Introduction aux systèmes d'exploitation-IS1

Examen de 1ère session 2012/2013

mercredi 9 janvier - durée: 3h00

A lire attentivement avant de commencer

L'énoncé est composé de 3 exercices. Le barême est donné à titre indicatif.

Le premier exercice est un QCM, vous rendrez les réponses sur la fiche de réponse qui se trouve à la fin de celui-ci. Pour préserver l'anonymat, mettez comme nom le numéro de la salle et comme prénom votre rang (en partant du tableau) et votre place (en partant de la droite) et reportez ces informations sur votre copie ainsi que le numéro du sujet.

Comme il y a plusieurs amphis/salle 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 expliquer 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.

Exercice 1.— Processus, variables et droits [6 points]

Soit Paul un utilisateur.

1. Une fois son script `monscript` écrit (par exemple grâce à un éditeur de texte), que Paul doit-il faire pour pouvoir l'éxécuter ? comment l'éxécute-t-il ?

   Correction. Le rendre exécutable et soit mettre le répertoire contenant `monscript` dans la variable PATH soit donner sa référence complète.
   `chmod +x monscript ; ./monscript arg1`

2. Comment exécuter `monscript` et rediriger sa sortie standard dans le fichier `toto` et ajouter sa sortie erreur au fichier `tata` ?

   Correction. ./monscript 2>> tata > toto

3. Comment faire en sorte que `monscript` soit exécutable quel que soit le répertoire de travail dans lequel Paul se trouve ?

   Correction. Mettre la référence absolue du répertoire contenant `monscript` dans la variable PATH (au début) et mettre cette affectation dans un fichier d'initialisation.

4. (a) Quels droits doivent avoir les répertoires et les fichiers ordinaires de Paul pour que tous les utilisateurs autres que Paul puissent lire et exécuter ses fichiers mais ne puissent ni les détruire ni les modifier et que Paul puisse lire, modifier et exécuter fichiers et répertoires.

   Correction. Lecture (r), Ecriture(w), Exécution (x) pour Paul. Lecture (r) et Exécution (x) pour le groupe et les autres.

(b) Donner la commande qui permet à Paul de positionner les droits ainsi pour tous les répertoires et fichiers qu'il possède dans son arborescence personnelle.

Correction. `chmod -R 755 $HOME`

(c) Donner la commande qui permet à Paul que tout nouveau répertoire soit créé avec ces droits.

Correction. `umask 022`

5. Paul veut que dans toutes ses sessions, l'invite de commande soit "Paul :". Que doit-il faire ?

   Correction. Mettre `PS1="Paul:"` dans un fichier d'initialisation par exemple `~/.bashrc`

6. Paul souhaite arrêter la commande `comSansFin` qu'il a lancé en arrière plan sans noter son numéro de processus. Il ne veut pas fermer la fenêtre où cette commande s'exécute. Que doit il faire (donner deux méthodes) ?

   Correction. Il doit chercher son numéro de processus soit à l'aide de `ps` (il a alors le pid) soit à l'aide de `jobs` (dans la fenêtre où cette commande s'exécute) (il a alors le num). Puis avec `kill pid` ou `kill %num`.

7. Paul souhaite arrêter la commande `comSansFin` qu'il a lancé au premier plan. Il ne veut pas fermer la fenêtre où cette commande s'exécute. Que doit il faire (donner deux méthodes) ?

   Correction. Il peut utiliser un des signaux d'interruption correspondant à `<quit>` ou `<intr>`. Depuis une autre fenêtre, il peut chercher le numéro de processus soit à l'aide de `ps -a` (il a alors le pid). Puis terminer la commande avec `kill pid` .

Exercice 2.— Filtres, redirections et scripts [6 points]

On dispose d'un fichier `users` qui contient des informations sur les utilisateurs d'un service. Chaque ligne du fichier concerne un utilisateur, et contient les informations suivantes, séparées par le caractère : (deux points).

- identifiant numérique unique (un numéro donné n'apparaît qu'une fois dans la liste) ;
- pseudonyme unique (idem) ;
- prénom ;
- nom de famille (la combinaison prénom+nom n'est pas nécessairement unique).

Par exemple, le fichier peut contenir les lignes suivantes :

```
19887:lolo75:Laurine:Mitchell
21260:mdr_ouf:Karim:Benani
29016:phil:Philippe:Lao
```

Toutes les lignes du fichier sont à ce format et aucune ne contient de ligne vide.

Les commandes

Pour chaque question ci-dessous concernant le fichier `users`, écrire une commande.

