**Linux et Bash - Introduction a Linux**

**Introduction à 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

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

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

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

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 :

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.

man ls

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

cd /

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 ~ .

**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.

**Créer une variable d'environnement**

export DATASCIENTEST=vincent.bulut

Lire les variables d'environnement existantes

echo $DATASCIENTEST

Nous avons également la commande printenv qui affiche les noms et les valeurs de toutes les variables d'environnement définies.

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.

**Supprimer une variable**

unset DATASCIENTEST

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Manipulation de fichier et dossier**

Création de fichier et dossier  
touch my\_file  
Si ce fichier existe déjà, cela ne fait que le mettre à jour sa dernière date de modification.

Dans le dossier maison (~), créez les fichiers file1, file2, file3 et vérifiez leur existence en utilisant ls .

Exécutez cette commande pour créer un dossier my\_directory .

mkdir my\_directory

touch ./directory1/file4

Vérifiez que le fichier a bien été créé en utilisant la commande ls -R ~ qui permet de lister les fichiers de manière récursive.

rm file3 ⬄ rm ./file3

Pour supprimer un dossier, il faut ajouter l'argument -r :

rm -r /home/ubuntu/directory1

Créez à présent un dossier directory2 dans le dossier maison. Nous pouvons bien évidemment copier/coller des fichiers/dossiers en utilisant cp ou les déplacer en utilisant mv.

Copiez le fichier file1 et collez le dans le dossier directory2 en utilisant la commande suivante :

cp ./file1 ./directory2/

Nous pouvons faire la même chose en donnant un nouveau nom au fichier de destination :

cp ./file1 ./directory2/new\_file1

Déplacez le fichier file2 dans le dossier directory2 en le renommant new\_file2 avec la commande :

mv ./file2 ./directory2/new\_file2

Nous pouvons utiliser mv pour renommer un fichier si le fichier de destination est dans le même répertoire que le fichier d'origine. Observez tous les changements avec la commande ls -R . Pour copier des dossiers, on peut utiliser l'argument -r comme nous pouvons le faire avec la commande rm .

Édition et lecture de fichier

**Imprimer dans un fichier**

Pour imprimer du texte, nous pouvons utiliser la commande echo.

Essayez cette commande avec cet exemple :

echo hello world

echo hello world > file1

Pour afficher le contenu d'un fichier, nous pouvons utiliser la commande cat.

cat file1

echo hello world >> file2

Exécutez la commande suivante pour imprimer la liste du contenu de la racine / dans un fichier appelé root\_content .

ls / > root\_content

Dans certains cas (fichiers trop volumineux), nous ne voulons afficher que le début ou la fin du fichier. Dans ce cas, nous pouvons utiliser head ou tail. Ces deux commandes peuvent prendre l'argument -n qui introduit un nombre de lignes à afficher. Essayez ces deux commandes avec le fichier root\_content :

head -n 2 root\_content

tail -n 3 root\_content

**Mon premier éditeur de texte : Nano**

Tapez nano pour ouvrir cet éditeur et ajoutez quelques lignes.

Pour fermer nano, il faut faire ctrl+x. Il est demandé alors si nous souhaitons sauvegarder les données modifiées. Ensuite, il nous est demandé de rentrer un nom de fichier. Nous pouvons alors valider le nom du fichier en tapant sur entrée. Notez enfin que pour modifier un fichier existant, nous pourrons faire : nano nom\_de\_fichier .

**Les flux de redirection**

Un mot sur les redirections

En utilisant > et >>, nous avons pu rediriger la sortie standard dans d'autres fichiers. La sortie standard est un outil qui contient ce que les commandes entrées dans l'entrée standard (standard input) impriment. Cependant il existe un troisième élément, l'erreur standard ou standard error qui affiche les messages d'erreur des commandes entrées dans l'entrée standard. Par défaut, la sortie standard et l'erreur sont imprimées dans la console.

