Formation Utilisateurs sur HP-UX

Corrigé des Exercices

2.2 Système de fichiers

1. Créez un répertoire dans votre répertoire hôte. Appelez le *essai* et faites en le répertoire courant. Quel est le chemin d'accès absolu de ce répertoire ?

$ cd
$ mkdir essai
$ cd essai
$ pwd
/users/cours0/essai

2. Revenez dans le répertoire hôte. Affichez toute la structure arborescente du répertoire *arbre*. Comment peut-on faire la distinction entre les répertoires et les fichiers ?

$ cd
$ ls -RF arbre
sous_dir1/ sous_dir2/ sous_dir3/

arbre/sous_dir1:
fichier1/ fichier2/

arbre/sous_dir1/fichier1:
f1 f2/
...
Les répertoires sont signalés par le /.

3. Quel est le chemin d'accès absolu du fichier *f1* ? Quel est son chemin d'accès relatif à partir de votre répertoire hôte ?

Chemin d'accès absolu: /users/cours0/arbre/sous_dir1/fichier1/f1
Chemin d'accès relatif: arbre/sous_dir1/fichier1/f1

4. A partir de votre répertoire hôte, placez-vous dans le répertoire *arbre/sous_dir1/fichier1*. Puis à l'aide d'un chemin d'accès relatif placez-vous dans le répertoire */arbre/sous_dir3/fichier4/d2*. Retournez dans votre répertoire hôte. Comment faire pour vérifier si on se trouve vraiment dans les répertoires souhaités ?

$ cd
$ cd arbre/sous_dir1/fichier1
$ pwd
/users/cours0/arbre/sous_dir1/fichier1
$ cd ../../sous_dir3/fichier4/d2
$ pwd
/users/cours0/arbre/sous_dir3/fichier4/d2
$ cd
$ pwd
/users/cours0

5. A partir de votre répertoire hôte, créez les répertoires suivants:
essai/dirB/dir1 essai/dirB/dir2/dirC

$ mkdir -p essai/dirB/dir1 essai/dirB/dir2/dirC
L'option -p permet de créer les répertoires intermédiaires non encore créés.

6. A partir de votre répertoire hôte, supprimez tous les répertoires sous *essai*.

$ rmdir essai/dirB/dir1 essai/dirB/dir2/dirC essai/dirB/dir2 essai/dirB

3 Commandes simples

1 Trouvez la fonction de la commande cd.

$ man cd
La commande cd permet de changer de répertoire de travail

2 Qui est connecté sur le système ?
Sur quel terminal êtes-vous connecté ?
Comment êtes-vous identifié par le système ?

$ who
$ who am i
$ whoami

3 Modifiez votre mot de passe.

$ passwd
Le nouveau mot de passe doit contenir au moins 6 caractères qui diffèrent de l'ancien d'au moins 3 caractères.
Il doit contenir au moins 2 caractères alphabétiques et un caractère non alphabétique.

4 Tapez les commandes suivantes et vérifiez les résultats:

$ echo Bonjour tout le monde
Bonjour tout le monde — 4 arguments
$ echo Bonjour Bernard
Bonjour Bernard — 2 arguments
$ echo "Bonjour Bernard"
Bonjour Bernard — 1 argument
$ echo
— 0 argument
$

