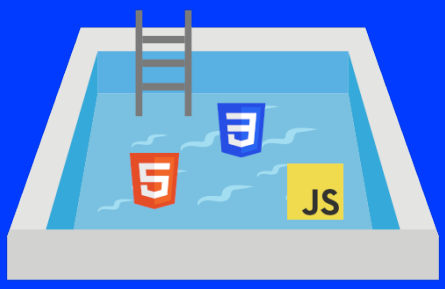
{EPITECH}

# WEB DEV SEMINAR_

## < DAY 01 - UNIX ENVIRONMENTS />



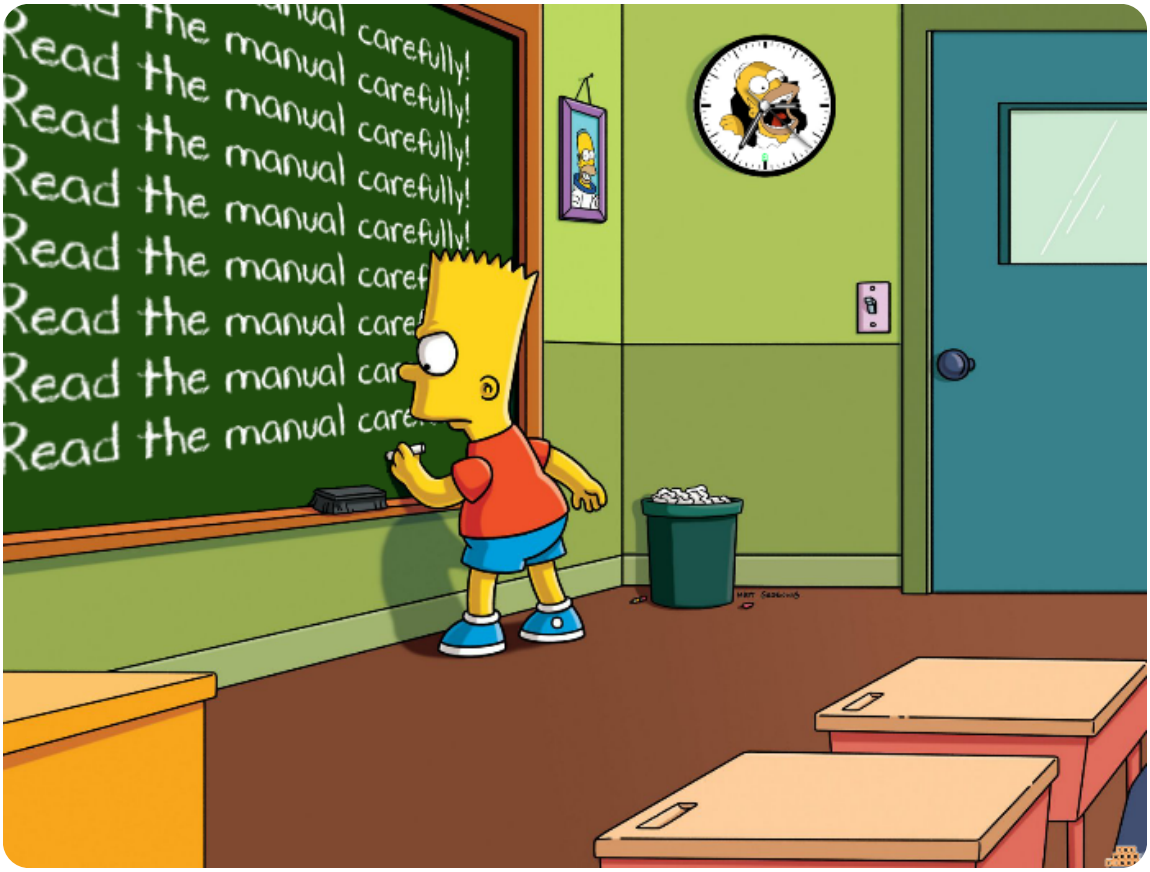
# WEB DEV SEMINAR

## Task 01

Create a directory named `task01`.
Within that directory, create:

- ✓ an empty file named `test01` with default permissions ;
- ✓ a file named `test02` containing "**If you don't struggle, you don't improve.**", you must add execution permissions for everyone for this file ;
- ✓ a symbolic link (symlink) named `test03`, which points to `test02`.