Exécutez cette commande qui génère une erreur : cat no\_file

Si nous utilisons le mot-clef 2> ou 2>>, nous pouvons rediriger l'erreur standard dans un fichier :

cat no\_file 2> errors\_file

head errors\_file

Notons que l'entrée standard, la sortie standard ainsi que l'erreur standard sont chacune associées à ce que nous appelons un descripteur de fichier, qui correspondent respectivement aux entiers 0, 1, 2. C'est pourquoi nous retrouvons 2> pour la sortie d'erreur, > est un diminutif de 1>, c'est-à-dire de la sortie standard. Nous pouvons aussi rediriger du texte vers l'entrée standard en utilisant < ou 0<.

head < root\_content

Il est difficile de percevoir l'utilité de l'entrée standard <, mais sa principale fonctionnalité est de pouvoir donner en argument le contenu d'un fichier.

Par exemple, créez le fichier nom.py avec les lignes suivantes :

name = input("Entrez votre nom\n")

print("Bonjour {} !".format(name))

Ensuite, écrivez Daniel dans un fichier prenom via la sortie standard.

echo Daniel > prenom

python3 nom.py < prenom

Sans l'entrée standard, le programme Python vous aurait invité à saisir votre prénom, puis aurait affiché Bonjour votre-prénom !

Il est aussi possible d'utiliser plusieurs flux en une seule commande, par exemple avec la commande ci-dessous, nous décidons de rediriger l'erreur standard dans un fichier log et la sortie standard dans un fichier fic :

ca 1>fic 2>log

cat fic

cat log

Par ailleurs, si nous voulons que la sortie standard ainsi que l'erreur standard soient imprimées dans le même fichier, nous pouvons utiliser le mot-clef 2>&1. Pour cela, il faut le placer à la fin de votre commande, comme ci-dessous :

ls -l > fic 2>&1

**Opérateur pipe**

Cet opérateur est très intéressant car il permet d'enchaîner les instructions. L'opérateur | prend la sortie d'une commande et l'utilise comme entrée pour une autre commande.

ls / | grep bin

La commande ls / permet d'afficher le contenu de la racine. La fonction grep est une fonction très utile qui permet de filter les lignes qui contiennent l'expression qui suit. Ici, on ne retournera que les lignes qui contiennent l'expression bin. Nous pouvons tout à fait les enchaîner :

Exécutez cette commande et essayez de comprendre ce qu'elle fait :

ls / | grep bin | head -n 1

Il existe un autre opérateur && qui lui ne permet que d'enchaîner les commandes, sans lien entre elles.

ls -l && python3 --version && mkdir Test

Cette commande utilise l'évaluation paresseuse, c'est à dire si une des premières commandes ne fonctionne pas alors la suite ne sera pas exécutée.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Utilisateurs et droits**

Linux est un système d'exploitation multi-utilisateurs auquel de nombreux utilisateurs peuvent accéder simultanément. Linux peut également être utilisé dans les ordinateurs centraux et les serveurs sans aucune modification (c’est souvent le cas pour la plupart des serveurs Web, où chaque site appartient à un utilisateur. Chaque utilisateur peut se connecter à distance pour maintenir son site).

Mais cela pose des problèmes de sécurité car un utilisateur non sollicité ou malveillant peut corrompre, modifier ou supprimer des données cruciales. Pour une sécurité efficace, Linux divise l'autorisation en 2 niveaux : La propriété (utilisateurs) et les permissions (droits). Le concept de permission et de propriété des fichiers Linux est crucial dans Linux. Ici, nous allons expliquer ces niveaux. Commençons par la propriété.

*Propriété*

*Utilisateur*

Un utilisateur est le propriétaire d'un fichier. Par défaut, la personne qui a créé un fichier en devient le propriétaire. Par conséquent, un utilisateur est aussi parfois appelé propriétaire.

*Groupe*

Un groupe d'utilisateurs peut contenir plusieurs utilisateurs. Tous les utilisateurs appartenant à un groupe auront le même accès au fichier avec les autorisations du groupe Linux. Supposons que vous ayez un projet dans lequel un certain nombre de personnes doivent avoir accès à un fichier. Au lieu d'attribuer manuellement des autorisations à chaque utilisateur, vous pouvez ajouter tous les utilisateurs à un groupe et attribuer des autorisations de groupe aux fichiers de telle sorte que seuls les membres de ce groupe et personne d'autre ne puisse lire ou modifier les fichiers.

