Introduction aux systèmes d'exploitation (IS1)

Examen du 16 Janvier 2008

Durée: 2 heures

- Les notes ainsi que les énoncés de TP sont autorisés.
- **Les livres sont interdits.**
- Le barème est donné à titre indicatif et est susceptible d'être modifié.
- La note prendra compte de la clarté des explications.
- Il ne faut pas recopier les énoncés.
- Ce sujet fait 4 pages (plus annexe).

Exercice – *Questions rapides - [10 points]*

Cet exercice comporte des questions courtes auxquelles vous devez vous efforcer de répondre de façon **justi e** mais **concise** (deux ou trois lignes).

Il suffira de répondre correctement à 20 questions pour obtenir la totalité des points. En cas de réponse a plus de 20 questions, seules les 20 premières réponses seront corrigées et les autres ne compteront pas dans la note.

Syst me de chiers.

. Si l'on se trouve dans le répertoire `/usr/local/eclipse`, quel est le nom relatif du fichier `/usr/local/bin/bash` ?

. En sachant que le répertoire personnel de l'utilisateur `martin` est situé dans le répertoire `/home/`, quel est le nom absolu le plus court possible du fichier `~martin/if1/../../martin/./dubois/is1/../chanson.mp3` ?

. Si l'on se trouve dans le répertoire `/tmp`, quelle commande taper pour recopier le fichier `mozart.mp3` (qui se trouve aussi dans `/tmp`) sous le nom `musique.mp3` dans le sous-répertoire MP3 de son répertoire personnel ?

4. Toujours depuis `/tmp`, quelle commande taper pour créer un sous-répertoire `examIS1` dans son répertoire personnel sans changer de répertoire courant ?

. Écrire une commande qui permet d'effacer le répertoire `/tmp/a-jeter/` si l'on suppose que ce répertoire n'est pas vide (et que l'on dispose des droits suffisants) ?

Syst me de chiers : droits. On donne le listing suivant :

```
$ ls -lR
---xr--r--  1 michel  etudiants   12  Jan 5  18:06 fic.sh
drwxr--r-x  3 michel  etudiants  512  Jan 5  18:06 rep

./rep:
-rw-r-----  1 michel  etudiants  306  Jan 5  18:06 fic.txt
drw-----w-  2 michel  etudiants  512  Jan 5  18:06 sous-rep

./rep/sous-rep:
```


6. La ligne concernant le répertoire `rep` contient un certain nombre d'informations. En particulier, le chiffre `3` désigne le nombre de liens physiques sur ce répertoire. A quoi correspondent les autres informations ?

7. Quelle commande doit émettre le propriétaire pour pouvoir modifier le contenu du fichier `fic.sh` ?

8. Quels utilisateurs peuvent entrer dans le répertoire `rep` ?

. Quels utilisateurs peuvent supprimer le répertoire `sous-rep` ?

. Quels utilisateurs peuvent exécuter le script `fic.sh` ?

Syst me de chiers : inoeuds et liens

. Dans l'arborescence suivante, quels sont les fichiers en lien physique ?

```
$ ls -iR
.:
1507992 a  1507999 c  1507992 e  1507999 g
1507992 b  1508001 d  1508003 f  1508004 rep

./rep:
1507992 a  1508008 b  1507999 c  1507992 d  1508006 e  1507999 f 1508010 g
```

Processus.

. Expliquez la notion d'interruption.

. Expliquez ce que fait la commande `kill`.

4. Expliquez la notion de valeur de retour.

. Expliquez en quoi la valeur de retour intervient dans une construction `if` ou `while`.

6. Écrivez une commande qui compile le fichier java `UneClasse.java`, puis exécute le programme correspondant **uniquement** si la compilation n'a pas produit d'erreurs.

Redirections.

7. Quel sera le contenu des fichiers `a.txt` et `b.txt` après la commande `ls -R /usr >a.txt 2>b.txt` ?

8. Que fait la commande `jobs | grep "xclock" > res.txt` ?

. Qu'affiche la commande `grep -ow "[A-Z]*" cyrano.txt | sort | uniq` ?

Tests et boucles.

. Que teste la commande suivante :
`test -r Main.java -a ( ! -e Main.class -o ( Main.java -nt Main.class ) )`

. Quelle variable d'environnement a pour valeur le chemin du répertoire personnel de l'utilisateur ? Écrire une commande qui teste si le répertoire courant est le répertoire personnel et si on a le droit d'écrire dans ce répertoire.

. En utilisant des tests appropriés, écrivez une commande qui ajoute un des préfixes suivants au nom du fichier `fic` selon sa nature :
  – `C_` si `fic` est un lien symbolique ;
  – `R_` si `fic` est un répertoire ;
  – `A_` pour les autres fichiers.

. En utilisant la question précédente écrivez une boucle qui ajoute les préfixes `C_`, `R_` ou `A_` au nom de chaque fichier du répertoire courant selon sa nature.

Expressions r guli res. Les questions suivantes portent sur les expressions régulières telles que celles qu'utilise la commande `grep`.

4. Donnez 5 mots différents qui correspondent à l'expression régulière `([mp]a)*[1-9]` ?

. En Java, il est de bon usage que les noms de classes commencent par une lettre majuscule, suivie d'une séquence quelconque de lettres minuscules ou majuscules et de chiffres. Écrivez une expression régulière qui représente les noms de classes de cette forme.

6. Écrivez une expression régulière qui représente tous les entiers multiples de 5.

Divers.

