Systèmes - Windows & Linux

OS - Shell 2 (Fichiers)

18 janvier 2022

Version 1

Fabrice BOISSIER <fabrice.boissier@epita.fr>

Copyright

Table des matières

1 Consignes Générales **IV**

2 Format de Rendu **V**

3 Aide Mémoire **VI**

4 Cours **1**

4.1 Noms, liens, type . . . 1
4.1.1 Nom et chemin - basename & dirname . . . 1
4.1.2 Liens - ln . . . 1
4.1.3 Type - file . . . 3
4.2 Recherche de fichiers . . . 4
4.2.1 Locate . . . 4
4.2.2 Find : Recherche . . . 4
4.2.3 Find : Exécution . . . 5
4.3 Date, Mesure de temps, Repos . . . 6
4.3.1 Date et heure - time, cal . . . 6
4.3.2 Mesure du temps d'exécution - time . . . 6
4.3.3 Mise en attente - sleep . . . 6
4.4 Partitions et disques . . . 7
4.4.1 Disk Usage - du . . . 7
4.4.2 Disk Free - df . . . 7
4.4.3 Copie de blocs/de contenu/d'octets - dd . . . 7
4.4.4 Formatage vers un système de fichiers - mkfs . . . 9
4.4.5 Point de montage - mount & umount . . . 10

1 Consignes Générales

Les informations suivantes sont très importantes :

Le non-respect d'une des consignes suivantes entraînera des sanctions pouvant aller jusqu'à la multiplication de la note finale par 0.

Ces consignes sont claires, non-ambiguës, et ont un objectif précis. En outre, elles ne sont pas négociables.

N'hésitez pas à demander si vous ne comprenez pas une des règles.

Consigne Générale 0 : Vous devez lire le sujet.
Consigne Générale 1 : Vous devez respecter les consignes.
Consigne Générale 2 : Vous devez rendre le travail dans les délais prévus.

Consigne Générale 3 : Le travail doit être rendu dans le format décrit à la section Format de Rendu.

Consigne Générale 4 : Le travail rendu ne doit pas contenir de fichiers binaires, temporaires, ou d'erreurs (`*~`, `*.o`, `*.a`, `*.so`, `*#*`, `*core`, `*.log`, `*.exe`, binaires, ...).

Consigne Générale 5 : Dans l'ensemble de ce document, la casse (caractères majuscules et minuscules) est très importante. Vous devez strictement respecter les majuscules et minuscules imposées dans les messages et noms de fichiers du sujet.

Consigne Générale 6 : Dans l'ensemble de ce document, **`nom1-nom2`** correspond à la combinaison des deux noms de votre binôme (par exemple pour Fabrice BOISSIER et Mark ANGOUSTURES, cela donnera **`boissier-angoustures`**).

Consigne Générale 7 : Dans l'ensemble de ce document, le caractère ␣ correspond à une espace (s'il vous est demandé d'afficher ␣␣␣, vous devez afficher trois espaces consécutives).

Consigne Générale 8 : Tout retard, même d'une seconde, entraîne la note non négociable de 0.

Consigne Générale 9 : La triche (échange de code, copie de code ou de texte, ...) entraîne **au mieux** la note non négociable de 0.

Consigne Générale 10 : En cas de problème avec le projet, vous devez contacter le plus tôt possible les responsables du sujet aux adresses mail indiquées.

Conseil : N'attendez pas la dernière minute pour commencer à travailler sur le sujet.

2 Format de Rendu

Responsable(s) du projet :	**Fabrice BOISSIER** `<fabrice.boissier@univ-paris1.fr>`
Balise(s) du projet :	**`[OS][TP2]`**
Nombre d'étudiant(s) par rendu :	1
Procédure de rendu :	Pas de rendu
Nom du répertoire :	`login.x-TP2`
Nom de l'archive :	`login.x-TP2.tar.bz2`
Date maximale de rendu :	Pas de rendu
Durée du projet :	Pas de rendu
Architecture/OS :	Linux - Ubuntu (x86_64)
Langage(s) :	sh
Compilateur/Interpréteur :	**`/bin/bash`**
Options du compilateur/interpréteur :	

L'arborescence attendue pour le projet est la suivante :

```
login.x-TP2/
login.x-TP2/AUTHORS
login.x-TP2/README
login.x-TP2/
login.x-TP2/src/
```

Les fichiers suivants sont requis :

`AUTHORS`	contient le(s) nom(s) et prénom(s) de(s) auteur(s).
`README`	contient la description du projet et des exercices, ainsi que la façon d'utiliser le projet.

