Guide technique pour Ubuntu et Linux Mint

Auteur : Christophe Masutti | Date : 21 04 2026

Ce document est une introduction à l'architecture logicielle et les mécanismes d'administration des distributions Ubuntu et Linux Mint.

1. Principes généraux des distributions

Ubuntu et Linux Mint reposent sur la distribution historique **Debian**. Cette filiation détermine l'organisation du système de fichiers et la méthode de gestion des logiciels.

- **Ubuntu (Canonical) :** orientée vers une standardisation des environnements de travail. Cette distribution utilise **GNOME** par défaut et suit un cycle de mise à jour rigoureux.
- **Linux Mint :** dérivé d'Ubuntu, elle vise à simplifier l'expérience utilisateur, notamment via l'environnement **Cinnamon**. Elle privilégie une ergonomie classique.

1.1. L'écosystème Linux

1.1.1. La nature du Noyau (Kernel)

Un système Linux est un empilement de couches logicielles. Le **noyau** gère le matériel, tandis que les **outils système** et l'**environnement de bureau** (GNOME sur Ubuntu, Cinnamon sur Mint) transforment ce noyau en un environnement de travail utilisable.

- Le noyau gère l'ordonnancement des tâches, les pilotes de périphériques et l'accès à la mémoire.
- Ce que nous appelons couramment « Linux » est souvent un assemblage du noyau Linux et des outils du **projet GNU** (compilateurs, interpréteurs de commandes). C'est pourquoi la dénomination rigoureuse est **GNU/Linux**.

1.1.2. Le développement communautaire et la Licence GPL

Le succès de Linux repose sur son modèle de licence, la **GNU GPL (General Public License)**.

- **Copyleft :** contrairement aux licences propriétaires, la GPL garantit la liberté d'étudier, de modifier et de redistribuer le code. Elle impose que toute modification distribuée soit elle-même sous licence GPL.
- **La Linux Foundation :** bien que le développement soit mondial et décentralisé, la Linux Foundation héberge le projet et emploie Linus Torvalds, le créateur original, qui conserve un rôle d'arbitre final.

1.1.3. Les cycles de développement du noyau

Le développement du noyau Linux suit une cadence extrêmement régulière, sans doute l'une des plus productives au monde.

- **Versions LTS :** Ubuntu publie une version à support long terme tous les deux ans. Linux Mint se base uniquement sur ces versions pour garantir une stabilité maximale.
- **Mise à jour du noyau (HWE) :** Pour supporter du matériel très récent sur une base stable, Ubuntu propose des noyaux HWE (*Hardware Enablement*), permettant de bénéficier de pilotes récents sans changer de distribution.

> 🔥 **Versions LTS**
>
> Selon les distributions, même si il est parfois proposé des versions non-LTS, il est préférable d'installer les versions LTS pour bénéficier de cette stabilité à long terme.

1.1.4. Le concept de distribution

Puisque le noyau est libre, n'importe qui peut l'assembler avec une interface graphique et des logiciels pour créer un système complet.

- Les familles : on distingue trois lignées majeures : **Debian** (dont descendent Ubuntu et Mint), **Red Hat / Fedora** (orientée entreprise) et **Arch Linux** (orientée vers une « simplicité » structurelle, sans trop d'outils de configuration automatiques, ce qui la réserve aux utilisateurs avancés souhaitant un contrôle total).
- Le choix : la différence entre les distributions réside principalement dans le **gestionnaire de paquets** (APT, DNF, Pacman) et la **philosophie de mise à jour** (versions fixes vs *rolling release*).

1.1.5. La gouvernance décentralisée

Linux n'est pas développé par une entreprise unique, mais par une communauté mondiale de développeurs bénévoles et d'entreprises qui contribuent au projet (dont IBM, Google, Microsoft, etc.). Sans éditeur unique,, le support repose sur l'entraide, la documentation, les formation, l'expertise.