*Autres*

Tout autre utilisateur qui a accès à un fichier. Cette personne n'a pas créé le fichier et n'appartient pas à un groupe d'utilisateurs qui pourrait posséder le fichier. En pratique, cela signifie tout le monde. Par conséquent, lorsque vous définissez l'autorisation pour les autres, on parle également d'autorisation pour le monde entier.

*Super utilisateur et sudo*

Nous avons pu voir que certaines permissions nous sont parfois refusées. Pour acquérir à nouveau ces droits, nous nous connectons en tant que super-utilisateur, aussi appelé root. Nous pouvons utiliser ce rôle de super-utilisateur de plusieurs façons différentes :

Connectez vous en tant que super-utilisateur en utilisant la commande sudo su.

Vous remarquerez que l'invite de commande a changé : l'habituel $ a été changé en #. D'ailleurs, nous remarquerons que la commande cd ne ramène plus à /home/ubuntu mais à /root . Nous sortons de ce mode super-utilisateur en utilisant la commande exit. Si nous utilisons les droits du super-utilisateur pour une seule commande, nous pourrons utiliser la clause sudo pour introduire la commande. Ici, notre utilisateur ubuntu n'a pas de mot de passe mais cette commande devrait générer une invite de mot de passe.

*Permissions*

Chaque fichier et dossier de votre système a les 3 permissions suivantes définies pour les 3 propriétaires mentionnés ci-dessus.

*Lire*

Cette autorisation vous donne le droit d'ouvrir et de lire un fichier. L'autorisation de lecture sur un répertoire vous donne la possibilité d'énumérer son contenu.

*Écriture*

Le droit d'écriture vous donne l'autorité de modifier le contenu d'un fichier. L'autorisation d'écriture sur un répertoire vous donne le pouvoir d'ajouter, de supprimer et de renommer les fichiers stockés dans le répertoire. Imaginez que vous avez le droit d'écrire sur un fichier mais pas sur le répertoire dans lequel le fichier est stocké. Vous serez en mesure de modifier le contenu du fichier. Mais vous ne pourrez pas renommer, déplacer ou supprimer le fichier du répertoire.

*Exécuter*

Vous ne pouvez pas exécuter un programme si la permission d'exécuter n'est pas définie. Si vous n'avez pas les droits d'exécution, vous pouvez toujours voir/modifier le code du programme (à condition que les autorisations de lecture et d'écriture soient définies), mais pas l'exécuter.

Passons à la pratique

ls -l /

Les sorties devraient ressembler à ça :

-rw-rw-r-- 1 ubuntu ubuntu 133 Apr 12 11:10 file

On peut voir différentes parties.

-rw-rw-r-- : listes des permissions sur l'objet ;

1 : nombre de liens vers cet objet ;

ubuntu : nom d'utilisateur du propriétaire ;

ubuntu : nom du groupe d'utilisateur du propriétaire ;

133 : taille de l'objet ;

Apr 12 11:10 : dernière date de modification ;

file : nom de l'objet.

La première partie indique les droits sur l'objet. Nous retrouvons le premier caractère et ensuite 3 chaînes de trois caractères.

- : Cette première partie correspond à la nature de l'objet: d pour un dossier, - pour un fichier, l pour un lien... ;

rw- : correspond aux permissions accordées au propriétaire de l'objet ;

rw- : correspond aux permissions accordées aux utilisateurs qui appartiennent au même groupe que le propriétaire de l'objet ;

r-- : correspond aux permissions accordées aux autres utilisateurs.

Ces permissions semblent un peu compliquées mais en réalité, les lettres indiquent que l'autorisation est donnée alors que - indique que la permission n'est pas accordée. Ainsi, on peut avoir les lettres suivantes :

droits en lecture : r ;

droits en écriture : w ;

droits en exécution : x ;

information concernant le fait que ce soit un dossier ou non : d .

L'utilisation de groupe d'utilisateurs est intéressante car elle permet de donner des droits à un groupe de personne d'un coup.

### **Changement de permission**

Pour changer les permissions d'un fichier, il faut utiliser la commande chmod .

chmod 777 file

chmod a+rwx file

La première façon consiste à utiliser une représentation binaire des permissions : chaque chiffre correspond à un groupe d'utilisateurs comme vu précédemment lors de la lecture des droits.

