TP n°2: I/O redirection and access permissions

1. Objectives

The aim of the first part of this practical course is to learn how to redirect input, output and errors to target files and the objective of the second part of this practical course is to learn how to protect files and directories by assigning access permissions to different categories of users (owner, group and other users).

2. I/O Redirection and pipe

The bash shell has three basic streams;

- *input from **stdin** (stream **0**),*
- *output to **stdout** (stream **1**),*
- *error messages to **stderr** (stream **2**).*

2.1 Output redirection

*stdout can be redirected using a greater than sign (>). When analyzing the line, the shell will see the **>** sign and clear the file.*

Example :

```
echo "Hello World" > test.txt
```

Erasing a file when using > can be avoided by setting the **noclobber** option and t*he **noclobber** can be overruled by using the symbol* **>|**.

>> append

We use the symbol >> to append output to a file

2> stderr

Redirection of stderr is done with 2>. This can be very useful for avoiding error messages cluttering up your screen. The standard file used for this operation is ***/dev/null.***

< stdin

Redirecting **stdin** is done with <.

2.2 Pipe

A pipe is a form of redirection in Linux used to connect the **STDOUT** of one command into the **STDIN** of a second command. It allows us to narrow the output of a string of commands until we have an easily digestible amount of data. The pipe character is the | symbol and is placed between any two commands

Example:

cmd1 arguments1 | cmd2 rguments2

3. File and directory access permissions

3.1 Overview

Access rights are used to protect files or directories against unauthorized reading, writing or execution. The main kinds of file and directory access rights are as follows:

- r (read): read rights. It is possible to view or copy the file and, in the case of a directory, to list its contents.
- w (write): write access. This access right allows you to modify, remove or rename a file and, in the case of a directory, to add or remove files in the directory concerned.
- x (execute): execution rights. These are the rights assigned to files that can be executed, such as programs. In the case of a directory, this gives you access to all its sub-directories.

There are three categories of users for whom access rights can be defined:

- Owner: the user.
- Group: other people belonging to the same group as the user. Groups are established and updated by the system administrator.
- Other: any other person.

Files are protected by a 9-bit binary protection code.

Owner (u)	**Group (g)**	**Other (o)**
3 bits	3 bits	3 bits
rwx	rwx	rwx

1st bit	2nd bit	3rd bit
r	w	x

3.2 Octal notation for permissions

There is another, simpler way of indicating permissions in each category, using octal numbering:

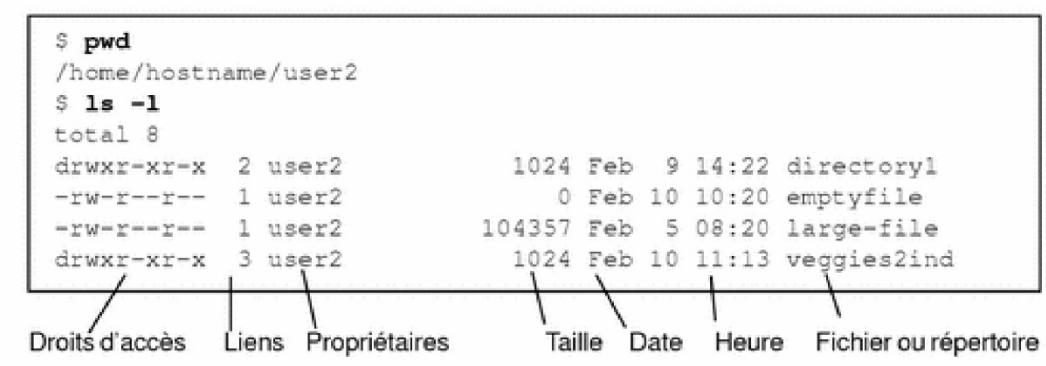
- 4 for read,
- 2 for write,
- 1 for execute.

Correspondence of permissions in binary/octal and their meanings:

string representation	numerical representation	single number representation
---	000	0
--x	001	1
-w-	020	2
-wx	021	3
r--	400	4
r-x	401	5
rw-	420	6
rwx	421	7

3.3 Displaying access rights

To find out the access rights of items (files, directories, etc.) in a given directory, simply use the command ***"ls -l"***, which displays a detailed list in the following form:



- The first character of the line indicates the type of file:
  - the dash (-) represents an ordinary file,
  - the letter ***d*** represents a directory,
  - other characters for special file types.
- The next nine characters indicate the access rights attributes for the corresponding file or directory (the first three for the owner, the next three for the group and the last three for the others).

