Shell tutorial

Amandine Decker

M1 TAL/NLP 2022–2023

Université de Lorraine, LORIA

An introductory question

How would you find the number of occurrences of a word in *one* document?

An introductory question

How would you find the number of occurrences of a word in *one* document? And in *one hundred*?

Shell/Bash

- Shell : command-line interface = makes the link between the user and the operating system ;
- Bash : implementation of shell
- (There exist many shells : *sh*, *zsh*, *bash*, ...)

→ Makes simple task (*e.g.*, simple NLP tasks) very easy.

Shell/Bash

During this session we will :

- Discover basic shell commands;
- Practice : Get the number of occurrences of each word in a document.

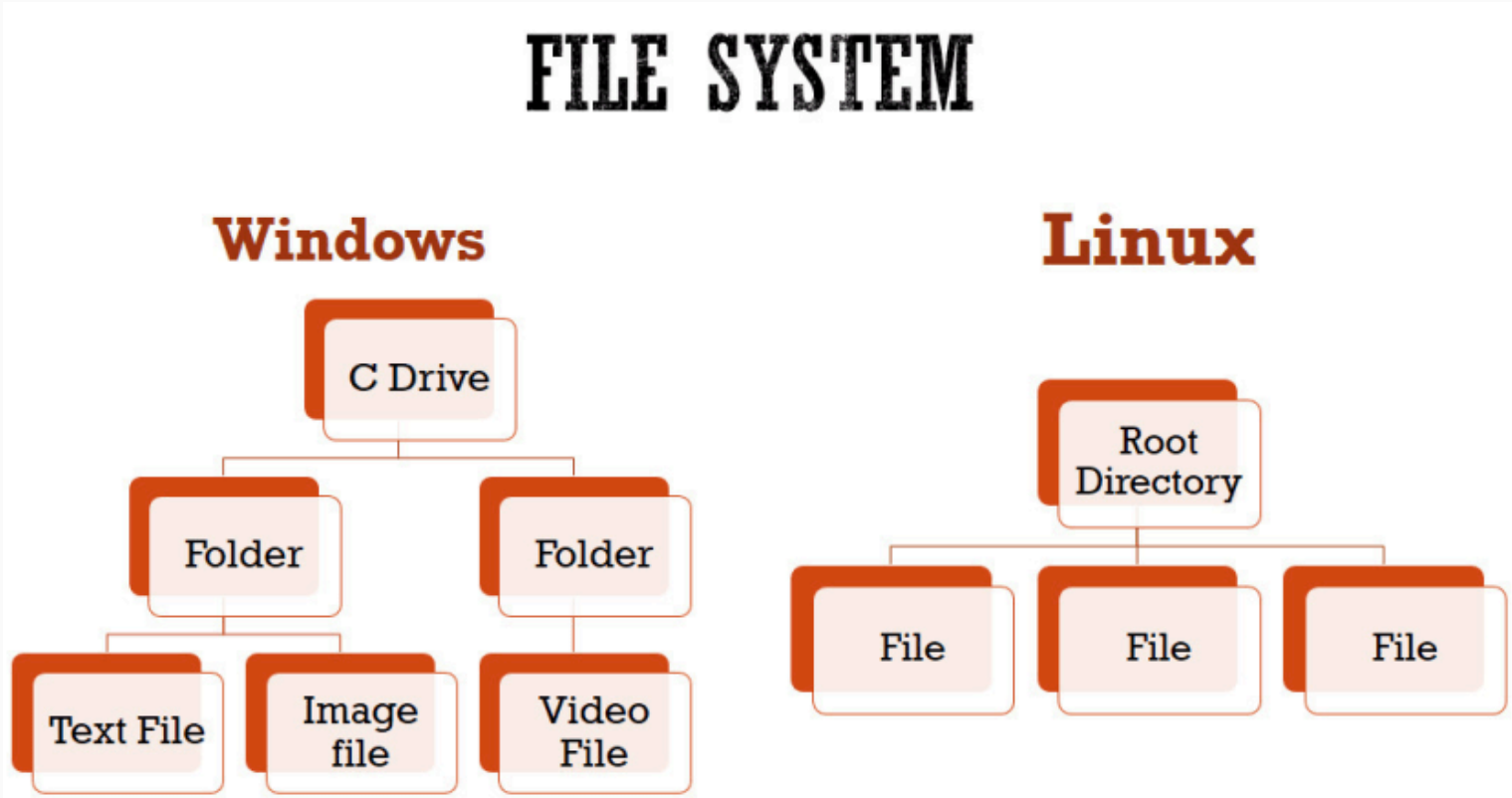
Open up the terminal

- Mac OS : “terminal” application (*Cmd+Space* + type “terminal”);
- Most Linux-based desktop OS have a dedicated shortcut, *e.g.*, Ubuntu : *Ctrl+Alt+T*;
- Windows : activate **Windows Subsystem for Linux** (WSL) and install a Linux system.
  - En : *https://itsfoss.com/install-bash-on-windows/*
  - Fr : *https://blog.ineat-group.com/2020/02/utiliser-le-terminal-bash-natif-dans-windows-10/*

Path

`.`, `~`, `pwd`, `ls`, `cd`, `tree`

Your filesystem is organised as a tree : *i.e.*, a *root*, where you find folders/directories, themselves containing folders/directories, etc. (You store files in folders/directories)



Path

`.`, `~`, `pwd`, `ls`, `cd`, `tree`

When using bash, you are always “somewhere”, *i.e.*, at a node of the filesystem. The path is the *address* of this node, it is a list of files, separated by forward slashes (/), indicating the way from the root (absolute path), or from your current node (relative path), to a certain directory.