0 : ---

1 : --x

2 : -w-

3 : -wx

4 : r--

5 : r-x

6 : rw-

7 : rwx

La deuxième façon consiste à désigner le ou les groupes à qui l'on veut attribuer ou enlever des droits :

a : les utilisateurs concernés par la modification a pour tous (all), u pour le propriétaire, g pour le groupe de l'utilisateur o pour les autres utilisateurs ;

+ : est ce qu'on donne ou enlève des droits: + pour donner et - enlever ;

rwx : Les droits à ajouter ou retirer.

Notez enfin que pour modifier les droits d'un fichier, il faut en être le propriétaire ou utiliser le super-utilisateur.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Script Shell**

**Langage Bash**

Bash est un langage installé par défaut sur les machines Linux et permet de piloter la manipulation des fichiers.

Exécuter des scripts Bash

Un script Bash est un simple fichier contenant des lignes de codes écrits en Bash qui peuvent être exécutés. Il contient à sa base, un shebang, qui correspond à un indicateur de l'emplacement du shell qui doit être utilisé pour exécuter le code et se présente sous cette forme :

#!/bin/bash

C'est quelque chose que vous trouverez dans de nombreux scripts, pas seulement dans les scripts Bash : par exemple, au début d'un script Python, vous pouvez trouver une ligne du type #!/bin/python3. Cela permet au fichier de s'exécuter de lui même.

Ouvrez un fichier nommé script.sh avec nano. Ajoutez le shebang correspondant à Bash et une autre ligne qui répertorie le contenu du répertoire /. Ensuite, quittez l'édition du fichier et enregistrez-le.

Il y a deux façons d'exécuter le fichier :

bash script.sh

./script.sh

Dans le premier cas, nous lisons uniquement le contenu du fichier et le transmettons à Bash. Dans le second cas, nous l'exécutons. Par défaut, cela devrait déclencher une erreur car vous n'avez pas le droit d'exécuter ce fichier.

Modifiez les droits pour rendre ce fichier exécutable par vous et essayez de le relancer.

**Les bases en Bash**

**Commentaires**

Tout d'abord, comme pour tout langage de programmation, vous pouvez commenter des parties de votre code. Pour ce faire, vous pouvez commenter une ligne en utilisant # au début de celle-ci.

Définir une variable

my\_variable=hello

echo $my\_variable

Notez que les variables ne sont pas typées et doivent apparaître sous forme de tableaux ou listes de caractères, mais certaines peuvent être utilisées pour l'arithmétique opérations, comparaisons, ...

**Guillemets**

Si vous souhaitez attribuer une phrase à une variable, vous rencontrerez quelques problèmes :

my\_variable=hello world

Nous nous rendons compte que l'espace rompt l'affectation de variable. Vous devez utiliser des guillemets pour correctement définir votre variable :

my\_variable="hello world"

echo $my\_variable

Il existe plusieurs types de guillemets : ', " et ` .

Pour comprendre les différences entre les deux premiers, exécutez les lignes suivantes :

echo 'the content of my variable is $my\_variable'

echo "the content of my variable is $my\_variable"

Le dernier est utilisé pour affecter le résultat d'une commande Bash à une variable :

my\_variable=`ls -l`

echo $my\_variable

Opérations mathématiques

Nous pouvons définir des opérations mathématiques en Bash en utilisant le mot-clé let :

let "a=1"

let b=2

let "c=b"

let "d = a + b \* c"

echo $d

Les opérations mathématiques sont assez similaires aux autres langages de programmation : +, -, \*, /, \*\*, ... .

Tableaux

Nous pouvons définir des objets similaires à des listes tableaux (array), pour stocker plusieurs valeurs à la fois :

my\_array=(hello world)

Pour accéder aux différentes valeurs du tableau, nous pouvons utiliser la syntaxe suivante, en tenant compte du fait que l'indexation d'un tableau commence à partir de 0 :

echo ${my\_array[0]}

Pour lui attribuer de nouvelles valeurs, nous pouvons utiliser :

my\_array[0]=Hi

echo ${my\_array[0]}

Et pour renvoyer le tableau complet, nous remplaçons simplement l'index par \* .

echo ${my\_array[\*]}

Pour ajouter des éléments à la liste, nous attribuons simplement des valeurs aux indices qui ne sont pas encore attribués dans le tableau :

my\_array[2]=or

my\_array[4]=hello

my\_array[1000]=world

echo ${my\_array[\*]}

Notez que l'index n'a pas besoin d'être complet en ce sens que les indices n'ont pas besoin de se suivre.

