SSH : Un Shell à Distance

Table des matières

Objectifs

Rappels de réseau

Découvrir le principe de SSH

Apprendre à se connecter à distance à un ordinateur

Apprendre à copier et éditer des fichiers à distance

Mettre en place les bonnes pratiques SSH

I Principe de fonctionnement des adresses IP

Une adresse IP (avec IP pour Internet Protocol) est un numéro d'identification qui est attribué de façon permanente ou provisoire à chaque appareil connecté à un réseau informatique utilisant l'Internet Protocol. L'adresse IP est à la base du système d'acheminement des messages sur Internet.

Il existe des adresses IP de version 4 (sur 32 bits, soit 4 octets) et de version 6 (sur 128 bits, soit 16 octets). La version 4 est actuellement la plus utilisée : elle est généralement représentée en notation décimale avec quatre nombres compris entre 0 et 255, séparés par des points. On aura donc des IPv4 qui ressemble à 172.31.128.1 et des IPv6 qui ressemblent à 2001:0db8:0000:85a3:0000:0000:ac1f:8001.

Les adresses IPv4 disponibles étant presque épuisées, les opérateurs incitent à la transition d'IPv4 vers IPv6.

Remarque

La plupart des adresses IP des serveurs peuvent être converties en un nom de domaine et inversement. Le nom de domaine est plus facilement lisible : pic.crzt.fr est le nom de domaine correspondant à 80.67.182.78.

Remarque

Chaque appareil connecté au réseau (pas seulement les ordinateurs) possède une adresse IP. D'ailleurs, si vous avez une carte wifi et une carte ethernet dans votre ordinateur, elles ont toutes deux une adresse différente étant donné qu'il s'agit de composant différent.

IP publique Définition

Une adresse IP publique est une adresse dite "tournée vers l'extérieur". C'est l'adresse qui est visible par les autres appareils lorsque vous êtes connecté à internet et qui permet aux appareils réseaux de communiquer entre eux. Cette adresse est unique et sert d'identifiant.

IP privée Définition

Une adresse IP privée est une adresse dite "tournée vers l'intérieur". Cette adresse permet de se connecter à un réseau local (donc pas sur internet). Elle n'est pas unique entre plusieurs réseaux locaux. Une IP privé se reconnait facilement par ses premiers chiffres : 10.x.x.x, 172.x.x.x et 192.168.x.x en général.

IP et serveur

Az Définition

Afin d'accéder à un serveur il faut donc au préalable connaître l'adresse IP publique de ce serveur (ou son nom de domaine). On pourra ainsi s'y connecter via le protocole SSH.

II SSH : Secure Shell

Objectif

- Se familiariser avec le protocole SSH (Secure SHell).

Mise en situation

SSH est une application d'Internet qui n'est pas aussi connue que le Web ou le mail.

Mais elle est très utile dans le monde du développement informatique car elle permet de se connecter à un serveur à distance, puis d'exécuter des commandes sur ce serveur. C'est aujourd'hui le mode de contrôle privilégié des informaticiens sur les serveurs qui font fonctionner Internet.

Terminal et shell

Rappel

Le **terminal** (aussi appelé invite de commande ou console) est le moyen le plus naturel pour interagir avec un ordinateur. Ce programme offre un **shell** (une interface système) donnant accès aux programmes de l'ordinateur. Il suffit d'un écran et d'un clavier pour utiliser ce shell.

- *Terminal et shell* (cf. p.18)

Le programme SSH (Secure SHell)

Az Définition

Le programme SSH (*Secure Shell*) permet d'interagir de manière sécurisée avec un ordinateur distant via un shell. C'est le programme de référence pour effectuer des opérations à distance.

- Toutes les commandes tapées depuis un clavier d'ordinateur à un emplacement A sont exécutées dans le shell d'un ordinateur à un emplacement B.
- Les informations qui transitent sur Internet via SSH sont les chaînes de caractères représentant les commandes à exécuter et les chaînes de caractères représentant les résultats de ces exécutions.

