La ligne de commande

- Conventions dans la documentation
- Memo en vrac
- SSH
    - Utilisation d'une YubiKey comme clé SSH avec FIDO2
- Screen
- Raccourcis claviers

Conventions dans la documentation

Convention des Symboles `$` et `#` dans les Commandes Linux

Introduction

Dans les documentations, tutoriels, et exemples de commandes Linux, vous verrez souvent les symboles `$` et `#` au début des lignes de commande. Ces symboles ne font pas partie de la commande elle-même, mais servent à indiquer dans quel contexte la commande doit être exécutée, que ce soit en tant qu'utilisateur normal ou administrateur (root).

Symbole `$`

Le symbole `$` est utilisé pour représenter l'invite de commande d'un **utilisateur non-root** (c'est-à-dire un utilisateur sans privilèges administratifs). Cela signifie que la commande peut être exécutée directement par un utilisateur classique.

Exemple :

```
$ ls -l
```

Dans cet exemple :

- `$` indique que la commande `ls -l` est exécutée par un utilisateur normal.
- La sortie de la commande (le résultat) apparaîtra après l'exécution.

Symbole `#`

Le symbole `#`, quant à lui, est utilisé pour représenter l'invite de commande d'un **superutilisateur (root)** ou lorsque la commande nécessite des privilèges administratifs. Si vous voyez une commande précédée de `#`, cela signifie que vous devez l'exécuter avec des privilèges élevés, généralement en utilisant `sudo` (si vous êtes un utilisateur non-root).

Exemple :

```
# apt update
```

Dans cet exemple :

- `#` signifie que la commande `apt update` nécessite des privilèges root pour s'exécuter correctement.
- Vous devrez soit vous connecter en tant que root, soit précéder la commande de `sudo` :

  ```
  $ sudo apt update
  ```

Résumé

- `$` : Invite de commande d'un utilisateur non-administrateur. Exécutez les commandes directement.
- `#` : Invite de commande pour un superutilisateur (root). Exécutez les commandes avec `sudo` si vous n'êtes pas root.

Exemple Comparatif :

```
$ lsmod | grep 8021q      # Commande exécutée par un utilisateur normal
# systemctl restart nginx  # Commande nécessitant des privilèges root
```

Remarque

Le symbole `$` ou `#` n'est **pas** à taper lors de l'exécution des commandes, il est uniquement là pour indiquer le contexte d'exécution.

Conclusion

Ces conventions sont très courantes dans les environnements Unix/Linux et aident à comprendre rapidement si une commande peut être exécutée par un utilisateur classique ou si elle nécessite des droits d'administrateur.

Memo en vrac

Pour quitter une session Telnet :
`Ctrl + $`

Pour afficher la version du système (Ubuntu):

```
lsb_release -a
```

Pour obtenir des informations sur le matériel :

```
sudo lshw
```

Pour obtenir des informations sur des composants précis :

```
sudo lshw -C cpu
sudo lshw -C memory
sudo lshw -C network
sudo lshw -C disk
```

SSH

SSH

Utilisation d'une YubiKey comme clé SSH avec FIDO2

Prérequis

Vérifier que OpenSSH 8.2 ou supérieur est installé sur le client et le serveur.

```
ssh -V
```

Vérifier la compatibilité de la YubiKey

Deux types de paires de clés sont compatibles avec FIDO2 : `ecdsa-sk` et `ed25519-sk`.

- L'extension -sk signifie "security key".
- Le type `ed25519-sk` nécessite une YubiKey avec le firmware 5.2.3 ou supérieur.

> Les clés `ed25519-sk` nécessitent une YubiKey avec un **firmware en version 5.2.3 ou supérieure**, compatible FIDO2.

Pour afficher la version du firmware de la YubiKey :

```
lsusb -v 2>/dev/null | grep -A2 Yubico | grep "bcdDevice" | awk '{print $2}'
```

Installer les dépendances nécessaires

```
sudo apt install libfido2-dev
```

Générer une paire de clés SSH avec FIDO2

Créer une clé SSH sécurisée par YubiKey :

```
ssh-keygen -t ed25519-sk -C "$(hostname)-$(date +'%Y%m%d')-yubikey1"
```

Une partie de la clé privée est stockée dans la YubiKey. Elle repose sur un mécanisme de défi-réponse sécurisé. La clé publique, un handle et des données d’attestation sont générés.

Copier la clé publique sur un serveur

Sous Linux :

```
ssh-copy-id -i ~/.ssh/id_ed25519-sk.pub user@serveur
```

Sous Windows (PowerShell) :

```
type $env:USERPROFILE.ssh\id_ed25519-sk.pub | ssh user@serveur "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

Remplacer user@serveur par les informations du serveur cible.

Sources

https://forums.lawrencesystems.com/t/ssh-with-yubikey-fido-u2f-authentication/13024

https://developers.yubico.com/SSH/Securing_SSH_with_FIDO2.html

Screen

La commande screen permet de gérer des sessions terminal détachables et persistantes. Utile pour lancer des processus longs sur un serveur distant et se déconnecter sans interrompre leur exécution.

Commandes de base

- Lancer une session
  screen
  ou avec un nom pour la session :
  screen -S nom_session
- Détacher une session
  Ctrl + a, puis d
- Lister les sessions existantes
  screen -ls
- Reprendre une session
  screen -r nom_session
  ou, si plusieurs sessions sont ouvertes :
  screen -r ID_session
- Fermer une session
  Dans le terminal de la session :
  exit
  Ou terminer une session détachée :
  screen -X -S nom_session quit

Commandes internes utiles

> (toutes commencent par Ctrl + a)

c : Créer une nouvelle fenêtre
n : Aller à la fenêtre suivante
p : Aller à la fenêtre précédente
" : Liste des fenêtres
k : Fermer la fenêtre active
A : Renommer la fenêtre active

Astuces

Démarrer un processus dans une session détachée directement :
screen -dmS nom_session commande

Reconnecter une session même si elle est attachée ailleurs :
screen -x nom_session

Personnaliser le fichier de configuration : ~/.screenrc

Raccourcis claviers

- `Ctrl + A` : aller au début de la ligne
- `Ctrl + E` : aller en fin de ligne
- `Ctrl + L` : effacer le contenu de l'écran (équivalent de `clear`)
- `Ctrl + U` : effacer la ligne jusqu'au début
- `Ctrl + K` : effacer la ligne jusqu'à la fin