De plus, les tableaux présentent deux fonctionnalités intéressantes :

$ {! my\_array [\*]} renvoie les indices des éléments.

$ {# my\_array [\*]} renvoie le nombre d'éléments.

Boucles et conditions

If

Pour vérifier une condition et effectuer une opération si celle-ci est remplie, nous pouvons utiliser une structure if-then-fi :

prenom="Daniel"

if [ $prenom = "Daniel" ]

then

echo "Salut Daniel !"

fi

Si vous souhaitez ajouter une instruction else, il est possible de le faire avec :

prenom="Daniel"

if [ $prenom = "Daniel" ]

then

echo "Salut Daniel !"

else

echo "Bonjour" + $prenom +"!"

fi

Si nous voulons enchaîner les conditions pour vérifier plusieurs cas :

prenom="Diane"

if [ $prenom = "Daniel" ]

then

echo "Salut Daniel ! "

elif [ $prenom = "Diane" ]

then

echo "Salut Diane"

else

echo "Bonjour" + $prenom +"!"

fi

Nous pouvons faire appel à de nombreux éléments pour créer une condition :

$var1 = $var2 teste l'égalité des tableaux de caractères ;

$var1 != $var2 teste l'inégalité des tableaux de caractères ;

-z $variable teste si le tableau de caractères est vide ;

-n $variable teste si le tableau de caractères n'est pas vide ;

$var1 -eq $var2 teste l'égalité de valeurs numériques ;

$var1 -ne $var2 teste l'inégalité de valeurs numériques ;

$var1 -gt $var2 teste var1 > var2 ;

$var1 -lt $var2 teste var1 < var2 ;

$var1 -ge $var2 teste var1 >= var2 ;

$var1 -le $var2 teste var1 <= var2 .

Pour combiner deux conditions, nous pouvons utiliser && (ET) si nous souhaitons que les deux conditions soient vérifiées et || (OU) si nous voulons qu'au moins une des deux conditions le soit :

prenom="Diane"

nom="Datascientest"

if [ prenom="Daniel" ] && [ nom="Datascientest" ]

then

echo "Bonjour Daniel Data"

else

echo "Bonjour" + $prenom + $nom

fi

While

Pour effectuer une boucle while (tant que...), la syntaxe est la suivante :

let i=0

while [ $i -lt 10 ]

do

let "i=i+1"

done

echo $i

For

Pour effectuer une boucle for (pour...), la syntaxe est la suivante :

for x in '1st iteration' '2nd iteration' '3rd iteration'

do

echo $x

done

Une fonction intéressante pour ces boucles est seq. Cette fonction fournit une suite d'entiers d'un entier de départ à un entier de fin comme suit :

Exécutez le code suivant :

seq 3 22

Fonctions

Il y a deux façons pour définir une fonction :

my\_function () {

echo "Nous pouvons faire quelque chose ici"

}

Ou

function my\_function {

echo "Nous pouvons faire quelque chose ici"

}

Les arguments à passer à la fonction sont donnés par leur numéro. Par exemple une fonction dans laquelle on affiche le premier puis le second argument ressemblera à ceci :

function my\_function {

echo "Premier argument"

echo $1

echo "Second argument"

echo $2

}

Enfin pour appeler ma fonction, il suffit simplement d'écrire :

my\_function "Daniel" "24"

Processus

Qu'est ce qu'un processus

Une instance d'un programme est appelée un processus. En termes simples, toute commande que vous donnez à votre machine Linux démarre un nouveau processus. Une application sur votre système, un script Bash, ou encore une simple commande ls crée un nouveau processus. Plusieurs processus peuvent s'exécuter en même temps. Il existe deux types de processus :

Foreground processus : Ils s'exécutent à l'écran et nécessitent une intervention de l'utilisateur (Office, votre navigateur web, application de retouche photos/videos...).

Background processus : Ils s'exécutent en arrière-plan et n'ont généralement pas besoin de l'intervention de l'utilisateur (Antivirus, programme Python de web scraping...).