> 🔥 **Un commun numérique**
>
> - Transparence : les bogues sont identifiés et corrigés publiquement.
> - Diversité : la communauté permet à Linux de fonctionner aussi bien sur un supercalculateur que sur un routeur, un téléphone (Android) ou un simple microcontrôleur.

2. Changement de paradigme : de Windows/Mac à Linux

Pour une transition efficace, il est nécessaire de comprendre certaines différences dans la gestion du système.

2.1. L'arbre unique (pas de lecteur C: ou D:)

Contrairement à Windows, Linux ne traite pas les disques comme des entités séparées désignées par des lettres.

- Tout périphérique (clé USB, second disque dur) est « monté » en tant que **volume** dans l'arborescence unique.
- Par exemple, vos données se trouvent dans `/home`, mais si vous ajoutez un disque ou une clé USB, il apparaîtra comme un simple dossier dans `/media`.
- Cette approche unifiée simplifie la navigation et la gestion des fichiers, mais nécessite une compréhension de l'arborescence du système (voir plus loin).

2.2. La configuration par le texte

Sous Linux, « tout est fichier ». La quasi-totalité de la configuration du système et des logiciels se fait par la modification de fichiers texte.

- Ces fichiers se trouvent majoritairement dans le répertoire `/etc`.
- Cela permet une transparence totale : il n'y a pas de « Base de registre » cachée ou binaire.

2.3. L'installation logicielle centralisée

Le réflexe consistant à chercher un installateur sur Internet est proscrit.

- **La Logithèque** est le moyen le plus sûr d'installer des logiciels. Ils sont vérifiés par les mainteneurs de la distribution. C'est l'équivalent du Microsoft Store ou de l'App Store d'Apple (qui s'inspirent d'ailleurs de ce qui se pratique depuis bien plus longtemps sous Linux), mais avec une garantie de sécurité et de compatibilité.

- **Sécurité :** en utilisant les dépôts officiels, vous réduisez à néant le risque de télécharger un logiciel contenant des logiciels espions (spywares), contrairement au téléchargement de `.exe` sur des sites tiers (cependant, voir plus loin pour les méthodes d'installation hors dépôts).

3. Installation des logiciels : les formats et les dépôts

Les logiciels sur ces distributions peuvent être installés via plusieurs méthodes, chacune ayant ses avantages et inconvénients en termes de sécurité, de maintenance et de compatibilité.

3.1. Le système de dépôts « classiques » (APT)

Un dépôt est un serveur distant contenant des logiciels compilés et signés numériquement par les mainteneurs de la distribution.

3.2. Fonctionnement du gestionnaire APT (Advanced Package Tool)

Le système maintient une base de données locale des logiciels disponibles. Lorsqu'un utilisateur installe un programme, APT effectue les opérations suivantes :

1. **Résolution des dépendances :** identifie les bibliothèques partagées indispensables au fonctionnement du logiciel.
2. **Téléchargement :** récupère les paquets au format `.deb`.
3. **Installation :** déploie les fichiers dans l'arborescence système.

> *i* **Les dépendances**
>
> On appelle « dépendances » les bibliothèques ou les composants logiciels dont un programme a besoin pour fonctionner. APT gère automatiquement ces dépendances, ce qui garantit que le logiciel installé dispose de tous les éléments nécessaires à son bon fonctionnement.

3.3. Ubuntu et le format Snap

En complément des dépôts APT, Ubuntu intègre **Snap**, un format de paquetage universel développé par Canonical.

- **Autonomie :** un Snap inclut toutes ses dépendances dans un volume compressé et isolé.
- **Confinement :** l'application s'exécute dans une **sandbox**, limitant ses interactions avec le système.
- **Mises à jour :** elles sont automatiques et gérées par le service `snapd`. (pour mettre à jour les snaps, on peut utiliser la commande `sudo snap refresh`).

3.4. Linux Mint et le format Flatpak

Linux Mint privilégie **Flatpak**, une alternative communautaire.

