

# 04 - INTRODUCTION AU TERMINAL & GIT

IPA 11x001 exercices - Université de Genève

---

Stéphane Nguyen

[https://github.com/Zenchiyu/11x001\\_tp](https://github.com/Zenchiyu/11x001_tp)

# CONTENTS

---

1. Semaine 2 - Introduction au terminal
2. Semaine 2 - Git clone et pull
3. Semaine 3 - Clés ssh et GitHub
4. Semaine 4 - Votre projet Git
5. Semaine 5 - Branches, git push et pull
6. Extra
  - Liens complémentaires

- **Terminal**: programme (souvent avec interface graphique ou en mode texte) permettant d'interagir avec le système via des **commandes textuelles**. Le terminal affiche l'entrée/sortie, mais il ne comprend pas les commandes lui-même. Les commandes sont transmises au **shell**.
- **Shell**: programme qui interprète les commandes saisies dans le terminal et demande au système d'exploitation de les exécuter (ex. Bash, Zsh, Fish, PowerShell).
- **Shell scripting language**: langage de programmation associé au shell, permettant d'écrire des scripts (fichiers .sh, .zsh, etc.) avec variables, boucles, conditions, fonction etc. pour automatiser des tâches.
- **PowerShell**: lorsque vous lancez l'application PowerShell, vous ouvrez une fenêtre de terminal qui exécute le shell PowerShell à l'intérieur. Le terminal est l'interface, et PowerShell est le programme qui interprète vos commandes et permet aussi d'écrire des scripts (langage PowerShell).

- **Git**: système de gestion de versions décentralisé (Version Control System VCS) installé localement sur votre ordinateur.

Plutôt que d'ajouter un numéro à chaque nouvelle version d'un fichier, Git enregistre les différences (deltas) à chaque modification. Vous pouvez donc

- Suivre l'historique de votre projet & revenir facilement aux anciennes versions.
- Travailler en parallèle sur différentes versions de votre projet à travers des “branches”.
- **GitHub et Gitlab**: plateformes web permettant d'héberger des dépôts Git (repositories, pensez à projets). Elles offrent des fonctionnalités collaboratives dont:
  - suivi des problèmes (issues),
  - revue de code,
  - intégration continue/déploiement (CI/CD) (ex. couverture de tests)

- Git est gratuit, et a été créé par Linus Torvalds en 2005
- Linux ou GNU/Linux est basé sur le noyau Linux développé par Linus Torvalds ([contro-](#)  
[versie](#))
- La majorité des développeurs utilisent Git ([source en 2023](#))
- Beaucoup de développeurs utilisent GitHub ou GitLab pour mettre en avant leur portfolio de projets afin d'obtenir un travail.

- Git devrait déjà être installé dans Ubuntu dans WSL 2
- Idem pour **macOS**. Cependant, si pas installé: <https://git-scm.com/downloads/mac>
- Vous allez voir quelques commandes durant le semestre mais **vous n'aurez juste besoin de deux durant le semestre: git clone et git pull.**
- Les commandes du terminal ainsi que Git et GitHub ne seront pas dans l'examen.

# SEMAINE 2 - INTRODUCTION AU TERMINAL

---

- Se déplacer dans le filesystem : `cd <chemin>`
- Revenir un répertoire au dessus : `cd ..`
- Voir les fichiers présents dans le répertoire courant : `ls` ou `ls -l`
- Voir aussi les fichiers cachés: `ls -a`
- Voir où est-ce que je me trouve : `pwd`
- Lancer un code C :
  - Compiler votre code source avec `clang <nom>.c -o <nom_executable>`
  - Exécuter avec `./<nom_executable>`
- Lancer un code Python : `python3 <nom>.py`



- Création
  - Créer un fichier vide: `touch <nom>`
  - Créer un dossier vide (make directory): `mkdir <nom>`
- Suppression (**Attention, pas de corbeille !**):
  - Supprimer le fichier (-i pour éviter des suppressions involontaires): `rm -i <nom>`
  - Supprimer un dossier vide: `rmdir <nom>`
- Déplacement ou copie (**Attention, cela peut écraser des fichiers !**):
  - Déplacer un fichier ou dossier: `mv <chemin_source> <chemin_destination>`
  - Copier un fichier: `cp <chemin_source> <chemin_destination>`
  - Copier un dossier ainsi que le contenu: `cp -r <chemin_source> <chemin_destination>`

- Afficher le contenu textuel d'un fichier dans le terminal: `cat <nom>`
- Revenir sur le chemin précédent: `cd -`
- Si vous entrez dans l'éditeur nano sans faire exprès, pour sortir, faites `Ctrl + X`.
- Si vous entrez dans l'éditeur vim sans faire exprès, pour sortir, faites `Esc` puis tapez `:q!` en bas.

