

LINUX et le WSL

1. Définitions

Lorsque nous travaillons sur notre machine, nous travaillons avec sur un environnement bien précis déterminé par notre **système d'exploitation**. Ce système d'exploitation est Windows, en l'occurrence **Windows 11**.

Mais il n'est pas le seul système d'exploitation disponible, loin de là. Au contraire, on peut en compter des centaines, tous regroupés dans certaines « familles », caractérisées par l'origine de leur « noyau ». Soit le noyau **UNIX**, soit le noyau **MS/DOS** pour Microsoft.

Si l'on souhaite travailler avec un noyau UNIX, on peut se tourner vers Linux, auquel cas il nous faudra choisir une des nombreuses **distributions** de Linux. Il en existe là aussi des centaines, voyez plutôt :

https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

Dans notre cas, nous allons nous tourner vers une distribution bien précise, la distribution **Ubuntu**, et pas n'importe laquelle : Celle embarquée dans Windows.

2. Le WSL, révolution en soi

Le WSL, ou **Windows Subsystem for Linux**, est une nouveauté qui a été ajoutée à Windows 10 et qui représente une véritable révolution pour nous, développeurs : En continuant à travailler sous un environnement Windows, nous avons accès à une distribution de Linux **virtualisée** sur la même machine.

La solution utilisée par WSL présente plusieurs avantages :

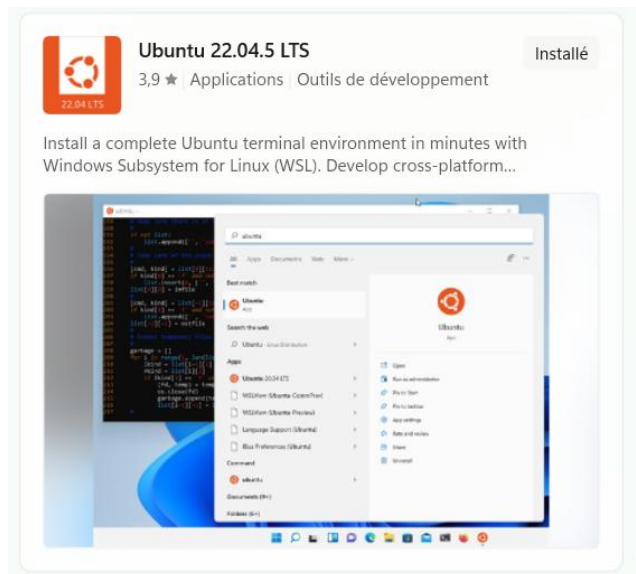
- des temps de démarrage très courts
- une exécution avec des ressources réduites
- des interactions possibles entre les systèmes de fichiers de Linux et de Windows dans les deux sens
- aucune configuration ou gestion de la machine virtuelle n'est requise

La distribution par défaut intégrée dans le WSL est Ubuntu 22.04.

3. Installation

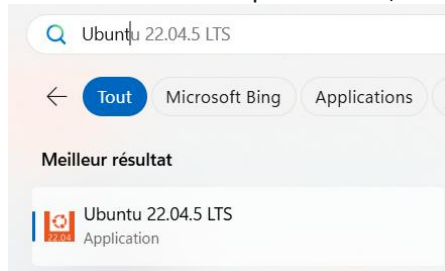
Méthode rapide : `wsl --install Ubuntu-24.04`

Méthode visuelle : En passant par l'application Microsoft Store, recherchez « Ubuntu 22.04.5 ». Vous devriez arriver sur ceci :

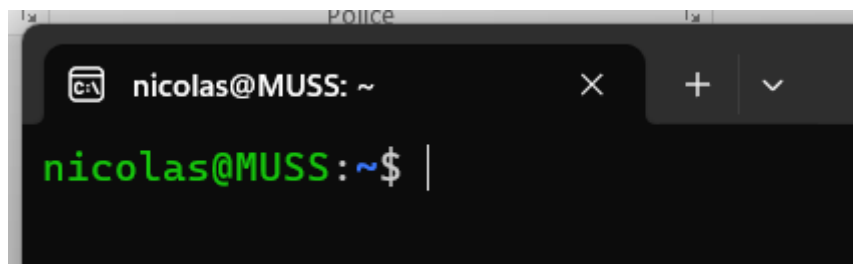


Installez cette version.

Une fois cette installation effectuée, vous devriez avoir l'application prête à lancer dans le menu Démarrer. Si vous tapez Ubuntu, vous devriez avoir ceci :



Lancez cette application. Un terminal va alors s'ouvrir avec cette police verte pour vous signifier que vous êtes bien sur Linux 😊



4. Commandes fondamentales

Contrairement à Windows, le WSL est un système d'exploitation **sans interface graphique**. Ce qui veut dire que l'on va naviguer dedans en utilisant des **commandes**.

