Corrigé du TP 2 Système

C. Pain-Barre

INFO - IUT Aix-en-Provence

version du 13/11/2012

Corrigé de l'exercice 1 (à traiter en TD)

1. La première ligne affichée par **ls** indique que `truc` est le propriétaire de `.` (répertoire de travail).
2. `truc` est le propriétaire de `fic`. Il possède donc les droits **r-x**.
3. `bidule` est membre du groupe de `fic`. Il possède donc les droits **rw-**.
4. `bidule` n'est ni le propriétaire de `fic`, ni membre de son groupe. Il a donc les droits **-wx**.
5. Pour supprimer `fic`, il faut au minimum les droits d'écriture et d'exécution sur le répertoire de travail. Ceux qui les possèdent sont `truc` et les utilisateurs qui ne sont pas membre du groupe `bidule`.
6. Non, s'il n'y a pas eu intervention de `root`, `rep` n'a pu être créé qu'avec des permissions différentes du répertoire de travail (ou alors par déplacement mais ce n'est pas une création). En effet, le propriétaire (et créateur) de `rep` est `machin`. Puisque le groupe de `rep` est `bidule` et que seul `machin` (ou `root`) peut le changer, on en déduit que `machin` est membre du groupe `bidule`. Or, pour créer un fichier il faut les droits **wx** sur le répertoire mais `machin` a les droits du groupe, soit uniquement **r**.
7. Aucun droit ! En effet, en tant que membre du groupe `bidule`, `machin` n'a pas les droits d'exécution sur le répertoire de travail et ne peut donc pas accéder à `rep` !

Corrigé de l'exercice 2

1. **cd ~/tp/tpunix**
2. **ls -l**
3. `$` **umask**
   `0022`

   ➪ Selon les versions, peut écrire **022** ou encore **22**, ce qui est équivalent. Notons que l'administrateur peut fixer un masque par défaut différent.

   (a) **rw-r--r--** (c.-à-d. lecture/écriture pour le propriétaire et seulement lecture pour le groupe et les autres)

   (b) **rwxr-xr-x** (i.e. lecture/écriture**/exécution** pour le propriétaire et lecture**/exécution** pour le groupe et les autres)

   (c) `$` **vi essai1.txt**
   (dans vi)
   i
   une ligne de texte quelconque Esc
   :wq
   `$`

   (d) **mkdir rep1**

   (e) L'utilisation de l'option **-d** permet d'avoir les informations détaillées sur `rep1` lui-même, et pas sur son contenu...

   (f) Si vous vous êtes trompés, peut-être devriez-vous revoir le rôle de **umask**...

4. **umask 077**
   (a) **vi essai2.txt**
   (b) **mkdir rep2**
   (c) **ls -ld rep1 essai1.txt rep2 essai2.txt**
   (d) les permissions de `essai2.txt` devraient être **rw-------**. Celles de `rep2` devraient être **rwx------**
   (e) *. . . pas besoin de corrigé pour cette question. . .*
5. (a) Les deux masques ont le même effet pour la création de ces fichiers qui sont créés sans l'exécution. Les permissions seront **rw-------**
   (b) Pour le masque `066`, le groupe et les autres auront le droit d'exécution. Les permissions seront donc **rwx--x--x**. Mais pour le masque `077`, ils n'auront pas le droit d'exécution et les permissions seront **rwx------**

   ✍ Notons que cela s'applique aussi aux "exécutables" créés par les compilateurs.

6. (a) **umask 066**
   (b) **vi essai3.txt**
   (c) **mkdir rep3**
   (d) **ls -ld rep1 essai1.txt rep2 essai2.txt rep3 essai3.txt**
   (e) *. . . pas besoin de corrigé pour cette question. . .*
   (f) *. . . pas besoin de corrigé pour cette question. . .*
7. *. . . pas besoin de corrigé pour cette question. . .*
8. *. . . pas besoin de corrigé pour cette question. . .*
9. Parce que la modification du masque n'est effective que depuis le shell sur lequel elle a lieu. Lorsque la nouvelle connexion est établie, le masque est remis à sa valeur par défaut (lue dans un fichier de configuration).