7. Comment définir une commande `lr` équivalente à `ls -lr` ? Comment faire pour pouvoir l'utiliser même après un redémarrage et depuis n'importe quel terminal ?

8. Un fichier `fic` contient un entier, une variable `var` a pour valeur un entier. Écrivez une commande qui calcule et affiche la somme de ces deux entiers.

Exercice – *Commandes simples et moins simples - [5 points]*

Supposons que l'on se situe dans le répertoire Biblio qui contient un grand nombre de fichiers et dossiers.

. Écrivez une commande qui permet d'afficher la liste des fichiers du répertoire Biblio dont le nom contient "Book".

. Écrivez une commande qui affiche cette fois-ci la liste des fichiers du répertoire Biblio et de sa sous-arborescence dont le nom contient "Book"

. Écrivez une commande qui affiche la liste des fichiers de Biblio dont le nom contient au moins deux majuscules.

4. Écrivez une commande qui permet de rajouter à la fin de chacun des fichiers correspondant à la première question la ligne "Bonne année à tous".

. Écrivez une commande afin de rajouter à chacun des fichiers précédents la ligne "Ce fichier me semble bien inutile !" si le fichier est vide.

Exercice – *Comparateur de répertoires - [5 points]*

L'objectif de cet exercice consiste à écrire un script bash qui compare le contenu de deux répertoires et affiche le résultat de comparaison.

. On commence par le script suivant (fichier `cmp_dir.sh`) :

```
1  #!/bin/bash

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 ..
```

Commentez chaque ligne du script, puis expliquez en une phrase ce que fait ce script précisément.

. Le répertoire `rep2` peut contenir un fichier qui a le même nom qu'un fichier du répertoire `rep1`. Cela ne veut pas dire que les deux fichiers ont le même contenu. Quelles commandes faut-il ajouter au script précédent pour qu'il compte aussi les fichiers ayant le même nom mais un contenu différent dans les répertoires `rep1` et `rep2`. N'oubliez pas d'afficher le résultat.

. Le script actuel ne vérifie pas l'existence des répertoires à comparer. Modifiez le script pour faire une telle vérification de la manière suivante :
  – avant de faire la comparaison, on vérifie que `rep1` et `rep2` sont bien des sous-répertoires du répertoire courant ;
  – Si ce n'est pas le cas, on affiche un message d'erreur et on termine l'exécution du script avec la valeur de retour 1. (Pour cela vous pouvez utiliser la commande `exit 1`).

4. Un script qui compare toujours les répertoires de noms `rep1` et `rep2` n'est pas très utile. Comment faut-il le modifier pour qu'il compare deux répertoires quelconques passés en paramètre de ligne de commande ? (Vous pouvez supposer qu'il s'agit de sous-répertoires du répertoire courant.)

Annexe

Expressions r guli res syntaxe de grep)

Voici une liste de symboles utilisables pour grep :

.	un seul caractère quelconque
*	répétition, zéro ou plusieurs fois, du caractère précédent. ab* représente l'ensemble { a,ab,abb,abbb,...}
+	répétition (au moins une fois) du caractère précédent ab+ représente l'ensemble { ab,abb,abbb,...}
?	l'élément précédent est facultatif ab?c représente l'ensemble {ac,abc}
^	début de ligne ^a représente toutes les lignes qui commencent par 'a'
$	fin de ligne a$ représente toutes les lignes qui finissent par 'a'
[...]	liste ou intervalle de caractères recherchés [abcd] représente l'ensemble {a,b,c,d} [a-d] représente l'ensemble {a,b,c,d}
[^ab]	listes des caractères interdits

Exemples :

- `grep abc fichier` recherche la chaîne `abc` dans toutes les lignes du fichier. Les lignes trouvées sont envoyées sur la sortie standard.
- `grep -o abc fichier` recherche toutes les occurrences de la chaîne `abc` dans toutes les lignes du fichier. Les occurrences trouvées sont envoyées sur la sortie standard
- `grep -w abc fichier` recherche le mot `abc` dans toutes les lignes du fichier (`abc` doit être un mot entier et non une partie d'un mot). Les lignes trouvées sont envoyées sur la sortie standard.
- `grep "^bonjour " fichier` recherche les lignes qui commencent par la chaîne `bonjour` suivie d'un espace dans `fichier`.

Manuel de la commande `test` extrait)

SYNOPSIS
`test [expr]`
DESCRIPTION
test renvoie une valeur `0` (vrai) ou `1` (faux) suivant l'évaluation de l'expression conditionnelle expr. Les expressions peuvent être unaires ou binaires. Les expressions unaires sont généralement utilisées pour examiner le statut d'un fichier. Il existe également des opérateurs de chaînes de caractères, et des opérateurs de comparaison numérique.

`-d fichier`
Vrai si le fichier existe et est un répertoire.

`-e fichier`
Vrai si le fichier existe.

`-f fichier`
Vrai si le fichier existe et est un fichier ordinaire.

`-L fichier`
Vrai si le fichier existe et est un lien symbolique.

`-p fichier`
Vrai si le fichier existe et est un tube nommé.

`-r fichier`
Vrai si le fichier existe et est lisible.

`-s fichier`
Vrai si le fichier existe et a une taille supérieure à zéro.

`-w fichier`
Vrai si le fichier existe et est accessible en écriture.

`-x fichier`
Vrai si le fichier existe et est exécutable.

`fichier1 -nt fichier2`
Vrai si fichier1 est plus récent (d'après les dates de modification) que fichier2.

`fichier1 -ot fichier2`
Vrai si `fichier1`