Langage C et Programmation Système
TP n° 1 : Utilisation d'un système Unix
(Correction)

Exercice 1 : Editeur de texte

Lancez `emacs`, et créez un nouveau fichier. Rédigez un texte et sauvegardez le fichier, sous le nom `foo.txt`, dans le sous-répertoire `test` de votre répertoire personnel que vous aurez créé au préalable. Fermez le fichier. Ouvrez-le à nouveau et rajoutez-y un texte. Sauvegardez puis fermez l'éditeur de texte.

Exercice 2 : Fichiers/Répertoires

1. Quelle est la référence absolue de votre répertoire privé ?
   ▷ `cd ~; pwd` ou `echo ~` ou `echo $HOME`
2. Créez une copie de `foo.txt` appelée `.foo` (le point est voulu) dans votre répertoire personnel. Supprimez ensuite le fichier `foo.txt`. Affichez le contenu de votre répertoire personnel avec `ls`. Que constatez-vous ? Réessayez en ajoutant l'option `-a`. Concluez.
3. Quels sont les droits d'accès associés à votre répertoire privé ?
   ▷ `ls -ld ~`
4. Qui peut consulter son contenu ?
5. Qui peut y déposer/créer un fichier ?
6. Sauriez-vous créer un répertoire dans lequel n'importe qui peut déposer un fichier mais personne ne peut consulter le contenu du répertoire ? Tester avec votre voisin.
   ▷ `mkdir test`
   `chmod 333 test` ou `chmod a=wx test`

Exercice 3 : Fichiers / regexp

1. Affichez la liste de tous les fichiers dans le répertoire `/usr/bin` dont le nom commence par `k` et contient exactement 6 caractères.
   ▷ `cd /usr/bin`
   `ls k?????` ou `ls | egrep '^k.{5}$'`
2. Affichez la liste de tous les fichiers dont l'extension est `.so` dans le répertoire `/usr/lib` (*note :* ces fichiers sont des bibliothèques).
   ▷ `cd /usr/lib`
   `ls *.so` ou `ls | egrep '.so$'`
3. Que fait la commande `find` ?
4. Utilisez-la pour retrouver les fichiers dont les noms contiennent la chaîne "conf" dans tout le système (on peut arrêter avec `^C` au bout d'un moment)
   ▷ `find / -name '*conf*'`

5. Recommencez de sorte que les messages d'erreurs soient "perdus" (`2>/dev/null`)
6. Créez un certain nombre de répertoires et des fichiers dont quelques-uns auront la chaîne "abc" dans leur nom. Pour cela, vous pouvez vous aider d'un script shell. Utilisez find pour les retrouver. Utilisez find pour afficher les informations les concernant (taille, dates, etc). Utilisez find pour les supprimer.

   ▷ `touch axbxcx abcxxx xxxabc; mkdir aXbXcX abcXXX XXXabc`

   ```
   find . -name '*abc*' -exec stat '{}' \;
   find . -name '*abc*' -delete
   ```

   ou `find . -depth -name '*abc*' -exec rm -r '{}' \;`

Exercice 4 : Redirection

1. Sauvegardez le résultat de la commande `ls -l /usr/lib` dans un fichier `liste`, puis affichez le contenu de ce fichier avec la commande `less` par exemple.

   ▷ `ls -l /usr/lib > liste`

   `less liste`

2. Redirigez l'entrée standard de la commande `cat` vers le fichier `liste` et observez ce qui se produit. Comme pour un certain nombre de commandes, le même résultat peut être obtenu en passant le fichier `liste` directement en argument de la commande `cat` (sans symbole de redirection). Essayez.

   ▷ `cat < liste`

   `cat liste`

3. Ajoutez au fichier `liste` une ligne de texte.

   ▷ `echo "..." >> liste`

4. Copiez `liste` dans `liste-bis` sans utiliser la commande `cp`.

   ▷ `cat liste > liste-bis`

Exercice 5 : Pipe

1. Si la commande `ls` fournit un résultat trop long, comment le consulter intégralement sans utiliser la souris ?

   ▷ `ls | less` ou `ls | more`

2. Ecrire dans un fichier le contenu de votre répertoire personnel trié par ordre alphabétique *inverse*.

   ▷ `ls -r ~ > liste` ou `ls ~ | sort -r > liste`

3. En une seule commande composée, cherchez dans `/bin` tous les noms de fichier contenant la lettre `a` et triez-les par ordre alphabétique inverse.

