Université Bordeaux I | LST Informatique 2^{eme} Semestre | Année 2003-2004

UE INF103 : Utilisation des Systèmes Informatiques

Tous documents autorisés. | Groupe 2 INF A6 | 2 heures

---

1 Travail à réaliser

Attention, respectez scrupuleusement les instructions suivantes.

1. Créez dans votre répertoire d'accueil, un répertoire dont le nom est constitué de vos nom de login.
   `mkdir login1_login2`
2. Placez vous dans ce répertoire.
   `cd login1_login2`
3. Recopiez l'arborescence `~agriffau/USI/CC/Exercices/` dans votre répertoire.
   `cp -r ~agriffau/USI/CC/Exercices/ .`
4. Répondez aux questions sans modifier l'arborescence.
5. À la fin de la séance, placez vous dans votre répertoire d'accueil, puis recopiez dans le répertoire `~agriffau/USI/` votre arborescence.
   ```
   cd
   cp -r login1_login2 ~agriffau/USI/
   ```

2 Emacs, Processus, Unix

Exercice 1 (1,5 point) *Répondez dans le fichier* `exercice1.rep` *aux questions :*

1. *Que fait la commande* `halt` *?*
2. *Expliquez le message que vous obtenez si vous l'exécutez ?*

Réponse

```
Question 1 : Que fait la commande "halt" ?
------------
Reponse : un extrait du man
---------
NOM
       halt, reboot - Arrêter le système.

SYNOPSIS
       /sbin/halt [-n] [-w] [-d] [-f] [-i]
       /sbin/reboot [-n] [-w] [-d] [-f] [-i]

DESCRIPTION
       Halt  enregistre  le  fait  que le système va s'arrêter dans le fichier
       /var/log/wtmp, et ensuite demande au noyau d'arrêter ou  de  redémarrer
       le  système.  Si  halt ou reboot est appellé quand le système n'est pas
       dans au niveau d'exécution 0 ou 6, shutdown(8) sera appelé à  la  place
       (avec l'option -h ou -r).



Question 2 : Expliquez le message que vous obtenez si vous l'exécutez ?
------------
Reponse : deux situations possibles
```

```
---------
1/ halt: command not found

Cela signifie que le chemin d'accès à la commande "halt" n'est pas dans votre
variable PATH.


2/ /sbin/halt: Permission non accordee

Cela signifie que vous n'avez pas les droits d'exécution pour cette commande.
En effet la commande "ls -l /sbin/halt" donne :

-rwx------    1 root     root        12645 jui 19  2002 /sbin/halt

Cela signifie que seul l'utilisateur "root" et ceux ayant les droits de
"super-utilisateur" peuvent exécuter la commande "halt".
```

Fin réponse

Exercice 2 (1,5 point) *Le fichier* `exercice2` *contient la ligne :*

```
kill -9 $$
```

Répondez dans le fichier `exercice2.rep` *aux questions :*

1. *Que fait la commande* `bash exercice2` *?*
2. *Que fait la commande* `source exercice2` *?*
3. *Expliquez la commande* `kill -9 $$` *et les différences de comportement ?*

Réponse

```
Question 1 : Que fait la commande "bash exercice2" ?
------------
Reponse :
---------
$ bash exercice2
Processus arrêté

Le script exercice2 s'execute, mais la fin est "anormale", car un méssage
indiquant l'arrêt d'un processus esy affiché à l'écran.


Question 2 : Que fait la commande "source exercice2" ?
------------
Reponse :
---------
$ source exercice2

Le terminal est tué. Le script exercice2 tue le shell qui l'exécute.


Question 3 : Expliquez la commande "kill -9 $$" et les différences de comportement précédentes?
------------
Reponse :
---------
$ man kill
NOM
       kill - Envoyer un signal à un processus.
```

```
$ kill - l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      32) SIGRTMIN    33) SIGRTMIN+1
34) SIGRTMIN+2  35) SIGRTMIN+3  36) SIGRTMIN+4  37) SIGRTMIN+5
38) SIGRTMIN+6  39) SIGRTMIN+7  40) SIGRTMIN+8  41) SIGRTMIN+9
42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13
46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMAX-15 49) SIGRTMAX-14
50) SIGRTMAX-13 51) SIGRTMAX-12 52) SIGRTMAX-11 53) SIGRTMAX-10
54) SIGRTMAX-9  55) SIGRTMAX-8  56) SIGRTMAX-7  57) SIGRTMAX-6
58) SIGRTMAX-5  59) SIGRTMAX-4  60) SIGRTMAX-3  61) SIGRTMAX-2
62) SIGRTMAX-1  63) SIGRTMAX

La valeur 9 correspond au signal SIGKILL

$ echo $$
1068

$$ contient le PID du processus qui exécute la commande.

$ kill -9 $$
envoie un signal de terminaison au processus qui exécute la commande.

Donc :
$ bash exercice2
tue le shell "bash" que l'on crée pour exécuter le script

$ source exercice2
tue le shell "courant" (donc le terminal).
```

