www.uib.cat

Introduction to Bash

Centre Balear de Biodiversitat

Tommaso Cancellario
Laura Triginer

Universitat de les Illes Balears

Julio 2023

What is the terminal? What is the shell?

The computer terminal is the set of devices that allow the user to interact with the system. They usually consist of screen, keyboard and computer.

Terminal: Device used to send commands to a computer and show its answer.

SHELL: interface or program that interprets the command lines sent to the terminal.

bash: shell programming language of UNIX (macOS and Linux)

A terminal is an environment where texts enter and exit. Shell is the one who interprets the data and commands sent by the terminal

Why is important to learn bash?

- Is important for anyone working with data.
- Is far more efficient and powerful than using a GUI (Graphical User Interface)
- Bash correlates heavily with other data science technologies such Python
- Commands are easily automated and replicated.
- Makes you more flexible.
- The interaction between programs is easier to adjust to command line.
- Helps you to handle text files quickly and efficiently.

How to use bash in a OS...

Windows

https://itsfoss.com/install-bash-on-windows/

macOS

Open a terminal and type ...
sudo port install bash
Do you need to pre-install macports?

Linux

Open a terminal and type ...

Sudo apt update
Sudo apt install bash-completion

Use BASH in WINDOWS

→ Tecla de Windows + I para abrir el menú de Configuración. Luego seleccionar Actualización y seguridad y posteriormente la opción Para programadores.
→ download Windows Subsystem for Linux.
→ Configuración → Actualización y seguridad → Actualización (Windows Subsystem for linux)
→ Configuración>Actualización y seguridad>Activación (Hyper-V).
Si no está hyper-v, se accede a BIOS) mayúscula al apagar, F2 en el inicio
(revisar acceso a BIOS según modelo).

→ download Ubuntu 22.04, ABRIR UBUNTU "INSTALLING", open as administrator! Nombre solo minuscula!

Restart computer

En la terminal, escribir ubuntu

CD / -- cd c cd Users cd aless cd Desktop

Let’s start!

In bash we use another kind of logical, different than in other programming languages.

$ command -flags

Flags are options that can be activated when using them, they can be also default values

Most common flags

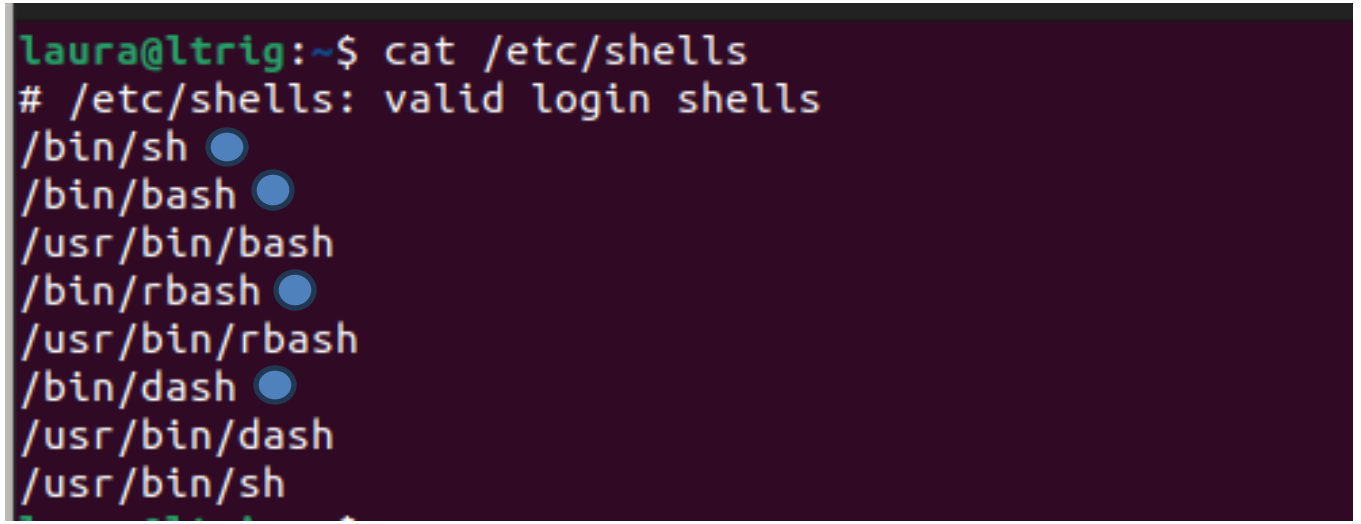
-h, --h or -help → will give yoo details of how to use the tool

-o file → is usually «output» or output directory

-i file → is usually an input directory

Let’s check our avaliable shell...

cat /etc/shells



If we would like to check which shell are we using, we can do «echo $SHELL». Eventhough the commands used will be the same.

