



# Linux d'après Napoléon

## C'est quoi un système d'exploitation ?

Un système d'exploitation constitue l'ensemble des logiciels nécessaire au fonctionnement basique d'un ordinateur. L'abréviation pour désigner le système d'exploitation est "OS", qui vient de l'anglais **Operating System**. L'OS qui a conquis le plus de territoire est Windows, de Microsoft, grâce à son ancienneté et à [l'appel d'IBM pour faire un système pour leurs premiers ordinateurs, qui a conduit Microsoft à devenir une référence dans le développement d'OS](#). L'OS que Microsoft a fait pour IBM est MS-DOS, ou Microsoft Dirty Operating System.

## C'est quoi "Linux" ?

Linux est un noyau ou **kernel** d'OS, soit la **brique** logicielle la plus proche du matériel, gérant les ressources disponibles et donnant une interface aux applications pour interagir avec ledit matériel. La première **release** (sortie) du noyau Linux date de 1991, son auteur est Linus Torvalds

## Linux ne suffit donc pas,

c'est pourquoi beaucoup d'OS sont de forme GNU/Linux, où GNU (se prononce "gnou") désigne des applications qui implémentent des fonctionnalités basiques dont a besoin tout utilisateur: créer un fichier, l'éditer, lister les fichiers dans un dossier, créer un dossier... [Les applications GNU](#) sont très nombreuses. Dire "un système Linux" est donc à la base un abus de langage et le vrai terme est "un système GNU/Linux".

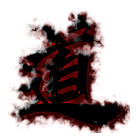
## Distributions Linux

Bien que GNU/Linux soit au centre de presque tous les systèmes d'exploitation qu'on dénomine par abus de langage "Linux", certaines "distribution", adhérant à des philosophies d'organisation du système et d'expérience utilisateur, existent. Des exemples de distributions Linux et de leurs philosophies sont:

- Debian, distribution impérative (la configuration se fait avec des commandes)



- Ubuntu, qui est Debian avec l'environnement de bureau **GNOME** préinstallé (offrant une expérience de bureau relativement proche de celle de MacOS). C'est la distribution Linux la plus renommée, le développement est dirigé par l'entreprise anglaise Canonical.



- Kubuntu, qui est Debian avec l'environnement de bureau **KDE Plasma** préinstallé (offrant une expérience de bureau relativement proche de celle de Windows)
- Archlinux, cherche à être le plus **base bones** possible
- NixOS, qui cherche à avoir une configuration complètement **déclarative** (donc qui se configure en modifiant des fichiers qui définissent la configuration globale du système)
- SteamOS, qui cherche à être optimisé pour le jeu vidéo avec des outils comme Proton pour permettre la compatibilité avec certains jeux vidéos et applications faites pour l'OS Windows

Pour résumer, la plupart des systèmes sous la dénomination "Linux" sont en fait des noyaux Linux packagés en distributions, qui peuvent elles-mêmes être basées sur d'autres distributions, et utilisant le plus souvent les applications de la suite GNU.

Toutes ces distributions sont FOSS (Fully Open Source Software) à cause de la license de Linux, la [GPL](#). Pour résumer, la GPL force les forks d'un programme développé avec cette license à utiliser aussi la GPL et à être open source, dans l'intérêt de l'utilisateur final ("all right reversed").

## NixOS

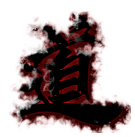
Nous allons ici nous intéresser à NixOS car une configuration déclarative a de nombreux avantages sur une configuration impérative. Pour rappel, une configuration impérative se fait en entrant des commandes tandis qu'une configuration déclarative se fait en modifiant des fichiers de configuration. Avec NixOS, qui est un système qui est construit au tour du **gestionnaire de paquets** Nix, il est possible de configurer toutes les applications, avec une syntaxe unifiée pour ne pas avoir besoin d'apprendre à configurer chaque application.

### Mais qu'est qu'un gestionnaire de paquets, ou package manager ?

Un gestionnaire de paquets est un programme qui permet d'installer et mettre à jour tous les programmes. Les gestionnaires de paquets sont omniprésents dans les systèmes GNU/Linux: **apt** dans les systèmes basés sur Debian, **pacman** sous Archlinux, **nix** sous NixOS... En revanche, sous Windows, le seul programme qui s'apparente à un gestionnaire de paquets est le Microsoft Store. La plupart des applications se mettent à jour de manière autonome sous Windows, ce qui conduit à un travail redondant supplémentaire et souvent bâclé pour les développeurs d'applications.

## Installation

Des instructions d'installation détaillées sont disponibles sur [le manuel](#). Il est fortement recommandé d'utiliser un flake (et donc de suivre la partie sur la configuration basée sur un flake), qui est une fonctionnalité de Nix qui permet de simplifier fortement une configuration. Il est recommandé de lire les commentaires et d'éditer le fichier de configuration du système `/etc/nixos/configuration.nix` avant d'installer le système. Certaines lignes seront commentées, ce qui se voit par un dièse # au début de la ligne. Une ligne commentée ne sera pas prise en compte dans l'évaluation de la configuration. Il est recommandé de décommenter la ligne qui active NetworkManager. L'outil recommandé dans le manuel, `nano`, est un bon exemple de programme GNU. En revanche, il pue la mort et l'utiliser revient à pactiser avec le Diable. Les commandes doivent être entrées dans une **shell**.



## Une shell ? Késako ?

Une shell est un interpréteur de ligne de commande. Elle permet à un utilisateur de contrôler son système en écrivant des commandes dans la shell.