/Users/Amandine/desktop

Path

`.`, `~`, `pwd`, `ls`, `cd`, `tree`

`.`	represents your working directory
`~`	represents your home directory
`pwd`	**p**rint **w**orking **d**irectory
`ls`	**l**i**s**t the content of a directory
`cd`	**c**hange **d**irectory
`tree`	print the **tree** structure of a directory

Path

`.`, `~`, `pwd`, `ls`, `cd`, `tree`

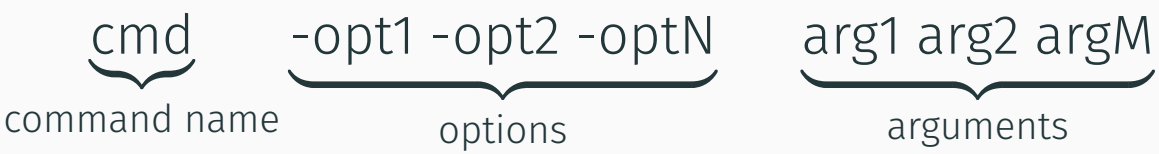
`.`	represents your working directory
`~`	represents your home directory

`pwd`	**p**rint **w**orking **d**irectory
`ls`	**l**i**s**t the content of a directory
`cd`	**c**hange **d**irectory
`tree`	print the **tree** structure of a directory

→ Where are you ?

→ Move to the directory you want to work in.

Syntax of a Command



Listing the Commands + Documentation

`compgen, man`

`compgen -c`	print all the available commands
`man cmd`	print the documentation for a command

(quit the manual by pressing *q*)

Files and Folders

`mkdir, touch, mv, cp, rm`

`mkdir DIR`	create (**m**a**k**e) a **dir**ectory DIR
`touch FILE`	create a file FILE
`mv SOURCE DEST`	**m**o**v**e a file SOURCE to another location DEST
`cp SOURCE DEST`	**c**o**p**y a file SOURCE to another location DEST
`rm FILE_1 ... FILE_N`	r**m** file(s) (or directories)

Files and Folders

`mkdir, touch, mv, cp, rm`

`mkdir DIR`	create (**m**a**k**e) a **dir**ectory DIR
`touch FILE`	create a file FILE
`mv SOURCE DEST`	**m**o**v**e a file SOURCE to another location DEST
`cp SOURCE DEST`	**c**o**p**y a file SOURCE to another location DEST
`rm FILE_1 ... FILE_N`	**r**emo**v**e file(s) (or directories)

→ Create a new directory for the practice and move the file *`book.txt`* to this new directory.