3 Aide Mémoire

Le travail doit être rendu au format ***.tar.bz2***, c'est-à-dire une archive **bz2** compressée avec un outil adapté (voir **`man 1 tar`** et **`man 1 bz2`**).
Tout autre format d'archive (zip, rar, 7zip, gz, gzip, ...) ne sera pas pris en compte, et votre travail ne sera pas corrigé (entraînant la note de 0).

Dans ce sujet précis, vous ferez du code en script shell, qui affichera les résultats dans le terminal (et pourra donc être redirigé dans un fichier texte).

4 Cours

Commandes étudiées :	`sh`, `bash`, `basename`, `dirname`, `ln`, `file`, `locate`, `find`, `date`, `cal`, `time`, `sleep`, `du`, `df`, `dd`, `mkfs`, `mount`, `umount`
Builtins étudiées :	
Notions étudiées :	Shell, Liens, Recherche de Fichiers, Dates, Disques & Partitions, Systèmes de Fichiers

4.1 Noms, liens, type

4.1.1 Nom et chemin - basename & dirname

Pour obtenir le nom de la ressource/cible pointée par un chemin, il faut utiliser la commande `basename`.

```
basename /usr/bin/ls
basename /home
basename ..
basename ~
basename ../../home/
```

L'inverse, pour afficher le chemin vers la ressource/cible pointée, se fait avec la commande `dirname`.

```
dirname /usr/bin/ls
dirname /home
dirname ..
dirname ~
dirname ../../home/
```

4.1.2 Liens - ln

Il est possible de référencer plusieurs fois un même fichier, pour cela, sur UNIX, on crée des *liens*. Il existe des liens *physiques* (*hard link*) et des liens *symboliques* (*symlink*).

Les **liens physiques** sont des références vers l'i-node du fichier visé : On crée dans le dossier une référence vers l'i-node du fichier visé (il n'y a qu'un seul i-node au total). Quand le compteur de références atteint 0 après des `rm`, on supprime l'i-node.
Les **liens symboliques** sont des i-nodes supplémentaires créés qui pointent vers un chemin précis (l'i-node pointera vers des blocs de données contenant le chemin). Si on déplace

(ou supprime) la cible d'un lien symbolique, le lien sera encore là, mais renverra une erreur. Inversement, si on supprime le lien, le fichier pointé n'est pas impacté.

Voici comment créer des liens physiques (`ln`) :

```
cd MonDossier
touch TestLink
echo "Pouet" > TestLink
ln TestLink liendur1
ln TestLink liendur2
echo "Glop" > liendur1
cat liendur2
rm TestLink
rm liendur1
cat liendur2
mv liendur2 TestLink
```

Voici comment créer des liens symboliques (`ln -s`) :

```
ln -s TestLink liensym1
ln -s TestLink liensym2
cat TestLink
echo "Toto" > liensym1
cat TestLink
cat liensym2
rm liensym1
cat TestLink
cat liensym2
rm TestLink
cat liensym2
rm liensym2
```

Il est bien évidemment possible de créer des liens sur des dossiers :

```
ls /usr/bin
ln -s /usr/bin lienusrbin
pwd
cd lienusrbin
pwd
ls
cd ..
pwd
rm lienusrbin
```

Le lien créé donne accès au dossier ciblé, tout en laissant l'accès "dossier parent" au dossier précédent (et non pas celui de l'arborescence réelle). Attention à ne pas faire de `rm -rf` sur des liens ! Mais bien *uniquement* sur le lien.

4.1.3 Type - file

Il est possible de tenter de déterminer le type de fichier grâce à la commande `file`. Celle-ci recherche plusieurs indices dans un certain ordre (par exemple, l'extension du nom de fichier, puis les premiers octets du fichier, et éventuellement d'autres octets) afin de déterminer le format des données contenues dans le fichier. *(dans l'exemple qui suit, nous allons générer des images JPEG, PNG, et bitmap avec ImageMagick que vous pouvez installer avec cette commande :* ***apt-get install imagemagick****)*

```
file /etc/passwd
convert -size 100x200 xc:gray +noise gaussian image.jpg
file image.jpg
convert -size 100x200 xc:gray +noise gaussian image.png
file image.png
convert -size 100x200 xc:gray +noise gaussian image.bmp
file image.bmp
```

4.2 Recherche de fichiers

4.2.1 Locate

La commande `locate` permet de retrouver des fichiers, en utilisant par défaut le mode globbing. Cette commande utilise une base de données à construire/mettre à jour régulièrement (voir les `updatedb` et autres scripts similaires décrits dans le manuel de `locate` en section *See Also*), elle ne reflète donc pas en temps réel le contenu du système de fichier.

```
locate ls
locate /usr/bi
locate passwd
locate pass
```