Le protocole SSH

Az Définition

SSH est aussi le nom du protocole de communication utilisé par le programme SSH. Ce protocole fonctionne sur une **architecture client-serveur** ; l'ordinateur qui fournit les commandes est le **client** et l'ordinateur qui exécute les commandes sur son système est le **serveur**.

SSH repose généralement sur TCP pour le transport et est par défaut associé au port **22**.

L'établissement d'une connexion SSH se fait en deux étapes :

- l'établissement d'une communication sécurisée,
- l'authentification du client.

Établir une connexion SSH

Méthode

La commande `ssh` se lance depuis un terminal sur votre machine (ou Putty[1] sous Windows) :

```
ssh root@80.67.182.1xx
```

avec 80.67.182.1xx l'IP de votre VPS et `root` l 'utilisateur avec qui vous connecter.

Lors de la première connexion, la commande vous demandera de valider si ce serveur est le bon (pour des raisons de sécurité que nous n'aborderons pas).

Ensuite, vous devrez rentrer votre mot de passe. Par défaut, le mot de passe `root` est `root`.

Attention

Cette combinaison de login/mot-de-passe n'est vraiment pas une bonne pratique. Nous corrigerons cela dans un prochain module.

Remarque

Si un serveur est associé à un nom de domaine, on peut utiliser ce domaine à la place de l'IP.

```
ssh root@1xx.picagraine.net
```

Chiffrement symétrique et chiffrement asymétrique

Complément

Le **chiffrement symétrique** repose sur le chiffrement et le déchiffrement de messages par une seule clé appelée **clé symétrique**. Ainsi, si deux ordinateurs veulent protéger leurs communications, ils peuvent se mettre d'accord sur une clé symétrique à utiliser et veiller à ce qu'ils soient les seuls à la détenir. Le chiffrement symétrique est rapide mais nécessite un canal sécurisé pour échanger la clé.

Le **chiffrement asymétrique** repose sur le chiffrement et le déchiffrement de messages par deux clés différentes (l'une chiffre les messages et l'autre déchiffre). Dans ce système, une clé est connue de tous : on parle de **clé publique** ; l'autre clé est gardée secrètement : on parle de **clé privée**. Le chiffrement asymétrique est plus lent mais ne nécessite par de canal sécurisé préalable.

Étape 1 : Établissement d'une communication sécurisée

Complément

Au début de la session et avant de commencer toute interaction entre le client et le serveur, c'est-à-dire avant même que le serveur vérifie que le client est légitime, les deux ordinateurs doivent se mettre d'accord sur un moyen de sécuriser leur communication.

Pour cela, le client et le serveur se mettent d'accord via le protocole de transport (TCP) sur la version du protocole SSH à utiliser et sur une méthode de **chiffrement symétrique** de leur communication.

1. https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Les deux ordinateurs vont utiliser des méthodes de chiffrement asymétrique pour échanger une clé symétrique. Cette clé leur permettra donc de protéger toutes les communications suivantes de la session.

À la fin de cette étape, une connexion sécurisée est établie entre le client et le serveur.

Étape 2 : Authentification de l'utilisateur côté client

Complément

Une fois une connexion sécurisée établie, le serveur doit **authentifier** l'utilisateur qui cherche à se connecter. Il existe deux méthodes d'authentification :

- **Méthode 1 : Par mot de passe**

  Puisque le but est ici d'ouvrir un shell à distance, le client cherche à se connecter à un utilisateur de la machine distante. Le serveur peut ainsi demander au client le mot de passe de l'utilisateur via lequel il veut se connecter.

- **Méthode 2 : Par clé**

  Chaque utilisateur présent sur le serveur possède une liste de clés publiques de clients de confiance, présentes dans un fichier `authorized_keys`.

  1. Le client envoie une des clés publiques de l'utilisateur avec lequel il veut se connecter.
  2. Le serveur vérifie le fichier `authorized_keys` de l'utilisateur pour s'assurer que cette clé publique existe.
  3. Si oui, le serveur chiffre un nombre aléatoire avec ladite clé publique et envoie le résultat au client.
  4. Le client utilise sa clé privée pour déchiffrer le nombre et y appliquer un certain traitement avant de le renvoyer au serveur.
  5. Le serveur applique le même traitement au nombre qu'il a envoyé et vérifie que le résultat du client est le même que le sien.
  6. Si le résultat est correct, et comme l'utilisateur est le seul à posséder la clé privée et déchiffrer le nombre envoyé par le serveur, il est authentifié.