- **Philosophie :** permet d'installer des versions récentes de logiciels sans créer de conflits avec les bibliothèques stables du système hôte (un peu comme les snap mais avec une approche plus ouverte).
- **Flathub :** principal dépôt décentralisé pour l'acquisition de ces paquets.
- **Gestion des droits :** l'utilisateur peut restreindre finement les accès d'une application (réseau, fichiers).

3.5. Acquisition et installation de logiciels hors dépôts

Il arrive que certains logiciels ne soient pas disponibles via les gestionnaires de paquets. Deux méthodes principales prévalent alors : l'installation par paquet `.deb` (via la CLI ou une méthode graphique) et l'extraction de fichiers à partir d'archives compressées (tar.gz, zip).

3.5.1. Installation de paquets .deb

L'installation d'un fichier `.deb` tiers nécessite de la vigilance. Contrairement aux dépôts officiels, ces paquets ne sont pas audités par la distribution et ne bénéficient pas de mises à jour automatiques via `apt upgrade`.

- **Méthode par ligne de commande :** `sudo apt install ./nom_du_fichier.deb` (permet la résolution automatique des dépendances).
- **Méthode graphique :** double-cliquer sur le fichier `.deb` ouvre un gestionnaire de paquets qui guide l'installation.

3.5.2. Extraction d'archives et compilation (.tar.gz, .zip)

Certains projets fournissent le code source ou des binaires compressés.

- **Extraction :** l'archive doit être décompressée pour accéder aux fichiers.
- **Scripts d'installation (.sh) :** certains logiciels incluent un script (`install.sh`) qui automatise le placement des fichiers. Il s'exécute généralement via la commande `sh install.sh` ou `./install.sh`.
- **Compilation via Makefile :** si le projet contient un fichier `Makefile`, l'installation nécessite souvent la suite de commandes `make` (compilation) puis `sudo make install` (déploiement système).

> ⚠️ **Se documenter**
>
> Il est impératif de consulter le fichier `README.md` ou `INSTALL` présent sur le dépôt Git du projet ou la documentation sur le site officiel pour identifier les dépendances nécessaires et la meilleure méthode d'installation.

4. L'arborescence du système Linux : Standard FHS (Filesystem Hierarchy Standard)

Contrairement à d'autres systèmes, Linux utilise une arborescence unique partant de la racine `/`.

- `/` (la racine) : point de départ de tout le système.
- `/bin` et `/usr/bin` : exécutables des commandes de base.
- `/etc` : fichiers de configuration du système et des services.
- `/home` : répertoires personnels des utilisateurs.
- `/root` : répertoire personnel du super-utilisateur.
- `/var` : données variables et journaux (**logs**).
- `/media` et `/mnt` : points de montage pour les périphériques externes.

> 🔥 **Ne rien mettre sur le Bureau**
>
> Il est recommandé de ne pas stocker de fichiers directement sur le Bureau (`~/Desktop`) pour éviter les problèmes de synchronisation avec les services de cloud ou les sauvegardes automatiques. Utilisez plutôt des dossiers dédiés dans votre répertoire personnel (`~/Documents`, `~/Téléchargements`, etc.) pour une meilleure organisation et sécurité des données.

5. La ligne de commande (CLI) : cas d'usage et exemples

5.1. La désignation des chemins : absolu et relatif

- **Le chemin absolu :** commence par la racine `/`. Définit l'emplacement exact. Exemple : `/home/dupont/Documents/rapport.pdf`.

- **Le symbole tilde `~` :** raccourci vers le répertoire personnel (`/home/[utilisateur]`). Exemple : `cd ~/Images` mène à `/home/dupont/Images`.
- **Le chemin relatif :** se réfère à la position actuelle dans l'arborescence. Exemple : si vous êtes dans `/home/dupont`, la commande `cd Documents` vous amène à `/home/dupont/Documents`.

5.2. Syntaxe et logique des commandes

Pour interagir efficacement avec le terminal, il est nécessaire de comprendre la structure universelle des instructions Unix. Une commande ne s'exécute pas de manière isolée ; elle est modulée par des options et s'applique à des arguments.

