TP : Processadmin

Réalisé par : Chams TMAR (GL3-2)

Dans ce TP, on voulait écrire une application shell pour gérer les processus du système Linux. Ainsi, on va afficher un menu à l'utilisateur, présentant différentes options.
En fonction du choix de l'utilisateur, le script exécute différentes actions à l'aide d'une structure de contrôle `case` :

- Si le choix est **1**, le script utilise la commande `ps -ef` pour lister tous les processus en cours d'exécution avec leurs PID et noms.
- Si le choix est **2**, le script utilise `ps -ef` pour lister les processus par utilisateur, triés par PID, avec les colonnes PID, Nom du Processus et PPID. Il filtre les résultats en fonction de l'ID utilisateur actuel avec `grep` et trie les résultats numériquement avec `sort`.
- Si le choix est **3**, le script utilise `ps -e` pour afficher le PID et le nom du processus de login de chaque utilisateur connecté en utilisant la commande `grep`.
- Si le choix est **4**, le script demande à l'utilisateur de fournir un nom d'utilisateur, puis utilise `pkill -p -i` pour déconnecter (tuer) tous les processus associés à cet utilisateur.
- Si le choix est **5**, le script affiche les processus fils du processus en cours d'exécution (le script lui-même). Il utilise `ps -p` pour obtenir les informations sur le processus actuel et itère ensuite sur les processus parents jusqu'à atteindre le processus racine.
- Si le choix est **6**, le script affiche les processus parents du processus en cours d'exécution (le script lui-même). Il utilise une approche similaire à celle de l'option **5**.
- Si le choix est **7**, le script quitte en utilisant la commande `exit`.

Voici le script expliqué :

```
echo "Application"

while true
do

echo ""
```

```
echo "1- Lister tous les processus"
echo "2- Lister les processus par utilisateur triés par PID (PID, Nom Process, PPID)"
echo "3- Afficher le PID du processus de login de chacun des utilisateurs connectés"
echo "4- Déconnecter un utilisateur"
echo "5- Lister les processus parents d'un processus donné (PID, Nom)"
echo "6- Lister les processus fils d'un processus donné (PID, Nom)"
echo "7- Quitter"


echo ""
echo "Tapez votre choix"
read choix

case $choix in
1)   ps -ef | awk '{printf("%d\t%s\n", $2, $8);}'  #afficher tous les processus actuels en extended format puis pour chaque ligne écrire le 2ème et le 8ème argument (qui sont le PID : entier et la commande : string)
     ;;
2)   user=$(id -u)  #assigner le user ID de l'utilisateur courant à la variable user
     ps -ef \  #afficher tous les processus en détail
     | grep -w "^$user" \  #rechercher exactement les lignes commençant par le user ID
     | awk '{printf("%d\t%s\t%d\n", $2, $8, $3);}' \  #pour chaque ligne écrire le 2ème, le 8ème et le 3ème argument (qui sont le PID : entier, la commande : string, le PPID : entier)
     | sort -n   #trier par ordre numérique des PIDs
     ;;
3)   ps -e | grep 'login'  #afficher tous les processus peu importe leur terminal et filtrer pour n'afficher que les lignes contenant 'login'
     ;;
4)   read -p "Donner l'utilisateur: " user  #lire le user saisi et l'assigner à la variable user
```

```
        if [[  -z  "$user" ]]  #si le user saisi est vide
        then
                user=$(whoami)  #on le remplace par le user actuel
        else
                :  #sinon on ne fait rien
        fi
        pkill -p -i $user  #on tue le processus de ce user afin
de le déconnecter
        ;;
5)      pid=$$  #assigner le pid du processus actuel à la
variable pid
        ps -p $pid -o pid,cmd  #extraire le pid et la commande du
processus ayant $pid comme pid
        pid=$(ps -p $pid -o ppid=) #on assigne le ppid de
l'ancien processus à la variable pid
        while [ $pid -gt 0 ]  #tant que $pid est supérieur à 0
        do
        ps -p $pid -o pid=,cmd --no-headers  #extraire le pid et
la commande du processus ayant $pid comme pid
        pid=$(ps -p $pid -o ppid=) #on assigne le ppid de
l'ancien processus à la variable pid
        done
        ;;
6)      pid=$$  #assigner le pid du processus actuel à la
variable pid
        echo "Tous les processus fils du processus $pid: "
        pgrep -P $pid | while read child_pid;do  #on trouve tous
les processus dont le ppid est $pid et on lit tous les fils
                ps -o pid,cmd h -p $child_pid  #pour chaque fils, on
affiche les champs pid et commande
        done
        ;;
7)      exit
        ;;


        ;;
esac

done
```