Corrigé de l'exercice 3

1. **chmod 700 essai1.txt** puis **ls -l essai1.txt**.
   Avec le mode symbolique, on aurait pu écrire **chmod u+x,go-r essai1.txt**
2. **chmod g+rw,o+x essai2.txt** puis **ls -l essai2.txt**.
   Avec le mode absolu, on aurait écrit **chmod 661 essai2.txt**
3. **chmod 404 essai3.txt** ou **chmod u-w,o+r essai3.txt**
   puis **ls -l essai3.txt**
4. **vi essai3.txt**
   Au moment de l'entrée en mode insertion, **vi** avertit qu'on va modifier un fichier qui n'est accessible qu'en lecture seule, et pas en lecture/écriture. Au moment de le sauvegarder, **vi** ne veut pas sauver le fichier qui est protégé en écriture. Il faut le forcer à le faire en utilisant le caractère **!**
5. **:wq!** pour le forcer à sauver et quitter.

Corrigé de l'exercice 4

1. **`cd`**
2. **`mkdir public prive`**
3. **`chmod 700 prive`**
4. Les permissions de `public` devraient être **`rwxr-xr-x`** et le droit d'exécution doit être accordé à tout le monde sur votre répertoire d'accueil. Si ce n'est pas le cas, il faut exécuter **`chmod 755 public`** et **`chmod a+x ~`** ainsi que remettre le bon masque avec **`umask 022`**
5. **`cp tp/tpunix/essai2.txt tp/tpunix/cigale.txt public`**
6. Car c'est une copie de `essai2.txt` de `tpunix`, donc créée avec les droits du fichier d'origine (**`rw-rw---x`**) moins ceux du masque (qui enlève l'écriture pour le groupe et les autres).
7. **`cp tp/tpunix/essai1.txt tp/tpunix/essai3.txt prive`**

Corrigé de l'exercice 5

1. Si votre voisin tape : **`ls ~`***vous*`/public`** cela doit réussir
2. **`vi ~`***vous*`/public/cigale.txt`** doit éditer le fichier (en lecture seule)
3. **`ls ~`***vous*`/prive`** doit échouer.
4. **`vi ~`***vous*`/public/cigale.txt`** ne permet pas d'éditer le fichier. Notons que la commande **`cd ~`***vous*`/prive`** aurait aussi échoué.
5. Non, cela ne changerait rien. Simplement, personne d'autre que vous ne pourrait connaître le contenu de votre répertoire d'accueil. Mais les utilisateurs connaissant l'existence de votre répertoire `public` y ont quand même accès (s'ils ont le droit d'exécution sur votre répertoire d'accueil).

Corrigé de l'exercice 6

1. (a) Le nombre d'espaces séparant les arguments n'a pas d'importance. Il y a 5 arguments à **echo** qui les écrits séparés par un seul espace
   (b) Les espaces sont écrits tels quels car ils sont protégés par les guillemets et il n'y a plus qu'un seul argument à **echo**
   (c) Les quotes et les guillemets peuvent être placés n'importe où. Ici, ils protègent les espaces supplémentaires. **echo** reçoit les deux arguments `Je` et `fais␣␣␣partie␣␣des␣␣␣␣␣␣arguments.` et on obtient le même affichage qu'à la question précédente.
2. **`cd ~/tp/tpunix`**
3. **`ls`**
4. (a) **echo** reçoit de nombreux arguments car bash remplace **`*`** par la liste triée des fichiers du répertoire (`cigale.txt`, `essai1.txt`, etc.) ; **`$PATH`** par `/usr/local/bin:/usr/bin:/bin:...`, et **`!!`** par la dernière commande (`ls`)
   (b) bash supprime les backslashs protégeant les caractères spéciaux qu'il ne traite pas, et **echo** reçoit 6 arguments
   (c) bash supprime les quotes protégeant les caractères spéciaux, et **echo** ne reçoit qu'un argument
   (d) bash supprime les guillemets qui ne protègent ni **`$PATH`** ni **`!!`** qui sont remplacés. Mais **echo** ne reçoit qu'un seul argument.

(e) bash supprime les guillemets dans lesquels **$PATH** est protégé par le backslash. À l'extérieur des guillemets, **!!** est aussi protégé par un backslash. Au total, **echo** reçoit deux arguments.