Read the **ln** man carefully!



## Task 02

Create a file `./task02/z`, which displays the character '**Z**' followed by a line feed (`\n`) when the binary `cat` is used to read it:

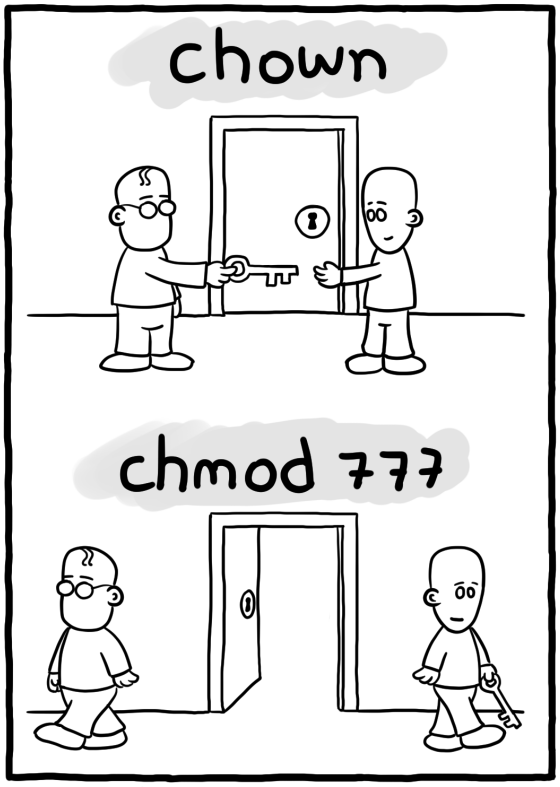
```
Terminal
$> cat -e z
Z$
```

## Task 03

Write in `./task03/midLS`, a command that lists the current repository's files and directories (without hidden files, ellipses, or files starting with a dot) sorted alphabetically.

Files and directories should be separated by commas. Directories must end with a slash.

Add execution permission to everyone.






{EPITECH}

# Git interlude

## Access your repositories

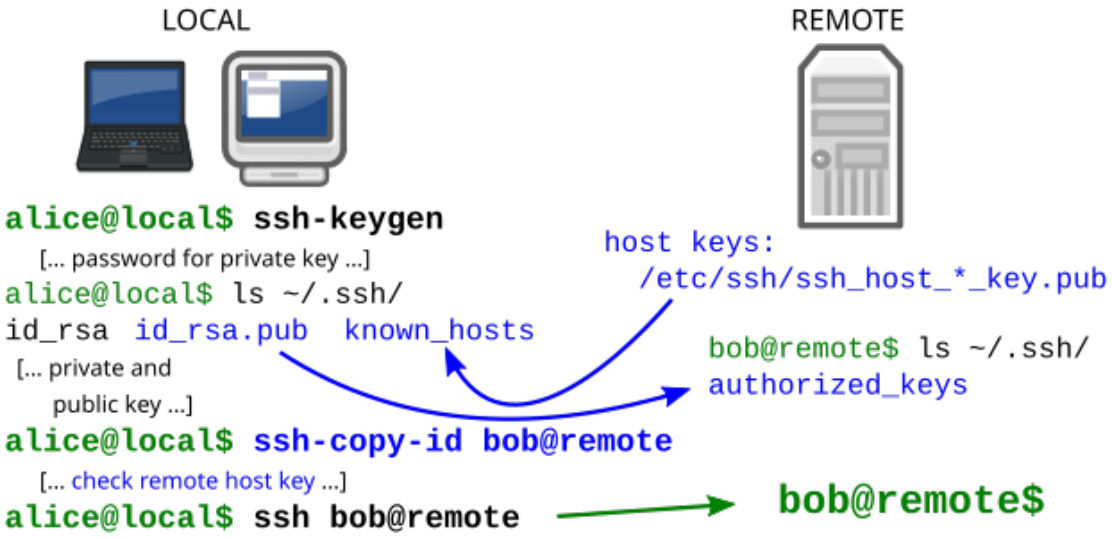
Most of your projects have to be turned-in as a Git repository hosted on GitHub. So, you must have a GitHub account using your Epitech email address. If you already have an account, you can simply add your Epitech email address to it.

