

28

Linux et Bash

En bref

Linux est l'un des systèmes d'exploitation importants aujourd'hui, notamment dans le domaine des serveurs. Nous allons nous initier à l'usage de Bash, l'interpréteur de commandes de ce système.

1 Le système Linux

1 Histoire

L'histoire du système d'exploitation (OS) Linux commence en 1991, date à laquelle l'étudiant finlandais Linus Torvalds propose le noyau d'un système d'exploitation libre et *open source*, bientôt nommé Linux en référence à son prénom et au système Unix existant. Avant lui, Richard Stallman, un des pionniers de l'*open source* avait déjà appelé de ses vœux la création d'un système libre baptisé GNU (acronyme récursif : Gnu is Not Unix). → **FICHE 25**.

2 Déclinaisons

Depuis lors, Linux s'est développé de manière très rapide et a donné lieu à de multiples systèmes partageant le même noyau dont Ubuntu et Debian sont aujourd'hui les plus connus. Le noyau de Linux contient des millions de lignes de code, pour la plupart écrites en langage C. Le système Android est également basé sur le noyau Linux.



debian ubuntu android

3 Installation

Au départ réservé à des initiés, Linux est aujourd'hui d'installation aisée sur la plupart des matériels courants à l'aide d'une simple clé USB. Il faut fournir quelques informations sur le partitionnement souhaité, le réseau et donner un nom d'utilisateur qui aura par défaut des droits d'administration via la commande sécurisée sudo.

II Le Bash

1 Le terminal

Quelle que soit la déclinaison de Linux, on trouve une application « terminal » qu'on peut lancer et l'interpréteur de commandes avec lequel on interagit est par défaut Bash (*Bourne Again Shell*) qui est l'interpréteur de commande le plus courant sous Linux et aussi sous Mac OSX. Il est également possible de l'activer sous Windows 10.

2 Les commandes de bases

Toutes ces commandes acceptent de nombreuses options dont on peut consulter la documentation en tapant : man ls par exemple pour la commande ls. Celle-ci comporte des options pour afficher les fichiers cachés ls -a ou encore pour afficher les détails et droits d'un fichier ls -l.

Commande	Description
ls	Lister le contenu du répertoire courant
cp	Copier des fichiers ou des répertoires
mv	Déplacer ou renommer des fichiers ou des répertoires
rm	Effacer des fichiers ou des répertoires
cd	Se déplacer dans l'arborescence
cat	Visualiser le contenu d'un fichier
echo	Afficher un message ou le contenu d'une variable
touch	Réinitialiser le timestamp d'un fichier ou créer un fichier vide

3 Les répertoires fondamentaux

Dans un système Unix, on dispose d'une arborescence de fichiers ancrée sur /, la « racine » du système de fichiers. Voici quelques points d'entrée de cette arborescence :

/	← Commandes de base du système
- bin	← Fichiers représentant les dispositifs matériels (devices) du système
- dev	← Fichiers de configuration du système
- etc	← Répertoires d'accueil (HOME) des utilisateurs
- home	← Librairies
- lib	← Points de montage (clé USB, etc.)
- mnt	← État du système et de ses processus
- proc	← Répertoire de l'administrateur du système, pas stocké dans /home
- root	← Logiciels installés avec le système, bases de données, etc.
- run	← Données fréquemment utilisées et réécrites
- sys	
- usr	
- var	

29

Arborescences et flux

En bref

Observons quelques moyens de naviguer dans l'arborescence d'un système Unix, puis voyons comment manipuler les entrées/sorties et les redirections.

I

Navigation et entrées/sorties en shell Bash

1 Naviguer dans une arborescence, les chemins

- Pour aller dans son HOME en utilisant un « chemin absolu », c'est-à-dire un chemin qui part de la racine, l'utilisateur bob peut faire : cd /home/bob/
- Il peut aussi utiliser le raccourci ~, et faire cd ~ ou plus simplement cd
- Pour remonter dans le répertoire parent, on utilise cd .. et on désigne le répertoire courant avec un point « . ».
- Ainsi, si on veut copier le fichier toto qui se trouve dans le répertoire parent vers le répertoire courant, on tape : cp ../toto .