4.2.2 Find : Recherche

Une autre commande (`find`) permet d'énumérer tous les fichiers de l'arborescence (ou une partie), et filtrer les résultats ou exécuter d'autres commandes à partir des résultats.

```
find .
find . -name 'f* '
find /usr
```

Les options `-atime` et `-amin` recherchent les fichiers *accédés* durant les N dernières 24h ou minutes.
Les options `-mtime` et `-mmin` recherchent les fichiers *modifiés* durant les N dernières 24h ou minutes.
Les options `-ctime` et `-cmin` recherchent les fichiers dont les statuts ont été *modifiés* durant les N dernières 24h ou minutes.
Les nombres après correspondent à la durée : -60 signifie *durant les 60 dernières 24h ou minutes*, +15 signifie *depuis au moins les 15 dernières 24h ou minutes.*

```
touch file
find . -atime +1
find . -mmin -60
chmod 755 file
find . -cmin -1
```

D'autres options permettent : `-uid` chercher les fichiers à l'UID en question, `-user` chercher les fichiers de l'utilisateur en question, `-nouser` chercher les fichiers dont l'UID n'existe pas.
De façon similaire : `-gid` chercher les fichiers au GID en question, `-group` chercher les fichiers du groupe en question, `-nogroup` chercher les fichiers dont l'UID n'existe pas.

```
find . -uid 1042
find . -user metalman
```

```
find . -gid 1000
find . -group users
```

Il est aussi possible de filtrer par type avec l'option `-type` et les arguments **f** (fichiers), **d** (répertoires), **l** (liens symboliques), **s** (sockets), **p** (tubes nommés), **b** (spéciaux mode blocs), **c** (spéciaux mode caractères).

```
find . -type d
find . -type l
```

La profondeur dans l'arborescence peut aussi être limitée avec `-maxdepth` (profondeur maximale recherchée) et `-mindepth` (profondeur minimale recherchée).

```
find / -maxdepth 1
find /var -mindepth 2
```

Évidemment, le but de `find` est de mélanger toutes ses options simultanément :

```
find .. -maxdepth 4 -type d -uid 1003 -atime -1 -name 'M* '
```

4.2.3 Find : Exécution

La commande `find` permet également d'appeler des programmes externes sur chacun des arguments listés.

```
find . -exec ls {} \;
```

Ici, le paramètre `-exec` permet d'appeler la commande `ls`. Il existe deux façons d'utiliser `-exec`, avec un **;** (point virgule) ou avec un + (plus). Dans les deux cas, on peut rappeler chaque résultat à l'aide des caractères `{}`.
Le **;** permet d'appeler une fois la commande pour chaque résultat.
Le + permet de créer une chaîne de caractères contenant tous les résultats en une seule ligne.

```
find . -exec echo {} \;
find . -exec echo {} +
```

Ainsi, il est possible de mixer de nombreuses options pour faire des commandes complexes :

```
find -atime -1 -exec echo -e {} {} \\n \;
find . -name "*~" -exec ls {} \;
find . -name "*~" -exec rm -f {} \;
find . -name ".svn*" -exec rm -rf {} \;
```

4.3 Date, Mesure de temps, Repos

4.3.1 Date et heure - time, cal

Récupérer l'heure dans un format précis est parfois très important pour pouvoir nommer ses fichiers de logs et les faire tourner (ou les supprimer) s'ils sont trop anciens. La commande `date` permet d'afficher la date et l'heure dans votre langue, avec certaines options, il est possible d'indiquer le format souhaité (`%Y` donne l'année, `%m` donne le mois, `%d` donne le jour du mois, `%H` donne l'heure au format 24h, `%M` donne la minute de l'heure courante, `%S` donne la seconde de la minute courante). L'option `-u` donne l'heure universelle (UTC).

```
date
date -u
date '+%Y-%m-%d-%H:%M:%S'
```

4.3.2 Mesure du temps d'exécution - time

Il est possible de mesurer le temps d'exécution d'un programme avec la commande `time`. On obtient également le temps processeur passé en *userland* (mode normal du processeur) et le temps processeur passé en *kerneland* (mode superviseur du processeur).

```
time ls
time pwd
```

4.3.3 Mise en attente - sleep

La commande `sleep` permet de lancer un programme qui ne fait... rien, hormis attendre. Le paramètre donné est le nombre de secondes où le programme ne fait rien/le processus se met en attente.

```
sleep 1
sleep 5
```

