Techniques for Big Data

First session

Yérali Gandica

Contemporary Issue Module. International Bachelor. CY Cergy Paris Université

19-23 October 2020

Program

- Bash-shell Unix commands
  - Basic bash-commands
  - grep
  - awk
  - sed
- Python
- Python-pandas

Basic bash-commands

- pwd — Print working directory.
- cd — Change directory.
- ls — List directory contents (ls -la gives time and size).
- mkdir — Create a directory.
- mv — Move or rename directory.
- rm — remove a file.
- rm -r — remove a directory.
- echo — Prints text to the terminal window.
- touch — Creates a file.

Be careful: Things that have been deleted will not go to the computer's trash

Once you have a BIG file in your directory

- wc File — number of lines on File.
- head File — Read the start of File (First 10 lines).
- tail File — Read the end of File (10 lines).
- less File — view the contents of File.

Once you have a BIG file in your directory

- wc File — number of lines on File.
- head File — Read the start of File (First 10 lines).
- tail File — Read the end of File (10 lines).
- less File — view the contents of File.
  - The less command allows you to view files without opening an editor

Once you have a BIG file in your directory

- wc File — number of lines on File.
- head File — Read the start of File (First 10 lines).
- tail File — Read the end of File (10 lines).
- less File — view the contents of File.
  - The less command allows you to view files without opening an editor
  - It's faster to use, and there's no chance of you inadvertently modifying the file

Once you have a BIG file in your directory

- wc File — number of lines on File.
- head File — Read the start of File (First 10 lines).
- tail File — Read the end of File (10 lines).
- less File — view the contents of File.
  - The less command allows you to view files without opening an editor
  - It's faster to use, and there's no chance of you inadvertently modifying the file
  - It starts with the visible windows of your console, but you can scroll down to explore your data

You can install Git BASH (https://gitforwindows.org/) if your OS is Windows

Bash-shell Unix commands

- **Bash** is a command language interpreter. It is widely available on various operating systems and is a default command interpreter on most GNU/Linux systems. The name is an acronym for the ‘Bourne-Again SHell’.

Bash-shell Unix commands

- **Bash** is a command language interpreter. It is widely available on various operating systems and is a default command interpreter on most GNU/Linux systems. The name is an acronym for the 'Bourne-Again SHell'.

- **Shell** is a macro processor which allows for an interactive or non-interactive command execution.

Bash-shell Unix commands

- **Bash** is a command language interpreter. It is widely available on various operating systems and is a default command interpreter on most GNU/Linux systems. The name is an acronym for the 'Bourne-Again SHell'.
- **Shell** is a macro processor which allows for an interactive or non-interactive command execution.
- **Scripting** allows for an automatic commands execution that would otherwise be executed interactively one-by-one.

You will need an editor

- Emacs runs in any OS
- You can use gedit in Linux
- You can use Notepad in Mac
- You can use vscode (https://code.visualstudio.com/) in Windows and Mac

My first Bash script

```
#!/bin/bash -e
# declare STRING variable
STRING="Hello World"
#print variable on a screen
echo $STRING
```

Navigate to a directory where your `hello_world.sh` is located and make the file executable:

```
$ chmod +x hello_world.sh
```

Now you are ready to execute your first bash script:

```
./hello_world.sh
```

Loops

```
#!/bin/bash -e
      for i in ‘seq 1 10‘;
      do
              echo $i
      done
```

Arrays

```
#!/bin/bash
#Declare array with 3 elements
ARRAY=( 'Debian Linux' 'Redhat Linux' 'Ubuntu Linux' )
# get number of elements in the array
ELEMENTS=${#ARRAY[@]}

# echo each element in array
# for loop
for (( i=0;i<$ELEMENTS;i++)); do
    echo ${ARRAY[${i}]}
done
```

More loops

```
#!/bin/bash
QUEY=( first second third )
for Y in "${QUEY[@]}"
do
  :    # for one or more instructions
  echo $Y
done
```

Loops with while: how can you see the output of this script?

```
 #!/bin/bash
  x=1
while [ $x -le 4 ]
do
   file="output$x.dat"
   cp output.dat $file
   x=$(( $x + 1 ))
done
```

Increasing variables

```
#!/bin/bash
COUNTER=0
        while [  $COUNTER -lt 10 ]; do
            echo El COUNTER es $COUNTER
            let COUNTER=COUNTER+1
        done
```

Loop over your files

```
#!/bin/bash

# bash for loop
for f in $( ls /home/yerali ); do
echo $f
done
```

Bash also has

- if / else / fi statements
- Arithmetic Comparisons
- String Comparisons
- Functions
- Case statement conditional
- and more....

