51IF1IS1 - Introduction aux systèmes d'exploitation

Examen de 1ère session 2008/2009

vendredi 16 janvier 2009 - durée: 3h00

A lire attentivement avant de commencer

L'énoncé est composé de 4 exercices. Vous devrez rendre deux copies à en-tête séparées (éventuellement complétées par des copies intermédiaires), ceci afin d'accélérer la correction de l'examen. La première copie comportera les exercices 1 et 2, la deuxième copie comportera les exercices 3 et 4.

Pour assurer l'anonymat des copies, nous vous demandons de :
– mettre votre nom et votre numéro d'étudiant dans les caches des copies à en-tête prévus à cet effet et de coller ces derniers,
– écrire un code que vous reporterez sur toutes vos copies.

L'énoncé comporte plusieurs variantes, merci d'indiquer **sur chaque copie à en-tête** le numéro de l'énoncé.

Comme il y a moins d'enseignants présents que d'amphis à surveiller et dans un souci d'équité il ne sera répondu à aucune question. Si vous pensez que l'énoncé comporte une erreur, vous pouvez la signaler (avec de brèves explications) sur votre copie ; si vous pensez que l'énoncé est ambigu, vous pouvez expliquez cette ambiguïté sur votre copie, et explicitez le choix que vous faites pour répondre à la question.

Les seuls documents autorisés sont les documents papiers autres que les livres. Les documents électroniques sont interdits, en particulier les téléphones, ordinateurs, PDA, etc.

Le barême est donné à titre indicatif.

Le décachetage et la consultation des copies se feront mercredi 21 janvier à 10h30 au département (salle à préciser).

énoncé A1

Exercice 1 – *Système de Gestion de Fichiers [3 points]*

Cet exercice comporte des questions courtes auxquelles vous devez vous efforcer de répondre de façon justifiée mais concise. Il suffira de répondre correctement à 6 questions

pour obtenir la totalité des points. En cas de réponse à plus de 6 questions, seules les 6 premières réponses seront corrigées et les autres ne compteront pas dans la note.

Les questions 1 à 5 portent sur la situation suivante :
Sur l'ordinateur `travail` (qui tourne sous un système de type `UNIX`) on a les renseignements suivants concernant certains utilisateurs (le symbole > est le prompt) :

```
> id watt
uid=234(watt) gid=23(puissance)
> id joule
uid=127(joule) gid=15(energie)
> id calorie
uid=128(calorie) gid=15(energie)
```

On a également les renseignements suivants concernant le système de gestion de fichiers :

```
> whoami
watt
> pwd
/home/energie/joule
> ls -ilad
300 drwxr-xr-x 6 joule energie   4096 2007-06-10 13:41 .
> ls -ilR
.:
total 16
310 drwxr-x--x 2 joule energie   4096 2009-01-03 22:13 bin
311 drwx--x--x 2 joule energie   4096 2008-01-30 13:27 Documents
307 drwxr-xr-x 2 joule energie   4096 2008-04-25 03:15 public_html
309 drwxr-x--- 2 joule energie   4096 2008-02-03 22:33 tmp

./bin:
total 0
312 -rw-r--r-- 1 joule energie   25 2008-02-01 15:15 mydisplay

./Documents:
total 0
315 -rw-r--r-- 2 joule energie   104102 2009-01-03 22:29 examenIS1.pdf

./public_html:
total 0
308 -rw-r--r-- 1 joule energie   5384 2008-09-08 12:45 index.html

./tmp:
total 0
314 -rw-r----- 1 joule energie   0 2009-01-03 22:14 brouillon
313 -rw-r--r-- 1 joule energie   40 2008-12-15 15:52 toto
>
```

```
> cd ~
> ls -ail
298 drwxr-xr-x 2 watt  puissance 4096 2007-08-03 14:49 .
132 drwxr-xr-x 3 root  root      4096 2007-06-02 14:43 ..
333 -rw-r--r-- 1 watt  puissance 103708 2009-01-03 15:15 examenIS1a.pdf
315 -rw-r--r-- 2 joule energie   104102 2009-01-03 22:29 examenIS1b.pdf
```