Complément

La méthode d'authentification par clé nécessite qu'une paire de clés publique/privée soit générée au préalable, et que le serveur ait stocké la clé publique du client dans le fichier `authorized_keys` de l'utilisateur.

Installer un serveur SSH sur un serveur

Complément

SSH était déjà installé sur le VPS. Si cela n'avais pas été le cas ou que vous vouliez tester SSH sur vos machines (Linux) personnelles, il suffit de rentrer les commandes suivantes :

```
1 sudo apt update
2 sudo apt install openssh-server
3 sudo systemctl start ssh
```

À retenir

- Le protocole SSH permet à un client d'ouvrir un shell sur un serveur distant.
- Ce protocole sécurise les communications en employant plusieurs méthodes de chiffrement, de l'authentification au transfert de données.

III Exercice : Se Mettre à l'Aise Chez Soi

Question

[solution n°1 p. 17]

- Regarder le nom de votre machine : `hostname`
- Regarder qui vous êtes : `whoami`
- Regarder où on est : pwd
- Regarder quel est le système installé : `lsb_release -a`
- Aller dans le dossier */tmp* : `cd /tmp`
- Créer un fichier avec votre lieu de naissance : `nano doujeviens`
- Regarder ses processus : `top`
- Regarder l'état de son disque : `df -h`
- Regarder qui est connecté au serveur : `who`
- Regarder l'état du service (*daemon*) SSH : `systemctl status ssh`
- Regarder les logs du service SSH :
  `journalctl -u ssh`
  `journalctl -n 20 -u ssh`
- Trouver votre IP : `curl https://ifconfig.me && echo`

IV Copie et Édition de Fichiers Par SSH

1. Copie de Fichiers

En plus de pouvoir se connecter à distance avec `ssh login@host` il est également possible de copier des fichiers depuis et vers le serveur grâce a un dérivé de SSH : `scp`. Cette commande prend la source et destination comme parametre soit

```
scp fichierlocal login@host:chemindistant
```

pour envoyer un fichier et

```
scp login@host:chemindistant fichierlocal
```

pour récupérer un fichier

> **Copie de Dossiers** — Remarque
>
> `scp` peut également copier des dossiers et leurs contenus. Auquel cas il ajouter `-r` à la commande.

Copie Intelligente

`scp` est limitée dans ce qu'elle peut faire. Si vous copiez un dossier lourd avec beaucoup de fichiers déjà présents de l'autre côté, tous seront envoyés. Il existe une autre commande qui permet de faire cela mais qui est plus complexe : `rsync`. Cette nouvelle commande va vérifier la présence et le *checksum* des fichiers avant de les envoyer. Sa syntaxe est proche de `scp` mais nécessite plus d'options :

```
rsync -PIavz source destination
```

Ici, pas besoin de `-r` pour copier des dossiers.

2. Édition de Fichiers à Distance

Nous avons jusqu'à maintenant deux manières d'éditer le contenu des fichiers sur le serveur :

- Se connecter en SSH et éditer avec `nano` (ou autre éditeur préféré) directement depuis SSH
- Modifier les fichiers en local puis les pousser avec `scp` ou `rsync`

Une autre méthode combine les meilleur des deux monde : éditer sur votre machine et répercuter automatiquement les modifications sur votre VPS : `sshfs`.

`sshfs` est une commande à lancer sur votre machine personnelle comme suit

```
sshfs server:path localpath
```

ce qui aura pour conséquence de *monter* le contenu de `server:path` dans le dossier `localpath`. Vous pourrez alors éditer le contenu de `localpath` sur votre machine personnelle et `ssh` se chargera de synchroniser automatiquement sur le VPS.