Nous utilisons cette fois un « chemin relatif », partant de la position actuelle.

2 Entrées et sorties

Entrées

La commande read permet d'effectuer une saisie utilisateur.

```
read var # Saisie de var (stdin)
echo $var # Réaffichage, remarquez le $ comme en PHP
```

Sorties

Une commande affiche normalement son résultat sur la sortie standard stdout. On peut aussi envoyer ce résultat de commande vers un fichier.

Exemple : echo "Bonjour" > salut.txt envoie le mot « Bonjour » dans le fichier salut.txt qui est créé (ou réinitialisé s'il existait préalablement), puis si on tape ensuite echo "Tout le monde" >> salut.txt, on ajoute une nouvelle ligne au fichier existant qui contient maintenant 2 lignes.

Sortie d'erreur

Les sorties d'erreur ne s'affichent pas sur la sortie standard, il existe un troisième flux, le flux de sortie d'erreur dénommé stderr.

Par exemple, s'il n'y a pas de fichier toto dans le répertoire courant, la commande :

```
cat toto
```

déclenche une erreur indiquant que le fichier toto n'existe pas. Si l'on ajoute

```
cat toto 2>/dev/null
```

GOUPS

EXERCICES ET PROBLEMES

CORRIGES

on effectue la même commande, mais on redirige vers le périphérique « null » les éventuelles sorties d'erreur pour qu'elles ne s'affichent pas.

3 Les filtres, tubes et redirections

- Unix permet aussi l'utilisation de **filtres** comme :
 - wc pour compter des lignes ou des caractères ;
 - sort pour trier ;
 - cut pour extraire des colonnes dans un fichier ;
 - tr pour transposer ou supprimer des caractères.
- Ces filtres s'utilisent généralement avec des tubes (pipes) notés « | ».

Exemple :

```
cat monfic | wc -l
```

compte le nombre de lignes dans le fichier monfic et :

```
cat monfic | sort > fictrie
```

trie les lignes du fichier monfic selon l'ordre lexicographique puis place le résultat correspondant dans le fichier fictrie.

II Scripts Bash

Une suite de commandes permet à l'administrateur d'automatiser certaines tâches, on parle alors de « script », stocké dans un fichier d'extensions .sh. Les arguments du script peuvent être invoqués avec \$1, \$2, etc., leur nombre avec \$# et leur liste avec \$*.

Voici par exemple un script efface.sh qui, au lieu d'effacer le fichier qu'on passe en argument, le déplace dans un dossier « poubelle » à la racine du HOME de l'utilisateur (la première ligne indique l'interpréteur utilisé) :

```
#!/bin/bash
# Ce script a un argument: un nom de fichier
# - crée si besoin un répertoire poubelle dans le répertoire
# HOME de l'utilisateur
# - déplace le fichier donné en argument dans ce répertoire
# poubelle
# vérifie d'abord si ce dossier existe dans ~ :
if [ -d ~/poubelle ]
then
    echo "Le répertoire poubelle existe déjà dans votre Home"
else # sinon, on le crée
    mkdir ~/poubelle
    echo "Répertoire poubelle inexistant. Il est créé."
fi
# On déplace dans poubelle le fichier donné en argument
mv $1 ~/poubelle
```

30

Droits et permissions sous Unix

En bref

Le système de droits et de permissions sous Unix est un des aspects fondamentaux de la gestion de la sécurité du système.

1

Droits et groupes

1 Le monde selon Unix

Unix sépare le monde en trois catégories du point de vue des droits :

- l'utilisateur (user) ;
- le groupe (group) ;
- le reste du monde (others).