5 Un autre utilisateur peut-il envoyer un message sur votre terminal ?

$ mesg
is y
Oui

6 Affichez la date du jour avec le format jour/mois/année.

$ date +%d/%m/%y
20/02/95

7 Entrez une commande qui vous permette de savoir quel jour de la semaine vous êtes né.

$ cal 06 1967

```
    June 1967
 S  M Tu  W Th  F  S
              1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```


4 Manipulation de fichiers

1 Visualisez le contenu du fichier *fable*.

$ cat fable Trop long pour tenir sur l'écran
$ more fable Affichage paginé

2 Faites une copie du fichier *fable* vers *fable.cp*. Vérifiez que ces deux fichiers sont identiques. Changez le nom de la copie en *fable.B*

$ cp fable fable.cp
$ diff fable fable.cp
$
L'absence de toute ligne différente assure que les fichiers sont identiques
$ mv fable.cp fable.B

3 Effacez les fichiers *fable.B* et *fable.A* en une seule commande.

$ rm fable.B fable.A

4 Copiez en une seule commande les fichiers suivants dans le répertoire *essai*: *mon_fichier ton_fichier son_fichier un fichier*

$ cp mon_fichier ton_fichier son_fichier un_fichier essai

5 Changez le nom de *mon_fichier* se trouvant dans le répertoire *essai* en *son_fichier*. Qu'en est-il du fichier *son_fichier* qui existait ?

$ mv essai/mon_fichier essai/son_fichier
Le fichier qui existait est écrasé.

6 Allez dans le répertoire *essai*. Déplacez les fichiers *son_fichier* et *ton_fichier* vers *arbre/sous_dir2*.

$ cd essai
$ mv son_fichier ton_fichier ../arbre/sous_dir2

7 De votre répertoire hôte supprimez tous les fichiers du répertoire *essai*.

*$ rm essai/**

8 Lancez une commande qui indique l'endroit de votre arborescence où se trouve *fichier3*.

$ cd
$ find . -name "fichier3" -print
./arbre/sous_dir2/fichier3
$

9 Lancez une commande qui indique le nom de tous les fichiers qui se trouvent dans l'arborescence de */users* dont le nom contienne la chaîne *cours*.

*$ find /users -name "*cours*" -print*

5 Permissions sur les fichiers

1 Qui est le propriétaire de *mon_fichier* ? A quel groupe appartient-il ? Qui est autorisé à changer le propriétaire et le groupe ?

$ ll mon_fichier
-rw-rw-rw- 1 cours0 cours 12 Feb 20 11:02 mon_fichier
Le propriétaire de mon_fichier est cours0. Son groupe est cours. Seuls l'administrateur et cours0 peuvent changer sa paternité.

2 Attribuez la paternité du fichier *ton_fichier* à un autre utilisateur. Pouvez-vous en récupérer la paternité ?

$ chown cours9 ton_fichier
$ chown cours0 ton_fichier
ton_fichier: Not owner
Il est impossible d'en récupérer la paternité si son nouveau propriétaire ne veut pas le faire.

3 Faites une copie de *son_fichier*. A qui appartient la copie ? Effectuez une copie de *son_fichier* d'un autre utilisateur. A qui appartient cette copie ? Comment changer les permissions de cette dernière copie afin que l'autre utilisateur ne puisse plus le modifier ?

$ cp son_fichier son_fichier.cp
Si la destination n'existait pas, vous en êtes le propriétaire
$ cp ../cours9/son_fichier son_fichier.cp2
Si la destination n'existait pas, vous en êtes le propriétaire
*$ chmod 644 son_fichier.**

4 Changez les protections de *son_fichier* de manière à ce que son propriétaire puisse lire et écrire, le groupe puisse lire et que les autres ne puissent rien faire.

$ chmod 640 son_fichier
On aurait pu faire aussi:
$ chmod go-w son_fichier
$ chmod o-r son_fichier

5 Copiez *son_fichier* vers le répertoire */bin*. Avez-vous des problèmes et pourquoi ?

$ cp son_fichier /bin
cp: cannot create /bin/son_fichier: Permission denied
$ ll -d /bin
drwxr-xr-x 4 root other 3072 Jan 5 13:59 /bin
Le répertoire /bin est protégé en écriture pour les autres.

6 L'éditeur vi

Entrez dans le fichier *fable* et effectuez les actions suivantes:

$ vi fable

1 Déplacez les six premières lignes à la fin du fichier.

On est déjà positionné en première ligne.

6dd	Efface les 6 lignes
:$	Commande ed pour se déplacer en dernière ligne
ou	
G	Commande vi pour se déplacer en dernière ligne
p	Recopie du buffer après la ligne courante

2 Effectuez une copie des dix premières lignes et placez les à la fin du fichier.

1G	Se positionner en première ligne
ou	
:1	Commande ed pour se positionner en première ligne
10yy	Copie des 10 lignes dans le buffer
:$	
p	Recopie du buffer après la ligne courante

3 Cherchez toutes le occurrences de *Corbeau* et *corbeau*. Au fur et à mesure changez les en *pinson*.

*/[Cc]**orbeau***	Rechercher Corbeau ou corbeau
*cw**pinson**Esc*	Change le mot courant en pinson
/	Rechercher l'occurrence suivante de Corbeau ou corbeau
.	Effectuer à nouveau la dernière commande de modification de texte

4 Cherchez toutes les lignes vides du fichier.

/^$/	Recherche de la ligne vide
/	Rechercher l'occurrence suivante de ligne vide

5 Cherchez les lignes qui se terminent par une virgule ou une double quote.

/[,"]$/	Recherche de la virgule ou de la double quote en dernier caractère
/	Refaire la même recherche

6 Cherchez les lignes qui commencent par une minuscule.

/^[a-z]/	Recherche de la minuscule en premier caractère
/	Refaire la même recherche

7 Sauvegardez le fichier et quittez.

ZZ
ou
:x
ou
:wq
Pour le sauvegarder sous un autre nom puis quitter:
*:w **nouveau***
:q

7 Le Shell - 8 Les Processus

1 Affectez votre prénom à une variable *XX*. Vérifiez son contenu.

$ XX=Xavier
$ echo $XX
Xavier

2 Lancez un Shell fils en tapant ksh. Affichez *XX*. Tuez le Shell fils. Affichez *XX*. Expliquez.

$ ksh
$ echo $XX

$ exit
$ echo $XX
Xavier
Le Shell fils ne peut pas hériter d'une variable locale.

3 Quelle commande faut-il taper pour que le Shell fils voit la variable *XX* ? Comment, depuis le Shell père voir toutes les variables dont le Shell héritera ?

$ export XX
$ env
ou
$ export

4 Démarrez un autre Shell fils. Affichez *XX*. Maintenant affectez à la variable *XX* votre nom de famille. *XX* est-elle une variable locale ou d'environnement ? Quelle est la valeur de *XX* ?

$ ksh
$ $echo $XX
Xavier
$ XX=Dupont
XX est devenue une variable d'environnement
$ echo $XX
Dupont

5 Effacez *XX* du Shell fils. *XX* existe-t-elle encore en local ou dans l'environnement ? Pourquoi ?

$ unset XX
$ env XX a disparu
$ set XX a disparu
On vient de l'effacer localement. Dans le Shell local toute trace de XX a disparu.

6 Tuez le Shell fils. *XX* existe-t-elle encore ? Pourquoi ?

$ exit
$ env XX y est bien
Aucune opération effectuée dans le Shell fils ne peut affecter de variables du Shell père.

8 Les Processus

1 Affichez votre numéro d'identifiant et votre numéro de groupe. Changez de compte en prenant celui d'un autre utilisateur. Affichez à nouveau les informations précédentes. Revenez à l'état initial.