Pour démonter le dossier `localpath` (ce qui est une bonne idée quand vous avez fini une session d'édition), il faut utiliser la commande

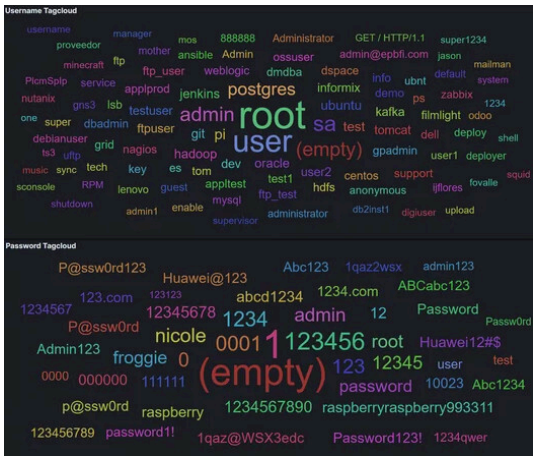
```
fusermount -u localpath
```

V Bonnes Pratiques de SSH

> ⚠ Attention
>
> Exposer un ordinateur sur l'internet public n'est pas sans danger. Des scripts tournent en permanence pour scanner les vulnérabilités de toutes les IP existantes. SSH est à ce titre une porte d'entrée privilégié car les configurations par défaut sont les mêmes chez tout le monde. Dans les prochaines parties, nous allons apprendre à nous protéger de 99% de ces attaques.

1. Arrêter de se Connecter en Root

Pour commencer, c'est une mauvaise idée de laisser `root`, le super-administrateur se connecter directement depuis l'extérieur, d'autant plus avec un mot de passe aussi faible. En effet, l'image suivant montre les combinaisons de login/mot-de-passe les plus brut-forcées.



On préférera donc d'abord se connecter avec un utilisateur normal puis se connecter avec le super utilisateur. Pour cela, nous aurons besoin de quatre étapes :

1. Changer le mot de passe `root` pour quelque chose de plus robutes
2. Créer un utilisateur normal
3. Désactiver les connexions `root`
4. Apprendre à se rendre temporairement `root`

1.1. Changement de Mot de Passe

Pour changer le mot de passe du compte courant, on utilisera la commande suivante :

```
1 passwd
```

1.2. Création d'un nouvel utilisateur

Pour créer un nouvel utilisateur, nous utiliserons la commande suivante qui vous posera quelques questions (mot de passe, nom complet, etc) :

```
1 adduser <VOTRENOM>
```

Ensuite, prenons un peu d'avance sur le point prochain et ajoutons notre nouvel utilisateur au groupe `sudo`, c'est-à-dire ceux capables de se rendre administrateurs :

```
1 usermod -aG sudo <VOTRENOM>
```

Déconnectez vous de cette session et reconnectez vous avec le nouvel utilisateur.

1.3. Se rendre root

Rendez-vous désormais `root` grâce à la commande suivante :

```
1 sudo su
```

Ce qui aura pour effet de vous connecter en tant que `root`. Le mieux est encore de ne se rendre root que quand cela est nécessaire en faisant.

```
1 sudo <COMMANDE>
```

1.4. Empêcher les connexions root

Finalement, pour désactiver les connexion `root`, il faudra éditer le fichier de configuration du démon SSH avec la commande suivante. Cette fois ci, cherchez par vous même quelle ligne éditer afin de désactiver les connexion `root`.

```
1 sudoedit /etc/ssh/sshd_config
```

On pourra ensuite relancer le démon SSH pour prendre en compte cette modification :

```
1 sudo systemctl restart sshd
```

2. Connexion par Clé

Jusqu'alors, nous nous somme connecté en rentrant notre mot de passe à chaque fois. En plus d'être rapidement fastidieux, la connexion par mot de passe n'est pas le plus sécurisé.

Pour faire d'une pierre deux coups, nous allons mettre en place la *connexion par clé*, c'est-à-dire que notre authentification se fera désormais à l'aide d'un couple clé privée - clé publique. Le serveur connaîtra notre clé publique qui, étant liée mathématiquement à notre clé privée, nous identifiera de manière sure sans jamais avoir à quitter notre machine.