   ▷ `find /bin -name '*a*' | sort -r > test.txt`

   ou `find /bin -name '*a*' -printf '%f\n' | sort -r > test.txt`

Exercice 6 : Identification

1. À quoi sert la commande 'id' ? (Indication : `man` *ma_commande* pour accéder à la description complète de *ma_commande*)
2. Quel est votre numéro d'identification ?
3. Quel est le numéro de votre camarade (obtenez cette information sans lui demander bien sûr) ?
   ▷ `id -u username`

Exercice 7 : Localisation machine

1. Sur quelle machine travaillez-vous ? (`hostname`)
   ▷ `hostname`
2. Ouvrez un terminal et connectez-vous sur la machine du voisin (`ssh`)
   ▷ `ssh user@machine_name`

Exercice 8 : Expansion

1. Que font les commandes `wc`, `wc -w`, `echo a | wc -w` ?
2. `echo * | wc -w` devrait vous fournir un résultat différent de 1 pourquoi ?
   ▷ `*` est interprété comme l'ensemble de noms de fichiers dans le répertoire courant.

Exercice 9 : Processus

1. Quel shell utilisez-vous actuellement ?
   ▷ `ps -p $$` (fonctionne dans Bash, mais pas dans tous les shells)
2. Quelles sont les options de ps qui permettent d'obtenir la liste des processus dont vous êtes le propriétaire ?
   ▷ `ps -fu $(whoami)` (ou, si le système le permet, `pstree -p $(whoami)`)
3. Retrouvez le processus père de votre processus shell et tuez-le. Qu'est ce qu'il se passe ?
   ▷ `kill $PPID`
4. Si vous lancez la commande `sleep 60 | grep toto &`, combien de processus sont utilisés par cette commande ? Quel est le père de ces processus ?
   ▷ `sleep 60 | grep toto &`
   `ps -f` (ou, si le système le permet, `ps -fH` ou `pstree -p`)
5. Quelle différence y a-t-il entre un "job" et un processus ?
   ▷

Exercice 10 : Script

1. Créez un script permettant de recevoir en argument un répertoire, une chaîne de caractères et qui à l'exécution permet de retrouver sous le répertoire indiqué tous les fichiers dont le nom contient la chaîne.

▷ Contenu de `monscript.sh` :

```
#!/usr/bin/bash
find "$1" -type f -name "*$2*"
```

Le rendre exécutable : `chmod u+x monscript.sh`
Exemple d'appel : `./monscript.sh . 'mon'`

2. Modifiez ce script de sorte que si l'option `-s` est présente les fichiers (et uniquement les fichiers, pas les répertoires) concernés sont supprimés, sinon leurs informations associées sont simplement affichées.

▷ Deuxième version de `monscript.sh` :

```
#!/usr/bin/bash
if [ "$1" = '-s' ]
then
  find "$2" -type f -name "*$3*" -exec rm '{}' \;
else
  find "$1" -type f -name "*$2*" -exec stat '{}' \;
fi
```

3. Rajoutez un deuxième argument optionnel qui prend les valeurs `-i` ou `-f`. L'option `-i` indique que l'on veut confirmer à la main la suppression de chaque fichier, l'option `-f` indique le contraire.

▷ Troisième version de `monscript.sh`, avec une ligne «shebang» (`#!`) plus portable :

```
#!/usr/bin/env bash
SET_S=false
SET_I=false
SET_F=false
while getopts sif OPT; do   # 'sif' : liste des options
  case "$OPT" in
    s) SET_S=true
       ;;
    i) SET_I=true
       ;;
    f) SET_F=true
       ;;
  esac
done
shift $((OPTIND-1))  # Renomme les arguments.

if [ "$SET_S" = true ]; then
  if [ "$SET_I" = true ]; then
    find "$1" -type f -name "*$2*" -exec rm -i '{}' \;
  elif [ "$SET_F" = true ]; then
    find "$1" -type f -name "*$2*" -exec rm -f '{}' \;
```

```
  else
    find "$1" -type f -name "*$2*" -exec rm '{}' \;
  fi
else
  find "$1" -type f -name "*$2*" -exec ls -l '{}' \;
fi
```

4. Modifiez le script de façon à ce que les erreurs soient correctement gérées : nombre d'argument correct, répertoire existant, etc. En cas d'erreur ce script renverra un code non nul et sinon un code nul (vérifiez-cela).

▷