**Linux – Fiche 17, 18, 19  
Scripts – arithmétique, conditions, boucles**

Connectez-vous en ssh à la machine courslinux.ipl.be. Si vous utilisez une autre machine, exécutez la commande suivante pour installer le jeu *cowsay* :

sudo apt install cowsay

1. Ajoutez les lignes suivantes à la fin de votre fichier de configuration ~/.bashrc :

# game cowsay

cows=($(ls /usr/share/cowsay/cows/))

num=$(($RANDOM % ${#cows[\*]}))

cowsay -f ${cows[num]} "Hello, $USER! I am $(uname -o) on $HOSTNAME"

Rechargez votre fichier avec la commande « source ~/.bashrc ». Puis réexécutez plusieurs fois cette commande.

Expliquez chacune des 3 lignes d’instructions que vous avez ajoutées à votre .bashrc.

* cows=($(ls /usr/share/cowsay/cows/))

La variable cows recoit comme valeur un tableau de liste des fichiers du répertoire

* num=$(($RANDOM % ${#cows[\*]}))

La variable num recoit comme valeur le modulo du nombre d’éléments dans le tableau et une valeur aléatoire.

* cowsay -f ${cows[num]} "Hello, $USER! I am $(uname -o) on $HOSTNAME"

Le personnage selectionné dans le tableau dit la phrase : « Hello corentin.dhaeyere ! I am GNU/Linux on courslinux.ipl.b

1. Écrivez une instruction qui permet d’afficher tous les personnages du jeu *cowsay*. Ils sont définis dans le répertoire /usr/share/cowsay/cows/. Utilisez le nom du personnage dans la phrase qui sera affichée.

for f in $(ls /usr/share/cowsay/cows/)

do cowsay -f $f "$f says hi!"

done

1. Ecrivez le script *somme.sh* qui affiche la somme des éléments d'une liste initialisée aux valeurs 1, -3, 45, 67.

#!/bin/bash  
#############################################################  
# somme-liste.sh  
# Calcule différentes sommes arithmétiques  
#   
#   
# corentin.dhaeyere  
# 04 May 2022  
#############################################################  
echo  
echo " Somme des nombres d’une liste"  
echo " -----------------------------"  
liste="1 -3 45 67"  
for nbr in $liste;  
 do((somme+=nbr))  
 echo "Avec $nbr, la somme partielle vaut $somme"  
done  
echo "La somme finale vaut $somme"

1. Modifiez le script précédent pour que la somme soit calculée sur les éléments d'un tableau et non d’une liste.

#!/bin/bash  
#############################################################  
# somme-tableau.sh  
# Calcule différentes sommes arithmétiques  
#   
#   
# corentin.dhaeyere  
# 04 May 2022  
#############################################################  
echo  
echo " Somme des nombres du tableau"  
echo "----------------------------"  
tab=(1 -3 45 67)  
cnt=${#tab[\*]}  
i=0  
#while (( i < cnt ))  
#while test$i -lt $cnt  
while [ $i -lt $cnt ]  
do  
 ((somme+=${tab[i]}))  
 ((i++))  
 echo "Avec ${tab[i-1]}, la somme partielle vaut $somme"  
done  
echo "La somme finale vaut $somme"

1. Ecrivez une nouvelle version du script *somme.sh* dans laquelle les nombres sont introduits en argument du programme. Si l'utilisateur n'indique aucun argument, le script affiche un message d’erreur et retourne un code de sortie valant 2.

#!/bin/bash  
#############################################################  
# somme-entree.sh  
# Calcule différentes sommes arithmétiques  
#   
#   
# corentin.dhaeyere  
# 04 May 2022  
#############################################################  
echo  
echo " Somme des arguments"  
echo " -------------------"  
if [ $# -eq 0 ]; then  
 echo "Pas d'arguments" > /dev/stderrexit 2  
else  
 somme=  
 for nbr in $\*; do  
 ((somme+=nbr))  
 echo "Avec $nbr, la somme partielle vaut $somme"  
 done  
 echo "La somme finale vaut $somme"  
fi

1. Dans cette troisième version, le programme doit lire les nombres au clavier sur *stdin*. La fin des données est indiquée par <Ctrl-D> ou une ligne vide.