Commands

pwd -> Print Working Directory
cd -> Change directory
mkdir -> make directory
touch -> create a new file (we can write the extension .sh to make easier for other people to know about the file)
cat -> view the contents of a file
ls -> lists the contents of a directory.
mv -> move $mv file1 file2
rm -> remove *$rm file* *$rm –r folder*

Exercises from the terminal

- Check in which directory are you.
- What do you have in that directory?
- Check the space of what is placed in the directory.
- Move between directories
- Create a directory called "Test_1"
- Access to the directory and create another one inside named "Test_2"
- Remove the directory Test_2.
- We will work in Test_1 directory

Creating an executable

Remember to ALWAYS write:

#! /bin/bash

In the first line of a shell script

1. Create a text file named helloworld.sh

   $ *nano helloworld.sh*

2. Write the program to have an echo saying «hello world»

3. $ *chmod +x helloworld.sh* to make the file executable.

4. Execute ./helloworld.sh

Creating more executables

1. Create a new .sh file
2. Exercices programmed:
    1. Send «hello» to a text file
    2. Input from the terminal sent to a text file
    3. Input from the terminal sent to a text file created in 2.
    4. See a text in the terminal.
    5. Concatenate text files
3. chmod +x *nameofthefile.sh* to make the file executable.
4. Execute ./*nameofthefile.sh*

Conditionals

```
If [ statement ]
then
        command
s
elif [statement]
        command
s
else
        command
s
fi
```

-eq =	Equal to
-ne !=	Not equal to
-gt >	Greater than
-ge >=	Greater or equal
-lt <	Less than
-le <=	Less or equal

Conditional exercices

1. Create a new .sh file
2. Exercices programmed:
    1. Set an age, print «el número es igual» en el caso que age sea igual a 10.
    2. Set an age and write a program that defines if the age is less or bigger than 10.
    3. Set an age, print 3 different options depending on it, using if, elif and else.
3. chmod +x *nameofthefile.sh* to make the file executable.
4. Execute ./*nameofthefile.sh*

Loops

While:

```
While [ condition ]
do
        command
s
        ...
done
```

For:

```
for [ condition ]
do
        command
s
        ...
done
```

Until:

```
until [ condition ]
do
        command
s
        ...
done
```

Exercice with penguins.csv

Download penguins.csv and place it in the working directory.

1. Check 10 first and 10 last lines of the csv typing:

$ *head -10 penguins.csv && tail -10 penguins.csv*

Check the difference between using & and &&.

2. Try to check only the first column values typing:

$ *cut -f1 penguins.csv*

What happens?

Exercice with penguins.csv

3. Let's separate the csv comma separated file to a file with as many columns as we have

    1. count how many ',' are there in the first row.

        $ *head -1 penguins.csv | grep -o ',' | wc -w*

    2. How many columns will we need?

    3. Let's split the different values into columns using:

        $ *awk -F ',' '{print $1 "\t" $2"\t" $3"\t" $4"\t" $5"\t" $6"\t" $7"\t" $8}' penguins.csv > output.txt*

    4. How can we do this in a more general way?

```
awk -F ',' '{
    for (i = 1; i <= NF; i++) {
        printf "%s\t", $i
    }
    printf "\n"
}' penguins.csv > spl_penguins.txt
```

awk → permite procesar y modificar el texto
-F →indica el delimitador
for (i=1; i<=NF; i++) → en todas las líneas.

printf "%s\t", $i → imprimir con un formato específico, en la misma línia con un tabulador entre valores

printf «\n» la siguiente línea en una nueva.

Now we have the different values in different columns...

1. Count how many different penguins species do we have.

$ *tail -n +2 spl_penguins.txt | cut -f1 | sort | uniq | wc -w*

2. Count how many of each specie do we have in our database.

$ *tail -n +2 spl_penguins.txt | cut -f1 | sort | uniq -c*

3. Count how many males and females do we have in our database.

$*tail -n +2 spl_penguins.txt | cut -f7 | sort | uniq -c*

WE CAN SEE THAT THERE ARE MANY NA numbers, let's clean a little bit our dataset so we can have all columns with values.

$ *awk '!/NA/' spl_penguins.txt > clean_penguins.txt*

There is another way to count how many times there's a pattern in a string.

$ grep -c 'Adelie' spl_penguins.txt

It might work in this case, however if there''s a word containing the other one, as male and female, it might not work.

$ grep -c "male" spl_penguins.txt

$ tail -n +2 spl_penguins.txt | cut -f7 | sort | uniq -c

Thank you very much for your attention!

Do not forget to fill out the survey