5. Voici une possibilité parmi tant d'autres :
   echo "~ C'est * bon * pour tout ?"' \" \'\'' !!! * ~ $PATH "'" ' ??"
6. **vi '*.txt'**

Corrigé de l'exercice 7

1. `$ type echo`
   `echo est une primitive du shell`
   ➪ **echo** est une commande interne
2. **help echo** affiche cette aide
3. Deux possibilités :
   `$ type -a echo`
   `echo est une primitive du shell`
   `echo est /bin/echo`
   ➪ où **echo** est recherché parmi les commandes internes, externes, alias, fonctions et les mots clés
   `$ type -P echo`
   `/bin/echo`
   ➪ où la recherche est limitée aux commandes externes
4. `$ echo $PATH`
   `/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games`
5. **echo** apparaît effectivement dans `/bin`
6. **man echo** affiche le manuel
7. **/bin/echo --help** affiche cette aide
8. **help umask**
9. **man bash**

Corrigé de l'exercice 8

1. . . . *pas besoin de corrigé pour cette question. . .*
2. . . . *pas besoin de corrigé pour cette question. . .*
3. **ls ~cpb/public/unix**
4. **echo ~root**

Corrigé de l'exercice 9

1. **cd ~cpb/public/unix**
2. **ls**
3. . . . *pas besoin de corrigé pour cette question. . .*

4. . . . *pas besoin de corrigé pour cette question. . .*
5. En effet, la commande exécutée devient (à peu près) :

   `ls dateng.bash dateng.tcsh dates.txt debian decale.txt des_lignes.txt`

   qui comporte `debian` qui est un répertoire. Or, sans option, **ls** affiche son contenu.
6. Car l'option **-d** désactive l'affichage du contenu des répertoires en arguments qui sont traités comme des fichiers ordinaires.
7. Le motif **`z*`** ne correspond à aucun fichier et est laissé tel quel par bash. **echo** reçoit donc **`z*`** en argument et l'affiche. En revanche **ls** affiche une erreur car il ne peut afficher les informations sur **`z*`** qui n'existe pas dans le répertoire de travail.
8. Produit l'affichage des entrées qui commencent par **`d`** suivies de celles qui commencent par **`i`**
9. **`ls -ld d* i*`**
10. (a) **`echo [aeiouy]*`** ou **`ls -d [aeiouy]*`**
    (b) **`echo [^aeiouyAEIOUY]*`** ou **`ls -d [^aeiouyAEIOUY]*`**
    (c) **`echo *.txt`** ou **`ls -d *.txt`**
    (d) **`echo *[es].txt`** ou **`ls -d *[es].txt`**
    (e) **`echo *a*`** ou **`ls -d *a*`**
11. Ce motif n'exclut que les fichiers dont le nom est une suite de **`a`** (`a`, `aa`, `aaa`, etc.). En effet, ce motif doit s'interpréter comme "tout fichier dont le nom contient au moins un caractère différent de **`a`**".
12. . . . *pas besoin de corrigé pour cette question. . .*
13. **`ls -d ~/tp/tpunix/*\**`
14. **`rm -i ~/tp/tpunix/\*.txt`**
15. . . . *pas besoin de corrigé pour cette question. . .*

Corrigé de l'exercice 10

1. **`cd ~/tp/tpunix`**
2. **`cp ~cpb/public/unix/*.txt .`**
3. **ls** devrait le confirmer.
4. **`cp c*.txt ..`**
5. **`rm ../*e*.txt`**
6. *A priori*, **`rm ../*.txt`** devrait suffire

Corrigé de l'exercice 11

La dernière commande **mv** échoue car `toto` ne dispose pas des droits de modification sur `rep1`.

fichier	propriétaire	groupe	permissions
`fic2`	titi	tata	**`rw-r-x-w-`**
`fic4`	truc	chose	**`rwxr-----`**
`fic5`	titi	chose	**`rwxrw--w-`**
`fic6`	toto	users	**`-wxr-xr--`**
`rep1/fic8`	toto	users	**`rwxrw-r-x`**
`fic9`	toto	users	**`-wxr-xr--`**