Commandes Linux

T'es pas encore assez matrixé ?

- [Commandes Linux] dd
- [Commandes Linux] Sudo
- [Commandes Linux] Grep / Awk / Cut / Sed
- [Commandes Linux] ls
- [Commandes Linux] Curl
- [Commandes Linux] Screen
- [Commandes Linux] Redimensionnement à chaud BTRFS
- [Commandes Linux] Script
- [Commandes Linux] du
- [Commandes Linux] Few
- [Commandes Linux] rtcwake
- [Commandes Linux] jq
- [Commandes Linux] Xan

[Commandes Linux] dd

Introduction

DD est un outil en ligne de commande qui permet d'écrire sur des périphériques.

Il est très utile pour flasher des clés USB à partir de fichier ISO ou pour faire l'opération inverse.

Attention : Cette commande est très dangereuse.

Utilisation

Syntaxe

```
dd if=<SOURCE> of=<CIBLE> bs=<BLOCKS_SIZE>
```

Flasher une clé USB

```
dd status=progress if=<SOURCE_FILE.iso> of=</dev/sdX> bs=4096
```

[Commandes Linux] Sudo

Introduction

Sudo est une commande linux permettant d'exécuter des commandes en tant que super-administrateur (root).



Syntaxe générale

Voici l'utilisation la plus basique de **sudo** :

```
sudo <COMMAND>
```

Et un exemple où l'on exécute une commande nécessitant les droits root :

```
sudo apt update
```

Remarque : Selon la configuration faite de **sudo**, le mot de passe de votre utilisateur peut vous être demandé lors l'utilisation de la commande **sudo**.

Remarque bis : La saisie de ce mot de passe ne sera pas affichée dans le terminal par sécurité.

Quelques options pratiques

- Ouvrir un shell avec les droits root :

```
sudo -i
```

- Lister les droits **sudo** de l'utilisateur actif :

```
sudo -l
```

- Exécuter une commande en tant qu'un autre utilisateur (potentiellement différent de root) :

```
sudo -u <USER> <COMMAND>
```

Installation et configuration

Afin de pouvoir utiliser **sudo**, il faut installer le paquet **sudo**.

Installation

- Debian / Ubuntu :

```
apt install sudo
```

- Fedora / CentOS / Alma Linux / Rocky Linux / RHEL :

```
dnf install sudo
```

- Arch Linux / Manjaro :

```
pacman -S sudo
```

Configuration

Pour donner les droits **sudo** à votre utilisateur, il suffit de l'ajouter au **groupe sudo** ou **wheel** (selon les systèmes) :

```
newgrp sudo
```

```
usermod -aG sudo <USER>
```

```
usermod -aG wheel <USER>
```

Cependant il peut être intéressant d'ajouter une entrée pour votre utilisateur dans le fichier **/etc/sudoers** :

```
<USER> ALL=(ALL:ALL) ALL
```

Pour ne plus demander le mot de passe de l'utilisateur lors de l'utilisation de **sudo**, remplacez l'entrez ci-dessus par celle-ci :

```
<USER> ALL=(ALL:ALL) NOPASSWD: ALL
```

Si vous souhaitez autoriser seulement l'exécution d'un programme spécifique à un utilisateur en tant que root :

```
<USER> ALL=(ALL) ALL: <PATH_TO_BIN>
```

Et voici sans demander le mot de passe de l'utilisateur :

```
<USER> ALL=(ALL) NOPASSWD: <PATH_TO_BIN>
```

[Commandes Linux] Grep / Awk / Cut / Sed

Introduction

Cette page montre l'utilisation d'outils en ligne de commande qui permettent de travailler avec les chaînes de caractères.

Grep

Permet de trouver un motif dans un fichier.

