

Unit 3.2. Basic GNU/Linux (Practising Shell)

Héctor D. Menéndez

Endless Science
endsci.net

Index

- 1 Execution and Process Control
- 2 Redirections
- 3 Permissions
- 4 Symbolic links
- 5 Exercises

Index

- 1 Execution and Process Control
- 2 Redirections
- 3 Permissions
- 4 Symbolic links
- 5 Exercises

Execution and Process Control

Linux is a multitasking OS. This means that it is possible to execute more than one command during the same period of time.

The shell considers two different execution modes: foreground and background.

By default, all commands are executed in foreground, but when the user indicates (adding '**&**' at the end of the command) that the command must be executed in background.

The background processes are running in parallel (except those processes which are synchronized).

Execution and Process Control

For example, compare:

```
$>sleep 5
```

```
$>sleep 5 &
```

```
$>who & ps -aux & ls -l
```

In this case, `who` and `ps` are running in background while `ls` is running in foreground.

Execution and Process Control

It is possible to chain several commands sequentially in foreground.

The following example shows how to chain the last 3 commands sequentially:

```
$>who ; ps -aux ; ls -l
```

It is usual to run process in “background” when they are going to take long to finish and the user wants to use other processes.

Execution and Process Control

It is possible to interrupt or terminate the execution of a process and also pass it to “background” from “foreground” or vice versa.

Command	Description
<CTRL>+C	It terminates a “foreground” process
<CTRL>+Z	It stops a “foreground” process
jobs	It lists the current jobs list. It also provides the job numbers “njob”
fg njob	It pass a job from background to foreground. The job is identified by “njob”
bg njob	It pass a job to background. The job is identified by “njob”
kill %njob	It terminates a job identified by (“njob”).
kill pid	It terminates a job identified by its process id (“pid”). the pid can be obtained using the ps command.

Execution and Process Control

The steps to pass a process from foreground to background are: stops the process (using `<Ctrl>+ Z`), and using the `njob` of the process, apply the `bg` command.

```
$>sleep 100
^Z
[1]+  Detenido          sleep 100
$>bg 1
[1]+  sleep 100 &
```


Index

- 1 Execution and Process Control
- 2 Redirections**
- 3 Permissions
- 4 Symbolic links
- 5 Exercises

Redirections

Through special characters you can control the flow of information that go to and from Linux commands.

Command	Description
>	Redirects the output of a command to a file.
>>	The same, but adding the text at the end of the file.
<	Specifies where to read the input for a command to execute.
<<	Allows you to provide data via the console until you type the character sequence indicated. Meanwhile you can enter text, a special prompt is displayed, usually preceded by ">".
	Redirects the output of a command to the input of another.

Redirection

```
$> ls -la > file.list  
$> ls -la ../*.txt >> file.list  
$> mail student < body.txt  
$> cat << stop  
$> ls -la * | more  
$> ls -la * | less
```

Redirections: Filters

Linux filters are commands that can be used as a flow manager (“stream”) of characters (from other command). These characters are processed and printed on the standard output (which can also be redirected to another command pipe).

Comando	Descripción
grep	Searches for strings in files or data streams.
sort	Sorts the words of a stream or file.
wc	Counts the number of bytes, words and lines in files or streams
uniq	Removes duplicate lines from files or streams sorted.
tr	Changes the specified characters in a file or stream, by others.

Redirections

```
$> ps -aux | grep student  
$> who | sort  
$> wc -l file  
$> uniq -c file  
$> cat file | tr a-z A-Z
```

Index

- 1 Execution and Process Control
- 2 Redirections
- 3 Permissions**
- 4 Symbolic links
- 5 Exercises

Permissions

To view the permissions of the elements of a system directory tree, you can use the command “`ls -l`”

```
-rwxr-xr-x  1 trabajo trabajo  13549 mar 30  2013 file
```

To change the permissions of a file the “`chmod`” command is used as follows:

```
chmod [{u, g, o}{+, -}{r, w, x}] file  
chmod opt file
```

Permissions

Permissions can be translated into numbers.

These values depend on the octet from 0 to 7 (000 a 111).

For example, if we have:

```
—rwxr—xr—
```

The binary value is 111/101/100 (1 means activated and 0 deactivated). The decimal equivalent number is 754. If you want to set the same permissions to another file, you would type:

```
chmod 754 file2
```


Index

- 1 Execution and Process Control
- 2 Redirections
- 3 Permissions
- 4 Symbolic links**
- 5 Exercises

Symbolic links

Allows access to an item (file or directory) of system directory tree from another location in the tree.

Linux allows users to set symbolic links between elements of the directory tree.

A symbolic link is a file that points to another file.

When you edit, run or read a symbolic link the system searches the element to which the link refers.

