys
drwxrwxrwt  23 root root        4096 déc.  10 17:19 tmp
drwxr-xr-x  12 root root        4096 août  27 17:37 usr
drwxr-xr-x  15 root root        4096 sept.  5 10:23 var
```

6 The variables

On Unix, the system is able to exchange or retrieve information by using variables. These variables are made accessible by the Shell. Some are operating system variables (environment variables), others are shell-specific. We will study some of them, how to modify them, create them and display them.

6.1 Declaration

The following is a template for declaring a variable :

```
name_of_var=v a l u e
```

Where "value" is a string, and "name_of_var" is a string consisting of basic ASCII characters and not starting with numbers. For example :

```
name="J e a n "
```

6.2 Viewing

Bash (which is a type of shell) has the command "echo". This command allows you to display the value associated with the name of a variable. Variables must be prefixed by the character "$". The following command will display the value associated with the variable named "name" :

```
echo  $name
```

Although this syntax works, the recommended syntax by bash is :

```
echo  $ {name}
```

6.3 Modification

To changing a variable one needs to overwrite its previous declaration.

6.4 Exercices

Q21) View the contents of the HOME variable. What do you think this variable represents ?

aryouko@aryouko-IdeaPad-1-14ALC7:/$ echo $HOME
/home/aryouko
La variable représente le chemin absolu de mon répertoire utilisateur

Q22) View the contents of the PWD variable. What do you think this variable represents ?

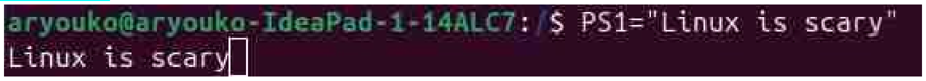
aryouko@aryouko-IdeaPad-1-14ALC7:/$ echo $PWD
/
La variable représente le chemin absolu de où je suis.

Q23) View the contents of the PATH variable. What do you think this variable represents ?

aryouko@aryouko-IdeaPad-1-14ALC7:/$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin:/home/aryouko/bin_perso
La variable englobe les chemins absolus de là où sont situés les commandes utiles.

4. View the contents of the PS1 variable. What do you think this variable represents ? Look in the help page of bash.

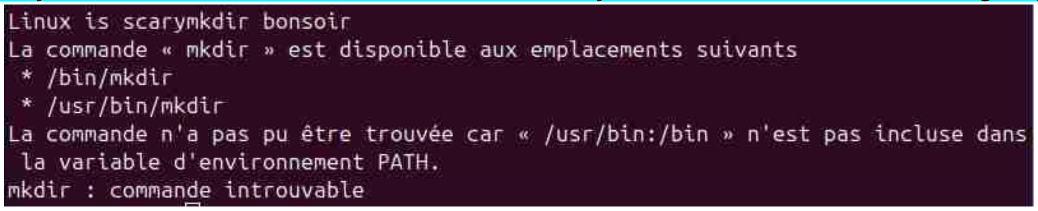
Q24) Try changing it to something else (for example "Linux is not scary :"). Re-launch a new terminal. What do you notice ?



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ PS1="Linux is scary"
Linux is scary
```

La chose qui s'affiche à gauche dans le terminal à changer mais n'ai pas resté sur les autres terminal.

Q25) Modify the contents of the PATH variable, then try to run a command. What is going on ?



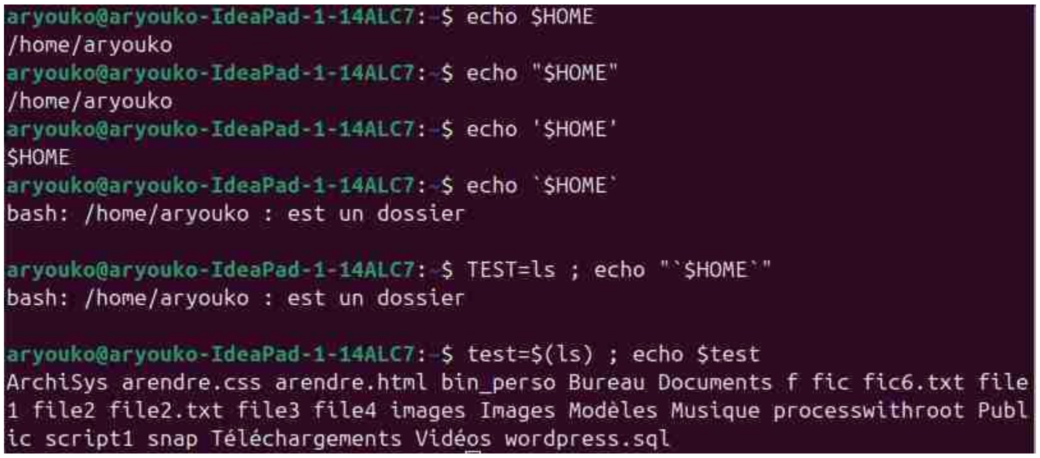
```
Linux is scarymkdir bonsoir
La commande « mkdir » est disponible aux emplacements suivants
 * /bin/mkdir
 * /usr/bin/mkdir
La commande n'a pas pu être trouvée car « /usr/bin:/bin » n'est pas incluse dans
 la variable d'environnement PATH.
mkdir : commande introuvable
```

Je ne peux plus accéder à la commande car le chemin pour n'existe plus.

6.5 String of characters

In the shell you can find strings between double quotes ("foo"), single quote ('foo') or anti-quote (‘foo‘).