## VI, VIM, NVIM

Un éditeur qui est hyper bien, en revanche, est **NVIM** (NeoVIM), une **fork** (= une implémentation qui se base sur un programme existant) de VIM (VI Improved), qui lui-même est une fork de l'historique VI.

Je recommande l'usage de NVIM, qui est fortement customisable par des plugins qu'on peut écrire en langage de programmation **Lua**.

Pour installer NVIM, il suffit d'ajouter le **package** nvim à la liste des packages du système. Pour ce faire, il suffit d'ajouter nvim à la liste des packages du système, dénommés par `environment.systemPackages`, dans `/etc/nixos/configuration.nix`.

```
{
  environment.systemPackages = with pkgs; [
    nvim
  ];
}
```

Listing 1: Un extrait de `/etc/nixos/configuration.nix`

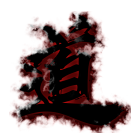
La vraie puissance de NVIM ne se révèle cependant seulement une fois customisé. Un bon départ est d'utiliser une configuration existante. Je recommande NvChad, accessible sous NixOS en suivant [ces instructions](#).

NVIM est un éditeur qui fonctionne en ligne de commande, qu'on appelle donc depuis une shell. Tente d'écrire nvim puis d'appuyer sur Entrée dans la ligne de commande affichée (celle ci s'appelle une TTY (teletypewriter)).

## GUI !

Une **GUI**, ou Graphical User Interface, est une interface non textuelle comme communes sous Windows. Pour utiliser des applications qui utilisent des GUI, comme par exemple la plupart des navigateurs Web, il faut d'abord avoir installé un compositeur. Je recommande ici [Niri](#), un compositeur qui a pour philosophie de mettre en valeur le pavé tactile et le défilement en général. La section [getting started](#) de sa documentation est très utile pour débiter. Une fois que niri est ajouté à vos `environment.systemPackages` (de la même manière que NVIM), il est trivial de le lancer en écrivant "niri" en ligne de commande.

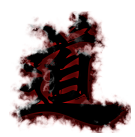
Cependant, sans applications, un compositeur est inutile. C'est pourquoi il est recommandé d'installer un terminal, qui donne accès à une GUI pour interagir avec la shell dans un compositeur, je recommande "kitty", mais aussi un navigateur web comme "firefox" et un lanceur d'applications comme "fuzzel". En lisant l'aide de Niri, qui aide à connaître les raccourcis clavier par défaut, il est possible de lancer fuzzel, pour ensuite lancer kitty, puis enfin NVIM depuis kitty (comme NVIM est une application qui s'utilise dans une shell). Tu peux aussi lancer firefox depuis fuzzel.



## Configurer des applications de manière déclarative

Quasiment toutes les applications sont déjà supportées pour être configurées avec la syntaxe unifiée. Voici un exemple de configuration pour kitty, un terminal:

```
{ pkgs, ... }:  
{  
  programs.kitty = {  
    enable = true;  
    font = {  
      package = pkgs.nerd-fonts.jetbrains-mono;  
      name = "JetBrainsMono Nerd Font";  
      size = 9;  
    };  
    shellIntegration.enableFishIntegration = true;  
    keybindings = {  
      "f1" = "launch --cwd=current --type=tab nvim";  
      "f2" = "launch --cwd=current --type=tab";  
      "kitty_mod+f1" = "show_kitty_doc overview";  
    };  
    settings = {  
      allow_remote_control = "socket-only";  
      listen_on = "unix:/tmp/kitty";  
      window_padding_width = 25;  
      hide_window_decorations = true;  
  
      foreground = "#CDD6F4";  
      background = "#1C2433";  
      selection_foreground = "#1C2433";  
      selection_background = "#F5E0DC";  
      cursor = "#F5E0DC";  
      cursor_text_color = "#1E1E2E";  
      url_color = "#B4BEFE";  
      active_border_color = "#CBA6F7";  
      inactive_border_color = "#8E95B3";  
      bell_border_color = "#EBA0AC";  
  
      active_tab_foreground = "#11111B";  
      active_tab_background = "#CBA6F7";  
      inactive_tab_foreground = "#CDD6F4";  
      inactive_tab_background = "#181825";  
      tab_bar_background = "#11111B";  
  
      mark1_foreground = "#1E1E2E";  
      mark1_background = "#87B0F9";  
      mark2_foreground = "#1E1E2E";  
      mark2_background = "#CBA6F7";  
      mark3_foreground = "#1E1E2E";  
      mark3_background = "#74C7EC";  
  
      color0 = "#43465A";  
      color8 = "#43465A";  
  
      color1 = "#F38BA8";  
      color9 = "#F38BA8";  
  
      color2 = "#A6E3A1";  
      color10 = "#A6E3A1";  
  
      color3 = "#F9E2AF";
```



```
color11 = "#F9E2AF";

color4 = "#87B0F9";
color12 = "#87B0F9";

color5 = "#F5C2E7";
color13 = "#F5C2E7";

color6 = "#94E2D5";
color14 = "#94E2D5";

color7 = "#CDD6F4";
color15 = "#A1A8C9";
};
};
}
```

Cependant, pour avoir accès à la plupart des applications que l'on voudrait pour un utilisateur et non pour le système, par exemple un navigateur, il faut utiliser le module Nix `home-manager`. Il permet donc de pouvoir avoir des configurations par-utilisateur et de pouvoir configurer bien plus d'applications. La configuration kitty ci-dessus, par exemple, n'est utilisable qu'avec `home-manager`.

L'installation de `home-manager`, comme tout le reste, est déclarative !

Dans `/etc/nixos/flake.nix`,