Print and Redirect Outputs

`cat, head, tail, >, >>, <, |`

`cat FILE_1 ... FILE_N`	con**cat**enate and print contents of files
`head FILE`	show beginning of FILE
`tail FILE`	show end of FILE

- `>` redirect the output of a command to a file
- `>>` append the output of a command to a file
- `<` use the content of a file as input for a command
- `|` redirect the output of a command to another command

Print and Redirect Outputs

`cat, head, tail, >, >>, <, |`

`cat FILE_1 ... FILE_N`	con**cat**enate and print contents of files
`head FILE`	show beginning of FILE
`tail FILE`	show end of FILE

- `>` redirect the output of a command to a file
- `>>` append the output of a command to a file
- `<` use the content of a file as input for a command
- `|` redirect the output of a command to another command

→ Have a look at the content of *`book.txt`*.

Manipulate Files

`grep, wc, sort, uniq`

`grep PATTERN FILE`	**g**lobal **r**egular **e**xpression **p**rint : print matches of a regular expression PATTERN found in FILE ($\geq$ 1 file)
`sed PATTERN FILE`	**g**lobal **r**egular **e**xpression **p**rint : print matches of a regular expression PATTERN found in FILE ($\geq$ 1 file)
`wc FILE`	print the number of characters, number of words (whitespace-delimited) and number of lines in the file
`sort FILE`	**sort** file FILE
`uniq FILE`	keep only **uniq**ue (distinct) adjacent lines in FILE

Manipulate Files

`grep, wc, sort, uniq`

1. Transform the text in *`book.txt`* in a list of words (you can store it in a new file);
2. Sort the list by alphabetical order (you can also store it in a new file to not overwrite the previous one);
3. Get the number of occurrences of each word and sort the result (you can save the result in a file).

Manipulate Files

`grep, wc, sort, uniq`

1. Transform the text in *`book.txt`* in a list of words (you can store it in a new file);
   → Replace the spaces by a new line (`\n`);

2. Sort the list by alphabetical order (you can also store it in a new file to not overwrite the previous one);
3. Get the number of occurrences of each word and sort the result (you can save the result in a file).

Manipulate Files

`grep, wc, sort, uniq`

1. Transform the text in *`book.txt`* in a list of words (you can store it in a new file);
   - → Replace the spaces by a new line (`\n`);
   - → *`sed 's/foo/bar/g' input_file > output_file`* replaces *`foo`* by *`bar`* in *`input_file`* and stores the result in *`output_file`*.
2. Sort the list by alphabetical order (you can also store it in a new file to not overwrite the previous one);
3. Get the number of occurrences of each word and sort the result (you can save the result in a file).

Manipulate Files

`grep, wc, sort, uniq`

1. Transform the text in *`book.txt`* in a list of words (you can store it in a new file);
   - → Replace the spaces by a new line (`\n`);
   - → *`sed 's/foo/bar/g' input_file > output_file`* replaces *`foo`* by *`bar`* in *`input_file`* and stores the result in *`output_file`*.
2. Sort the list by alphabetical order (you can also store it in a new file to not overwrite the previous one);
3. Get the number of occurrences of each word and sort the result (you can save the result in a file).
   - → *`uniq -c input_file > output_file`* counts the number of occurrences of each adjacent line appearing in *`input_file`* and stores the result in *`output_file`*.

Write and Run a Script

Write your script in a *.sh* file with *#!/bin/bash* on the first line and '$n' for the n-th argument.

Execute with *bash file.sh arg1 ... argN*.

Write and Run a Script : Count word occurrences

```
#!/bin/bash

sed 's/ /\n/g' $1 > words.txt
sort words.txt > sorted.txt
uniq -c sorted.txt | sort > counts.txt
tail -n 30 counts.txt
```

```
>> bash count_words.sh book.txt
```

Write and Run a Script : Count word occurrences, case insensitive

```
#!/bin/bash

sed 's/ /\n/g' $1 > words.txt
sort -f words.txt > sorted.txt
uniq -i -c sorted.txt | sort > counts.txt
tail -n 30 counts.txt
```

```
>> bash count_words.sh book.txt
```

The usual tip

Questions?

Some websites with shell commands for NLP

- https ://medium.com/nlp-technology/nlp-technology-text-processing-in-bash-linux-90e96c13781a
- https ://github.com/motazsaad/ShellCMDs4NLP
- https ://nlp.stanford.edu/ johnhew/bash-for-nlp-tutorial-basic.html