On GitHub, all your Epitech projects will be grouped under an organization. If you check your email, you should have already received an invitation to your promotion's Epitech organization.

If you've not received an invitation email, contact your local pedagogical team about it.

Okay, now you're in the organization. And you're part of a team within this organization, which gives you access to the project's dedicated repository.

You must generate **a ssh key** named `id_rsa`. Then, add add it to your GitHub account. It allows you to authenticate with GitHub to access your repositories.




Be sure to authorize the previously added ssh key for use with GitHub single sign-on.

For more information, see the document ”How to turn-in”.

## Clone a repository, then push your work to deliver

Now is time to check if the previous part really works. To do so, clone the repository for today's delivery, it should be available at an address that looks like this: `git@github.com:<$PROMOTIONTEAM>/<$MODULECODE>-day01-<$GROUPNAME>.git`, where:

- ✓ `<$PROMOTIONTEAM>` is the github team corresponding to your promotion,
- ✓ `<$MODULECODE>` is the code of the project's module,
- ✓ `<$GROUPNAME>` is the name of your group for this project.

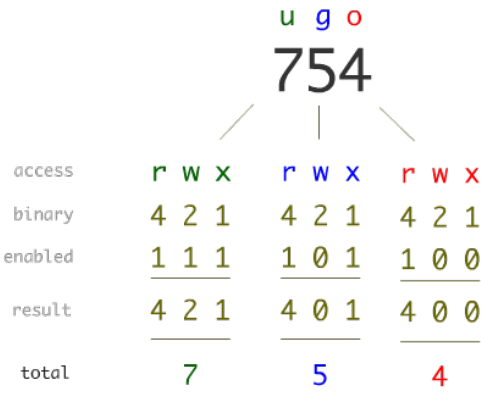
Visit the organization's GitHub page to view all your accessible repositories.
On a repository webpage, click on the ”Code” button to get its SSH URL.

If everything worked correctly, you'll have a result similar to:

```
Terminal
$>  git clone git@github.com:EpitechBachelorPromo2042/T-WEB-500-day01-LIL_john.doe.git into
'T-WEB-500-day01-LIL_john.doe'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done
```

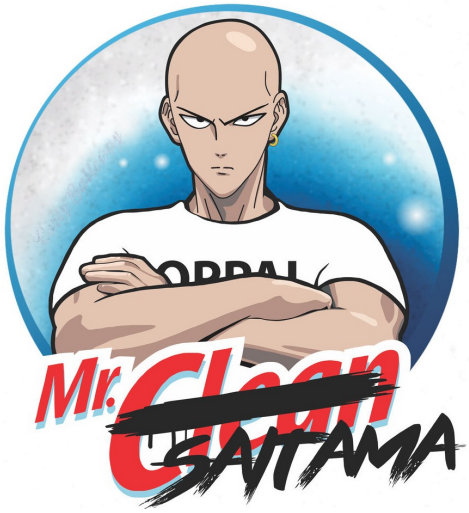
Now, move all your previously created task folders into the repository of the day.
Then, you have to inform `git` of your wish to **add** these files.
After that, you have to **commit** a new local revision which contains all these modifications.
Finally, you have to **push** this revision to the remote server.
From now on, commit and push at the end of each task.