1. Compter le nombre de lignes du fichier.

   Correction. `wc -l users`

2. Afficher les trois premières lignes du fichier.

   Correction. `head -3 users`
   ou `head -n 3 users`

3. Afficher les trois dernières lignes du fichier.

Correction. `tail -3 users`
ou `tail -n 3 users`

4. Afficher les lignes de tous les utilisateurs ayant le nom de famille `Dubois`.

   Correction. `grep ':Dubois$' users`

5. Extraire la liste des pseudonymes et la mettre dans un fichier `pseudo`, un pseudonyme par ligne.

   Correction. `cut -d ':' -f 2 users > pseudo`

6. Trier le fichier par la première colonne, et mettre le résultat dans un fichier `users-by-id`. Selon la manière dont vous utiliserez la commande, vous préciserez si le tri est fait par ordre alphabétique ou par ordre numérique.

   Correction. pour un tri par ordre alphabétique : `sort users > users-by-id`
   pour un tri par ordre numérique : `sort -n users > users-by-id`
   Note : si tous les identifiants ont le même nombre de chiffres, les deux ordres sont identiques.

Les combinaisons de commandes

Ici, vous pourrez avoir besoin de plusieurs commandes pour chaque question.

1. Afficher les informations de l'utilisateur dont le pseudonyme est `gogo`.

   Correction. `grep ':gogo:' users`

2. Afficher seulement l'identifiant numérique de cet utilisateur.

   Correction. `grep ':gogo:' users|cut -f1 -d':'`

3. Afficher seulement le prénom et le nom de famille de cet utilisateur, en les séparant par un espace (exemple : Philippe Lao).

   Correction. `grep ':gogo:' users|cut -f3,4 -d':' |tr ':' ' '`

4. Compter le nombre d'utilisateurs ayant le nom de famille `Dubois`.

   Correction. `grep ':Dubois$' users|wc -l`

5. Afficher uniquement les prénoms de ces utilisateurs.

   Correction. `grep ':Dubois$' users|cut -f3 -d':'`

6. Indiquer ce qu'il faut ajouter à la commande précédente pour que cette liste de prénoms soit triée par ordre alphabétique.

   Correction. `grep ':Dubois$' users|cut -f3 -d':'|sort`

7. Même question pour afficher une liste où ces prénoms sont précédés de leur nombre d'occurrences, et triés par nombres d'occurrences décroissants.

   Correction. `grep ':Dubois$' users|cut -f3 -d':'|sort| uniq -c|sort -r -n`

8. Proposer une méthode pour vérifier que les identifiants numériques sont bien tous différents.

   Correction. Compter le nombre d'identifiants et le comparer avec le nombre d'utilisateurs :
   `NBIDS=‘cut -f 1 -d':'users |sort|uniq|wc -l‘`
   `NBUSERS=‘wc -l <users‘`
   `test $NBIDS -eq $NBUSERS || echo certains identifiants egaux`

Scripts

Pour chacune des opérations suivantes, vous écrirez un script.

1. Afficher le premier argument du script (sans vérification préalable).

   Correction.
   ```
   echo $1
   ```

2. Si le script a un paramètre, afficher ce paramètre sur la sortie standard, sinon afficher `Erreur` sur la sortie erreur.

   Correction.
   ```
   if test $# -eq 1
   then echo $1 >&1
   else echo Erreur >&2
   fi
   ```
   ou bien :
   ```
     test $# -eq 1 || { echo Erreur >&2; exit; }
   echo $1
   ```

3. Si le script a un paramètre, utiliser ce paramètre comme nom de fichier et afficher le nombre de lignes de ce fichier, en redirigeant la sortie erreur de la commande. Si la commande a échoué (par exemple si le fichier n'existait pas), afficher `Erreur` sur la sortie erreur.

   Correction.
   ```
   if test $# -ne 1
   then echo "erreur il faut un argument" >&2; exit
   fi
   if wc -l $1 2> /dev/null
   then echo "est le nombre de ligne de $1"
   else echo "erreur $1 n'existe pas">&2
   fi
   ```
   ou bien :
   ```
    test $# -eq 1 || { echo "erreur il faut un argument" >&2; exit; }
   wc -l $1 2> /dev/null && exit
   echo "erreur $1 n'existe pas">&2
   ```
   il était possible aussi d'utiliser `test $# -ge 1`

4. Si le script a un paramètre, utiliser ce paramètre comme identifiant d'un utilisateur, pour afficher la ligne contenant ses informations dans le fichier `users`. Sinon, ne rien faire.

   Correction.
   ```
   if test $# -eq 1
   then grep "^$1:" users
   fi
   ```
   ou bien :
   ```
   test $# -eq 1 && grep "^$1:" users
   ```