TP n°5 & 6
Processus & Variables d'environnement
Introduction à la programmation shell

Ceci est un double TP qui est destiné à vous familiariser avec la gestion de processus et les variables d'environnement sous Unix ainsi qu'à la programmation shell.
En ce qui concerne la programmation shell, vous avez un rappel fourni avec ce TP.

1 Gestion de processus

- Exécuter l'application xemacs. Pouvez-vous exécuter une autre commande dans le même shell ? Justifier votre réponse.

- Suspendre l'application xemacs en appuyant sur les touches <ctrl>+Z. Peut-on exécuter une autre commande dans le même shell ? Justifier.

- Reprendre l'exécution de l'application suspendue en premier plan.

- Suspendre à nouveau l'application puis reprendre son exécution en tâche de fond. Peut-on exécuter une autre commande dans le même shell ? justifier.

- Donner une commande qui permet d'arrêter l'application graphique exécutée.

- Exécuter l'application gedit comme tâche de fond. Fermer la fenêtre de l'interpréteur de commandes à partir de laquelle l'application est lancée. Quel est l'effet de cette action sur l'application gedit ? Justifier votre réponse.

- Dans un nouveau terminal, exécuter la commande nohup gedit ; quel est le processus père de gedit ? Ensuite, arrêter le processus du shell. Quel est l'effet sur l'application gedit ? Quel est le (nouveau) processus père de l'application gedit ?

- Afficher en temps réel les informations sur l'ensemble de processus exécutés sur votre machine.

- En utilisant la commande ps, afficher les informations détaillées sur tous vos processus.

- Lancer à nouveau l'application xemacs en tâche de fond. Donner une commande qui renvoie uniquement le PID du processus qui exécute cette application.

2 Variables d'environnements

- Afficher l'ensemble des variables d'environnements définies dans votre shell avec leurs valeurs associées.

- Donner une commande qui renvoie la liste des noms des variables d'environnement sans les valeurs associées. La liste doit être triée en ordre alphabétique.

- Afficher la valeur associée à la variable *PATH*. Quel est le rôle de cette variable ?

- Donner une commande qui renvoie le nombre de répertoires déclarés dans la variable *PATH*.

- À l'aide de la commande *which*, localiser le compilateur gcc.

- A l'aide de la commande *which*, essayer de localiser la commande ifconfig. Justifier le résultat obtenu.

- A l'aide de la commande alias, renommer la commande *ls* pour que le résultat de son appel soit équivalent à la commande *ls -l*.

TP n°5 & 6
Processus & Variables d'environnement
Introduction à la programmation shell

3 Affichage et interprétation des variables

- Dans un shell, exécutez la commande *X="ls -l"*. Ensuite exécuter les commandes suivantes et pour chacune justifier le résultat obtenu :

  o echo X
  o echo $X
  o echo '$X'
  o echo "$X"
  o echo `$X`

4 Portée Des Variables

- Dans un shell bash taper les commandes suivantes et justifier les affichages obtenus :

  o X1=3
  o Y1=10
  o Z1=4
  o export Y1
  o env | grep X1= >echo $X1 >echo $x1
  o env | grep Y1= >unset Y1 >export X1 >bash o env | grep X1= >echo $Z1 >exit
  o echo $Z1

5 Premier script shell

Développer un script shell *bonjour.zsh* qui :

- Affiche le message Hello suivi de la liste des noms passés comme paramètres. Par exemple, l'appel *bonjour.zsh Yann Lori* a le message Hello Yann, Lori

- Modifier le script pour qu'il affiche en première ligne le nom du script et le numéro du processus qui l'exécute, et en deuxième ligne le message suivant : Salut à x personnes où x est le nombre de noms passés comme paramètres.

6 Gestion d'un drapeau

Un drapeau est un simple fichier texte qui peut contenir soit la valeur 1, soit la valeur 0. On dit que le drapeau est positionné si le fichier contient la valeur 1. On voudrait développer une commande nommée flag qui a le fonctionnement suivant :

o *flag <nom_du_drapeau>* : retourne l'état actuel du drapeau. Si le fichier correspondant n'existe pas, il affiche un message d'erreur.

o *flag <nom_du_drapeau> on* : positionne le drapeau (c.-à-d. met dans le fichier correspondant la valeur 1). Si le fichier n'existe pas, la commande doit le créer.

o *flag <nom_du_drapeau> off* : enlève le drapeau (c.-à-d. met dans le fichier correspondant la valeur 0). Si le fichier n'existe pas, la commande doit le créer.

o *flag <nom_du_drapeau> flop* : inverse l'état du drapeau. Si le fichier correspondant n'existe pas, la commande affiche un message d'erreur.

TP n°5 & 6
Processus & Variables d'environnement
Introduction à la programmation shell

7 Gestion des sauvegarde

Écrire un programme shell nommé *saveTXT.zsh* qui permet de copier dans un répertoire nommé *~/backup* tous les fichiers qui se trouvent dans l'arborescence du répertoire de connexion et qui se terminent par le suffixe « *.txt* ». Si le répertoire *~/backup* n'existe pas alors la commande doit le créer.
Modifier le programme précédent afin de ne pas écraser les fichiers existant dans le répertoire *~/backup*.

8 Poubelle

Développer une commande nommée *poubelle* qui permet de transférer les fichiers à effacer dans un répertoire nommé *~/trash*. La syntaxe de cette commande est la suivante :

- *poubelle f1 f2 f3 . . . fn* a pour effet de transférer les fichiers de *f1* à *fn* dans le répertoire *~/trash*.

- *poubelle -f* a pour effet d'effacer le contenu du répertoire *~/trash*.

- *poubelle (sans argument)* a pour effet d'afficher un message d'aide décrivant la syntaxe correcte de la commande.