1. Parmi les utilisateurs connus, qui peut lire le fichier `/home/energie/joule/tmp/toto` ? Qui peut le modifier ?
2. Le répertoire `/home/energie/joule/bin` est plus récent que le fichier `mydisplay` qu'il contient. Est-ce possible ? Si oui, comment ; sinon, pourquoi ?
3. A qui appartient le fichier `~watt/examenIS1b.pdf` ? Qui peut le supprimer ? Expliquez comment ce fichier a pu arriver dans ce repértoire.
4. A quelles conditions un utilisateur peut-il voir le contenu d'un fichier du répertoire `/home/energie/joule/bin` ?
5. Que signifie le chiffre 2 dans le troisième champs de la ligne décrivant `/home/energie/joule/Documents/examenIS1.pdf` ?
6. Quels sont les renseignements contenus dans un i-nœud ?
7. Donnez une raison pour justifier qu'on ne peut pas faire de lien physique sur un répertoire.
8. Décrivez un procédé utilisé par un système de type `UNIX` pour s'assurer de l'obtention **rapide** d'un numéro d'i-nœud libre.

Exercice 2 – *La commande* `find` *[6 points]*

Vous trouverez dans l'annexe, des extraits de la page de manuel de la commande `find`, traduits en français.

1. Que font les lignes de commandes suivantes ?
   (i) `find / \! -name "*.c" -print`
   (ii) `find ~ -regex ".*[A-Z].*" -exec ls -l {} \;`
   (iii) `find /home -name "*~" -mmin +240 -exec echo {} \;`
2. En utilisant la commande `find`, écrivez une ligne de commandes affichant tous les répertoires de votre arborescence personnelle qui ont été créés il y a moins d'une heure.
3. En utilisant la commande `find`, écrivez une ligne de commandes effaçant tous les fichiers réguliers vous appartenant, dont la taille est supérieure à 1 Go (*giga-octet*) . Faites en sorte qu'il y ait une demande de confirmation avant l'effacement de chaque fichier.
4. Ecrivez une commande permettant d'enlever à tous les fichiers réguliers vous appartenant, le droit d'exécution pour le groupe propriétaire et le reste du monde. Faites en sorte que cette commande n'agisse que sur les fichiers dont le mode octal n'est pas 777.

Pensez à prendre une nouvelle copie à en-tête et à y reporter le numéro de l'énoncé.

Exercice 3 – *Processus et petits scripts [3,5 points]*

1. Expliquez la différence entre un processus en avant plan (*foreground*) et un processus en arrière plan (*background*).
2. Expliquez la différence entre valeur de retour, sortie et sortie erreur d'un processus.
3. On suppose que la commande `dictionnaire` prend un mot en argument et écrit `CORRECT` ou `INCORRECT` sur sa sortie suivant que le mot se trouve ou non dans le fichier `/usr/share/dict/words`. La valeur de retour d'un appel à `dictionnaire` indique si l'appel a fonctionné (nombre et type d'arguments corrects, existance du fichier `words`)
   Comment utiliser la commande `dictionnaire` dans une construction `if` ou `while` pour décider si un mot existe ou non ?
   Ecrivez la commande `dictionnaire`.
4. Que fait la commande :
   `ls -l /etc | grep "^d" | wc -l`
5. Réécrivez la commande de la question 4 avec une boucle `for`.
6. Que fait la commande :
   `cd /home/martin 2>/dev/null && (pwd ; echo OK) || (pwd ; echo ECHEC)`

Exercice 4 – *Transport ferroviaire [7,5 points]*

Vous avez un fichier de données `.voyages` situé à la racine de votre répertoire personnel. Ce fichier donne les horaires de tous les trajets possibles en France sur une journée : chaque ligne présente un trajet au format `hd:vd:nt:va:ha`, où `hd` et `vd` sont l'heure et la ville de départ, `ha` et `va` l'heure et la ville d'arrivée, et `nt` est le numéro du train. On suppose que les heures sont au format `..h..`, que les noms de villes ne contiennent pas d'espace et que le fichier `.voyages` n'est pas trié. Par exemple, la ligne `10h30:Paris:9356:Lyon:12h30` correspond au train 9356 qui part de Paris à 10h30 et arrive à Lyon à 12h30.

Dans cet exercice, on développe quelques exécutables pour manipuler ce fichier de données. Vos exécutables sont situés dans un dossier `~/bin/voyages/`, lui-même présent dans votre variable `PATH`. Pour qu'ils soient lus par le correcteur, les exécutables que vous proposez doivent être *présentés lisiblement* (en particulier *indentés*), et *commentés brièvement*.