5.2.1. La structure universelle

La syntaxe d'une commande suit presque systématiquement ce schéma :

```
commande [options] [arguments]
```

- La commande : l'action à effectuer (ex : `cp` pour copier).
- Les options : modifient le comportement de la commande (ex : copier de manière récursive).
- Les arguments : les cibles de l'action (ex : le fichier source et la destination).

5.3. Options courtes vs Options longues

Il existe deux conventions pour appeler une option :

- Option courte (`-`) : composée d'une seule lettre. Elles peuvent être combinées. Exemple : `ls -l -h` peut s'écrire `ls -lh`.
- Option longue (`--`) : composée d'un mot complet, souvent plus explicite. Exemple : `cp -r` est identique à `cp --recursive`.

En milieu académique, les options longues sont privilégiées dans les scripts pour leur lisibilité, tandis que les options courtes sont préférées pour l'usage interactif rapide.

5.4. Le manuel : `man`

L'une des plus grandes forces de Linux est son auto-documentation. La commande `man` (pour *manual*) permet d'accéder à la documentation officielle de n'importe quelle commande installée.

Usage : `man [nom_de_la_commande]`

Exemple : `man cp` détaillera toutes les options disponibles pour la copie.

Navigation :
- Utilisez les flèches du clavier pour faire défiler le texte.
- Appuyez sur la touche `q` (quit) pour quitter le manuel et revenir au terminal.

> 🔥 **Rigueur**
>
> Le recours systématique au manuel (man) permet de comprendre précisément l'impact d'une commande sur le système. Copier-coller une instruction complexe trouvée en ligne sans en vérifier les options via le manuel expose l'administrateur à des effets de bord imprévus ou à des compromissions de sécurité.

5.5. Gestion des fichiers et navigation

- `ls` : liste les fichiers et dossiers du répertoire courant. Exemple : `ls -l` affiche les détails (permissions, taille, date).

- `cd [dossier]` : change le répertoire de travail. Exemple : `cd /home/dupont/Documents` ou `cd ..` pour remonter d'un niveau.
- `pwd` : affiche le chemin du répertoire de travail actuel. Exemple : `/home/dupont/projets`.
- `cp -R [source] [destination]` : copie récursivement un répertoire. Exemple : `cp -R ~/projets /media/sauvegarde/`.
- `mv [source] [destination]` : déplace ou renomme un élément. Exemple : `mv rapport.txt ~/Documents/archives/`.
- `mv -R [source] [destination]` : déplace un dossier et son contenu. Exemple : `mv -R ~/projets /media/sauvegarde/`.
- `mkdir [nom_dossier]` : crée un nouveau dossier. Exemple : `mkdir ~/NouveauProjet`.
- `rm [fichier]` : supprime un élément (utiliser `-r` pour un dossier).
- `cat [fichier]` : affiche le contenu d'un fichier texte. Exemple : `cat README.md`.
- `nano [fichier]` : éditeur de texte en ligne de commande. Exemple : `nano notes.txt` pour créer ou modifier un fichier.
- `grep [motif] [fichier]` : recherche un motif dans un fichier. Exemple : `grep "erreur" /var/log/syslog` pour trouver les lignes contenant « erreur » dans le journal système. Ou bien `grep -r "bidule" /Documents/` pour rechercher dans tous les fichiers du dossier.

5.6. Gestion des paquets avec APT

- `sudo apt update` : synchronise l'index local avec les dépôts.
- `sudo apt upgrade` : applique les mises à jour des paquets installés.
- `sudo apt install [paquet]` : installe un nouveau logiciel. Exemple : `sudo apt install vlc` pour installer le lecteur multimédia VLC.
- `sudo apt remove [paquet]` : désinstalle le logiciel.
- `sudo apt autoremove` : purge les dépendances inutiles.

> ⚠ **Un ordre à respecter**
>
> Lorsqu'on souhaite mettre à jour le système ou installer un nouvau logiciel via la ligne de commande, il est important de mettre à jour avant tout l'index des paquets disponibles pour s'assurer d'avoir accès à la dernière version. C'est pourquoi on commence par `sudo apt update` avant de lancer `sudo apt upgrade` ou `sudo apt install [paquet]`.