Exécution de processus

Foreground

Pour lancer un processus d'avant-plan, vous pouvez soit l'exécuter à partir du tableau de bord, soit l'exécuter à partir du terminal.

Si vous utilisez le terminal, vous devrez attendre que le processus de premier plan s'exécute.

Background

Si vous démarrez un programme/processus en avant-plan à partir du terminal, vous ne pouvez pas travailler sur le terminal tant que le programme n'est pas opérationnel.

Les tâches particulières, à forte intensité de données, nécessitent beaucoup de puissance de traitement et peuvent même prendre des heures. Vous ne voulez pas que votre terminal soit bloqué pendant un tel laps de temps.

Pour éviter une telle situation, vous pouvez exécuter le programme et l'envoyer en arrière-plan afin que le terminal reste à votre disposition. Voyons comment faire.

Mise en situation

Imaginons que le script Python suivant permet de faire du webscraping. Chaque seconde, il récupère des données et les écrit dans le fichier data.txt.

import time

i = 0

while i < (60 \* 20):

fichier = open("data.txt", "a")

fichier.write(str(i) + "\n")

fichier.close()

i += 1

time.sleep(1)

Recopiez ce script dans un fichier webscraping.py et exécutez.

python3 webscraping.py

Vous venez de lancer votre script en avant-plan. Le problème c'est que le script va tourner pendant 20 minutes. Nous aimerions bien avoir un certain contrôle sur notre terminal pendant ce temps. Pour cela, il existe plusieurs commandes pour interagir avec notre processus.

À chaque commande que nous allons voir, prenez le temps d'observer les dernières lignes du fichier data.txt (par exemple tail -3 data.txt).

Mise en pause

Avant tout pour récupérer le contrôle sur notre terminal il faut mettre notre processus en pause. Pour mettre un processus qui tourne en avant-plan en pause il suffit d'appuyer sur les touches de votre clavier Ctrl+z.

[1]+ Stopped python3 webscraping.py

La ligne ci-dessus devrait s'afficher (observez le contenu de data.txt).

Mise en Foreground

Nous pourrions vouloir mettre un processus en avant-plan pour cela vous pouvez utiliser la commande fg. Exécutez la commande pour voir que celle-ci fonctionne bien puis remettez vous en pause.

Mise en Background

Vous vous en doutez certainement la commande pour mettre en arrière-plan c'est bg. Grâce à celle-ci le processus continue de tourner en laissant le terminal entre vos mains.

[1]+ python3 webscraping.py &

La ligne ci-dessus devrait s'afficher. Nous observons le caractère & à la suite de notre commande d'exécution. Pour généraliser, si nous souhaitons directement lancer un script en arrière-plan, il suffit d'ajouter & à la suite de notre commande.

Outils supplémentaires

Il existe différentes commandes pour avoir des informations sur les processus en cours :

Htop

Cet utilitaire informe l'utilisateur sur tous les processus en cours sur la machine Linux. Lancez la commande htop, énormément d'informations s'affichent devant vous. Et oui ! Beaucoup de processus tournent sans que vous le sachiez afin de faire fonctionner votre système. Explorez l'outil afin de retrouver notre processus qui doit certainement être en train de tourner via la commande man.

Appuyez sur q sur votre clavier pour sortir de l'affichage du processus htop.

PS

Cette commande signifie "Process Status" (État du processus). Cette commande est similaire à la commande htop mais les informations affichées sont différentes.

Pour vérifier tous les processus en cours d'exécution sous un utilisateur, utilisez la commande ps ux. On observe ainsi plus facilement les informations concernant les processus lancés par nous-même.

PID

Avec les outils précédents, une des informations concernant notre processus est le PID pour Process ID. Cet identifiant nous permet de pouvoir interagir directement avec le processus. Vous pouvez par exemple faire un ps PID pour avoir seulement les informations du processus concerné, ou bien vous pouvez décider de mettre fin au processus.

Pour trouver facilement le PID d'un processus vous pouvez utiliser la commande pidof Process\_Name. Dans notre exemple : pidof python3 webscraping.py

Mettre fin au processus : Kill

La commande kill met fin aux processus en cours d'exécution sur une machine Linux. Pour l'utiliser, vous devez connaître le PID (identifiant du processus) du processus que vous voulez tuer et exécuter la commande kill PID.