```
$ id
uid=208(cours0) gid=202(cours)
$ su cours9
$ id
uid=217(cours9) gid=202(cours)
$ exit
```

2 Quels sont les processus vous appartenant en cours d'exécution sur le système ?

Processus en cours attachés au terminal

```
$ ps
PID TTY      TIME COMMAND
 20522 ttyp2    0:00 telnetd
 20523 ttyp2    0:01 ksh
 21028 ttyp2    0:00 ps
```

Processus en cours appartenant à cours0

```
$ ps -u cours0
  PID TTY      TIME COMMAND
 20523 ttyp2    0:01 ksh
 21029 ttyp2    0:00 ps
```

3 Quels sont les processus en cours d'exécution sur le système ?

```
$ ps -ef
```

9 Redirections d'Entrée / Sortie

1 Redirigez la sortie de la commande date vers un fichier *date.out*.

```
$ date > date.out
$ more date.out
Mon Feb 20 13:45:25 MET 1995
```

2 Ajoutez la sortie de la commande ls au fichier *date.out*.

```
$ ls >> date.out
$ more date.out
Mon Feb 20 13:45:25 MET 1995
arbre
date.out
fable, ...
```

3 Exécutez la commande cp sans argument. Que se passe-t-il ? Redirigez la sortie de cette commande vers *cp.out*. Que se passe-t-il ? Que faut-il faire alors pour rediriger les messages d'erreur vers *cp.err* ?

