**453---SYSTEME D'EXPLOITATION - e-learning - Séquence 15**

**Programmation de scripts – 4**

**Légende :**

Explications sur fond blanc

Exercice à accomplir sur fond rouge clair

Une ou plusieurs solutions possibles pour l’exécution de commande sur fond vert clair

Une solution possible pour la rédaction du script sur fond jaune clair

**Quelques commandes :**

IF

NAME

if - Conditionally perform a command (as test-commands has an exit status of zero)

SYNTAX

if test-commands

then consequent-commands

[elif more-test-commands

then more-consequents]

[else alternate-consequents]

fi

WHILE

NAME

while - Execute consequent-commands as long as test-commands has an exit status

of zero

SYNTAX

while test-commands

do consequent-commands

done

FOR

NAME

for - Expand words, and execute commands once for each member in the

resultant list, with name bound to the current member

SYNTAX

for name [in words ...]

do commands

done

for (( expr1 ; expr2 ; expr3 ))

do commands

done

CASE

NAME

case - Conditionally perform a command, case will selectively execute the

command-list corresponding to the first pattern that matches word

SYNTAX

case word in

[ [(] pattern [| pattern]...) command-list ;;]...

[ \*) command-list ;;]

esac

SLEEP(1)

NOM

sleep - Endormir un processus pour une durée déterminée

SYNOPSIS

sleep DURÉE[UNITÉ\_DE\_TEMPS]...

sleep OPTION

DESCRIPTION

Effectuer une pause d'une DURÉE définie. L'UNITÉ\_DE\_TEMPS peut être « s » pour

des secondes (par défaut), « m » pour des minutes, « h » pour des heures ou « d » pour

des jours. Contrairement à la plupart des applications qui requièrent un nombre entier,

la DURÉE peut être ici un nombre décimal arbitraire.

RM(1)

NOM

rm - Effacer des fichiers et des répertoires

SYNOPSIS

rm [OPTION]... FICHIER...

DESCRIPTION

Le programme rm efface chaque fichier listé.

Par défaut, il n'efface pas les répertoires.

TOUCH(1)

NOM

touch - Modifier l'horodatage d'un fichier

SYNOPSIS

touch [OPTION]... FICHIER...

DESCRIPTION

Mettre à jour les dates (dates et heures) d'accès et de dernière modification

de chaque FICHIER selon la date actuelle. Un paramètre FICHIER qui n'existe pas

sera créé vide, à moins que les options -c ou -h ne soient fournies.

EVAL

NAME

eval - Evaluate several commands/arguments

SYNTAX

eval [arguments]

LET

NAME

let - Perform arithmetic on shell variables

SYNTAX

let expression [expression]

Création d’une Variable numérique et affectation à cette Variable d’une VALEUR :

let "Variable=VALEUR"

((Variable=VALEUR))

EXPR(1)

NOM

expr - Évaluer des expressions

SYNOPSIS

expr EXPRESSION

expr OPTION

DESCRIPTION

Afficher la valeur de EXPRESSION sur la sortie standard.

Affectation à une Variable numérique d’une expression arithmétique :

Variable=$(expr $variable1 + $variable2)

Variable=`expr $variable1 + $variable2`

Variable=$(expr $variable1 \\* $variable2) # \ annule la signification du caractère

# spécial \* afin qu’il soit interprété

# comme l’opérateur de multiplication

**LABORATOIRE – Programmation de scripts – 4**

**Dans la distribution Debian GNU/Linux :**

Dans la suite, le nom de l’utilisateur ordinaire sera représenté par le mot « user ».

