Introduction aux systèmes d'exploitation (IS1)

Corrigé succinct de l'examen du 16 Janvier 2008

Exercice 1 – *Questions rapides*

Système de fichiers.

1. `../bin/bash`
2. `/home/martin/dubois/chanson.mp3`
3. `cp mozart.mp3 ~/MP3/musique.mp3`
4. `mkdir ~/examIS1`
5. `rm -R /tmp/a-jeter` (en effet, `rmdir` ne permet d'effacer que des répertoires vides).

Système de fichiers : droits.

6. `rep` est un répertoire, le propriétaire (michel) peut en voir le contenu, en changer le contenu, et le traverser. Le groupe (etudiants) peut en voir le contenu. Tout le monde peut en voir le contenu et le traverser. Il occupe 512 kilo-octets sur le disque. il a été modifié pour la dernière fois le 5 janvier de cette année à 18 :06.
7. `chmod u+w fic.sh` ou `chmod 344 fic.sh`
8. Michel et tous les non-membres du groupe etudiants car il faut avoir le droit d'exécution sur `rep`.
9. Michel car il faut avoir le droit d'écriture sur `sous-rep`.
10. Personne car il faut avoir le droit d'exécution ET le droit de lecture sur le fichier `fic.sh`.

Système de fichiers : i-noeuds et liens

11. Les fichiers en lien physique sont ceux qui possèdent le même numéro d'i-noeud :
    `a, b, e, rep/a, rep/d,` d'une part et `c, g, rep/c, rep/f,` d'autre part.

Processus.

12. Une `interruption` est un événement au niveau matériel qui interrompt le processus en cours d'exécution dans le processeur et passe la commande à un programme spécial appelé `traitant`. Ce mécanisme est essentiel pour gérer la communication entre processus dans un OS multi-tâche.
13. `kill -sig pid` : envoie le signal `sig` au processus identifié par le `pid`
    (ce qui permet notamment d'endormir, de relancer ou de tuer un processus).
14. Chaque commande renvoie une valeur de retour à la fin de son exécution : `0` en cas de succès, une autre valeur en cas de problème lors de l'exécution.

15. `if commande1; then commande2; else commande 3;fi`
    Si la valeur de retour de `commande1` est `0`, alors `commande2` s'exécute, sinon c'est `commande3` qui s'exécute.
    `while commande1; do commande2; done`
    Tant que la valeur de retour de `commande1` est `0`, `commande2` s'exécute. La boucle s'arrête lorsque la valeur de retour de `commande1` n'est plus `0`.
16. `javac UneClasse.java && java UneClasse`
    ou `if javac UneClasse.java; then java UneClasse; fi`

Redirections.

17. `a.txt` contiendra la liste des fichiers de tous les sous-répertoires de `/usr` que l'utilisateur a le droit de lister.
    `b.txt` contiendra la liste des messages d'erreur (tentative de lister un répertoire dont l'utilisateur ne possède pas le droit de lecture).
18. Cette commande écrit dans le fichier `res.txt` la liste de tous les processus `xclock` lancés depuis le shell courant.
19. Cette commande affiche par ordre alphabétique et sans répétition, tous les mots écrits en majuscules contenus dans le fichier `Cyrano.txt`.

Tests et Boucles.

20. Cette commande teste si le fichier source `Main.java` existe dans le répertoire courant, est lisible, et n'a pas encore été compilé (i.e le fichier `Main.class` n'existe pas ou est plus vieux que `Main.java`).
21. La variable d'environnement ayant pour valeur le chemin du répertoire personnel de l'utilisateur est la variable `HOME`.
    La commande permettant de tester si le répertoire courant est le répertoire personnel, et si on a le droit d'écriture sur ce répertoire est : `test $PWD=$HOME -a -w $HOME`
22. `echo $((‘cat fic‘ + $var))`
23. ```
    if test -L fic;
    .    then mv fic C_fic;
    .    elif test -d fic;
    .        then mv fic R_fic;
    .        else mv fic R_fic;
    . fi;
    ```
24. ```
    for fic in * do
    .    if test -L fic;
    .        then mv fic C_fic;
    .        elif test -d fic;
    .            then mv fic R_fic;
    .            else mv fic R_fic;
    .    fi;
    .done
    ```

Expressions régulières.

25. `8, ma7, papapama2, mapa3, mama`
26. `[A-Z][a-zA-Z0-9]*`
27. `[0-9]*[05]`

Divers.

28. `alias lr='ls -lr'`
Si l'on veut que cette commande soit exécutée à chaque fois que l'on lance un shell bash, il suffit de l'écrire dans le fichier `.bashrc` situé dans le répertoire personnel.

Exercice 2 – *Commandes simples et moins simples*

1. `ls *Book*`

2. `find . -name "*Book*"`

3. `ls *[A-Z]*[A-Z]*`

4.
```
for fic in *Book* ; do
.      echo "bonne année a tous" >>$fic ;
.done
```

5.
```
for fic in *Book* ; do
.      if test !-s $fic ;
.          then echo "Ce fichier me semble bien inutile !" >>$fic ;
.      fi ;
.done
```

Exercice 3 – *Comparateur de répertoires*

1. Commentaires des différentes lignes :
1. spécifie le chemin (absolu) du shell qui exécutera ce script,
2. on se déplace dans le sous-répertoire `rep1`,
3. on initialise à 0 le compteur de fichiers (variable `manque`)
4. début de la boucle `for` : la variable `nomfichier` parcourt tous les fichiers de `rep1`,
5. premier test : si ce fichier est un fichier ordinaire,
6. second test : et s'il n'y a pas de fichier de même nom dans `rep2`,
7. alors on incrémente le compteur,
8. fermeture du second test,
9. fermeture du premier test,
10. fermeture de la boucle `for`,
11. on affiche le nombre de fichiers qui manquent (valeur finale de la variable `manque`),
12. on revient dans le répertoire de départ.

Ce script détecte les noms de fichiers qui sont présents dans `rep1` mais pas dans `rep2`. Il affiche le nombre total de tels noms à la fin.

2.
```
1  #!/bin/bash

2   cd rep1
3   manque=0
+   differents=0
4   for nomfichier in *; do
5     if [ -f "$nomfichier" ]; then
6         if [ ! -f "../rep2/$nomfichier" ]; then
7                manque=$(($manque + 1))
+           elif  cmp $nomfichier ../rep2/$nomfichier;
+               then echo " $nomfichier identique dans rep1 et rep2"
+               else echo " $nomfichier differents dans rep1 et rep2"
+                    differents=$(($differents+1))
8         fi
9     fi
10   done
11   echo "Il manque $manque fichiers"
+    echo "Il y a $differents fichiers différents"
12   cd ..
```

Les lignes rajoutées sont précédées d'un +.

3.
```
1  #!/bin/bash

+ if  test (!-d rep1) -o (!-d rep2);
+  then echo "erreur"
+  exit 1
+ fi

2   cd rep1
3   manque=0
4   for nomfichier in *; do
5     if [ -f "$nomfichier" ]; then
6         if [ ! -f "../rep2/$nomfichier" ]; then
7                manque=$(($manque + 1))
8           fi
9       fi
10   done
11   echo "Il manque $manque fichiers"
12   cd ..
```

4. Il suffit de remplacer dans le script, `rep1` par $1 et `rep2` par $2, qui désignent respectivement le premier et le deuxième arguments passés au script.