TP 6++ – promo IG3

V. Berry & J. Fortin

26 septembre 2011

Résumé

Vous choisirez un des sujets présentés ci-dessous et travaillerez par groupe de deux. Ce sujet peut vous prendre plus d'une séance.

1 Aide personnalisée

On désire gérer un fichier `~/memo.hlp` contenant des aides sur des commandes diverses. Son format est le suivant : pour chaque sujet, on trouve une succession de quatres "#" puis une description du thème de l'aide, suivit de deux "-" suivit de l'aide proprement dite sur ce thème. Par exemple :

```
#### gnuplot : tracer des courbes
Commande : gnuplot
Aide : gnuplot puis help
faire afficher : gnuplot puis plot 'fichier-donnees' using 3:4 : tracera la
                 courbe a partir de la ligne 3 et 4 du fichier fichier-donnees
sortie postscript : faire "set terminal postscript" puis voir l'aide sur 'set'
                    puis 'output'
#### tar :  compresser , decompresser des fichiers et des arborescences
Archivage de fichiers :
tar cvf - repertoire_a_archiver | gzip -9c > nom_archive.tar.gz

Pour dearchiver une archive nom_archive.tar.gz :
gzip -dc nom_archive.tar.gz | tar xvf -

#### Suffixe d'un fichier avec basename
commande : basename fic suff
enleve le path du fichier fic et lui enleve le suffixe designe par suff
```

1. Donnez le code d'un script shell permettant de récupérer la liste des thèmes sur lesquels on dispose d'aide. Dans l'exemple précédent, on afficherait (sans les #### !) :

   ```
   gnuplot : tracer des courbes
   tar  compresser , decompresser des fichiers
   suffixe d'un fichier avec basename
   ```

   Indice : commandes grep et cut (bien lire le man).

2. Donnez le script `aide.sh` permettant de récupérer l'aide correspondante à un thème. Pax exemple, pour le thème `"gnuplot : tracer des courbes"` on affiche

   ```
   Commande : gnuplot
   Aide : gnuplot puis help
   faire afficher : gnuplot puis plot 'fichier-donnees' using 3:4 :
   tracera la courbe a partir de la ligne 3 et 4 du fichier
   fichier-donnees
   sortie postscript : faire "set terminal postscript"
   puis voir l'aide  sur 'set'  puis 'output'
   ```

   Ingrédients : utiliser les commandes `cat, if, egrep -q, rm, split` (dans le bon ordre et avec les bons paramétres), les arguments donnés en entrée à un shell ($1 pour la première, etc), et la variable $? qui contient 0 seulement si la commande egrep précédente a trouvé ce qu'elle cherchait.

3. Ecrivez un script `ajoute.sh` correspondant à l'ajout d'une information dans le fichier d'aide : lecture au clavier du thème puis de l'aide correspondante sur 4 lignes ; transformation au format d'aide correct ; insertion en fin de fichier d'aide.

   Indice : commandes `echo, read, cat` et redirections.

4. Choisissez une syntaxe de ligne de commande permettant, par l'intermédiaire d'un seul et même script `aide-totale.sh`, d'obtenir toutes les fonctionnalités précédentes. Pour se faire on aura recours à des drapeaux (flags), par exemple : `-i` pour insérer un nouveau thème, `-t` pour obtenir de l'aide sur un thème indiqué, etc.

   Indices : commandes `if, -eq, ...`, les arguments (1,2, ...) reçus par le script et les bouts de scripts précédents.

2 Nettoyage d'arborescence

Ecrivez un script `nettoie.sh` qui permet de faire le nettoyage dans une arborescence indiquée par sa racine :

1. le script doit lister dans l'arborescence indiquée, tous les fichiers de `core`, tous les fichiers dont le nom commence par `#` ou `.saves-`, tous ceux dont le nom finit par `~`, `.bak`, ou `.class`.

   Indice : créez une sous-arborescence dans laquelle vous ajouterez des sous-répertoires et des fichiers à supprimer un peu partout. Puis testez votre script sur cette arborescence. Pour tout ça : commandes `touch, find`, utiliser $1 pour le paramètre donné en entrée à un script.

2. Faîtes une copie de votre arborescence de test dans un fichier d'archive (commande `tar`). Maintenant modifiez le script pour qu'il détruise tout fichier des types précédemment listés. Attention : testez uniquement sur votre arborescence de test (dont vous pourrez rétablir le contenu entre chaque test depuis l'archive !).

3. Modifier le script afin que pour chaque fichier identifié comme à détruire, le script demande à l'utilisateur une confirmation en affichant le nom du fichier. En cas de réponse positive, il détruit effectivement le fichier. Le script doit afficher ce qu'il a fait avec le fichier (supprimé ou pas).

   Ingrédients : commandes `find, read, for $fic in ... , egrep -q, rm, if` , la syntaxe `$(find ....)` qui rcupre la liste des rsultats du find dans une variable, et la variable $? qui contient 0 seulement si la commande egrep précédente a trouvé ce qu'elle cherchait.

4. Modifier le script afin qu'il trouve aussi tous les fichiers dont le suffix est `.ps` ou `.tar` et les compresse automatiquement (commande `gzip -9`).

5. Modifier le script afin qu'il localise tous les répertoires ne contenant aucun fichier et les détruise après confirmation de l'utilisateur.

6. Modifier le script pour qu'il puisse être appelé avec une option `-s taille` (ou `taille` est un nombre entier entre 1 et 2000). Dans ce cas, le script donne le nom et la taille de tous les fichiers dont la taille dépasse `taille` K (rappel : $1K = 1024$ octets). Exemple :

   ```
   >  nettoie.sh -s 1000
   ...
   Fichier de taille > 1000 K :
   [1] ~/shuttle.bmp : 2120 K
   [2] ~/.netscape/cache/1AFD343 : 1504 K
   [3] ~/.netscape/cache/1ADE187 : 3400 K
   ```

7. Modifier le script précédent pour qu'il accepte une option `-d` lui demandant de détruire (avec confirmation et sélection) la liste des fichiers repérés par l'option `-s`. Note : il faudra bien sûr vérifier que cette option a été demandée.

8. Modifier le script pour qu'il calcule la place gagnée en ayant supprimé les fichiers indiqués des différents types et en ayant compressé d'autres fichiers. Exemple :

   ```
   >  nettoie.sh -s 2000
   ...
   BRAVO : vous avez gagne 7894 K par cette operation de nettoyage.
   Merci les scripts !
   ```