Pour mettre en place une identification par clé, il suffit de nous créer une paire de clés sur votre machine (et non VPS) avec

```
1 ssh-keygen
```

qui après nous avoir posé quelques questions générera une paire `~/.ssh/id_rsa` et `~/.ssh/id_rsa.pub`. Pour plus de sécurité on peut associer la clé à un mot de passe. La première est la clé privée à ne jamais dévoiler, tandis que la `.pub` peut et doit l'être. Pour envoyer notre clé publique sur le VPS :

```
1 ssh-copy-id login@host
```

Il n'y aura, par la suite plus jamais à rentrer de mot de passe pour s'authentifier.

> 💬 Remarque
>
> Dans cette API nous n'aurons besoin que d'une seule clé, même pour plusieurs VPS : un seul `ssh-keygen` et autant de `ssh-copy-id` que nécessaire. D'autres cas d'usage pourraient nécessiter plusieurs clés (une personnelle, professionnelle, associative, etc) mais cela dépasse le cadre de cette API.

Empêcher les Connexions par Mot-de-Passe

Maintenant que nous pouvons nous connecter par clé, on peut désactiver la connexion par mot de passe afin d'empêcher toutes tentatives de brut-force. Pour cela, nous allons modifier la configuration du démon SSH :

```
1 sudoedit /etc/ssh/sshd_config
```

et dé-commenter les lignes

```
#PubkeyAuthentication yes
#PasswordAuthentication no
```

> ⚠ Attention
>
> Ne perdez jamais votre clé ou vous ne pourrez plus vous connecter à cette machine. Vous ne pourrez également vous connecter que depuis une machine ayant le bon jeu de clé. Il existe des options pour autoriser la connexion par mot de passe depuis certaines adresses IP mais nous ne couvrirons pas cet aspect ici.

3. Changer le Port SSH

> ⚠ Attention
>
> Cette partie est très importante à faire dans des cas réels, mais le fonctionnement du réseau à l'UTC nous empêche de la réaliser.
>
> La partie n'est donc pas à faire pour cette API, mais tâchez de la garder à l'esprit si vous le refaites dans un autre cadre.

Le port SSH par défaut est 22, et ce sera le principal point d'attaque de la plupart des scripts. En bougeant simplement notre port SSH vert un autre port arbitraire, on se protège de la majorité des attaques automatisées. Pour cela, il faudra changer la configuration du démon SSH et changer la manière dont on se connecte au VPS.

Changement de la Configuration SSH sur le VPS

Nous allons retourner changer la configuration du démon avec :

```
sudoedit /etc/ssh/sshd_config
```

et éditer la ligne

```
#Port 22
```

avec le port de notre choix (dé préférence > 1000), que l'on dénotera <PORT> par la suite.

Se Connecter sur le Nouveau Port

Maintenant que le serveur expose SSH sur un autre port, il faudra s'y connecter avec l'option `-p`

```
ssh -p <PORT> login@host
```

> 💬 Remarque
>
> Pour se simplifier la vie, on peut se créer un fichier de configuration sur votre machine qui contiendra le login, hôte et port pour un nom donné. Pour cela :
>
> ```
> nano ~/.ssh/config
> ```
>
> et ajouter les lignes suivantes en remplaçant par les bonnes valeurs
>
> ```
> Host <PETITNOM>
>     HostName <VOTREVPS>
>     User <LOGIN>
>     Port <PORT>
> ```

4. Emprisoner

Conclusion

Dans ce module nous avons pris en main les terminaux à distance avec

```
ssh login@host
```

et comment copier ou éditer des fichiers distants avec

```
rsync -PIavz login@host:remotepath localpath # Receive
rsync -PIavz localpath login@host:remotepath # Send
sshfs login@host:remotepath localpath # Mount
```

Solutions des exercices

Solution n°1

[exercice p. 10]

Contenus annexes

1. À la découverte du terminal

Objectif

- Savoir ouvrir un terminal et exécuter une commande.

Mise en situation

On a l'habitude d'interagir avec les ordinateurs en mode graphique, c'est-à-dire en utilisant essentiellement la souris pour cliquer sur des éléments d'interface pour effectuer des actions. Mais il existe une autre manière d'interagir avec un ordinateur, très utilisée en informatique : le mode texte ou **interface en ligne de commande** (CLI en anglais).