3.4 Changing access rights (chmod)

The command ***chmod*** is used to change access rights to a file or directory. To be able to perform this operation, you must be the owner of a file or directory, or be a superuser. The command chmod is presented as follows:

chmod access_rights filename

where the argument ***access_rights*** indicates the access rights to be modified and ***filename***, the name of the file or directory to be modified.

Access rights can be indicated in different ways. For example, the following is one of the forms used:

- Use one or more letters to indicate the type of user to which access rights apply:
  - u (for user)
  - g (for group)
  - o (for other users)
  - a (for all categories u, g and o)
- Indicate whether access rights should be added (+), removed (-) or assigned (=).
- Use one or more letters to indicate access rights.
  - r (to read)
  - w (to write)
  - x (to execute)

Examples

In this example, write permission is added to the directory ***Rep1*** for users belonging to the same group using the command in bold.

```
$ ls -l  Rep1
drwxr-xr-x   3  user1        1024 Feb 10  11:15 Rep1
$ chmod  g+w  Rep1     or    $ chmod  775  Rep1
$ ls -l  Rep1
drwxrwxr-x   3  user1        1024 Feb 10  11:15 Rep1
```

To prevent users belonging to a group other than yours from being able to read and execute this directory, use the following command (the command in bold):

```
$ ls -l Rep1
drwxrwxr-x  3 user1        1024 Feb 10 11:15 Rep1
$ chmod o-rx Rep1   ou   $ chmod 770 Rep1
$ ls -l Rep1
drwxrwx---  3 user1        1024 Feb 10 11:15 Rep1
```

If we want to assign the same operation simultaneously for all three categories of users, we use option ***a***. For example, if we want the file toto.exe to be executable by all users, we need to enter the following command ((the command in bold) :

```
$ ls -l toto.exe
-rw-r--r--  3 user1        1024 Feb 10  11:31 toto.exe
$ chmod a+x toto.exe    or   $ chmod  751  toto.exe
$ ls -l toto.exe
-rwxr-xr-x  3 user1        1024 Feb 10  11:31 toto.exe
```

3.5 The commands chown and chgrp

It is possible to change the owner of a file or directory on Linux using the command chown or to change the owner group of a file or directory using the command chgrp.

The syntax of the command chown is as follows:

```
chown [OPTIONS] UTILISATEUR[:GROUPE] FICHIER(s)
```

For example, to change the owner of ***file1*** with user ***TOTO***, simply use the following command:

```
chown TOTO file1
```

We can also change the owner of a file or directory, by specifying both.

```
chown TOTO file1 Folder1
```

As it is possible to change the owner and group of a file (the user name and group are separated by the : character) using the following syntax:

```
chown USER:GROUP File1
```

To change the group of a file, use the separator : followed by the group name using the following syntax:

```
chown :GROUP File1
```

The above operation can be performed using the command ***chgrp*** with the following syntax:

```
chgrp GROUP File1
```

To change the owner of a directory recursively, use the -R option (all files and sub-directories in the tree will be modified):

```
chown -R USER:GROUP Folder
```

Exercises

Exercise n° 1

1) Create a directory called ***Rep1*** in your home directory.
2) Create two empty files ***"t1.txt"*** and ***"t2.txt"*** in the directory ***Rep1*** using a single line of commands.
3) Use the command ***echo*** to add the message "TP 2" in the file "***t1.txt"***.
4) Add the message "Redirection and pipe" in the file ***"t1.txt"*** without deleting the first message.
5) What happens if you use a greater than sign (>) in an output redirection operation?
6) What shell option can be used to avoid clearing a file in an output redirection operation? And how can this option be activated?
7) How can this option be overruled (ignored) even if it is activated?
8) How do to deactivate this option?
9) What is the standard file used for the stderr error stream?
10) Give an example of commands that can be used to redirect stdout to the "***file1.txt***" file and stderr to the standard file for errors?
11) Give an example of commands that can be used to redirect stdout and stderr to the same file?

Exercise n° 2

Continuation of exercise 1

1) List the contents of the directory ***Rep1*** with details of each item.
2) Modify the access rights for the file ***"t1.txt"*** by granting only the owner read and write permissions.
3) Remove all access permissions to the file ***"t1.txt"*** and try to display its contents afterwards. What do you notice?
4) Remove execution rights for the directory ***Rep1*** and try to access it. What do you notice?
5) Remove the read permission for the directory ***Rep1*** and try to list its contents. What do you notice?