# SEMAINE 2 - GIT CLONE ET PULL

---

- Nous allons vous montrer les prochain(e)s outils et commandes en direct:
  - Divers boutons sur le dépôt des exercices d'IPA
  - `git clone` en utilisant un lien HTTPs (on verra plus tard avec SSH)
  - `git pull`

- Tapez cette commande dans votre terminal:

```
git clone https://github.com/Zenchiyu/11x001_tp.git
```

ce qui clone le dépôt des exercices sur votre ordinateur. En particulier, vous aurez un dossier appelé 11x001\_tp.

**Ne modifiez rien dans ce dossier 11x001\_tp !**

- A chaque fois que nous ferions des modifications, allez dans le dossier et tapez `git pull` dans votre terminal. Cela mettra à jour les fichiers.
- **Lors de vos exercices, veuillez copier les TPs dans un autre dossier. Pas dans 11x001\_tp.**

# SEMAINE 3 - CLÉS SSH ET GITHUB

---

- Encrypter un message le rend illisible pour des tiers ne disposant pas de la clé privée.
- Avoir la clé privée permet de décrypter le message.
- Pour encrypter un message, un fichier ou autre, on peut le faire de façon symétrique (une clé) ou asymétrique (deux clés)
- Dans le cas asymétrique, une clé est utilisée pour encrypter (publique) et l'autre pour décrypter (privée).
- Dans le cas d'une authentification, on utiliserait la clé privée pour prouver son identité alors qu'une clé publique serait utilisée pour la vérifier.

- Dans le cas de GitHub, on peut s'authentifier soit à travers HTTPs, soit à travers SSH.
- A travers HTTPs, cela vous demandera de rentrer votre nom d'utilisateur et un mot de passe ([L'access token remplace votre mot de passe de nos jours](#)).
- A la place, on vous fera passer à travers SSH pour votre projet Git (Semaine 4).
  - Cela demande la création d'une paire de clés et de donner à GitHub votre clé publique
  - Une fois que c'est bon, vous n'aurez plus besoin de rentrer un mot de passe.



- Allez dans votre terminal (dans Ubuntu dans WSL ou dans macOS) et tapez:

```
ssh-keygen -t ed25519 -C "votre_email_ici@example.com"
```

- Tapez dans le terminal

```
cat ~/.ssh/id_ed25519.pub
```

puis copiez ce qui est affiché (c'est la clé publique).

- Sur GitHub, allez dans les “Paramètres > clés SSH et GPG” et “Créer une nouvelle clé SSH”.
- Mettez un titre et collez le contenu de `~/.ssh/id_ed25519.pub` dans la boîte de texte “Clé”

# SEMAINE 4 - VOTRE PROJET GIT

---

- Nous allons vous montrer les prochain(e)s outils et commandes en direct:
  - Création d'un dépôt/projet GitHub
  - Composition minimale: README.md et LICENSE ([explication](#))
  - Fichier gitignore et pourquoi le “.” devant
  - `git init`
  - `git status`
  - `git add`
  - `git commit`
  - `git diff`
  - `git log`
  - `git clone` mais avec SSH.
- Jusqu'à là, votre projet n'est pas mis à jour sur GitHub ! Voir Semaine 5.

# **SEMAINE 5 - BRANCHES, GIT PUSH ET PULL**

---

- Nous allons vous montrer les prochain(e)s outils et commandes en direct:
  - `git push` (avec `set upstream`)
  - `git pull`
  - `git checkout`
  - `git branch`
  - `git merge` et `pull request` sur GitHub

**EXTRA**

---

Si vous le souhaitez, vous pouvez lire plus sur Git et GitHub.

Voici quelques liens intéressants de concepts dont certains n'ont pas été expliqué:

- [Explication de Git par Fireship](#)
- `git stash` (l'[explication de git stash ainsi que d'autres commandes](#))
- `git rebase` (l'[explication de merge vs rebase](#))

Sophia Koulen a contribué à l'amélioration des slides.