Pour chaque question, vous devez réutiliser au maximum les exécutables développés dans les questions précédentes. Vous pouvez réutiliser un exécutable même si vous n'avez pas réussi à l'écrire.

1. (i) En utilisant la commande `sed`, écrivez un exécutable `afficherVoyage.sh` qui prend en argument une ligne au format `hd:vd:nt:va:ha` et affiche le voyage correspondant dans un format lisible par l'utilisateur : par exemple, le voyage `10h30:Paris:9356:Lyon:12h30` doit être affiché au format

```
Train 9356 - Départ de Paris à 10h30 - Arrivée à Lyon à 12h30
```

(ii) Écrivez un exécutable `afficherVoyages.sh` qui prend en argument une liste de voyages et les affiche au format précédent.

2. Écrivez un exécutable `depart.sh` qui prend en argument un nom de ville et affiche tous les trains qui partent de cette ville, dans l'ordre de leur heure de départ.

3. (i) Écrivez un exécutable `inverser.sh` qui prend en argument un chemin vers un fichier dont les lignes sont constituées de 5 champs séparés par des deux points (*i.e.*, au format `a:b:c:d:e`) et affiche le fichier en inversant l'ordre de ces champs (*i.e.*, au format `e:d:c:b:a`).

(ii) Écrivez un exécutable `arrivee.sh` qui prend en argument un nom de ville et affiche tous les trains qui arrivent à cette ville, dans l'ordre de leur heure d'arrivée.

4. (i) Écrivez un exécutable `departImmediat.sh` qui prend en argument deux noms de villes `vd` et `va` et une heure `h` (au même format que précédement `..h..`) et affiche le premier voyage direct entre `vd` et `va` qui parte après l'heure `h`.

(ii) Quelle commande pouvez-vous rajouter au début de votre exécutable pour que si l'utilisateur ne spécifie pas l'heure `h`, il affiche le prochain voyage entre `vd` et `va` (c'est-à-dire le premier voyage consécutif à l'heure de la demande) ?

5. Écrivez un exécutable `correspondance.sh` qui prend en argument trois noms de villes `v1`, `v2` et `v3` et affiche le prochain voyage de `v1` à `v3` en passant par `v2`.

6. Pour des raisons techniques, tous les trains dont le numéro est compris entre 6000 et 8999, ou entre 16000 et 18999 sont supprimés. Quelle ligne de commande vous permet de mettre à jour votre base de données `.voyages` ?

7. Suite à l'installation de la ligne TGV entre Paris et Strasbourg, l'heure d'arrivée de tous les trains entre Paris et Strasbourg ou entre Strasbourg et Paris doit être avancée d'une heure trente.

(i) Écrivez un exécutable `avancerHeure.sh` qui prend en argument deux heures `h1` et `h2` (au format `..h..`), vérifie si `h1` est supérieure à `h2` (et sinon, renvoie une erreur sur la sortie erreur), puis affiche l'heure `h1` avancée de `h2`. Par exemple, `avancerHeure.sh 10h20 02h30` doit renvoyer `07h50`.

(ii) Quelle ligne de commande vous permet de mettre à jour votre base de données `.voyages` pour prendre en compte la nouvelle ligne TGV ?

Annexe : Manuel de la commande `find` (extraits)

`SYNOPSIS`

```
find [chemin...] [expression]
```

`DESCRIPTION`

- `find` parcourt les arborescences de répertoires commençant en chacun des *chemins* mentionnés, en évaluant les *expressions* fournies pour chaque fichier rencontré.
- Le premier argument commencant par '-', '(', ')', ',', ou '!' est considéré comme le début de l'expression, tous les arguments précédents sont considérés comme des chemins à parcourir.
- Tous les arguments ultérieurs sont considérés comme le reste de l'expression régulière.
- Si aucun chemin n'est mentionné, le repertoire en cours sert de point de depart.
- Si aucune expression n'est fournie, find utilise l'expression '-print' par défaut.
- `find` renvoie une valeur de retour égale à 0 si tous les fichiers ont pu etre examinés correctement, et supérieure a 0 si une erreur s'est produite.