Fin réponse

Exercice 3 (1,5 point) *Répondez dans le fichier* `exercice3.rep` *aux questions :*
Dans le shell `bash`*, quelles sont les combinaisons de touches qui permettent de :*

1. *se positionner en début de commande,*
2. *revenir à la commande précédente,*
3. *revenir à la dernière commande commençant par* `unexpand`

Réponse

```
Dans le shell "bash", quelles sont les combinaisons de touches qui permettent de:

Question 1 : se positionner en début de commande ?
------------
Reponse :
---------
CTRL a

mais aussi
flèche "origine" (flèche nord-est)


Question 2 : revenir à la commande précédente ?
```

```
-----------
Reponse :
---------
CTRL p

mais aussi
flèche "haut"


Question 3 : revenir à la dernière commande commençant par "unexpand" ?
------------
Reponse :
---------
!unexpand
mais aussi
CTRL r unexpand
```

Fin réponse

Exercice 4 (2,5 points) *Le fichier* `exercice4` *contient les lignes :*

```
#!/bin/sh
VAR=who
set -x
echo $VAR
echo "$VAR"
echo '$VAR'
echo `$VAR`
eval $VAR
set +x
```

Répondez dans le fichier `exercice4.rep` *à la question :*

1. *Expliquez le résultat de la commande* `source exercice4` *?*

Réponse

```
Question : Expliquez ce que fait la commande "source exercice4" ?
----------
Reponse :
---------
$ source exercice4
+ echo who
who
+ echo who
who
+ echo '$VAR'
$VAR
++ who
+ echo alain :0 May 5 10:11
alain :0 May 5 10:11
+ eval who
++ who
alain    :0           May  5 10:11
+ set +x


Commentaires:

La commande "set -x" provoque l'envoi sur la sortie standard de toutes les
```

```
lignes (avec un préfixe indiquant un niveau d'imbrication) de commandes
interprétées par le shell après les substitutions des variables.

donc :

echo $VAR            (substitution de $VAR par who)
+ echo who           (echo de la chaine de caractère)
who

echo "$VAR"          ("" implique substitution de $VAR par who)
+ echo who           (echo de la chaine de caractère)
who

echo '$VAR'          ('' implique aucune substitution)
$VAR                 (echo de la chaine de caractère)

echo `$VAR`          (`` implique exécution de la commande après substitution)
++ who               (la commande "who" est exécutée)
+ echo alain :0 May 5 10:11 (le resultat devient parametre de "echo")
alain :0 May 5 10:11        (echo du resultat de "who")

eval $VAR                              (substitution de $VAR par who)
+ eval who                             (evaluation de "who")
++ who                                 (exécution de who)
alain    :0           May  5 10:11     (resultat de la commande "who")
```

Fin réponse

Exercice 5 (1 point) *Répondez dans le fichier* `exercice5.rep` *à la question :*

1. *Avec l'éditeur* `Emacs`*, si le buffer est dans le mode* `Fundamental`*, comment passez-vous en mode* `shell-script-mode` *afin de pouvoir profiter d'une mise en page automatique ?*

Réponse

```
Question : Quelle commande emacs passe en mode "shell-script-mode" ?
----------

Reponse :
---------

M-x shell-script-mode

mais aussi utiliser des noms de fichiers suffixés par ".sh"
```

Fin réponse

3 Scripts shell

Pour les exercices suivants, vous utiliserez les fichiers contenus dans le répertoire `Photos`, à savoir :
– des fichiers de photos : `photo1.jpeg photo2.jpeg photo3.jpeg photo4.jpeg photo5.jpeg`
– un fichier `description` associant pour chaque photo une liste d'attributs

```
photo1.jpeg mer famille
photo2.jpeg montagne famille
photo2.jpeg ski
photo4.jpeg anniversaire copains
photo5.jpeg anniversaire famille mer
```

L'objectif des scripts suivants est de classer par thème les photos sans gaspiller trop d'espace mémoire sur le disque.

Exercice 6 (2 points) *Modifiez le fichier* `exercice6` *en ajoutant une commande* `sed` *qui prend en paramètre un attribut et qui affiche sur la sortie standard la liste des photos ayant cet attribut.*

Réponse

```
#!/bin/sh

USAGE="echo Usage : ‘basename $0‘ attribut; exit 1"

if [ $# = 0 ]
    then
    eval $USAGE
fi

ATTRIBUT=$1

echo 'une premiere solution qui affiche les lignes'
sed -n /$ATTRIBUT/p  Photos/description

echo 'une seconde solution qui affiche seulement les noms des fichiers'
# Il ne faut pas que la ligne commence par un espace
sed -n /$ATTRIBUT/s/' '.*//gp  Photos/description
```

Fin réponse

Exercice 7 (6 points) *Modifiez le fichier* `exercice7` *afin d'avoir un script qui prend en paramètres un répertoire* `nomRep`*, un fichier* `nomPhoto` *et une liste d'attributs, et qui :*

