www.uib.cat

Introduction to Bash

Centre Balear de Biodiversitat

Tommaso Cancellario
Karen Schöninger
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 such as screen, keyboard and computer that allow the user to interact with the system.

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

Shell: An 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 and R
- 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 files quickly and efficiently.

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 you details of how to use the tool

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

-i file → is usually an input directory

Let's play with our directory

1. Check your current directory using pwd.
   -> Print Working Directory

2. Do you want to work from another directory? Use cd
   -> Change directory (Test cd .. cd ../.., cd /)

3. Create a new directory to have everything organized using mkdir.

4. Create a new file using touch. Test it

Let's play with our directory

1. Check what's inside of a directory using "ls"
2. cat, head, tail -> view the contents of a file
3. Move a file between directories using mv
   -> $mv file1 file2
4. Remove a file using rm
   -> $*rm file* *$rm –r folder*

More 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.
- Create two different directories, TestA, TestB.
- Create a file inside TestA called fileA.txt.
- Move fileA.txt to the folder TestB.
- Access to TestB, check that fileA.txt is inside and inside named "Test_2"
- We will work in Test_1 directory

Some nice commands to start working

grep

Powerful text search tool. Primarily used to search for a specific pattern or string of characters within files or streams of text. It helps locate lines in files that match a given pattern.

Usage: grep [options] file

grep '>' file.fasta
grep –c '>' file.fasta
grep –n '>' file.fasta
grep –v '>' file.fasta

Some nice commands to start working

sed

Powerful text processing utility in Unix and Unix-like operating systems. It is used to perform text transformations on an input stream (a file or input from a pipeline). It doesn't alter the original file.

Usage: sed –flags ....

sed 's/old/new/' file.txt --> replaces the first occurrence
sed 's/old/new/g' file.txt --> replaces the occurrence in all file

Some nice commands to start working

sort

Used to sort lines of text files or input streams. It is a powerful and flexible tool that can be used for various sorting tasks. Here's a basic explanation of how to use the sort command

Usage: sort [options] [file]

- sort file
- sort –r file
- sort –u file

Some nice commands to start working

uniq

command is used to filter out repeated lines in a sorted text file or input stream. It is often used in combination with the sort command.

Usage: uniq [options] [file]

- uniq –c file
- uniq –d file
- uniq –u file

Some nice commands to start working

awk

The awk command is a versatile text processing tool in Unix and Unix-like operating systems. It operates on a per-line basis and is used for pattern scanning and processing.

Usage: awk '/pattern/ { action }' file.txt

- awk '/pattern/' file.txt
- awk '{ print $1 }' file.txt
- awk '{ total += $1 } END { print "Total:", total }' file.txt

Examples of use, penguins.csv

- Count how many different species are there.
- How many individuals of each species are there in the file.
  - How many females and males are there?
  - In which years was the experiment conducted?

More exercises with penguins

- Clean a little bit our dataset, so we can have all columns with values and no "NA"

$ *awk '!/NA/' penguins.csv > clean_penguins.txt*

- Count how many "Adelie" individuals do we have.

$ grep -c 'Adelie' clean_penguins.txt

- Test how many males and females do we have in this dataset.

$ grep -c "male" clean_penguins.txt

More exercices with penguins.csv

1. Check 10 first lines of the csv.

2. Check the last lines of the csv.

3. How many lines are there?

4. How many columns are there?

5. Display the three first columns and create another file with them.

6. Display the lines where the bill_length is bigger than 39.1 mm.

7. Calculate the sum of the body mass.

More exercises with penguins

- *As* there's a word containing the other one, it might not work properly, so we will have to think another way to do it.

```
$ tail -n +2 penguins.csv | cut -d ' ' -f 7 | sort | uniq -c
```

Create a script that does the different commands and creates a new file for the output of each of them.

Steps:

1. Remove columns with NA
2. Count how many individuals of each specie there are.
3. Add to the output of 2. the number of males and females.

Examples of use, fasta.fas

- Count how many sequences are there in a fasta file.
- Copy the headers of the sequences to another file called sequences.txt
- Change the word 'size' in the sequences.txt to 'length'

Creating a script

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

```
#! /bin/bash

echo "hello world"
```

Creating more scripts

1. Create a script file called example.sh

2. Send «hello» to a text file using cat

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

4. Execute ./*nameofthefile.sh*

```
#! /bin/bash


echo 'hello' > cat text1.txt
```

Creating more scripts

1. Create a new .sh file named example2.sh

2. Input from the terminal sent to a text file

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

4. Execute ./*nameofthefile.sh*

```
#! /bin/bash

cat > text2.txt
```

Creating more scripts

1. Create a new .sh file

2. Input from the terminal sent to a text file created in 2

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

4. Execute ./*nameofthefile.sh*

```
#! /bin/bash

# >> will concatenate with what's already written in the text file.

cat >> test3.txt
```

Conditionals

```
If [ statement ]
then
        commands
elif [statement]
        commands
else
        commands
fi
```

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

Exercise with conditionals and input from terminal

Example 1. Write a script that asks your name and then the output it's «this is your name: »

Example 2. Compare two passwords and check if they are the same. (string_compare.sh)

Example 3. Create a script that when you introduce a value, it compares to 5 and if it's 5, it says it's correct. (conditionals1.sh)

Example 4. Create a script that tells you if you're an adult, a child or a teenager. Conditionals2.sh

Example 5. Create an script that you introduce a mark and it says your grades. Operators.sh

Loops

While:

```
While [ condition ]
do
        commands
        ...
done
```

For:

```
for [ condition ]
do
        commands
        ...
done
```

Exercise with loops

Example 1. Create a script that once you introduce a number between 0 and 10, prints all numbers and says the number introduced except of the one introduced.

Example 2. Use the previous script, but now it will print all numbers except the one introduced.

Example 3. Print the values between 0 and 20, two by two.

Example 4. Print all the values smaller than 10 and a given number.

Exercise with loops

Example 1. Create a script that once you introduce a number between 0 and 10, prints all numbers and says the number introduced except of the one introduced. for_continue.sh

Example 2. Use the previous script, but now it will print all numbers except the one introduced. for_continue.sh

Example 3. Print the values between 0 and 20, two by two. (loop_for.sh)

Example 4. Print all the values smaller than 10 and a given number.(loops.sh)

Thank you very much for your attention!

Do not forget to fill out the survey