Shell useful commands

https://www.youtube.com/playlist?list=PLVQYiy6xNUxxhvwi0PGmXb5isUdVwmsg8

---

man

used to show the manual page of the specified command

---

pwd

used to show the current working directory

---

clear (or just ctrl + L)

clears all output written in the terminal

---

ls

used to show all the files and directories in the current directory

---

ls -a

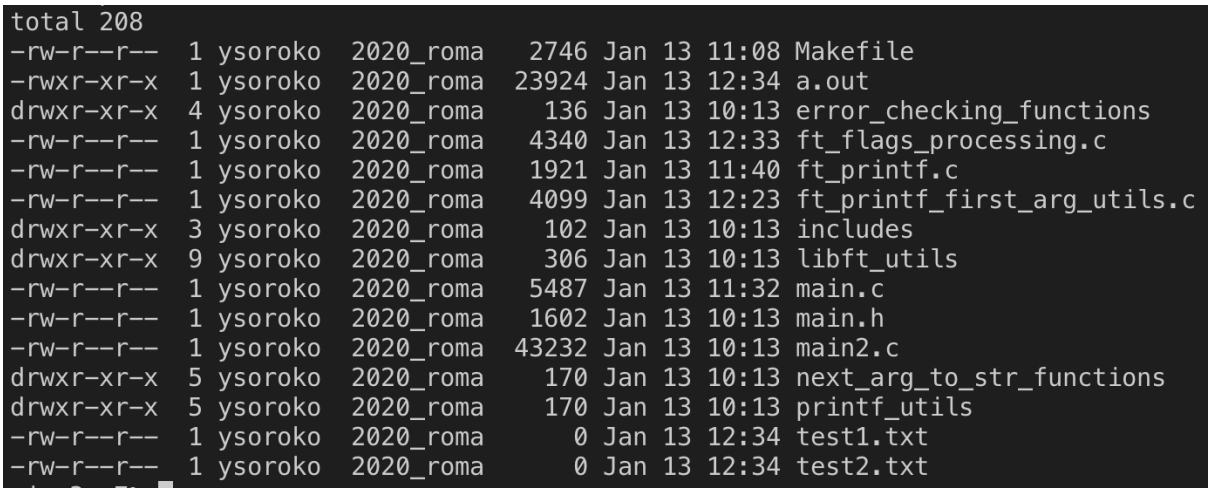
ls with ‘-a’ flag also shows hidden files

---

ls -l

ls with '-l' flag shows a detailed format:

(ls -la can also show hidden files under such format)



```
total 208
-rw-r--r--  1 ysoroko  2020_roma   2746 Jan 13 11:08 Makefile
-rwxr-xr-x  1 ysoroko  2020_roma  23924 Jan 13 12:34 a.out
drwxr-xr-x  4 ysoroko  2020_roma    136 Jan 13 10:13 error_checking_functions
-rw-r--r--  1 ysoroko  2020_roma   4340 Jan 13 12:33 ft_flags_processing.c
-rw-r--r--  1 ysoroko  2020_roma   1921 Jan 13 11:40 ft_printf.c
-rw-r--r--  1 ysoroko  2020_roma   4099 Jan 13 12:23 ft_printf_first_arg_utils.c
drwxr-xr-x  3 ysoroko  2020_roma    102 Jan 13 10:13 includes
drwxr-xr-x  9 ysoroko  2020_roma    306 Jan 13 10:13 libft_utils
-rw-r--r--  1 ysoroko  2020_roma   5487 Jan 13 11:32 main.c
-rw-r--r--  1 ysoroko  2020_roma   1602 Jan 13 10:13 main.h
-rw-r--r--  1 ysoroko  2020_roma  43232 Jan 13 10:13 main2.c
drwxr-xr-x  5 ysoroko  2020_roma    170 Jan 13 10:13 next_arg_to_str_functions
drwxr-xr-x  5 ysoroko  2020_roma    170 Jan 13 10:13 printf_utils
-rw-r--r--  1 ysoroko  2020_roma      0 Jan 13 12:34 test1.txt
-rw-r--r--  1 ysoroko  2020_roma      0 Jan 13 12:34 test2.txt
```

---

cd "name"

change current directory to the specified one

special symbols: "." (= dot) is the current directory

".." (= dot dot) is the previous directory

(example: cd folder will move me into "folder")

afterwards, cd . will move me into current folder, so I won't move

afterwards, cd .. will move me back to the starting folder

echo

show on output what follows, followed by a newline '$'

(example: echo Coucou will print "Coucou" on the standard output)

echo -n

same thing, but no newline character '$' at the end

---

rm "name"

used to remove a file with "name" name

rm -r "name"

used to remove the directory "name and everything inside.
asks for confirmation before deleting

rm -rf "name"

same as rm -r, but deletes without asking for confirmation

---

mv "a" "b"

1) renames "a" to "b" (if "b" doesn't exist)
2) moves "a" inside "b" (if "b" exists)

---

touch “name”

creates a file named “name”

---

mkdir “name”

creates a directory named “name”

---

cat “name”

displays the content of the file “name” on standard output

Example: cat test.txt shows all the text written in “test.txt” file

---

open “name”

opens “name” as if you double clicked on it in Finder (/!\ Mac only)

Example: open . command will open the current folder in finder

---

chmod “arg” “name”

changes the access rights of the “name” file depending on the specified “arg”

Example:

1) Create a “test.sh” file using “touch test.sh”, then use “ls -l”

Result of “ls -l” command before “chmod 777 test.sh”

```
mi-r3-p7% ls -l test.sh
-rw-r--r--  1 ysoroko  2020_roma  0 Jan 14 08:42 test.sh
```

Result of “ls -l” command after “chmod 777 test.sh”

```
mi-r3-p7% chmod 777 test.sh
mi-r3-p7% ls -l test.sh
-rwxrwxrwx  1 ysoroko  2020_roma  0 Jan 14 08:42 test.sh
```

You can see how in the first part of the list, the access right are added for each user (r = read / w = write / x = execute)

“777” is used to give all rights to all the users. More information about “arg” in man or video on youtube

---