Crontab

Qu'est ce que Crontab ?

Cron tire son nom du mot grec "Chronos", qui désigne le temps.

Il s'agit d'un processus qui exécute automatiquement des tâches selon un programme spécifique. C'est un ensemble de commandes qui sont utilisées pour exécuter des tâches régulières de planification. Crontab est également le nom du programme qui est utilisé pour modifier cette planification. Il est piloté par un fichier crontab, un fichier de configuration qui indique les commandes shell à exécuter périodiquement pour le programme spécifique.

Crontab est souvent utilisé pour l'automatisation de tâches telles que la gestion des logs, le nettoyage des données dans le cache, envoyer des emails de newsletter ou encore automatiser votre script de scraping de données.

Syntaxe d'un Cron Job

Un Cron Job est une automatisation de tâches à réaliser. La syntaxe est la suivante :

\* \* \* \* \* command/script

De gauche à droite :

Le premier \* correspond aux Minutes (0-59) ;

Le deuxième \* correspond aux Heures (0-23) ;

Le troisième \* correspond au Jour du mois (1-31) ;

Le quatrième \* correspond au Mois de l'année (1-12) ;

Le cinquième \* correspond au Jour de la semaine (0-6, du dimanche au samedi).

Pour spécifier plusieurs valeurs dans un champ, utilisez les symboles d'opérateur suivants :

Astérisque (\*) : Pour spécifier toutes les valeurs possibles pour un champ ;

Tiret (-) : Pour spécifier une plage de valeurs ;

La virgule (,) : Pour spécifier une liste de valeurs ;

Le séparateur (/) : Pour spécifier une valeur d'étape.

Exemple

Voici quelque exemple grâce à la syntaxe ci-dessus :

Exécutez un Cron Job à 5h15 tous les jours.

15 5 \* \* \* command/script

Exécutez un Cron Job à 5h15 chaque 2eme jour du mois.

15 5 2 \* \* command/script

Exécutez un Cron Job toutes les 5h.

0 \*/5 \* \* \* command/script

Exécutez un Cron Job chaque lundi et mercredi du mois de janvier et février.

0 0 \* jan,feb mon,wed command/script

Mise en pratique

Créer le script cron.sh suivant :

#!/bin/bash

echo "Cron Job" >> ~/data\_cron.txt

Ensuite, donnez les droits d'exécution au script cron.sh via la consigne suivante :

chmod +x ~/cron.sh

À présent, exécutez la commande crontab -e pour ajouter un job.

Insérez dans le fichier la ligne : \*/1 \* \* \* \* ~/cron.sh .

Le Cron Job défini exécute notre script à intervalle de 1 minute : Observez l'évolution du fichier data\_cron.txt au fil des minutes.

Commandes usuelles

crontab -e : Pour modifier le fichier crontab de l'utilisateur actuel ;

crontab -l : Pour afficher le contenu du fichier crontab ;

crontab -u [nom d'utilisateur] : Pour modifier le fichier crontab d'un autre utilisateur ;

crontab -r : Pour supprimer le fichier crontab de l'utilisateur actuel ;

crontab -i : Pour afficher une invite avant de supprimer le fichier crontab de l'utilisateur actuel.

Conclusion

Dans ce cours, nous avons vu comment fonctionne un système Linux. Nous aurions pu aller encore plus en détails mais nous avons construit la base pour la suite du parcours. Pour la validation de ce module vous devez suivre les instructions suivantes pour réaliser l'exercice.

Exercice à réaliser obligatoirement sur la machine Linux mise à votre disposition.

Connectez vous à votre machine et exécutez la commande suivante pour récupérer l'api

wget --no-cache https://dst-de.s3.eu-west-3.amazonaws.com/bash\_fr/api.tar

Vous avez maintenant un fichier d'extension .tar. Il s'agit simplement d'une archive à la manière d'un fichier compressé zip, mais spécifique à Linux. Pour manipuler ce fichier, nous passons par la commande tar (pour tape archiver). Pour tous les formats à base de tar, vous verrez que les options de tar sont les mêmes :

c : crée l'archive

x : extrait l'archive

f : utilise le fichier donné en paramètre

v : active le mode verbeux.