Ce mode d'interaction est très utile pour pouvoir **utiliser un ordinateur à distance**, ce qui est généralement le cas lorsque l'on souhaite administrer un serveur web.

À travers ce module vous allez découvrir le **shell** qui permet de dialoguer avec le système d'exploitation d'une machine. Il existe beaucoup de shells, le plus connu vient du monde Linux et se nomme bash. Vous allez découvrir les commandes de base pour parcourir des répertoires ou éditer des fichiers.

Terminal ou CLI — Az Définition

Un **terminal**, ou **interface en ligne de commande** (*CLI* en anglais), est une interface homme-machine dans laquelle l'utilisateur interagit avec la machine en mode **texte**. L'utilisateur écrit des **lignes de commande**, la machine les exécute et affiche le résultat des commandes.

Le terme terminal est très général : un terminal peut servir à dialoguer avec un programme informatique, à donner des ordres à un ordinateur, ou à exécuter d'autres programmes.

Shell — Az Définition

Un **shell** est une interface en ligne de commande permettant de dialoguer avec le système d'exploitation de la machine. On dit qu'il **interprète** les commandes.

Langages CLI — Exemple

Il existe au moins autant de shells que de systèmes d'exploitation. Chaque shell propose des commandes spécifiques. Le choix d'un shell se fait surtout par rapport à des critères pratiques (de quoi ai-je besoin ?) et des critères subjectifs (quel shell me semble le plus ergonomique ?).

- Sur Linux, le shell le plus connu et installé par défaut sur la plupart des systèmes s'appelle `bash` (pour "Bourne Again Shell").
- Sur Windows, il existe trois shells :
  - `cmd`, le shell historique,
  - `powershell`, une version plus moderne,
  - et il est possible d'utiliser `bash` avec le sous-système Linux.
- Sur macOS, le shell installé par défaut s'appelle `zsh`. Il partage de grandes similarités avec `bash`.

Shells et Windows

Remarque

Le shell historique `cmd` de Windows n'est presque plus utilisé. `powershell` utilise une syntaxe assez différente de la plupart des autres shells. Il est possible d'installer `bash` sous Windows depuis peu, mais il n'est pas inclus par défaut dans les installations.