```
$ cp                                          Erreur de syntaxe
Usage:  cp [-f|-i] [-p] source_file target_file
        cp [-f|-i] [-p] source_file ... target_directory
        cp [-f|-i] [-p] -R|-r source_directory ... target_directory
$ cp > cp.out                                 Redirection stdout n'affecte pas stderr
Usage:  cp [-f|-i] [-p] source_file target_file, ...
$ cp 2> cp.err                                Redirection stderr
$ more cp.err
Usage:  cp [-f|-i] [-p] source_file target_file, ...
```

4 Triez le fichier */etc/passwd* sur le premier champ.

```
$ sort -t":" +0 -1 < /etc/passwd
```

Il pourrait en fait s'agir d'un tri alphabétique tout simple:

```
$ sort /etc/passwd
```

5 Sélectionnez les lignes du fichier */etc/passwd* qui concernent les membres de la formation (cours0, ..., cours9) et dirigez ces lignes vers le fichier *form.out*. Comptez le nombre de lignes de ce fichier.

```
$ grep "cours[0-9]" < /etc/passwd > form.out
$ wc -l < form.out
10
```

6 En suivant le même principe, combien y a-t-il d'utilisateurs connectés ?

```
$ who > who.out
$ wc -l < who.out
4
```

7 Combien y a-t-il de noms de compte configurés dans le système ?

```
$ wc -l < /etc/passwd
25
```

8 Triez le fichier *form.out* et sauvegardez le résultat dans *form.tri*. Triez ce fichier dans l'ordre inverse et sauvegardez le dans le fichier *form.out*.

```
$ sort < form.out > form.tri
```


$ sort -r < form.tri > form.out

10 Les tubes

1 Refaites l'exercice numéro 5 précédent sans redirection.

$ grep "cours[0-9]" /etc/passwd| wc -l
10

2 Refaites suivant le même principe l'exercice numéro 6 précédent.

$ who | wc -l
4

3 Affichez la liste triée des fichiers (et répertoires) de votre répertoire courant. Affichez cette liste sur trois colonnes sans message d'en-tête ni de fin.

$ ls | sort | pr -t -3

4 Produisez une version non triée du fichier *fable* dans *fable.out* et une version triée dans *fable.tri*

$ cat < fable | tee fable.out | sort > fable.tri

5 Produisez une liste triée alphabétiquement des noms des utilisateurs connectés.

$ who | cut -c-8 | sort

6 Affichez la liste des fichiers du répertoire courant dont le nom commence par *f*.

*$ ls f**
ou
$ ls | grep "^f"

7 Produisez une liste des processus actifs du groupe de formation en ne conservant que leur propriétaire et leur nom.

$ ps -ef | grep "cours[0-9]" | cut -c-9,48-

11 Les caractères d'échappement

1 Saisissez une commande qui produise le résultat suivant:
$1 million dollars . . . c'est impressionnant !

$ echo "\$1 million dollars... c'est impressionnant !"

2 Affichez les lignes qui contiennent la chaîne de caractères ":*:" dans le fichier */etc/passwd*

$ grep "::" /etc/passwd*

3 Expliquez la différence des commandes

$ echo $abc	Parameter non set
$ echo '$abc'	
$abc	Pas d'interprétation du dollar
$ echo "$abc"	Parameter non set

12 Le mode multi-tâches

1 Tapez la commande suivante:
$ find / -print > /$HOME/grosfichier 2>&1 &
Tapez
$ ps -ef|grep find
Déconnectez vous, reconnectez vous et relancez la commande ps. Que constatez-vous ?

$ ps -ef|grep find
cours0 21500 21498 1 14:29:34 ttyp2 0:00 grep find
Le processus find a disparu

2 Effacez le fichier grosfichier. Tapez la commande
$ nohup find / -print > /$HOME/grosfichier 2>&1 &
puis effectuez la même manipulation que précédemment. Que constatez-vous ? Quel est le processus père de find ?

$ ps -ef|grep find
```
  cours0 21231     1 50 14:34:06 ?         0:07 find / -print
  cours0 21248 21233  1 14:34:27 ttyp6     0:00 grep find
```
Le processus init (1) est le processus père du find. Au moment de la déconnexion, l'init a repris à sa charge le processus qui tournait. Il n'a donc pas disparu.

3 Allez dans le répertoire *$HOME/arbre/sous_dir3/fichier4/d2/f3*. Définissez une variable *pdir* contenant le chemin d'accès du répertoire courant. Retournez dans votre répertoire hôte. Comment pouvez-vous retourner dans *f3* en utilisant la variable *pdir* ?

$ cd arbre/sous_dir3/fichier4/d2/f3
$ pdir=`pwd`
$ cd
$ pwd
/users/cours0
$ cd $pdir
$ pwd
/users/cours0/arbre/sous_dir3/fichier4/d2/f3

4 Définissez une variable appelée *qui* contenant la liste triée des noms des utilisateurs connectés.

$ qui=`who | cut -c-8 | sort`
$ echo $qui

13 Le Korn Shell

1 Définissez un alias appelé *go* qui permet de se déplacer dans le répertoire *arbre* et lance la commande ls.