2 Exemple de lecture de droits

■ En utilisant la commande `ls -l monfic.sh` par exemple, on obtient :

```
-rwxr--r-- 1 roza staff 0 6 mai 11:56 monfic.sh
```

■ La partie `-rwxr--r--`, indiquant les droits du fichier, se lit en omettant le tiret du début, puis en décomposant en trois parties :

- `rwx` (utilisateur) ;
- `r--` (groupe) ;
- `r--` (autres).

■ Chaque partie est elle-même composée de trois lettres :

- droit de lecture `r` ;
- droit d'écriture `w` ;
- droit d'exécution `x` : on peut exécuter le fichier en l'invoquant par son nom, dans cet exemple : `./monfic.sh`.

■ On sait donc que `monfic.sh` est accessible en lecture au groupe Staff et aux autres.

■ On sait en outre que le fichier appartient à l'utilisateur « `roza` ».

3 Les droits d'un répertoire

■ Créons un répertoire `www` dans notre `HOME` et lisons les droits correspondants avec la commande `ls -l www`, on obtient par exemple :

```
drwxr-xr-x 2 roza staff 64 6 mai 14:17 www
```

■ Le `d` initial signifie qu'il s'agit d'un répertoire.



À NOTER

Le droit `x` pour un répertoire est le droit de traverser ce répertoire.

II Changer des droits

1 La commande chmod

Seul le propriétaire d'un fichier (ou l'utilisateur « `root` ») peut changer ses permissions d'accès. Il le fait avec la commande `chmod` dont voici quelques exemples d'utilisation.

Droits	Syntaxe
Donner les droits de lecture au groupe <code>g</code>	<code>chmod g+r monfic.sh</code>
Donner les droits d'écriture au propriétaire <code>u</code>	<code>chmod u+w monfic.sh</code>
Donner les droits d'exécution aux autres (<code>others</code>) <code>o</code>	<code>chmod o+x monfic.sh</code>
Donner les droits d'exécution à tous	<code>chmod ugo+x monfic.sh</code>



À NOTER

On peut aussi utiliser la lettre `a` (all) en raccourci à la place de `ugo`.

2 Droits symboliques et numériques

Il est également possible d'utiliser un codage octal pour les droits.

L'écriture symbolique : `rwx r-x r-x`

correspond à l'écriture binaire : `111 101 101` soit `755` en octal.

Par exemple : `chmod 755 monfic.sh` donne les droits `-rwxr-xr-x` au fichier `monfic.sh`.

3 Lancement d'un script et fichiers exécutables

■ On peut lancer un script en l'invoquant par son nom s'il est exécutable. On précise parfois que le script est dans le répertoire courant en ajoutant `./` devant son nom :

```
./monfic.sh
```

■ Si le script n'est pas exécutable, on peut toujours le lancer en tapant :

```
source monfic.sh
```

■ Ces deux méthodes ne sont pas équivalentes : dans le premier cas, un nouveau shell est créé tandis que dans le second, les commandes du script s'exécutent dans le shell courant.



À NOTER

Tous les fichiers exécutables posent des problèmes potentiels de sécurité, tout fichier exécutabile pouvant se transformer en éventuel « cheval de Troie » (logiciel malveillant). Dans un site web par exemple, les fichiers HTML, CSS, images, JavaScript ou PHP n'ont pas à être exécutables, le droit de lecture suffit !

SE TESTER QUIZ

Vérifiez que vous avez bien compris les points clés des **fiches 25 à 31**.

