

# UNIX

## TP5

N. Delomez – N. Le Sommer – F. Lesueur - X. Roirand

<b>Prénom Nom :..... Date :.....</b>
<b>Groupe :.....</b>

**REMARQUES : Pas de copier / coller ou de copie d'écran à rendre dans ce document.**

Ce TP5 a un énoncé en anglais, vous pouvez y répondre en anglais ou en français. Tout ce qui sera sur fond bleu signifiera qu'il faut répondre à une question. Le reste est de l'information ou des éléments 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 Technology 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.

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

```
1 xterm &  
  nano monfichier &
```

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

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

Q3) Does the nano and xterm processes still running ?

Example of a possible output, after running xterm& :

```
[1] 1360
```

and after running nano & :

```
1 [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 «^».

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".

Perform and answer the following questions :

- In a terminal launch a new terminal "xterm" ;
- run the command "ls" in the new terminal ;
- Q4) put the new terminal to sleep ;
- Q5) to run the ls command in the new terminal, what's going on ? Why ?
- Q6) switch the terminal in the background
- Q7) relaunch "ls" what's going on ? Why ?

### 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".

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

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

Q10) Toggle the process in the background. What happens if you type "ls" when ">>" is displayed ?  
Re-take control of the script process and stop it.

## 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.

```
1 emacs &
2 xterm &
3 firefox &
4 nano
5 Ctrl + C
6 jobs
7 fg %3
8 Ctrl + Z
9 bg
10 fg %2
11 Ctrl + C
12 kill %1
13 jobs
```

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

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

## 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

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

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

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

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.

### 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.

## 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 ?

Q16) Which group does your homedir belong to ?

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

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

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

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

## 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 :

```
1 name_of_var=value
```

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

```
1 name="Jean"
```

### 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" :

```
1 echo $name
```

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

```
1 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 ?

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

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

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 ?

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

## 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 :

```
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 ?

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

Q29) How to mix these two redirects ?

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

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

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

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



## 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.

---