There are plenty of Bash -scripting- tutorials in internet, for example:

`https://linuxconfig.org/bash-scripting-tutorial`

Bash quotes. What is the input and output of this program?

```
CASES=( firms1 firms2 )

for WHATDATA in "${CASES[@]}"
do
  :
  if [ $WHATDATA = firms1 ]
      then YINI=125
           YFIN=503
      fi
  if [ $WHATDATA = firms2 ]
      then YINI=252
           YFIN=503
      fi
  cp "${QUEDATA}_""$XINI"_"$XFIN".txt new_files.txt
  export YINI YFIN
done
```

Bash quotes. What is the role of export?

```
CASES=( firms1 firms2 )

for WHATDATA in "${CASOS[@]}"
do
  :
  if [ $WHATDATA = firms1 ]
      then YINI=125
           YFIN=503
      fi
  if [ $WHATDATA = firms2 ]
      then YINI=252
           YFIN=503
      fi
  cp "${QUEDATA}_""$XINI"_"$XFIN".txt new_files.txt
  export YINI YFIN
done
```

Exercises

1. Print the elements in a directory of your computer without a loop and with a loop.
2. For the following i words, create a file with this name and add i lines to the file: elephant, dog, cat, fish.

grep, awk and sed

They are command-line utilities, each one with its own syntax, which help users to achieve further utilities when using bash-scripting.

My explanation will be based on

`https://www-users.york.ac.uk/~mijp1/teaching/2nd_year_Comp_Lab/guides/grep_awk_sed.pdf`

by Matt Probert from the University of York

grep = global regular expression print

- grep will search input files for a search string, and print the lines that match it.

grep = global regular expression print

- grep will search input files for a search string, and print the lines that match it.
- Beginning at the first line in the file, grep copies a line into abuffer, compares it against the search string, and if the comparison passes, prints the line to the screen.

grep = global regular expression print

- grep will search input files for a search string, and print the lines that match it.
- Beginning at the first line in the file, grep copies a line into abuffer, compares it against the search string, and if the comparison passes, prints the line to the screen.
- Grep will repeat this process until the file runs out of lines.

grep = global regular expression print

- grep will search input files for a search string, and print the lines that match it.
- Beginning at the first line in the file, grep copies a line into abuffer, compares it against the search string, and if the comparison passes, prints the line to the screen.
- Grep will repeat this process until the file runs out of lines.
- Notice that nowhere in this process does grep store lines, change lines, or search only a part of a line.

grep = global regular expression print

```
grep "boo" a_file

boot
book
booze
boots

grep -n "boo" a_file #giving line number
1:boot
2:book
3:booze
5:boots
```

grep = global regular expression print II

```
grep -vn "boo" a_file   #all of the lines not matching

4:machine
6:bungie
7:bark
8:aaradvark
9:robots

grep -c "boo" a_file #only display the number of lines
4                    #that match the query
```

grep = global regular expression print III

```
grep -l "boo" *  #searching through multiple files
grep -i "BOO" a_file  # -i ignores case
grep -x "boo" a_file # -x looks for eXact matches only
grep '\^^^_20' Special characters with '\ and finishing '
grep "e$" a_file # ending with the string
grep '\<WORD' file.txt # Starting with WORD
```

grep: Modifying files

- `grep WORD file.txt > file.dat`

  this just copies the lines containing the word WORD

grep: Modifying files

- `grep WORD file.txt > file.dat`

  this just copies the lines containing the word WORD

- `grep -i -Ev 'Boo' File.dat > cleaned.dat`

  It passes only lines without boots in any combination of upper or lower case.

grep: Modifying files

- `grep WORD file.txt > file.dat`

  this just copies the lines containing the word WORD

- `grep -i -Ev 'Boo' File.dat > cleaned.dat`

  It passes only lines without boots in any combination of upper or lower case.

- `grep -e word1 -e word2 -e word3 FILE.txt> NEW.txt`

  Looking for more than one word

Exercises

1. Search for (i) your name in every directory on your computer, (ii) for one year.

2. Copy in a new file removing the bots in any combination of case, from the WP-raw file (The first column is the wikipedia page, the second is the editors, the third is the time in Linux time).

```
https://www.dropbox.com/s/0qisllsrt2cr55x/raw.edit-huwiki?dl=0

https://www.dropbox.com/s/bg0suewysq03xw7/raw.edit-arwiki?dl=0

https://www.dropbox.com/s/5m4za5p8b5mpr9v/raw.edit-jawiki?dl=0
```