5.7. Surveillance du système

5.7.1. Avec `htop`

- **Installation :** `sudo apt install htop`
- **Utilisation :** Saisir `htop`. Permet de monitorer la charge système. Affiche les processus en cours, l'utilisation du CPU, de la mémoire et du swap. Les processus peuvent être triés par différentes colonnes (CPU%, MEM%, etc.) pour identifier les applications les plus gourmandes.

5.7.2. `ps -x` et `kill`

- `ps -x` : affiche les processus en cours d'exécution. Exemple : `ps -x` pour afficher tous les processus en cours d'exécution. `ps -x | grep firefox` pour trouver les processus liés à Firefox.
- `kill [PID]` : termine un processus identifié par son PID (Process ID). Exemple : `kill 12345` pour arrêter le processus avec le PID 12345. Utiliser `kill -9 [PID]` pour forcer l'arrêt si le processus ne répond pas.

5.8. A propos de Timeshift (à configurer plus tard)

Timeshift est un outil de sauvegarde et de restauration du système. Il permet de créer des instantanés (snapshots) du système à des points précis dans le temps, facilitant la restauration en cas de problème. Il est particulièrement utile avant d'appliquer des mises à jour majeures ou d'installer de nouveaux logiciels susceptibles de causer des conflits.

Lors de la première installation, Timeshift propose de configurer les paramètres de sauvegarde, notamment la fréquence des instantanés et les dossiers à inclure ou exclure. Il est recommandé de créer un instantané avant toute modification importante du système pour pouvoir revenir à une configuration stable en cas de besoin.

Cet instantané peut être restauré via l'interface de Timeshift ou en utilisant le mode de récupération du système en cas de défaillance critique.

Le stockage des instantanés peut consommer de l'espace disque mais cela reste raisonable si vous configurez correctement les paramètres de rétention et d'exclusion des dossiers volumineux (comme `/home` ou `/var`). Par défaut, Timeshift se concentre sur les fichiers système et les configurations, ce qui permet de limiter l'espace utilisé tout en assurant une restauration efficace du système.

> 🔥 **Inutile avec Timeshift**
>
> Timeshift n'est pas conçu pour sauvegarder les fichiers personnels (documents, images, etc.) mais plutôt pour les fichiers système et de configuration. Par conséquent, il est recommandé d'utiliser des outils de sauvegarde dédiés (ou sauvegarder « à la main » sur un disque externe) pour vos données personnelles, tandis que Timeshift se chargera de la restauration du système en cas de besoin. Cela permet d'optimiser l'espace de stockage utilisé par les instantanés tout en assurant une protection efficace du système.

- Il copie les fichiers système vers un dossier `/timeshift`.
- Il utilise des « liens en dur » (hard links) pour économiser de la place, mais le premier instantané prendra une place réelle équivalente à votre système (10 Go). Les instantanés suivants ne prendront que l'espace des fichiers modifiés depuis le dernier instantané, grâce à l'utilisation de liens en dur.
- Localisation : sur un disque unique, ces fichiers instantanés résident sur votre partition racine.

Généralement on stocke les instantanés sur une partition dédiée ou un second disque interne pour éviter de perdre ces données en cas de défaillance du disque. Cependant, il est important de s'assurer que le support de stockage est fiable et suffisamment rapide pour les opérations de sauvegarde et de restauration. Si vous devez stocker les instantanés sur la même partition que le système, gardez en tête qu'en cas de défaillance du disque vous perdrez ces informations. Cela dit, la réinstallation d'une distribution Linux prend peu de temps et vos usages ne sont pas si complexes que cela, donc ce n'est pas un drame en soi. De plus, si vous avez une bonne organisation de vos fichiers personnels (en les stockant dans des dossiers dédiés et en les sauvegardant régulièrement), vous pourrez facilement les restaurer après une réinstallation du système.