Make sure that the file permissions have not been altered during the copy/move.




## Task 04

Create a shell script named `mr_clean` at the root of the repository of the day. It recursively finds and deletes every file from the current directory and all subdirectories which end with `~` or start **and** end with `#`. Give execution rights to the owner of the file.




Only one command is allowed (no ";", neither "&&" or anything...).

Read carefully the manual of `find`.

## Task 05

At the root of your repository, write a shell script named `push_that.sh`, that :

- ✓ takes a commit message as parameter ;
- ✓ adds all of the current folder's files and pushes them to the repository ;
- ✓ should be able to handle simple problems and still push your files.

`man bash` to figure out how to retrieve parameters.

Don't forget to add a **shebang** and to set the proper execution rights to your scripts.
Don't use this script if you share a repository with other people.

## Task 06

Replicate the folder structure exactly as displayed below.

```
"task06"/
|-- "Electro"/
|   |-- "Aphex Twin"/
|   |   `-- "Come to Daddy"/
|   |       `-- "Bucephalus Bouncing Ball"
|   `-- "Fatboy Slim"/
|       `-- "You've Come a Long Way, Baby"/
|           `-- "1-02 - The Rockafeller Skank"
|-- "HipHop"/
|   |-- "Beastie Boys"/
|   |   `-- "Singles"/
|   |       `-- "Sabotage-Get It Together"/
|   |           `-- "Sabotage (LP Version)"
|   |-- "MF DOOM"/
|   |   `-- "Madvillain - Madvillainy"/
|   |       `-- "03 - Meat Grinder"
|   `-- "Notorious BIG"/
|       `-- "Life After Death"/
|           `-- "Hypnotize"
|-- "Rock"/
|   `-- "Rage Against The Machine"/
|       `-- "Rage Against The Machine"/
|           `-- "Killing In The Name Of"
`-- "Soundtracks"/
```



{EPITECH}

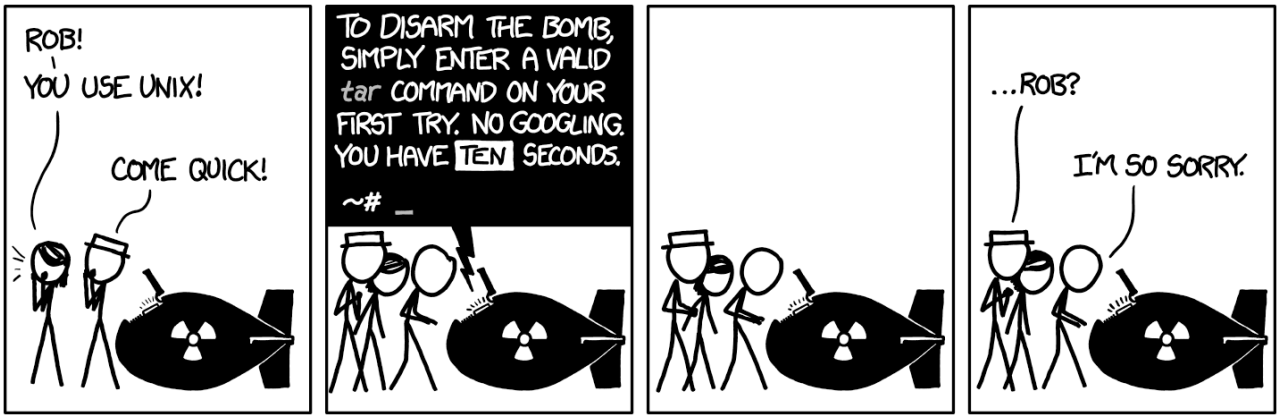
Git handles empty directories differently.


## Task 07

Using Gzip, create a compressed tarball of the previous task's directory content, named `task06.gz`. Then, deliver it inside the folder `task07`.

man tar






**Always respect the naming conventions, or you gonna have a bad day!**

v 2.0.1