4.4 Partitions et disques

4.4.1 Disk Usage - du

Pour connaitre l'espace consommé par un dossier, la commande `du` est très utile. Plusieurs arguments utiles : `-a` détaille les fichiers, `h` résultat affiché dans un format humainement compréhensible (approximativement...), `--block-size=N` affiche le résultat sous forme de multiple de la taille N, `-s` affiche un total pour chaque argument.

```
mkdir DossierVide
du DossierVide
du -h .
du -ah .
mkdir DossierPlein
echo "AAAAAAAAAAAAH" > DossierPlein/BigFile
i=0
while [ $i -ne 10000 ]; do
echo "AAAAAAAAAAAAH" >> DossierPlein/BigFile
i=$(( $i + 1 ))
done
du DossierPlein
du -h DossierPlein
du -ah --block-size=32 DossierPlein
```

4.4.2 Disk Free - df

Pour connaitre l'espace libre sur son/ses disques, il faut utiliser la commande `df`. L'option `-a` affiche l'espace sur tous les disques, l'option `-h` donne le résultat sous forme lisible, `-i` affiche le résultat en terme d'i-nodes, et `-l` (lettre L) permet de ne gérer que les disques locaux.

```
df -h
df -ah
df -il
```

4.4.3 Copie de blocs/de contenu/d'octets - dd

La commande `dd` permet de copier le contenu d'un fichier, voire d'une partition, vers un autre fichier (ou une autre partition). Ses paramètres principaux sont : `-if=`*input_file* (le fichier lu en entrée), `-of=`*output_file* (le fichier vers où écrire). Les fichiers mis en entrée et en sortie peuvent également être des périphériques tels que `/dev/sdX` (un disque) ou encore `/dev/zero` (un périphérique virtuel n'envoyant que des 0). *Afin de copier des données vers une partition stockée sur un disque (donc un périphérique aux yeux d'UNIX),*

il est nécessaire d'être administrateur.

Concrètement, il est possible de copier un fichier vers un autre de deux façons :

```
touch file_in file_out # Creation de l'exemple
cp -f file_in file_out
dd -if=file_in -of=file_out
```

On peut potentiellement extraire une image ISO depuis un CD ou un DVD ainsi :

```
dd if=/dev/cdrom of=image-cd.iso
```

D'autres paramètres important permettent d'optimiser la lecture/écriture depuis/vers différents périphériques : `bs=`*taille_bloc* (taille des blocs lus/écrits en octets), `count=`*nb_blocs* (combien de blocs seront lus/écrits). Ces paramètres permettent de transférer des blocs de données (une suite d'octets) selon une taille définie par l'utilisateur. La taille des blocs peut être très importante selon l'usage (certains périphériques communiquent en lisant les données bloc par bloc, d'autres caractère par caractère).

Pour remplir un fichier avec exactement 1024 octets de 0, il suffit par exemple d'envoyer 2 fois (`count=2`) un bloc de 512 octets (`bs=512`) ne contenant que des 0, c'est-à-dire dupliquer une source de 0 (`/dev/zero`) :

```
dd if=/dev/zero of=fichier-vide.txt bs=512 count=2
```

On peut également extraire le MBR (Memory Boot record) de son disque, si l'ordinateur ne démarre pas avec un UEFI et GPT :

```
dd if=/dev/sda of=mbr.bin bs=512 count=1
```

On peut également sauter les N premiers blocs du fichier d'entrée avec `skip=`*nb_blocs*, et démarrer l'écriture en sortie à partir d'un certain bloc avec `seek=`*nb_blocs*.

Par exemple pour découper un fichier en tranches de 10 Ko, on peut répéter ces opérations :

```
dd bs=10K count=10K if=big_file of=big_file-part1
dd bs=10K count=10K skip=10K if=big_file of=big_file-part2
dd bs=10K count=10K skip=20K if=big_file of=big_file-part3
dd bs=10K count=10K skip=30K if=big_file of=big_file-part4
```

1. Scriptez le découpage d'un fichier de n'importe quelle taille en 4 tranches égales (la dernière peut ne pas être égale aux 3 autres). Vous ajouterez le numéro de la partie en fin de nom tel que : `"-partX"` (où `X` correspond au numéro de la partie).
2. Scriptez le découpage d'un fichier de n'importe quelle taille en autant de tranches de 10 Ko que nécessaire (la dernière peut ne pas faire 10 Ko).