Syntaxe globale

```
grep [options] motif [fichier...]
```

Voici les options les plus courantes :

```
-i : Ignorer la casse (recherche insensible à la casse).
-r : Rechercher récursivement dans les répertoires.
-l : Afficher uniquement les noms de fichiers contenant des correspondances.
-v : Inverser la recherche (afficher les lignes qui ne contiennent pas le motif).
-n : Afficher le numéro de ligne de chaque correspondance.
-c : Afficher le nombre total de correspondances trouvées.
-A num : Afficher num lignes après chaque correspondance.
-B num : Afficher num lignes avant chaque correspondance.
-C num : Afficher num lignes avant et après chaque correspondance.
-e motif : Rechercher plusieurs motifs.
-f fichier : Lire les motifs à partir d'un fichier.
--exclude=pattern : Exclure les fichiers correspondant au modèle donné.
--include=pattern : Inclure uniquement les fichiers correspondant au modèle donné.
--color=auto : Surligner les correspondances en couleur (par défaut).
--color=never : Désactiver la coloration.
```

Exemples :

- Rechercher un motif dans un fichier :

```
grep "motif" fichier.txt
```

- Rechercher un motif dans plusieurs fichiers récursivement :

```
grep -r "motif" répertoire/
```

- Rechercher un motif en ignorant la casse :

```
grep -i "motif" fichier.txt
```

- Afficher uniquement les noms de fichiers contenant des correspondances :

```
grep -l "motif" fichier1.txt fichier2.txt
```

- Inverser la recherche pour afficher les lignes sans correspondances :

```
grep -v "motif" fichier.txt
```

- Afficher le numéro de ligne de chaque correspondance :

```
grep -n "motif" fichier.txt
```

- Rechercher plusieurs motifs :

```
grep -e "motif1" -e "motif2" fichier.txt
```

- Utiliser un fichier contenant des motifs à rechercher :

```
grep -f motifs.txt fichier.txt
```

- Exclure certains fichiers de la recherche :

```
grep "motif" --exclude=*.log répertoire/
```

- Activer/désactiver la coloration des correspondances :

```
grep --color=always "motif" fichier.txt
grep --color=never "motif" fichier.txt
```

Awk

Il s'agit d'un outil de traitement de texte principalement utilisé pour manipuler et analyser des données tabulaires.

Syntaxe globale

```
awk 'pattern { action }' fichier
```

Il existe deux principales utilisation **Awk** :

- **Pattern** : Condition pour appliquer une action.
- **Action** : Commandes à exécuter si le pattern est vrai.

Exemples

- Afficher une colonne spécifique d'un fichier CSV :

  ```
  awk -F ',' '{print $2}' fichier.csv
  ```

- Calculer la somme d'une colonne :

  ```
  awk '{sum += $1} END {print sum}' fichier.txt
  ```

- Filtrer les lignes qui correspondent à un motif :

  ```
  awk '/motif/ {print}' fichier.txt
  ```

- Afficher les lignes avec plus de 3 champs :

  ```
  awk 'NF > 3 {print}' fichier.txt
  ```

- Utiliser des opérations mathématiques :

  ```
  awk '{total += $1 * $2} END {print total}' fichier.txt
  ```

- Remplacer du texte dans une colonne :

  ```
  awk '{gsub("ancien", "nouveau", $3); print}' fichier.txt
  ```

- Regrouper et compter les occurrences :

  ```
  awk '{count[$1]++} END {for (item in count) print item, count[item]}' fichier.txt
  ```

- Utiliser des variables avec awk :

```
awk '{total += $1} END {average = total / NR; print "Moyenne :", average}'
fichier.txt
```

- Personnaliser le séparateur de champ (-F) :

```
awk -F ':' '{print $1}' fichier.txt
```

- Appliquer awk à plusieurs fichiers :

```
awk '{print FILENAME, $0}' fichier1.txt fichier2.txt
```

- Utiliser des conditions multiples :

```
awk '$1 > 50 && $2 == "A" {print}' fichier.txt
```

- Redirection de la sortie vers un fichier :

```
awk '{print $1, $2}' fichier.txt > sortie.txt
```

Cut

Cut st utilisée pour extraire des colonnes ou des champs de texte à partir de fichiers ou de données en entrée. Il est plus rapide et performant que awk et doit être utilisé, dans la mesure du possible, sur des grandes quantités de données.