```
$ alias go='cd /users/cours0/arbre;ls'
$ go
sous_dir1  sous_dir2  sous_dir3
```

2 Rééditez l'alias à l'aide des commandes vi de manière à ce qu'il réalise l'affichage de Salut !

A l'aide de Esc k on remonte jusqu'à la commande de l'alias
A l'aide des commandes vi l R et x on peut le remplacer ainsi:

```
$ alias go='echo  "Salut !"'
$ go
Salut !
```

3 Récupérez l'historique des commandes et réexécutez, grâce à l'historique, la commande *go*.

```
$ history
ou
$ fc -l
561    exit
562    ps -ef|grep find
563    cd
564    cd arbre/sous_dir3/fichier4/d2/f3
565    pdir=`pwd`
566    cd
567    pwd
568    cd $pdir
569    pwd
570    qui=`who | cut -c-8 | sort`
571    echo $qui
572    alias go='cd /users/cours0/arbre;ls'
573    go
574    alias go='echo  "Salut !"'
575    go
576    fc -l
$ fc -e - g
ou
$ fc -e - 575
Salut !
```

4 Affichez le contenu de *fable.A* alors que vous ne vous rappelez que des 3 premières lettres du nom. Utilisez la génération automatique des noms de fichier et affichez la liste des noms possibles. Complétez la commande.

```
$ more fab<b>EscEsc</b>
$ more fable<b>Esc</b>=
1) fable
2) fable.A
$ more fable.A
```

5 Définissez l'alias *qui* permettant d'obtenir la liste triée des noms des utilisateurs connectés.

```
$ alias qui='who|cut -c-8|sort'
$ qui
```


14 Programmation du Shell

Arguments et variables des programmes Shell

1 Si on lance le programme:
$ prog abc def -d -4 +900 xyz
qu'affichera *prog* s'il contient:

echo $#	6
echo $3	-d
echo $7	
echo $*	abc def -d -4 +900 xyz
echo $0	prog

2 Si *prog* contenait la commande shift 2 en première ligne, quel serait le résultat de:

echo $#	4
echo $3	+900
echo $7	
echo $*	-d -4 +900 xyz
echo $0	prog

3 Ecrivez un programme Shell appelé *inverse* qui recevra jusqu'à 4 arguments et les affichera dans l'ordre inverse.

```
echo "vous avez tape $# arguments"
echo $4 $3 $2 $1
```

4 Ecrivez un programme Shell appelé *alpha* qui affichera le premier et le dernier argument de la ligne de commande.

```
echo "le premier argument est $1"
A=$#
A=`expr $A - 1`
shift $A
echo "le dernier argument est $1"
```

5 Ecrivez un programme Shell appelé *double* qui demandera un nombre entier et affichera son double.

```
echo "Entrez un nombre entier "
read n
echo le double de $n est `expr $n \* 2`
```

6 Ecrivez un programme Shell appelé *hote* qui demandera un nom d'utilisateur et affichera son répertoire hôte (sixième champ du fichier */etc/passwd*).

```
echo "Entrez un nom d'utilisateur"
read u
D=`grep $u /etc/passwd | cut -d":" -f6`
echo le répertoire hote est: $D
```


Branchements

1 Positionnez une variable *X* à la valeur abc. Saisissez une commande de test renvoyant 0 si la valeur de *X* est abc.