Q26) Test the following commands, and explain what happens :



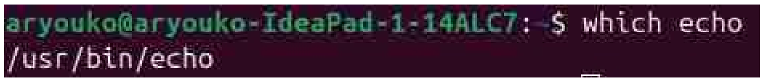
```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ echo $HOME
/home/aryouko
aryouko@aryouko-IdeaPad-1-14ALC7:~$ echo "$HOME"
/home/aryouko
aryouko@aryouko-IdeaPad-1-14ALC7:~$ echo '$HOME'
$HOME
aryouko@aryouko-IdeaPad-1-14ALC7:~$ echo `$HOME`
bash: /home/aryouko : est un dossier

aryouko@aryouko-IdeaPad-1-14ALC7:~$ TEST=ls ; echo "`$HOME`"
bash: /home/aryouko : est un dossier

aryouko@aryouko-IdeaPad-1-14ALC7:~$ test=$(ls) ; echo $test
ArchiSys arendre.css arendre.html bin_perso Bureau Documents f fic fic6.txt file
1 file2 file2.txt file3 file4 images Images Modèles Musique processwithroot Publ
ic script1 snap Téléchargements Vidéos wordpress.sql
```

Les " " servent à dire que c'est une chaine de caractère mais laisse passer les variables, les ' ' servent juste à dire que c'est une chaine de caractère il ne laisse pas les commandes passer et les ` ` permettent de savoir la valeur de la variable est quelle est le type du fichier au bout du chemin de HOME.

```
echo $HOME
echo  "$HOME"
echo    '$HOME'
echo    ‘$HOME‘
TEST= ls  ;  echo  " ‘ $TEST ‘ "
 test=$ ( ls )  ;  echo  $ test
```

7 Bonus

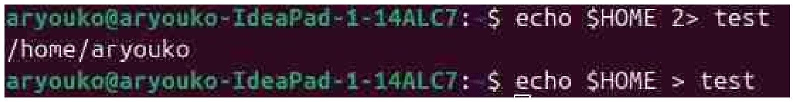
Q27) What does the command **which** do ?



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ which echo
/usr/bin/echo
```

La commande sert à savoir le chemin absolu d'une commande.

Q28) What is the difference between **>** and **2>** ?



```
aryouko@aryouko-IdeaPad-1-14ALC7:~$ echo $HOME 2> test
/home/aryouko
aryouko@aryouko-IdeaPad-1-14ALC7:~$ echo $HOME > test
```

Le > permet de l'enregistrer (Plain text document) tandis que le 2> permet juste la création d'un fichier (empty document) en affichant le résultat de la commande

Q29) How to mix these two redirects ?

echo $HOME > test 2> test1

Q30) What does the command **pkill** do ?

Pkill permet de tuer plusieurs precessus en même temps

Q31) What does the commands **less** and **more** ? do ?

Less et more servent pour la pagination

Q32) What does the command **yes** do ? What do you think it can be used for ?

yes permet d'afficher en boucle la même chose jusqu'à ce qu'on la tue

Q33) Search the internet for an answer that explains why the names of hidden files start with a dot.

Les fichiers cachés servent à l'identifier plus facilement pour ne pas avoir à le chercher et à ne pas surcharger l'affichage.

Bonus (part 2)

This bonus is made of a game where you have level to pass. For passing each level, you must properly use Linux commands and proper understand the given explanation.

Level 11:

The password for the next level is stored in the file **data.txt**, which contains base64 encoded data

Level 12:

The password for the next level is stored in the file **data.txt**, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions

Level 13:

The password for the next level is stored in the file **data.txt**, which is a hexdump of a file that has been repeatedly compressed. For this level it may be useful to create a directory under /tmp in which you can work using mkdir. For example: mkdir /tmp/myname123. Then copy the datafile using cp, and rename it using mv (read the manpages!)

Level 14:

The password for the next level is stored in **/etc/bandit_pass/bandit14 and can only be read by user bandit14**. For this level, you don't get the next password, but you get a private SSH key that can be used to log into the next level. **Note: localhost** is a hostname that refers to the machine you are working on.

Level 15:

The password for the next level can be retrieved by submitting the password of the current level to **port 30000 on localhost**.

Level 16:

The password for the next level can be retrieved by submitting the password of the current level to port 30001 on localhost using SSL encryption.

Helpful note: Getting "HEARTBEATING" and "Read R BLOCK"? Use -ign_eof and read the "CONNECTED COMMANDS" section in the manpage. Next to 'R' and 'Q', the 'B' command also works in this version of that command…

Level 17:

The credentials for the next level can be retrieved by submitting the password of the current level to **a port on localhost in the range 31000 to 32000**. First find out which of these ports have a server listening on them. Then find out which of those speak SSL and which don't. There is only 1 server that will give the next credentials, the others will simply send back to you whatever you send to it.

Level 18:

There are 2 files in the homedirectory: passwords.old and passwords.new. The password for the next level is in passwords.new and is the only line that has been changed between passwords.old and passwords.new

NOTE: if you have solved this level and see ‘Byebye!’ when trying to log into bandit18, this is related to the next level, bandit19

Level 19:

The password for the next level is stored in a file **readme** in the homedirectory. Unfortunately, someone has modified **.bashrc** to log you out when you log in with SSH.

Level 20:

To gain access to the next level, you should use the setuid binary in the homedirectory. Execute it without arguments to find out how to use it. The password for this level can be found in the usual place (/etc/bandit_pass), after you have used the setuid binary.