Syntaxe globale

```
cut [OPTIONS] [FICHIERS]
```

Options courantes

```
-c LISTE : Extraire des caractères (par numéro de colonne).
-f LISTE : Extraire des champs (délimités par un séparateur).
-d CARACTERE : Spécifier le séparateur de champ (par défaut : tabulation).
-s : Ignorer les lignes sans délimiteur, ne pas les afficher.

Liste :
- Une liste de numéros de colonnes (pour l'option -c) ou de numéros de champs (pour l'option -
f), séparés par des virgules.
- Les plages de numéros de colonnes ou de champs sont spécifiées avec un tiret (par exemple,
1-3).
```

Exemples

- Extraire une colonne spécifique d'un fichier CSV (séparé par des virgules) :

```
cut -d ',' -f 2 fichier.csv
```

- Extraire une colonne spécifique d'un fichier TSV (séparé par des tabulations) :

```
cut -f 3 fichier.tsv
```

- Extraire plusieurs colonnes d'un fichier CSV :

```
cut -d ',' -f 1,3,5 fichier.csv
```

- Extraire une plage de caractères d'une ligne de texte :

```
echo "ABCDEFG" | cut -c 2-4
```

- Ignorer les lignes sans délimiteur :

```
cut -d ',' -f 2,3 -s fichier.csv
```

- Extraire une colonne spécifique d'un fichier avec des espaces comme séparateur :

```
cut -d ' ' -f 2 fichier.txt
```

- Extraire plusieurs plages de caractères d'une ligne de texte :

```
echo "12345ABC67890" | cut -c 1-5,7-9
```

- Extraire tous les champs d'un fichier de configuration (séparés par des espaces ou des tabulations) :

```
cut -d ' ' -f - fichier.conf
```

Sed

C'est un outil puissant de traitement de texte en ligne de commande.

Syntaxe globale

```
sed [OPTIONS] 'commande' [FICHIERS]
```

Options courantes

```
-e 'script' : Ajouter un script de commande à exécuter.
-i[SUFFIXE] : Modifier le fichier en place (crée une sauvegarde avec un SUFFIXE si spécifié).
-n : Désactiver la sortie automatique, nécessite l'utilisation de p (impression explicite).
-r : Activer les expressions régulières étendues (compatibles avec les regex modernes).
```

Commandes courantes

```
- s/regex/remplacement/g : Remplacer les occurrences de regex par remplacement (g pour global, sinon seulement la première occurrence).
- p : Imprimer la ligne courante.
- d : Supprimer la ligne courante.
- 1,10s/regex/remplacement/g : Appliquer la commande de remplacement uniquement aux lignes 1 à 10.
- /motif/cmd : Appliquer la commande cmd aux lignes contenant le motif.
- !cmd : Appliquer la commande cmd à toutes les lignes sauf celles spécifiées.
- a\ : Ajouter une ligne après la ligne courante.
- i\ : Insérer une ligne avant la ligne courante.
- c\ : Remplacer la ligne courante par une nouvelle ligne.
```

Exemples

- Remplacer toutes les occurrences d'un mot dans un fichier :

  ```
  sed 's/motif/remplacement/g' fichier.txt
  ```

- Supprimer les lignes vides d'un fichier :

  ```
  sed '/^$/d' fichier.txt
  ```

- Remplacer une ligne spécifique par du texte dans un fichier :

  ```
  sed '7c\Nouveau texte' fichier.txt
  ```

- Ajouter du texte à la fin de chaque ligne :

```
sed 's/$/ Texte à ajouter/' fichier.txt
```

- Imprimer uniquement les lignes contenant un motif :

```
sed -n '/motif/p' fichier.txt
```

- Supprimer la dernière ligne d'un fichier :

```
sed '$d' fichier.txt
```

- Modifier un fichier en place avec sauvegarde :

```
sed -i.bak 's/motif/remplacement/g' fichier.txt
```

- Utiliser des expressions régulières étendues avec -r :

```
sed -r 's/(motif1|motif2)/remplacement/g' fichier.txt
```

- Ajouter une ligne avant chaque ligne contenant un motif :

```
sed '/motif/i\Nouvelle ligne' fichier.txt
```

- Supprimer toutes les lignes sauf celles contenant un motif :

```
sed -n '/motif/p; /motif2/p' fichier.txt
```

[Commandes Linux] ls

Introduction

La commande **ls** permet de lister ou plutôt d'afficher les fichiers avec leurs différentes informations sur les systèmes Unix.