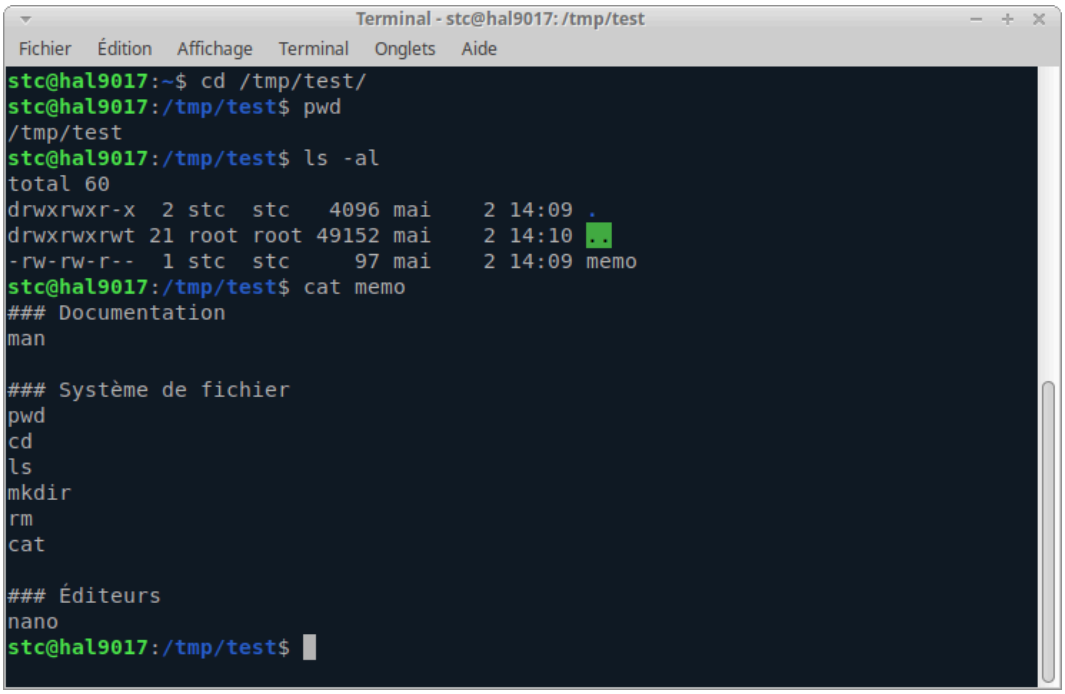
Bash et Windows

Conseil

La plupart des shells adhèrent à des standards communs, mais ce n'est pas le cas des shells disponibles sur Windows. Pour suivre les exemples et exercices qui suivent, il est recommandé **d'activer le sous-système Linux pour Windows 10** et d'utiliser `bash`, en suivant la partie "Installer le sous-système Windows pour Linux" de cette documentation : docs.microsoft.com/fr-fr/windows/wsl/install-win10

Commandes et Bash

Exemple



Cette image montre quelques commandes de base, exécutées par un shell `bash` sur un système Linux.

- `cd /tmp/test` se rend dans le dossier `test` qui se trouve dans le dossier racine `/tmp`.
- pwd affiche le dossier où on est situé dans le système de fichier.
- `ls -al` affiche la liste des fichiers dans le dossier courant.
- `cat memo` affiche le contenu du fichier `memo`.

Terminal et environnement graphique

Remarque

- Un ordinateur personnel moderne PC dispose d'une interface graphique et d'une interface terminal : les deux permettent d'effectuer à peu près les mêmes opérations : visualiser des fichiers, les supprimer, ouvrir des applications, etc.
- Un serveur n'offre en général qu'une possibilité d'accès à distance via un terminal. C'est une des raisons pour lesquelles savoir utiliser le terminal est utile.

Ouvrir un terminal sous Linux

Méthode

En général (et en particulier sur les systèmes Ubuntu), le raccourci `Ctrl+Alt+T` ouvre un terminal. Une alternative consiste à chercher `Terminal` dans la liste des applications.

Ouvrir un terminal sous Windows 10

Méthode

Ouvrir la fenêtre `Exécuter` à l'aide du raccourci `Super+R` (la touche `Super` est en général représentée par un logo Windows sur le clavier). Entrer :

- `cmd` ou `powershell` dans la fenêtre qui s'est ouverte pour démarrer un terminal Windows
- `bash` pour ouvrir un shell Bash s'il a été installé

Ouvrir un terminal sous macOS

Méthode

Depuis le Launchpad, chercher `Terminal` et cliquer sur l'icône qui s'affiche.

Quelques commandes de base

Syntaxe

Ces commandes sont des commandes Bash qui fonctionnent également sous macOS et avec la plupart des autres shells Linux.

- `ls -al` : lister les fichiers dans le répertoire courant
- pwd : afficher le répertoire courant
- `cat` *`fichier`* : afficher le contenu d'un fichier
- `cd` *`dossier`* : se rendre dans un dossier fils du dossier courant
- `cd ..` : se rendre dans le dossier parent
- `echo` *`message`* : afficher un message

- `man` *`commande`* : afficher la documentation détaillée d'une commande

Copier/coller dans un terminal

Méthode

Les actions « copier » et « coller » sont accessibles en effectuant un clic droit sur la fenêtre d'un terminal. Il est plus rapide d'utiliser des raccourcis clavier pour copier et coller, respectivement :

- Sur Bash et la plupart des shells Linux :`Ctrl+Shift+C` et `Ctrl+Shift+V`
- Sur macOS : `Command+C` et `Command+V`
- Sur Windows (`powershell`) : `Ctrl+C` et `Ctrl+V`

À retenir

- Un shell, ou par abus de langage un terminal, permet de dialoguer avec le système d'exploitation d'une machine.
- Il existe beaucoup de shells. Le plus connu vient du monde Linux et se nomme `bash`.
- Un shell permet de se passer du mode graphique, ce qui est souvent indispensable pour travailler sur une machine à distance.