Coursework 1: Basic GNU/Linux (Practising Shell) Part 2

Héctor Menéndez[1]

AIDA Research Group
Computer Science Department
Universidad Autónoma de Madrid

October 3, 2013

[1] based on the original slides of the subject

Index

Index

Execution and Process Control

- Linux is a multitasking OS. This means that it is possible to execute more than command during the same period of time.
- Shell considers two different execution modes: foreground and background.
- By default, all commands are execute in foreground, though 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 fininsh and the user wants to use other process.

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

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 data input of a command to execute.
<<	Allows you to provide data via the console until you type the character sequence indicated. While you can enter text, a special prompt is displayed, usually as ">".
|	Redirect output of one command as input to another.

Redirection

```
$> ls -la > lista.archivos
$> ls -la ../*.txt >> lista.archivos
$> mail alumno < texto.txt
$> cat << parate
$> 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 system).

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 alumno
$> who | sort
$> wc -l archivo
$> uniq -c archivo
$> cat archivo | tr a-z A-Z
```

Index

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

Symbolic links

- Allows having 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:

```
ln -s fileName linkName
```

Index

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 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 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`) 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 March, April and May 2013. Look at this statement as an example:

```
$>(date; echo hola) > mensaje.txt.
```

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

```
$> cat > archivo
ESTA ES LA LINEA 1
ESTA ES LA LINEA 2
<CTRL>+<D>
```

4. Displays the files on the screen and concatenates the first file at the end of the second. Displays 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 > archivo1.txt`
c)
```
$>echo '12343' >> fich2.txt
$>echo '234234' >> fich2.txt
$>echo '50923' >> fich2.txt
$>echo '231234' >> fich2.txt
$>cat fich2.txt | sort -d
$>cat fich2.txt | sort -n
$>cat fich2.txt | sort -nr
```

Exercise 7: Redirections

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

```
$>cat << parate > fich3.txt
> asdasdasdasd
> xcvxcvxcvxcv
> qweqweqweqwe
> asd asd qwd asd
> 123' 09 098239
> zcvzx zxcv zxcv zxcv
> 123OIU OPIUPOIU POIU
> parate
$>cat fich3.txt | tr -cs '[a-zA-Z0-9]' '[\n*]'
```

2. Use grep and wc with libros.txt file to count the number of books that belong to the category Ensayo.
3. Use grep and wc with libros.txt file to count the number of books that belong to the writer Arraval.

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 file containing the files of `backup` folder uncompressed, called it backup.tar.
4. Check the contents 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 and see if its size is smaller. Record the sizes in your report.
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. Performs 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 of the owner in the `other` directory. Try to copy any file in the `other` directory. What happens?.

Exercise 10: Symbolic links

1. Create two files in a directory and two symbolic links to them in another directory.
2. What happens to the original file if the corresponding file link is deleted?.
3. What about the link file if you delete the file it points to?.
4. How do you know using `ls` that a file is a symbolic link and where it points?