Elle dispose d'une multitude d'options intéressantes.

Manuel

Syntaxe globale

```
ls [OPTION] [FILE|DIRECTORY]
```

Afficher le dossier courant

```
ls
```

Afficher sous forme de liste

```
ls -l
```

Afficher les fichiers et dossiers cachés

```
ls -la
```

Afficher avec un format de taille de fichier lisible

```
ls -lh
```

Afficher la date de création d'un fichier ou d'un dossier

```
ls --time=creation <FILE|DIR>
```

[Commandes Linux] Curl

Introduction

Curl est un outil en ligne de commande qui permet d'effectuer des requêtes HTTP. Il est très souvent utilisé dans les scripts.



Manuel

Syntaxe globale

```
curl <URL>
```

> Sans option, curl affiche le code source de la page.

Options

Options	Descriptions
-O	Téléchargement d'un fichier en utilisant le nom distant.
-o <output>	Téléchargement d'un fichier en utilisant un nom spécifique.
-X <GET|POST> -d "<PARAM1>=<VALUE1>;<PARAM2>=<VALUE2>"	Envoi de données via une requête GET ou POST.

-X OPTIONS	Permet d'afficher les méthodes disponibles.
-L	Suit les redirections.
-k	Ignore les vérifications SSL.
-#	Affichage de la progression du téléchargement (à combiner avec le -O).
-s	Active le mode silencieux (verbeux par défaut).
-h "<HEADER>"	Ajoute un header.
-I	Affiche la bannière.
-T <FILE>	Permet d'upload un fichier (si autorisé).
-sSk <LINK> | bash	Télécharge un script et l'exécute directement.

Afficher son IP publique

```
curl -4 ifconfig.me
```

[Commandes Linux] Screen

Introduction

La commande **screen** sur Linux permet de créer des sessions et ainsi de pouvoir laisser tourner des tâches en arrière plan.

Manuel

Créer une session

```
screen -S <NAME>
```

Lister les sessions

```
screen -ls
```

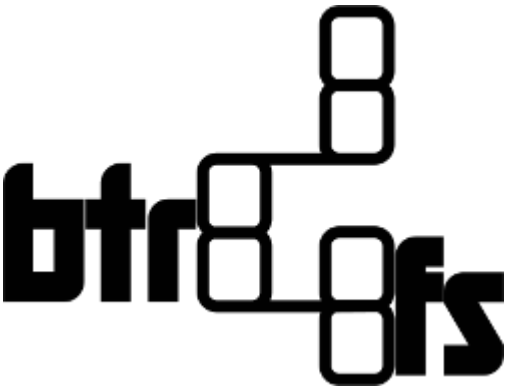
Se rattacher à une session existante

```
screen -r <NAME>
```

[Commandes Linux] Redimensionnement à chaud BTRFS

Introduction

Il est parfois utile de savoir redimensionner à chaud les partitions. Un système de fichiers qui le supporte bien est BTRFS



Manuel

Si parted n'est pas installé sur votre système :

```
apt install -y parted
```

Un fois avoir ajouté de l'espace sur le disque virtuel, redimensionnez la partition avec **parted** :

```
parted /dev/sda resizepart 2 100%
```

Si vous souhaitez le scripter, voici la commande qui ne vous demandera pas de confirmation :

```
echo Yes | parted /dev/sda ---pretend-input-tty resizepart 2 100%
```

Ensuite, lancez la commande suivante :

```
btrfs filesystem resize max /
```

[Commandes Linux] Script

Introduction

Cette commande permet d'enregistrer les commandes et la sortie de ces commandes directement dans un fichier.

Manuel

Syntaxe globale

```
script -a <OUTPUT>
```

> Puis exécutez la commande **exit** (ou **CTRL+D**) pour arrêter l'enregistrement.

[Commandes Linux] du

Introduction

La commande **du** permet d'afficher la taille d'un dossier sous Linux

Cheat-sheet

Syntaxe globale

```
du <DIR>
```

Afficher en format plus lisible

```
du -h <DIR>
```

Afficher la somme des tailles des fichiers

```
du -sh <DIR>
```

Afficher les plus gros dossiers

```
du -h --max-depth=1 /home/ | sort -rh
```

> Ici, les plus gros dossiers présents dans le **/home** seront affichés dans l'ordre.