1 Histoire des machines

→ FICHE 25

1. Les tubes à vide, volumineux et peu fiables, ont été remplacés au milieu des années 1950 par :
 - a. les cartes perforées
 - b. les bouteilles à oxygène
 - c. les transistors
 - d. les écrans à tubes cathodiques
2. Le premier microprocesseur (Intel 4004) a permis :
 - a. l'ouverture du marché aux particuliers
 - b. de miner des Bitcoins
 - c. de déchiffrer le code de la machine Enigma
3. Le premier langage informatique de haut niveau (autre que le langage machine) a été :
 - a. Python
 - b. C
 - c. Fortran
 - d. Java
 - e. Kotlin

2 Architecture de von Neumann

→ FICHE 26

1. Un pipeline d'instruction permet :
 - a. de faire passer des instructions dans un tunnel
 - b. d'exécuter des instructions séquentiellement
 - c. d'exécuter des instructions simultanément
2. L'UAL :
 - a. permet de gérer la mémoire
 - b. permet de gérer les entrées/sorties
 - c. est une adresse internet
 - d. est l'endroit où les calculs sont effectués

3 Mémoire et langage machine

→ FICHE 27

- Parmi ces affirmations, lesquelles sont vraies ?
- a. La mémoire morte s'efface lorsque le courant est éteint.
 - b. La mémoire centrale est une mémoire vive.

4 Bash

→ FICHES 28 à 30

1. La commande `ls -al` :
 - a. permet de lister les fichiers du répertoire courant sans détail.
 - b. permet de lister les fichiers standards et cachés du répertoire courant.
 - c. permet d'afficher des détails sur un fichier comme son propriétaire ou ses droits.
 - d. n'existe pas en Bash.
2. La commande `mv` :
 - a. sert à copier des fichiers ou répertoires.
 - b. sert à déplacer des fichiers ou répertoires.
 - c. peut servir à renommer un fichier ou un répertoire.
 - d. n'existe pas en Bash.
3. La commande `ls -l toto.sh` affiche :
`-rwxr--r-- 1 john staff 128 18 mai 11:56 toto.sh`
 - a. toto.sh appartient à john.
 - b. toto.sh appartient à staff du groupe john.
 - c. Personne n'a le droit d'écriture sur toto.sh.
 - d. Personne n'a le droit de lire toto.sh.
4. Logé sous Linux, dans un terminal on tape `cd` pour se placer dans HOME. Quelle commande doit-on ensuite taper pour déplacer dans le répertoire courant le fichier `exo1.py` qui se trouve dans `Documents/python/` en sachant que `Documents` est dans HOME ?
 - a. `cp Documents/python/exo1.py`
 - b. `mv /Documents/python/exo1.py`
 - c. `mv ./Documents/python/exo1.py`
 - d. `rm Documents/python/exo1.py`

5 Réseaux

→ FICHE 31

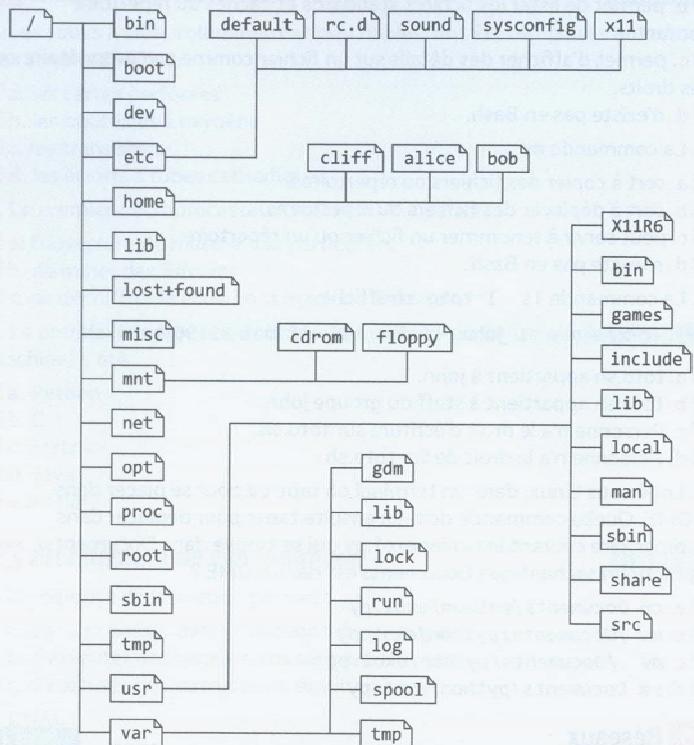
1. Une adresse IP est une adresse :
 - a. physique
 - b. logique
2. Une adresse MAC permet de repérer un appareil Apple.
 - a. vrai
 - b. faux