`EXPRESSIONS`

- L'expression est constituée d'options, de tests et d'actions, tous ces éléments étant séparés des opérateurs. Quand un opérateur est manquant, l'opération par defaut `-and` est appliquée.

`OPTIONS`

- Les options ont toujours un effet global plutôt que de s'appliquer uniquement à leur emplacement dans l'expression. Neanmoins, pour améliorer la lisibilité, il est préférable de les placer au début de l'expression.

  `-daystart`
  Mesurer les temps (avec `-amin`, `-atime`, `-cmin`, `-ctime`, `-mmin`, et `-mtime`) depuis le debut de la journée plutôt que depuis 24 heures.

  `-depth`
  Traiter d'abord les sous-répertoires avant le répertoire lui-même.

  `-help, -help`
  Afficher un message d'aide sur la sortie standard et terminer normalement.

  `-maxdepth n`
  Ne pas appppliquer les tests à des niveaux de profondeur supérieure à n.

  `-mindepth n`
  Ne pas appppliquer les tests à des niveaux de profondeur inférieure à n.

`ARGUMENTS NUMERIQUES`

Les arguments numériques peuvent être indiqués comme suit :

`+n` supérieur à `n`,
`-n` inférieur à `n`,
`n` égal à `n`.

`Opérateurs`

- `( expr )`
  Force la précédence.
- `! expr`
  Vrai si `expr` est fausse.

- `-not expr`
  Comme `! expr`.
- `expr1 expr2`
  ET (implicite) ; `expr2` n'est pas évaluée si `expr1` est fausse.
- `expr1 -a expr2`
  Comme `expr1 expr2`.
- `expr1 -o expr2`
  OU ; `expr2` n'est pas évaluée si `expr1` est vraie.

TESTS

Voici la liste des tests que vous pouvez effectuer sur les fichiers :

`-amin n`
Dernier accès au fichier il y a `n` minutes.

`-anewer file`
Dernier accès au fichier plus récent que la dernière modification du fichier `file`.

`-cmin n`
Dernière modification du statut du fichier il y a `n` minutes.

`-cnewer file`
Dernière modification du statut du fichier plus récente que la dernière modification du fichier `file`.

`-gid n`
Fichier de GID numérique égal à `n`.

`-inum n`
Fichier de numéro d'i-noeud égal à `n`.

`-links n`
Fichier possédant `n` liens.

`-mmin n`
Fichier dont les données ont été modifiées il y a `n` minutes.

`-name motif`
Fichier dont le nom de base (sans les répertoires du chemin d'accès), correspond à l'expression shell `motif`.

`-newer file`
Fichier modifié plus récemment que le fichier `file`.

`-path motif`
Fichier dont le nom complet correspond à l'expression shell `motif`.

`-perm mode`
Fichier dont les droits d'accès correspondent au mode indiqué.

`-regex motif`
Fichier dont le nom complet correspond à l'expression régulière `motif`.

`-size n`
Fichier utilisant `n` blocs de 512 octets sur le disque.

`-type x`
Selon le caractère `x` :
- `b` fichier spécial en mode bloc (avec buffer)
- `c` fichier spécial en mode caractère (sans buffer)
- `d` répertoire

- `p` tube nommé (FIFO)
- `f` fichier régulier
- `l` lien symbolique
- `s` socket

`-uid n`
Fichier dont le propriétaire possède l'UID `n`.

`-used n`
Fichier dont le dernier accès date de `n` jours après la dernière modification de son statut.

`-user toto`
Fichier appartenant à l'utilisateur `toto`.

ACTIONS

`-exec cmd \;`
Exécute la commande `cmd`. La chaîne '{}' est remplacée par le nom du fichier en cours de traitement.

`-fprint file`
Ecrit le nom complet du fichier en cours de traitement dans le fichier `file`. Si fichier n'existe pas au démarrage de find, il est créé. S'il existe, il est écrasé. Les noms de fichiers `/dev/stdout` et `/dev/stderr` sont traités de manière spécifique, ils correspondent respectivement aux sorties standard et erreur.

`-ok cmd \;`
Comme -exec mais interroge d'abord l'utilisateur (en utilisant l'entrée standard). Si la réponse ne commence pas par 'y' ou 'Y', la commande n'est pas exécutée, et le test devient faux.

`-print`
Affiche le nom complet du fichier sur la sortie standard, suivi d'un saut de ligne.