Décompressez l'archive à l'aide de la commande suivante :

tar xvf api.tar

L'extrait de l'archive vous dévoile le script api

Lancez le script api après avoir donné les droits d'exécution :

chmod +x api

./api

Notre API tourne maintenant en localhost (0.0.0.0) sur le port 5000.

Contexte de l'exercice

Cette API nous dévoile les ventes par minutes du plus gros revendeurs de cartes graphiques sur les modèles rtx3060, rtx3070, rtx3080, rtx3090 et rx6700. Il est possible de récupérer ces informations à l'aide de la commande cURL. Toutefois, il se peut que vous n'ayez pas cURL sur votre machine, pour remédier à cela, nous utilisons apt sur Linux.

Commande apt

apt est un gestionnaire de paquets qui contiennent différents logiciels que vous pouvez installer assez facilement avec une seule ligne de code. Pour ce faire, nous pouvons faire comme suit :

apt install software\_name

Dans les anciennes versions d'Ubuntu, vous aviez besoin d'utiliser apt-get au lieu de apt. Dans la plupart des cas, vous avez besoin de sudo pour forcer les droits d'installation d'un logiciel.

Pour vous assurer que les paquets sont à jour, vous pouvez utiliser sudo apt update . Pour mettre à jour les logiciels, vous pouvez utiliser sudo apt upgrade . Vous pouvez ajouter ou supprimer certains paquets et supprimer complètement un logiciel utilisant la fonction apt purge.

Installez curl avec apt(vous pouvez installer d'autres paquets comme default-jdk, qui est relatif à Java).

Maintenant que nous avons curl, expliquons l'outil.

Commande curl

cURL, qui signifie client URL est un outil de ligne de commande pour le transfert de fichiers avec une syntaxe URL. Il prend en charge un certain nombre de protocoles (HTTP, HTTPS, FTP, et bien d'autres). HTTP/HTTPS en fait un excellent candidat pour interagir avec les APIs !

On peut par exemple récupérer les ventes de rtx3060 a l'aide de la commande suivante.

curl "http://0.0.0.0:5000/rtx3060"

Créez un dossier exam\_NOM ou NOM est le nom de la personne en majuscule

Créez à l'intérieur du dossier un script bash nommé exam.sh dont les droits d’exécution sont -rwx---r-x.

Le script bash devra récupérer les chiffres correspondant au ventes des différentes carte graphique et écrire les information dans un fichier sales.txt sous la forme :

Date de scraping

rtx3060:Nb de vente

rtx3070:Nb de vente

...

Voici un exemple de fichier sales.txt:

Thu Apr 1 00:05:01 UTC 2021

rtx3060:18

rtx3070:20

rtx3080:20

rtx3090:2

rx6700:12

Thu Apr 1 00:06:01 UTC 2021

rtx3060:6

rtx3070:15

rtx3080:15

rtx3090:1

rx6700:15

...

...

Contrainte : utilisation d'une fonction et d'une boucle (for ou while) obligatoire.

Créez un Cron Job qui exécute votre script toutes les minutes seulement durant les mois de votre formation (copier le Cron Job dans un fichier cron.txt dans le dossier)

Rendu

on a donc les fichiers, dossiers suivant:

exam\_NOM/exam.sh

exam\_NOM/sales.txt

exam\_NOM/cron.txt

Créez une archive exam\_NOM.tar

Commande scp

La commande scp permet les transferts sécurisés via connexion SSH.

Vous pouvez télécharger votre archive en exécutant la commande suivante depuis votre propre machine. Vous nous transférez celle-ci par la suite.

scp -i "data\_enginering\_machine.pem" ubuntu@VOTRE\_IP:~/exam\_NOM.tar .

Une fois que vous avez téléchargé votre archive sur votre machine locale, vous pouvez l'envoyer au support DataScientest.

Bon courage !

**#!/bin/bash**

function write\_sales {

r=$(curl -X GET "http://0.0.0.0:5000/$1")

r2="$1:$r"

echo $r2 >> ~/exam\_bash/sales.txt

}

date >> ~/exam\_bash/sales.txt

gc=(rtx3060 rtx3070 rtx3080 rtx3090 rx6700)

for x in "${gc[@]}"

do

write\_sales "$x"

done