S'ENTRAÎNER

6 Se déplacer dans le système de fichiers

→ FICHES 28 et 29

Observer l'arborescence suivante :

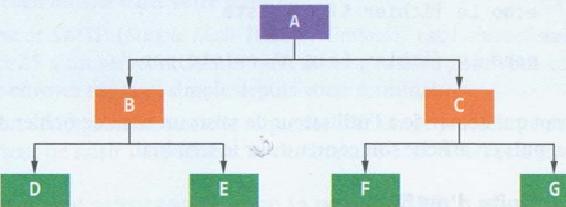


- Proposer une commande qui permette de se déplacer du répertoire HOME de Alice à celui de Bob :
 - en utilisant un chemin relatif ;
 - en utilisant un chemin absolu.
- Alice est à la racine /. Proposer trois commandes qui peuvent lui permettre de se déplacer dans son répertoire d'accueil (HOME).
- Bob est dans son HOME. Aidez-le à :
 - lister le contenu de son HOME ;
 - lister le contenu de son HOME y compris les fichiers et répertoires cachés ;
 - lister le contenu du répertoire share sans quitter son HOME.

7 Créer une arborescence et se déplacer dedans

→ FICHES 28 et 29

- Créer dans le répertoire d'accueil l'arborescence ci-dessous dans laquelle A, B, C, D, E, F et G sont des répertoires :



- Quelle commande Unix permet de créer un fichier vide ?
Créer deux fichiers vides appelés « un » et « deux » dans votre répertoire d'accueil.
- Quelle commande permet de copier des fichiers ou répertoires ? Copier le fichier « un » dans le répertoire « A » en lui donnant le nom « trois ».
- Comment réaliser la copie précédente en utilisant un chemin relatif si vous êtes :
 - dans le répertoire A ?
 - dans le répertoire B ?
- Si votre nom d'utilisateur est alice, comment réaliser cette copie en utilisant un chemin absolu ?
- Comment renommer le fichier « trois » en « quatre » ?

8 Mettre en majuscules

→ FICHE 29

- Expliquer ce que fait la commande suivante (respecter les espaces) :

`echo 'bonjour' | tr [a-z] [A-Z]`



À NOTER
Consultez la page de manuel (`man tr`) correspondante.

- Écrire un script à un argument qui met en majuscules l'unique argument fourni.

9 Tester l'existence d'un fichier et en afficher le contenu

→ FICHES 29 et 30

- Quelle option de la commande echo faut-il utiliser pour « rester sur la même ligne » ?
- Comment faut-il faire pour afficher un message sur plusieurs lignes avec cette même commande ?
- Rappeler la commande permettant de lire au clavier et de stocker le résultat dans une variable Bash.
- Comment affiche-t-on le contenu d'un fichier sur le terminal ?

5. Pour tester l'existence d'un fichier, en Bash on utilise le test suivant :

```
fic='monfic'
if [ -f $fic ]
then
    echo Le fichier $fic existe
else
    echo Le fichier $fic n'existe pas !
fi
```

Écrire un script qui demande à l'utilisateur de saisir un nom de fichier, teste si ce fichier existe, puis en affiche son contenu sur le terminal.

10 Lire les droits d'un fichier

→ FICHE 30

Pour chacun des fichiers suivants, répondre aux deux questions.

```
-rwx----- 1 alice etu 43M 14 jui 11:55 fichier1
-rw-r--r-- 1 roza staff 54K 14 jui 11:56 fichier2
-rwx--x--x 1 bob admin 3M 14 jui 11:57 fichier3
-r-xr----- 1 john john 1B 14 jui 11:58 fichier4
```

- a. Donner le nom de l'utilisateur auquel il appartient, les droits qu'il a sur le fichier, ceux du groupe et des autres.
b. Quel est l'équivalent en octal du droit correspondant ?

