TP5 – Shell – Scripts

OBJECTIFS :

- Comprendre la structure et le fonctionnement des scripts Shell au travers d'exemples progressifs.
- Pouvoir écrire et tester des scripts utiles.

RESSOURCES : document de TD intitulé « **Programmation de Scripts** »

1- Premier script : for et if

Enregistrer le script suivant dans un fichier portant le nom « ls_fic_blanc.sh »

```
#shabang indiquant le shell d'exécution
# !/bin/sh
for f in * ; do                 # pour tout objet f (fichier, dossier) contenu dans le dossier courant
                                # les éléments de la liste peuvent être exprimés en dur s'ils sont connus
                                # (for f in fic1 fic2 fic3) ou générés par une commande (for nb in `seq 1  10`)
                                # ; (séparateur) nécessaire entre 2 commandes sur la même ligne
  if test  -f $f  ; then        # si f est un fichier. If s'intéresse ici au résultat de la commande test mais
                                # peut être utilisé avec d'autres commandes. Ex :  if grep <motif>  <fichier>
                                # La commande test expression peut être remplacée par [ expression ]
    echo $f
  fi                            # fermeture du if
done                            # fermeture du for
```

Rendre ce fichier exécutable (+x) (u : seulement pour le propriétaire) en exécutant la commande :

chmod u+x ls_fic_blanc.sh

Créer 2 ou 3 fichiers en leur donnant des noms comprenant un ou plusieurs blancs :

touch 'fic1 blanc' ***touch 'fic2 blanc'*** ***touch 'fic3 blanc'***

Exécuter le script ls_fic_blac.sh et vérifier le résultat.
Corriger le script (sans ajout de commande) pour qu'il liste tous les fichiers. Vérifier le résultat.
Ajouter une ligne au script pour remplacer toute suite de blancs par un '_' dans les noms qui en contiennent. Exécuter et vérifier le résultat.
Compléter le script pour changer le nom de chaque fichier avec blanc (blancs remplacés par '_').

2- Deuxième script : arguments et while

Enregistrer le script suivant et tester le en fournissant une suite quelconque d'arguments.

```
# !/bin/sh
Iarg=0
while [ $# -gt 0 ] ; do         # tant que le nombre d'arguments ($#) est supérieur à 0
                                # [ -n "$1" ]  fonctionne aussi ("$1" non vide)
   let iarg=$iarg+1             # ou iarg=`expr $iarg + 1`
   echo "Argument num $iarg : $1"
   shift                        # décalage des arguments vers la gauche (sortie du 1er)
done
```

Ecrire le script **rec_motif_fichiers.sh** <motif> <ficfier1> [<fichier2>] … qui vérifie si le motif indiqué se trouve dans le ou les fichiers fourni(s), en affichant le nom du fichier et la ligne contenant le motif (avec son numéro).

3- Troisième script : case

Compléter le script suivant de manière à ce qu'il liste les caractéristiques complètes des dossiers présents dans le répertoire courant (option –d), des fichiers (option –f), des fichiers exécutables (option –x), ou de tous les objets (sans option). Syntaxe : lobj [-dfxa]

```
# !/bin/sh
usage="$0 [-dfx]"
if [ $# -gt 1 ] ; then                # si nombre d'arguments > 1
   echo "Erreur : $usage"; exit 1;
fi
case $1 in
   -d)          #compléter avec les actions à exécuter
        ;;      # fin du bloc
   -f)
        ;;
   -x)
        ;;
   *)           #les autres cas
esac            #fermeture du case
```

Transformer le script de manière à ce qu'il puisse prendre un nom de répertoire en argument. Penser à traiter tous les cas : pas d'argument, un seul argument (option ou nom de répertoire), deux arguments.

4- Transformation d'une ligne de commande

Une commande a la forme suivante : ***nom_commande –v –o <nom de fichier> -T <valeur> -S –l –e –t***
Mais une nouvelle version du logiciel n'accepte plus les options –T et -e, et a remplacé l'option –t par –x

Ecrire un script qui transforme l'ancienne ligne de commande (fournie en argument) et affiche la nouvelle forme.

5- Ecrire un script ***tuer_proc <motif>*** qui tue le processus dont le nom contient le motif donné en paramètre. Si plusieurs processus correspondent au motif fourni, une erreur est signalée et le processus n'a pas d'effet.

6- Script avec fonction

On peut utiliser des fonctions dans un script shell. Soit, par exemple, la fonction suivante qui reçoit trois paramètres : $1 $2 et $3

```
Lister() {
for obj in * ; do
   [ $1 = -f -o \( $1 = -d -a -d "$obj" \) ] && echo $2$obj
   If [ -d "$obj" ]; then
        cd "$obj" ; Lister $1 $2$3 $3          #  $2$3 est la concaténation de $2 et $3
        cd ..
   fi
done
}
```

Que fait cette fonction si elle est appelée avec Lister -f ' ' '…'
Que fait-elle si elle est appelée avec Lister -d ' ' '…'

Ecrire un script ***arbre [-d] [<nom de répertoire]*** qui liste de façon indentée tous les fichiers et dossiers de l'arborescence du répertoire spécifié (répertoire courant par défaut). Si l'option –d est spécifiée, seuls les répertoires sont listés.