Linux et Bash - Introduction a Linux

45 minutes Normal

Machine status

Ubuntu Server 18.04 LTS SSD Volume Type 64-bit x86

Online

34.245.135.23

Connect

Reset Stop

DataScientest • com

Introduction à Linux

Dans cette leçon, nous allons voir les bases du langage de programmation `Bash` et le fonctionnement des systèmes **Linux**.

Un peu d'histoire de Linux

Dans les années 80, Richard Stallman, chercheur au MIT, crée le système d'exploitation gratuit et ouvert GNU basé sur Unix. Au même moment Linus Torwalds, étudiant finlandais développe le système Linux. Les deux projets finissent par fusionner, donnant naissance à GNU/Linux plus simplement appelé Linux. Sur le coeur de ce système d'exploitation, plusieurs distributions ont été créées :

- Debian
- Red Hat
- ...

Par dessus ces distributions, nous retrouvons des distributions de distributions. Par exemple la distribution la plus connue construite sur Debian est **Ubuntu** que nous utiliserons dans cette leçon. Grâce à une large communauté de développeurs et d'utilisateurs, Ubuntu est bien documentée et régulièrement mise à jour.

Architecture du système

Les données d'un système Linux sont réparties selon une structure arborescente. Ainsi, si un dossier `folder1` contient deux sous-dossiers `subfolder1` et `subfolder2`, la structure de ces dossiers peut être représentée comme suit :

```
folder1

- subfolder1
- subfolder2

```

Et si le dossier `subfolder1` contient les fichiers `file1` et `file2`, la structure devient :

```
folder1

- subfolder1
- - file1
```

En particulier, l'architecture du système démarre à la racine désignée par le symbole `/`.

Dans ce répertoire, nous retrouvons les sous-répertoires suivants :

- `/bin` : contient les fichiers binaires essentiels (Bash shell, files manipulation, ...) ;
- `/boot` : contient les fichiers nécessaires pour démarrer le système ;
- `/cdrom` : répertoire temporaire utilisé pour les CD-ROM ;
- `/dev` : contient des fichiers spéciaux pour certains outils (console, stdout, ...) ;
- `/etc` : contient les fichiers de configuration du système ;
- `/home` : contient les fichiers de données des utilisateurs ;
- `/lib` : contient les librairies nécessaires pour faire tourner les fichiers binaires contenus dans `/bin` et `/sbin` ;
- `/lost+found` : contient les fichiers corrompus après une chute du système ;
- `/media` : contient la localisation des dossiers externes temporaires (clef USB, ...) ;
- `/mnt` : contient les répertoires montés ;
- `/opt` : contient certains fichiers qui ne respectent pas la hiérarchie classique des fichiers ;
- `/proc` : contient les informations système ;
- `/root` : dossier maison pour l'utilisateur `root` ;
- `/run` : contient des fichiers temporaires utilisés par des programmes qui ne doivent PAS être supprimés ;
- `/sbin` : contient des fichiers binaires utilisés par l'utilisateur `root` ;
- `/srv` : contient les données nécessaires à des processus qui fournissent des services (fichiers web par exemple, ...) ;
- `/tmp` : contient des fichiers temporaires utilisés par des processus qui peuvent être supprimés (ils le sont généralement à chaque redémarrage) ;
- `/usr` : contient les applications pour les utilisateurs. Ce répertoire est ensuite divisé en `/bin`, `/lib`... qui ont la même utilité qu'à `/` mais avec les applications utilisateurs ;
- `/var` : contient les fichiers de log.

Premières commandes

Quand vous ouvrez un terminal ou une console, vous allez voir plusieurs choses par défaut :

- `username` : le nom de l'utilisateur que vous utilisez ;
- `@` : un séparateur annonçant le nom de la machine que vous utilisez ;
- `machine_name` : le nom de la machine que vous utilisez ;
- `:` : un séparateur annonçant le chemin de l'endroit où vous vous trouvez ;
- `~` : le chemin de l'endroit où vous vous trouvez ;
- `$` : un séparateur indiquant les droits que vous avez à ce moment.

Si l'utilisateur est en fait le super-utilisateur, c'est-à-dire, l'utilisateur `root`, alors le `$` est remplacé par `#` .

Par défaut, pour un utilisateur normal, le terminal s'ouvre à `/home/username` généralement simplifié en `~` pour l'utilisateur `username`. Pour vérifier où vous êtes, vous pouvez entrer la commande `pwd` (qui signifie `print working directory`) :

```
pwd

```

Cette commande devrait retourner :

```
/home/ubuntu

```

Pour renvoyer le contenu d'un dossier, on peut utiliser la commande `ls` :

Pour le moment, il n'y a rien dans ce fichier.

Nous affichons aussi le contenu d'un autre dossier en utilisant le chemin qui mène à ce dossier comme argument. Ainsi, si nous souhaitons afficher le contenu du répertoire `/`, nous pouvons faire :

```
ls /

```

Vous devriez voir les dossiers mentionnés précédemment.

Pour voir le contenu du dossier `/usr/bin`, on peut faire :

```
ls /usr/bin

```

Dans ces deux exemples, nous avons utilisé le **chemin absolu**, c'est-à-dire le chemin depuis le répertoire `/` jusqu'au répertoire qui nous intéresse.