4.4.4 Formatage vers un système de fichiers - mkfs

Afin qu'un support physique soit lisible et manipulable par un système d'exploitation, il faut que le support soit organisé d'une façon compréhensible par le système d'exploitation. En effet, il faut bien séparer l'organisation des données et des fichiers (la logique qui indique comment placer quoi/comment lire quoi), du support physique (un disque, une bande, une mémoire, ...), et des données en elles-mêmes (une liste de noms, une liste de notes, un ensemble d'images, un texte, ...). L'organisation des données sous forme de fichiers et dossiers (ou autre chose encore) est permise grâce au système de fichiers. De très nombreux systèmes de fichiers ont existé, dont certains sont encore en usage, et d'autres plus récents sont apparus pour des usages précis.
Les plus connus actuellement traitant des disques durs sont *FAT32*, *exFAT*, *NTFS*, *ext4*, *ext3*, *Reiser4*, *HFS+*.
D'autres systèmes de fichiers plus spécialisés existent : *NFS*, *AFS*, *ZFS*, ...

Sur Linux, la commande `mkfs` permet de formater une partition (une partie d'un disque), pour préparer celle-ci à être manipulée par un système d'exploitation ou tout autre logiciel ou bibliothèque. Plusieurs extensions à `mkfs` existent et permettent de préparer une partition pour divers systèmes de fichiers. Il suffit littéralement de lire l'extension associée au programme `mkfs` : `mkfs.fat`, `mkfs.ext3`, `mkfs.ext4`, ...

Il est possible de simuler une partition dans un fichier. Pour cela, il suffit de préparer un fichier vide de 4 Mo par exemple. Étant donné que les disques actuels et autres mémoires travaillent avec des blocs de taille 4096 octets, nous utiliserons cette valeur. Pour constituer un exemple lisible, nous ferons également une toute petite partition de 64 Ko.

```
dd if=/dev/zero of=partition.img bs=4096 count=1000
dd if=/dev/zero of=partition-test.img bs=4096 count=16
```

Pour observer les différences qui apparaitront dans le fichier, on peut utiliser un éditeur hexadécimal. Sur Linux, n'hésitez pas à installer `xxd` (CLI), `hexedit` (CLI) , ou encore `ghex` (GNOME Hex Editor - GUI).

- `xxd` affiche un dump en hexadécimal
- `hexedit` est un éditeur intéractif dans un terminal (F1 pour l'aide, F2 pour sauvegarder, F3 pour charger, Ctrl-X pour sauvegarder et quitter, Ctrl-C pour quitter sans sauvegarder)
- `ghex` est un éditeur intéractif en mode graphique

Observer le contenu de votre fichier avec un de ces éditeurs : tous les octets sont à 0.

Nous pouvons maintenant formater le fichier en FAT32 :

```
mkfs.fat partition-test.img
```

Observez le résultat avec un éditeur hexadécimal, et vous verrez certaines modifications apportées au fichier.

Il est évidemment possible de formater un périphérique en donnant un chemin tel que `/dev/sdX`.

4.4.5 Point de montage - mount & umount

Afin d'utiliser le contenu d'une partition dans l'arborescence UNIX, il est nécessaire de la "monter" avec le programme `mount`. Ce programme prend au moins deux paramètres (et essaye d'en déterminer d'autres automatiquement) : le chemin vers la partition contenant le système de fichier à monter (potentiellement `/dev/sdX` ou encore un fichier image), et l'endroit dans l'arborescence UNIX où attacher le système de fichier contenu (généralement `/mnt`, ou `/media/sdX`).

Essayez de monter la partition nouvellement créée. Si nécessaire, créez un dossier `/mnt` en tant qu'administrateur, et autorisez tous les utilisateurs à pouvoir y accéder au moins en lecture et exécution. Les opérations de montage/démontage demande souvent à être administrateur : passez en mode administrateur avant d'exécuter les lignes suivantes :

```
mount partition-test.img /mnt
```

Vous pouvez maintenant modifier le contenu de la partition en vous y déplaçant.

```
cd /mnt
touch fichier1
echo "ABCD" > fichier1
cat fichier1
```

Afin de démonter la partition, et libérer l'accès à notre fichier "*partition-test.img*", utilisez la commande `umount` en administrateur également. Attention, si vous avez encore un programme ouvert utilisant un fichier sur la partition, ou simplement que votre shell est positionné dedans, le démontage sera refusé.

```
cd
umount /mnt
```

Vous pouvez maintenant quitter le mode administrateur, et observer le résultat dans le fichier "*partition-test.img*" avec un éditeur hexadécimal. Vous trouverez une en-tête (un *header*) décrivant la partition, puis beaucoup plus loin, la description du fichier "*fichier1*", et enfin, dans une troisième zone de la partition, le contenu "*ABCD*" du "*fichier1*".