11 Rendre exécutable un fichier

→ FICHE 30

- a. Rappeler la commande qui permet de rendre exécutable un fichier pour tous les utilisateurs.
b. Rappeler la manière de tester l'existence d'un fichier.
c. Utiliser ces connaissances pour construire un script rendExecutable à un argument qui teste si cet argument désigne un fichier existant, et le rend exécutable si besoin.

Pour tester si un fichier est exécutable, on peut utiliser le test similaire :

```
if [ -x $fic ]
then
    echo Le fichier $fic est exécutable
else
    echo Le fichier $fic n'est pas exécutable !
fi
```

12 Rendre exécutables plusieurs fichiers

→ FICHE 30

- a. Rappeler comment on accède à la liste des arguments d'un script.
b. Comment parcourir cette liste dans un script ?
c. Utiliser ce parcours pour proposer un script ajouteDroits.sh qui attend en entrée une liste de fichiers et les rend tous exécutables (si besoin).

13 Utiliser telnet ou netcat

→ FICHE 29

a. Un serveur web est habituellement hébergé sur le port 80 d'un ordinateur. Utiliser telnet, netcat ou gnetcat pour vous connecter sur ce port et récupérer la page d'accueil du site dans votre terminal.

b. Un serveur SMTP (*Simple Mail Transfer Protocol*) est habituellement hébergé sur le port 25 d'un serveur. Utiliser netcat ou gnetcat pour vous connecter sur ce port et envoyer un mail simple depuis votre terminal.

c. Écrire un script automatisant la séquence d'échange précédente et permettant à l'utilisateur de saisir l'expéditeur, le destinataire et le message à envoyer.

14 Utiliser des adresses IP avec la norme CIDR

→ FICHE 29

À l'intérieur d'un même réseau, toute machine reçoit une adresse IP unique. Il existe deux formats :

- une adresse de longueur 4 octets → IPv4 ;
- une adresse de longueur 6 octets → IPv6.

1. Expliquer pourquoi un codage sur 6 octets a été introduit il y a quelques années.
2. Les adresses IPv4 sont souvent notées en décimal, la valeur de chaque octet étant séparée par un point. Voici par exemple une adresse IP : 188.72.99.81. Écrire une fonction Python qui convertit en binaire une adresse IPv4 donnée sous forme d'une chaîne de caractères.

On pourra utiliser la méthode split et son homologue join, et commencer par fabriquer une fonction qui transforme un entier inférieur à 255 en une chaîne de 8 bits.

3. L'adresse IPv4 contient deux parties : la première partie identifie le réseau auquel appartient la machine et la seconde identifie la machine dans le réseau. Deux machines peuvent ainsi savoir si elles sont dans le même réseau. L'adresse du réseau commun est connue car elles sont données en suivant la méthode CIDR (*Classless Inter-Domain Routing*). Par exemple, si j'ai besoin dans mon école de 100 adresses, on peut me donner l'adresse 206.65.12.0/25 : cela signifie que les 25 bits de poids forts seront les mêmes pour toutes les machines du réseau, et cela nous laisse 7 bits pour distinguer les machines dans ce réseau. La valeur 25, ici, est appelée le masque.

- a. Est-ce assez pour donner une adresse aux 100 machines ?
- b. Imaginer un moyen de vérifier que deux machines sont dans le même réseau si l'on connaît leur adresse selon la méthode CIDR. Le coder en Python.

15 Tester et écrire du code assembleur

→ FICHE 27

Chaque processeur a son langage assembleur. Un langage assez simple et bien documenté est celui des processeurs MIPS qui ont été principalement utilisés dans des systèmes embarqués. Vous pouvez installer le simulateur MARS (<http://courses.missouristate.edu/KenVollmar/MARS/>) si vous disposez de Java.