1. *teste si le fichier* `nomPhoto` *existe (*`if [ -f nomFichier ]`*) dans le répertoire* `Photos`*,*
2. *si oui pour chaque paramètre* `attribut`*,*
   (a) *crée (*`mkdir`*) dans le répertoire* `nomRep` *un sous répertoire ayant pour nom* `attribut` *s'il n'existe pas déjà (*`if [ ! -d repertoire ]`*),*
   (b) *si le fichier* `nomRep/attribut/nomPhoto` *existe déjà, affiche un message indiquant son existence, sinon crée un lien physique (*`ln`*) de nom* `nomRep/attribut/nomPhoto` *vers le fichier* `nomRep/nomPhoto`

Les lignes suivantes correspondent à un exemple d'utilisation.

```
$ ls Photos/*
Photos/description  Photos/photo2.jpeg  Photos/photo4.jpeg
Photos/photo1.jpeg  Photos/photo3.jpeg  Photos/photo5.jpeg
$ ./exercice7 Photos photo3.jpeg foo bar
$ ls Photos/*
Photos/description  Photos/photo2.jpeg  Photos/photo4.jpeg
Photos/photo1.jpeg  Photos/photo3.jpeg  Photos/photo5.jpeg

Photos/bar:
photo3.jpeg

Photos/foo:
photo3.jpeg
$ ./exercice7 Photos photo3.jpeg foo bar
Photos/foo/photo3.jpeg deja traite
Photos/bar/photo3.jpeg deja traite
```

Réponse

```
#!/bin/sh

USAGE="echo Usage : ‘basename $0‘ repertoire fichier attribut1, attribut2,...; exit 1"

if [ $# -lt 3 ]
    then
    eval $USAGE
```

```
fi

REPERTOIRE="$1"
PHOTO="$2"

if [ -f $REPERTOIRE/$PHOTO ]
    then
    shift 2
    for ATTRIBUT in $*
      do
      if [ ! -d $REPERTOIRE/$ATTRIBUT ]
  then
  mkdir $REPERTOIRE/$ATTRIBUT
      fi
      if [ -f $REPERTOIRE/$ATTRIBUT/$PHOTO ]
  then
  echo $REPERTOIRE/$ATTRIBUT/$PHOTO deja traite
      else
  ln $REPERTOIRE/$PHOTO $REPERTOIRE/$ATTRIBUT/$PHOTO
      fi
    done
fi
```

Fin réponse

Exercice 8 (2 points) *Modifiez le fichier* `exercice8` *afin d'avoir un script qui prend en paramètres un répertoire* `nomRep`*, et un fichier* `descriptions`*, et qui :*

– *pour chaque ligne du fichier* `descriptions`*, crée s'ils n'existent pas déjà, les répertoires et les liens comme dans l'exercice précédent.*

Réponse

```
#!/bin/sh

USAGE="echo Usage : ‘basename $0‘ repertoire fichier; exit 1"

if [ ! $# -eq 2 ]
    then
    eval $USAGE
fi

( while read ligne
    do
    ./exercice7 $1 $ligne
done
) < $1/$2
```

Fin réponse

Exercice 9 (2 points) *Répondez dans le fichier* `exercice9.rep` *aux questions :*

1. *Que fait la commande* `ln` *?*
2. *Expliquez pourquoi ici il est préférable d'utiliser la commande* `ln` *à la place de la commande* `cp` *?*
3. *Expliquez pourquoi ici il est préférable d'utiliser la commande* `ln` *à la place de la commande* `mv` *?*

Réponse

```
Question 1 : Que fait la commande "ln"
------------
Reponse :
```

```
---------
$ man ln
NOM
       ln - Créer des liens entre fichiers.

       Un lien matériel est simplement une manière de nommer  un  fichier.  Un
       fichier  peut  avoir plusieurs noms. Un fichier n'est effacé réellement
       que lorsque son dernier nom  est  supprimé.  Le  nombre  de  noms  d'un
       fichier  est  indiqué par la commande ls(1).  Il n'y a pas de notion de
       nom 'original' : Tous les noms d'un  fichier  ont  exactement  la  même
       importance.


Question 2 : Pourquoi "ln" est-elle préférable à "cp"
------------
Reponse :
---------
"ln" ne fait que donner un autre nom à un fichier existant,
alors que "cp" duplique le fichier.

L'espace mémoire nécessaire pour "ln" est quasi nul,
alors que "cp" demande un espace libre au moins de la taille du fichier copié.

Ici un fichier peut apparaître dans de nombreux thèmes,
la place mémoire gagnée grâce à "ln" est donc très importante.


Question 3 : Pourquoi "ln" est-elle préférable à "mv"
------------
Reponse :
---------
"mv" déplace un fichier.
"mv" ne peut donc pas être utile pour notre problème.
Le fichier n'existera qu'en un seul exemplaire.
Il ne pourra pas être dans plusieurs thèmes simultanément.
```

Fin réponse