#!/bin/bash  
#############################################################  
# somme-lecture.sh  
# Calcule différentes sommes arithmétiques  
#   
#   
# corentin.dhaeyere  
# 04 May 2022  
#############################################################  
echo  
echo " Somme des nombres lus"  
echo " ---------------------"  
echo "Donnez les nombres à lire (un par ligne; Ctrl-D pour terminer)"  
somme=  
while read nbr; do  
 if [ -z "$nbr" ] ; then  
 break  
 fi  
 ((somme+=nbr))  
 echo "Avec $nbr, la somme partielle vaut $somme"  
done  
echo "La somme finale vaut $somme" > /dev/stderr

1. Modifiez le script précédent pour qu’il vérifie que la chaîne rentrée est bien un nombre entier (attention, celui-ci peut être négatif). Si ce n’est pas le cas, le programme lit une nouvelle ligne.

if [[ ! "$nbr" =~ ^-?[0-9]+$ ]] ; then  
 echo Error: not a number  
 continue  
fi  
((somme+=nbr))

1. Reprenez votre script entete.sh de la fiche 15 afin que la description du script lue à l’entrée standard ne soit plus limitée à trois lignes maximum. Pour ce faire, utilisez une boucle de la même manière que dans l’exercice précédent : des lignes sont lues et ajoutées à l’entête du script tant que l’utilisateur n’entre pas <Ctrl-D> (= end-of-file).

#!/bin/bash  
#########################################################################\*/  
# entete.sh  
# genere un entete dans un script, comprenant  
# une description saisie au clavier et rend le script executable  
#########################################################################\*/  
ligne="#############################################################"  
entete='#!'$SHELL  
entete=$entete"\n$ligne"  
entete="$entete\n# $1"  
echo "Donnez la description du script (<Ctrl-D> pour terminer)"  
while read line; do  
 entete="$entete\n#\t$line"  
done  
entete="$entete\n# $USER"  
entete="$entete\n# $(date +'%d %B %Y')"  
entete=$entete"\n$ligne"  
echo -e $entete > $1  
chmod u+x $1

1. Rédigez un script *lsd.sh* qui donne la liste des répertoires du répertoire courant.

#!/bin/bash  
#########################################################################\*/  
# lsd.sh# affiche la liste des répertoires du répertoire courant  
#########################################################################\*/  
# version avec liste  
for f in $(ls); do

if [ -d "$f" ]; then  
 echo "$f"  
 fi  
done

1. Complétez le script précédent de telle sorte qu'il puisse recevoir en argument le nom d'un répertoire.

Par défaut (s’il n'y a pas d'argument), le script traite le répertoire courant.

S’il y a un argument et que ce n'est pas un répertoire, le script affiche un message d'erreur et quitte avec un code valant 1.

Si l'utilisateur introduit plus d'un argument, le script rappelle l'usage

Usage: *nomDuScript* [directory]

et se termine avec le code 2.

Ex : à l'appel

lsd.sh ..

le script affiche la liste des répertoires du répertoire parent.

Ex : à l'appel

lsd.sh data.txt

le script affiche

data.txt is not a directory

et retourne le code 1.

Ex : à l'appel

lsd.sh rep1 rep2

le script affiche

Usage: lsd.sh [directory]

et retourne le code 2.

#!/bin/bash  
#########################################################################\*/  
# lsd.sh  
# affiche la liste des répertoires passés en argument  
#########################################################################\*/  
case $# in  
0) rep="."  
;;  
1) if [ ! -d "$1"]; then  
 echo "$1 n'est pas un repertoire" > /dev/stderr  
 exit 1  
fi  
rep=$1  
;;  
\*) echo "Usage: $0 [rep]" > /dev/stderr  
 exit 2  
esac  
for f in $(ls $rep); do  
 if [ -d "$rep/$f" ]; then  
 echo " $rep/$f"  
 fi  
done

1. Créez un script *smartfind.sh* qui sera appelé avec deux ou trois paramètres. Le premier paramètre désigne un répertoire dans l’arborescence duquel rechercher tous les fichiers qui ont pour extension le second paramètre. S’il n’y a que deux paramètres, il faut juste afficher les chemins (avec les noms) de ces fichiers. S’il y a trois paramètres, il faut, en plus d’afficher les noms des fichiers, lister les lignes de ces fichiers sélectionnées par la regex fournie par le 3ème paramètre.

#!/bin/bash  
if [ $# -eq 2 ]; then  
 find "$1" -name "\*.$2" 2> /dev/null  
elif [ $# -eq 3 ]; then  
 find "$1" -name "\*.$2" 2> /dev/null -exec egrep -H -n --color=always"$3" {} \; | less -R  
else  
 echo Erreur: nombre de parametres incorrect> /dev/stderr  
fi

1. Améliorez *smartfind.sh* afin qu’il numérote les lignes affichées dans les fichiers et utilise une pagination (commande less) pour afficher le résultat.