Syntax:

```
In -s fileName linkName
```

Index

- 1 Execution and Process Control
- 2 Redirections
- 3 Permissions
- 4 Symbolic links
- 5 Exercises**

Exercise 5: Execution and Process Control

- 1 Execute the following instructions and clearly explains what they do (the command “sleep” serves to make the system wait for a number of seconds which is the parameter, and the command “echo” displays a message on screen).

```
$>(sleep 5; echo -n uno; date) ; (sleep 2; echo -n dos; date)
$>(sleep 5; echo -n uno; date) & (sleep 2; echo -n dos; date)
```

- 2 Run the following in the command line:

```
$>sleep 30
```

Stop the instruction and put it in “background”, run `ls`, and relocate it in “foreground”. Wait until the instruction is complete, Where do we get the “# job” id used by “fg” and “bg”? How do we discover that a “background” process has ended?

Exercise 5: Execution and Process Control

- 1 Run the following in the command line:

```
$>sleep 50
```

- 2 stop the instruction, put it in “ background” and terminate it using “kill” pid. Is there a faster way to end the execution of a process in “foreground”?

```
$>sleep 30
```

Exercise 6: Redirections

- 1 Changes the current directory to the directory Months created earlier.
- 2 Create in a single instruction (with `>` and `cal`) and a file containing the calendar of June 2013, called `summer2013`.
- 3 Use `>>` to add to the file `summer2013`, July 2013 and finally add August 2013.
- 4 Check with `cat` the file content.
- 5 Change the current directory to the Months directory created earlier.

Exercise 6: Redirections

- 1 Create, in a single instruction, the file `spring2013` containing the calendars of March, April and May 2013. Look at this statement as an example:

```
$>(date; echo hello) > message.txt.
```

- 2 Check with `cat` the file content.
- 3 Use “`cat`” and `>` to create two files called `one.txt` y `two.txt`. To create them, use `<CTRL> + <D>` to choose the end of the files. For example:

```
$> cat > archivo  
This is Line 1  
This is line 2  
<CTRL>+<D>
```

- 4 Display the files on the screen and concatenate the first file at the end of the second. Display the resulting file.

Exercise 7: Redirections

- 1** Run and check what exactly the following groups of commands do, step by step:

```
a) $>touch a.dat ; touch b.dat; touch c.dat ; ls | grep dat | wc -l
b) $>echo 'abcde abcde 12345 12345' | tr a-z A-Z > file1.txt
c) $>echo '12343' >> file2.txt
d) $>echo '234234' >> file2.txt
e) $>echo '50923' >> file2.txt
f) $>echo '231234' >> file2.txt
g) $>cat file2.txt | sort -d
h) $>cat file2.txt | sort -n
i) $>cat file2.txt | sort -nr
```


Exercise 7: Redirections

- 1 Run and check what exactly the following groups of commands do step by step:

```
$>cat << parate > file3.txt
> asdasdasdasd
> xcvxcvxcvxcv
> qweqweqweqwe
> asd asd qwd asd
> 123' 09 098239
> zcvzx zxcv zxcv zxcv
> 123OIU OPIUPOIU POIU
> parate
```

```
$>cat file3.txt | tr -cs '[a-zA-Z0-9]' '[\n*]'
```

- 2 Use `grep` and `wc` with `books.txt` file to count the number of audio books.
- 3 Use `grep` and `wc` with `books.txt` file to count the number of books containing the work `War`.

Exercise 8: Backups

- 1 Created a directory called `backup` in your user directory.
- 2 Copy the following files to the `backup` directory:

```
/etc/issue  
/etc/hostname  
/etc/fstab  
/etc/hosts  
/etc/group  
/etc/passwd
```

- 3 Create a packed file containing the files of the `backup` folder uncompressed, called it `backup.tar`.
- 4 Check the content of the file `backup.tar`.
- 5 Delete the original files from `backup`. Finally, extract the files from `backup.tar` in the `backup` directory.

Exercise 8: Backups

- 1 Create a file with `cat`. Compress it and see if its size is smaller.
- 2 Unzip the file to get the original.
- 3 Create three files with `cat`. Create a file `.tar` uncompressed from these three files and compresses the file obtained using `gzip`.
- 4 Perform the inverse operation to obtain the three files.
- 5 Run the previous section on a single command line using the `z` option with `tar`.

Exercise 9: Permissions

- 1** Create a file called `a.dat` and deny permission to the owner. What happens when you try to delete it with `rm`?
- 2** Create a new directory named `other`. Deny the read permission to the owner and then list its contents with `ls`. What happens? Allow the read permission for the owner and try again, list the contents with `ls`.
- 3** Deny the permission to the owner in the `other` directory. Try to copy any file into the `other` directory. What happens?.

Exercise 10: Symbolic links

- 1 Create two files inside of a directory and two symbolic links to them from another directory.
- 2 What happens to the original file if the corresponding file link is deleted?.
- 3 What about the linked file if you delete the file it points to?.
- 4 How do you know, using `ls`, whether a file is a symbolic link and where does it point to?