I. Savoir où on est

pwd	Pour savoir où on est dans cette interface graphique, on peut utiliser la commande pwd . Ça ressemble à password mais ça n'a rien à voir, cela veut dire « Print Working Directory » : Afficher le répertoire en cours de travail.
ls	Permet de dresser la liste des fichiers présents dans le répertoire où l'on est. On peut lui passer différents paramètres pour obtenir des informations supplémentaires sur les fichiers du répertoire, comme par exemple ls -la qui affiche les informations détaillées de TOUS les fichiers
tree	Permet d'afficher l'arborescence du répertoire courant y compris tous ses sous-dossier. Il peut être intéressant d'y ajouter un paramètre -L suivi de X pour limiter le nombre de niveaux affichés. Exemple : tree -L 3 Attention, cette commande doit être installée avec sudo apt install tree

II. Se déplacer

cd	<p>cd signifie « change directory », littéralement changer de répertoire. On lui précise un chemin vers un répertoire que l'on souhaite visiter, comme par exemple cd /home/vagrant/dynamvc</p> <p>Utilisez la tabulation pour profiter pleinement de l'auto-complétion, qui peut vous aider à reconstituer des chemins complexes.</p> <p>Pour remonter d'un cran dans l'arborescence, on peut utiliser ../ dans le chemin donné à cd</p>	cd ./dev/Homestead
-----------	--	---------------------------

Attention, avec Linux, les chemins sont **sensibles à la casse**. Desktop n'est pas desktop...

my folder : **Downloads**

me : **cd downloads**

Linux :



III. Créer

mkdir	Permet de créer un dossier à l'endroit où l'on est. On tape mkdir suivi du nom du dossier à créer	mkdir monDossier
touch	Permet de créer un fichier à l'endroit où l'on est. On tape touch suivi du nom du fichier à créer	touch monFichier.php
nano	Permet d'éditer le contenu d'un fichier. En général on lui passe le paramètre « -m » pour autoriser le fonctionnement de la souris dans l'éditeur. Sans ça, on est obligé d'utiliser les flèches pour naviguer. Pour quitter l'édition, on va faire Ctrl + X, puis confirmer ou non les modifications et le nom de fichier.	nano -m index.php

IV. Renommer / Déplacer / Supprimer

mv	Mv comme move permet de déplacer un fichier vers une destination. On lui donne deux paramètres : le fichier à déplacer, et sa destination. On peut également s'en servir pour renommer un fichier, on le déplace vers son nouveau nom	mv index.php public/index.php (Ici on déplace le fichier index.php dans un dossier public)
cp	Permet de copier un fichier. On lui donne deux paramètres : le nom du fichier à copier, et le nom de sa copie	cp index.php indexBackup.php
rm	Permet de supprimer un répertoire ou un fichier. On peut lui passer des paramètres, notamment -rf comme remove force , pour FORCER une suppression	rm index.php

V. Chercher

find	Find permet de chercher un fichier donné. On lui passe en paramètre un chemin dans lequel chercher, par exemple . pour chercher dans le dossier courant, ou encore -name pour avoir un nom de fichier précis.	find . -name "filename"
grep	GREP permet de chercher dans le contenu d'un fichier. Par exemple si je veux vérifier que je n'ai pas oublié de remplacer une adresse e-mail sur ton un site, je peux chercher toutes les occurrences de cette adresse mail dans le serveur avec de nombreux paramètres que je vous invite à découvrir ici : https://docs.rockylinux.org/10/books/sed_awk_grep/2_grep_command/	grep -rl "monTexte" .

tail	Tail permet d'afficher la fin d'un fichier. Il est particulièrement utilisé pour regarder les fichiers de logs , que ça soit les logs d'accès ou les logs d'erreur. On le couple souvent avec le paramètre -f qui permet de laisser le fichier de log ouvert dans le terminal et donc de voir ce qui s'écrit dedans en temps. Particulièrement utile pour surveiller une erreur que l'on souhaiterait provoquer afin de la déboguer.	tail -f /tmp/php-error.log
-------------	--	-----------------------------------

Si jamais vous avez besoin d'une autorisation administrateur, sachez que toutes ces commandes peuvent être précédées de **sudo**

sudo veut littéralement dire « Super user » do, soit « exécuter en tant qu'administrateur ».

5. APT

Comment gérer des installations de composants sous Linux ? Admettons que je veuille utiliser Node pour un projet quelconque.

Nous allons ici pouvoir utiliser un **gestionnaire de package** intégré à Linux, qui s'appelle **APT** (pour Advanced Package Tool). Un gestionnaire de package est un composant logiciel capable de coordonner différentes installations et surtout de vérifier leur **compatibilité**. Si je cherche à installer une version d'un composant qui n'est pas compatible avec une autre, alors le gestionnaire de package va m'en avertir, voire même me proposer des solutions.

Sous Ubuntu, on peut donc nativement utiliser apt. Pour reprendre notre exemple, si je voulais installer Node, je ferais d'abord une mise à jour du package manager :

```
sudo apt update
```

Puis je demanderais à apt de m'installer node :

```
sudo apt install nodejs
```

Si je veux en plus installer npm avec Node, je peux passer plusieurs packages d'un coup à apt, comme ceci :

```
sudo apt install nodejs npm
```

Installez htop ou tree pour la démo

Exercice

Installez tree avec `sudo apt install tree`

Créez vous un répertoire **wsldev** dans votre dossier `home/{username}`

Créez un dossier **projects** dans le dossier `wsldev` et vous aurez une structure semblable à celle que l'on utilise sur Windows.

Créez un dossier **pratique** dans votre dossier `projects`, et créez cette structure de fichiers (peu importe le contenu des fichiers, utilisez **touch**) :

```
nicolas@MUSS:~/wsldev/projects/pratique$ tree
.
├── assets
│   ├── css
│   │   └── main.css
│   └── js
│       └── main.js
├── images
│   ├── backgrounds
│   │   ├── background_homepage.png
│   │   └── menu.jpg
│   └── logos
│       ├── mamarque
│       │   ├── monLogo.png
│       │   └── monLogoFooter.png
│       └── partenaires
│           ├── logoOpel.png
│           ├── logoPeugeot.png
│           └── logoRenault.png
└── index.php

8 directories, 10 files
```