[Commandes Linux] Few

Introduction

Cette page donne des outils qui vont changer votre vie au quotidien sur Linux dans votre shell.

Yazi

Mieux que **Ranger**, cet outil vous permettra de naviguer à travers vos fichiers comme dans un explorateur.

Vous pourrez avoir une prévisualisation des fichiers et même effectuer des opérations basiques dessus.

- https://github.com/sxyazi/yazi

Bat

Cet outil est un fork de **Cat** mais qui ajoute la coloration syntaxique.

fzf

Permet de rechercher des fichiers de manière dynamique et puissante (oubliez find).

Pour chercher un mot :

```
fzf -q <WORD>
```

- https://github.com/junegunn/fzf

> Ajoutez l'option **-e** pour chercher seulement les occurences exactes !

Attuin

Permet de chercher de manière dynamique dans votre historique de commande et même de synchroniser celui-ci avec un serveur :

- https://github.com/atuinsh/atuin

Dust

Comparable à l'analyseur de disque, vous allez mettre **df** et **du** à la poubelle :

- https://github.com/atuinsh/atuin

> Utilisez l'option **-b** pour ne pas afficher le graphique.

tldr

Cette commande affiche une sorte de manuelle en résumé et en couleur avec des exemples d'utilisation pour votre commande :

```
tldr <CMD>
```

eza

Une bonne alternative à **ls** mais en couleur et des options :)

[Commandes Linux] rtcwake

Introduction

La commande rtcwake permet d'éteindre ou de mettre en veille un système Linux pour une durée déterminée.

Manuel

Extinction

```
rtcwake -m off -s <TIME>
```

> Le temps saisis doit être un nombre en **secondes**.

Mise en veille

```
rtcwake -m mem -s <TIME>
```

Hibernation

```
rtcwake -m disk -s <TIME>
```

[Commandes Linux] jq

Introduction

Cette commande permet de parser du json et d'afficher le résultat en couleur ce qui est très pratique pour l'analyse.