```
$ X=abc
$ test "$X" = "abc"
$ echo $?
0
```

2 Créez une commande de test renvoyant 1 si la valeur de *X* est abc.

```
$ test "$X" != "abc"
$ echo $?
1
```

3 Ecrivez un programme Shell qui compte le nombre d'arguments de la ligne de commande, affiche un message d'erreur s'il n'y a pas 3 arguments et qui sinon les affiche.

```
if [ $# -ne 3 ]
then
        echo "Il faut trois arguments"
else
        echo $*
fi
```

4 Modifiez *hote* de l'exercice 6 précédent de façon à ce qu'un message d'erreur soit affiché si le nom d'utilisateur est invalide.

```
echo "Entrez un nom d'utilisateur"
read u
if grep $u /etc/passwd > /dev/null 2>&1
then
        D=`grep $u /etc/passwd | cut -d":" -f6`
        echo le répertoire hote est: $D
else
        echo "Cet utilisateur n'existe pas"
fi
```

5 **Avancé**: Modifiez *double* de l'exercice 5 précédent pour que le programme vérifie
la saisie du caractère NULL
une valeur non entière
la saisie de plus d'une valeur.

```
echo "Entrez un nombre" ; read n
if [ "$n" = "" ]
then
        echo Entrez une valeur !
else
        num=`echo $n | wc -w`
        if [ $num -gt 1 ]
        then
                echo Entrez une seule valeur !
        elif echo $n | grep "[^0-9]" > /dev/null
        then
                echo Entrez un entier !
        else
                echo "Le double de $n est `expr $n \* 2`"
        fi
```


fi

Boucles

1 Ecrivez un programme Shell affichant un par ligne, les arguments de la ligne de commande.

```
for i in $*
do
    echo $i
done
```

ou

```
for i
do
        echo $i
done
```

2 Ecrivez un programme Shell qui affiche "Je suis intelligent" 100 fois.

```
i=1
while [ $i -le 100 ]
do
    echo $i "Je suis intelligent !"
    i=`expr $i + 1`
done
```

3 Ecrivez un programme Shell comptant les options (arguments débutant par -) et les paramètres (tous les autres arguments) de la ligne de commande.

```
o=0
a=0
for i in $*
do
    case $i in
    -*)   o=`expr $o + 1`;;
    *)    a=`expr $a + 1`;;
    esac
done
echo "Il y a $o options et $a arguments"
```

4 Créez un répertoire *$HOME/.poubelle*. Ecrivez un programme destruct qui déplacera tous les fichiers que vous détruirez dans le répertoire poubelle. Vous inclurez les options suivantes:
-l affichage du contenu de poubelle
-r vidage de la poubelle.

```
if test -f $1
then
    mv $1 $HOME/.poubelle
    echo $1 a ete mis dans la poubelle
else
    case $1 in
        -l)   echo Il y a dans la poubelle
              ls $HOME/.poubelle;;
        -r)   echo "Destruction de la poubelle ..."
              rm $HOME/.poubelle/*
              echo "Fait";;
        *)    echo "Erreur de syntaxe"
    esac
fi
```


Exercice Pratique

```
# Programme statistique de décompte des objets Natural

# Test qu'il n'y a bien qu'un argument
if [ $# -ne 1 ]
then
      echo "usage is program_name <library_name>"
      exit 1
fi

# Test que la librairie existe bien
if ls /appli/fusr/$1/SRC > /dev/null 2>&1
then

      DIR="/appli/fusr/$1/SRC"
      DIR_COUR=`pwd`
      cd $DIR
      A=""
      X=0

      # Pour tous les fichiers du répertoire triés par extension
      for B in `ls | cut -d"." -f 2|sort`
      do
       if [ "$B" = "$A" ]
            then
                  X=`expr $X + 1`
            else                                    # Rupture

                  # Pour la première occurrence ne pas afficher le résultat
                  if [ "$A" != "" ]
                  then
                        echo $X occurences de $A
                  fi
                  A=$B
                  X=1
            fi
      done

      # Affichage du résultat pour la dernière occurrence
      echo $X occurences de $A

      # Retour au répertoire initial
      cd $DIR_COUR

else
      echo "/appli/fusr/$1/SRC n'existe pas"
      exit 2
fi
```