Nous aurions pu utiliser d'autres options :

- `-a` : pour lister tous les fichiers, même les fichiers cachés (dans ces systèmes, il suffit d'ajouter un `.` au début du nom d'un fichier pour le rendre caché) ;
- `-l` : pour ajouter des informations sur les fichiers (droits, taille, ...) ;
- `-R` : pour lister les fichiers de manière récursive dans les sous-dossiers ;
- `-h` : pour afficher la taille des fichiers avec un format lisible pour les humains ;
- ...

Vous pouvez aussi afficher les commandes disponibles ainsi que leur documentation en utilisant la commande :

Exécutez cette commande pour afficher l'aide de la fonction :

```
ls --help

```

Plus généralement, lorsque nous souhaitons obtenir des informations sur une commande du shell, nous utilisons la commande `man` suivie du nom de la commande. Ainsi dans notre shell le manuel de la commande s'affiche que nous défilons avec les flèches de notre clavier. Pour quitter celui-ci appuyez sur `q`.

Lire le manuel de la commande ls :

```
man ls

```

Listez les fichiers dans le dossier `~` :

```
ls -a

```

Nous devrions à présent voir deux choses :

```
.
..

```

Ces deux choses représentent en fait le dossier courant et le dossier parent. Ainsi `.` est une représentation de `/home/ubuntu` et `..` est une représentation de `/home` . Cela nous permet d'utiliser le **chemin relatif** : pour indiquer un fichier ou un dossier,nous

Exécutez la commande suivante pour afficher le contenu du dossier parent :

```
ls ..
```

Nous pouvons même afficher le contenu du dossier `/usr/bin` en utilisant :

Exécutez cette commande pour afficher le contenu de ce dossier :

```
ls ../../usr/bin
```

Ici nous cherchons donc le dossier parent du dossier parent du dossier courant. Nous cherchons alors le dossier `usr` puis le dossier `bin`. Les chemins absolus et relatifs peuvent presque tout le temps être utilisés de la même façon. Maintenant que nous savons nous repérer depuis le dossier courant, nous allons apprendre à se déplacer depuis ce dossier courant. Pour cela, nous allons utiliser la commande `cd` qui signifie `change directory` .

Déplacez vous à la racine du système en utilisant la commande suivante :

```
cd /
```

Vérifiez que nous sommes bien à la racine en utilisant la commande `pwd` :

- Déplacez vous dans le dossier `/usr/bin`. On pourra utiliser plusieurs commandes.
- Listez les fichiers disponibles dans ce dossier.

Notez aussi qu'on peut revenir au dossier maison de l'utilisateur, c'est-à-dire `/home/ubuntu` dans notre cas en utilisant simplement `cd` ou `cd ~` .

Retournez dans le dossier maison.

Variables d'environnement

Les variables d'environnement constituent un moyen d'influencer le comportement des logiciels sur votre système. Par exemple, la variable d'environnement `LANG` détermine la langue que les logiciels utilisent pour communiquer avec l'utilisateur.

Les variables sont constituées de noms auxquels on assigne des valeurs.

Créer une variable d'environnement

La syntaxe pour créer une variable d'environnement est la commande Bash suivante :

```
export NOM=VALEUR
```

Créez une variable d'environnement nommé `DATASCIENTEST` ayant pour valeur *prenom.nom* .

Lire les variables d'environnement existantes

La valeur de la variable peut également être récupérée en utilisant le signe `$` devant son nom, comme dans l'exemple suivant :

```
echo $DATASCIENTEST
```

> Affichez toutes les variables à l'aide de cette commande.

Vous pouvez observer qu'il y en a beaucoup (au passage vous pouvez observer que notre variable `DATASCIENTEST` est bien dans la liste). La signification d'une variable d'environnement et le type de valeur qui peut lui être assigné sont déterminés par l'application qui utilise celle-ci. Il existe un petit nombre de variables d'environnement bien connues, dont le sens et le type de valeur sont bien déterminés, et qui sont utilisés par de nombreuses applications :

- `PATH` : Lorsque vous tapez une commande, le système la recherche dans les dossiers spécifiés par la variable PATH, dans l'ordre où ils sont indiqués.
- `USER` : C'est le nom de l'utilisateur actuellement connecté. Cette variable est définie par le système.
- `PWD` : C'est le répertoire de travail courant de l'interpréteur de commande.
- `HOME` : C'est l'emplacement du répertoire personnel de l'utilisateur actuellement connecté.

C'est grâce à certaines de ces variables que l'intégrité de l'architecture du système est maintenue. Par exemple, lorsque nous exécutons la commande `pwd`, il nous suffit d'imprimer la variable d'environnement `PWD` ; Lorsque nous changeons de répertoire courant il suffit de modifier la valeur de PWD.

Supprimer une variable

La commande `unset` peut être utilisée pour supprimer complètement une variable d'environnement :

```
unset DATASCIENTEST

```

Conclusion

Nous avons pu ainsi voir comment fonctionne Linux et l'utilisation des fonctions primaires à connaître afin de pouvoir naviguer dans celui-ci. Dans le prochain cours nous verrons comment manipuler des fichiers et des dossiers.

Validated