Manuel

Utilisation basique

```
cat myfile.json | jq
```

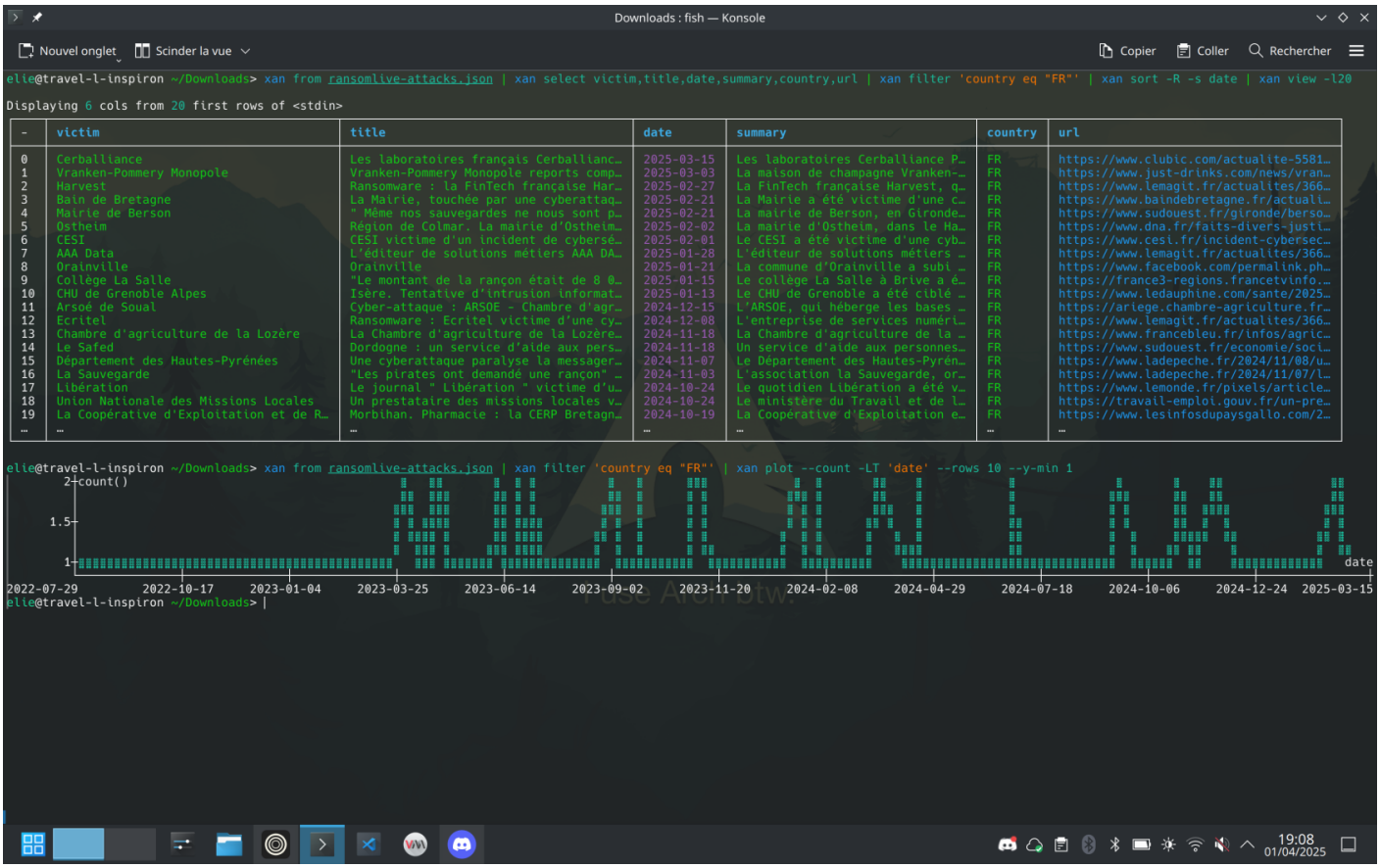
Chercher un champ spécifique

```
jq '.[] | select(.<FIELD> | test("<VALUE>"; "i"))' myfile.json
```

[Commandes Linux] Xan

Introduction

Cet outil permet de travailler avec les fichiers CSV en ligne de commande. En plus il est développé en Rust ce qui le rend beaucoup plus performant qu'un excel, ou libre office.



Manuel

Documentations

La doc est vraiment complète :

- https://github.com/medialab/xan/tree/master/docs/cmd
- https://korben.info/xan-couteau-suisse-manipulation-csv-ligne-commande.html

View

Pour afficher un fichier csv basiquement :

```
xan view file.csv
```

Headers

Pour afficher les noms des colonnes :

```
xan headers
```

Count

Pour savoir le nombre d'entrées :

```
xan count
```

Select

```
xan select col1,col3,col5
```

Sort

Pour trier sur un colonne (par exemple les dates) :

```
xan sort -R -s 'date'
```

> Dans cet exemple, on tri dans l'ordre décroissant avec le **-R** sur la colonne **date** .

Filter

```
xan filter 'country eq "FR"'
```

> Vous pouvez consulter la documentation des expressions avec **xan help cheatsheet** .

Agg

Pour effectuer une aggrégation sur une colonne comme une somme des valeurs :

```
xan agg 'sum(Revenu) as total_revenus'
```

Groupby

Il s'agit d'une alternative à Agg qui permet d'effectuer une expression (comme une somme) pour chaque valeur d'une autre colonne.

Voici un exemple où on calcul le total des revenus pour chaque entreprise :

```
xan groupby 'Entreprise' 'sum(Revenu) as total_revenus'
```

From

Pour convertir d'autre formats de fichiers en CSV notamment du **json** ou **jsonl** :

```
xan from -f <FORMAT> <FILE>.json
```

Plot

Permet de générer des graphiques :

```
xan plot -LT 'date' 'revenu' --rows 10 --y-min 1 -c 'entreprise'
```

Autre exemple si vous souhaitez afficher le nombre d'occurence d'un champ dans le temps (si vous n'avez pas de Y) :

```
xan plot --count -LT 'date' --rows 10 --y-min 1
```