Connectez-vous dans un terminal

Affichez le manuel de la commande sleep

man sleep

Dans ~, enregistrez sous le nom « fff\_existe » un fichier de script qui vérifie toutes les trois secondes si le fichier fff existe

gedit fff\_existe

#!/bin/bash

# Un script qui vérifie l’existence du fichier fff

while [ ! -e fff ]

do

sleep 3

done

echo

echo "Maintenant, le fichier fff existe"

echo

Exécutez un dump du fichier fff\_existe sur le terminal

cat fff\_existe

Vérifiez que le fichier fff n’existe pas

ls -l fff

Si un fichier fff existe, supprimez-le

rm fff

& Séparateur de commandes, mais la commande qui le précède est exécutée en tâche de fond

Lancez un shell bash qui exécute fff\_existe en tâche de fond

bash fff\_existe &

Avec touch, créez dans ~ un fichier nommé : « fff », et constatez l’effet au niveau de l’exécution du script fff\_existe

touch fff

Dans ~, enregistrez sous le nom « Rebours » un fichier de script qui effectue un compte à rebours, de 10 à 0, toutes les secondes, puis qui affiche : « FIN DU COMPTE A REBOURS »

gedit Rebours

#!/bin/bash

# Un script qui effectue un compte à rebours

echo

for top in 10 9 8 7 6 5 4 3 2 1 0

do

printf "=====> %2d\n" $top

sleep 1

done

echo

echo "FIN DU COMPTE A REBOURS"

echo

Exécutez un dump du fichier Rebours sur le terminal

cat Rebours

Lancez un shell bash qui exécute Rebours

bash Rebours

Exemple de script avec for et une variable numérique :

#!/bin/bash

for (( i = 0 ; i <= 5 ; i++ ))

do

echo "Welcome $i times"

done

Réfléchissez à la tâche accomplie par l’exemple de script précédent

Dans ~, enregistrez sous le nom « Menu » un fichier de script qui exécute un menu permettant à l’utilisateur d’afficher sous quel compte il est connecté, ou l’identification système de son compte, ou la liste des utilisateurs connectés, ou le chemin complet du current working directory, ou le nom d’hôte de la machine locale, ou la date et l’heure système (avec traitement du cas où l’utilisateur effectue un choix non prévu)

gedit Menu

#!/bin/bash

# Un script qui exécute un menu

clear

echo

echo "1 >>> Afficher sous quel compte vous etes connecte"

echo "2 >>> Afficher l’identification systeme de votre compte"

echo "3 >>> Afficher la liste des utilisateurs connectes"

echo "4 >>> Afficher le chemin complet du current working directory"

echo "5 >>> Afficher le nom d’hote de la machine locale"

echo "6 >>> Afficher la date et l’heure systeme"

echo

echo -n "Quel est votre choix ? "

read choix

echo

case "$choix" in

1)

whoami

;;

2)

id

;;

3)

who

;;

4)

pwd

;;

5)

hostname

;;

6)

date

;;

\*)

echo "Votre choix n’est pas prevu. Reexecutez Menu."

;;

esac

Exécutez un dump du fichier Menu sur le terminal

cat Menu

Lancez un shell bash qui exécute Menu

bash Menu

Exemple de script avec case et le premier paramètre du script :

#!/bin/bash

case "$1" in

1) echo 'Vous avez choisi 1'

;;

2) echo 'Vous avez choisi 2'

;;

\*) echo "Vous n'avez choisi ni 1 ni 2"

;;

esac

read

Réfléchissez à la tâche accomplie par l’exemple de script précédent

Essayez d’afficher une page de manuel concernant la commande let

man let

Lancez un navigateur et, dans un moteur de recherche sur Internet, recherchez une page de manuel concernant la commande let

Affichez le manuel de la commande expr

man expr

Création d’une Variable numérique et affectation à cette Variable d’une VALEUR :

let "Variable=VALEUR"

((Variable=VALEUR))

Affectation à une Variable numérique d’une expression arithmétique :

Variable=$(expr $variable1 + $variable2)

Variable=`expr $variable1 + $variable2`

Variable=$(expr $variable1 \\* $variable2) # \ annule la signification du caractère

# spécial \* afin qu’il soit interprété

# comme l’opérateur de multiplication

Exemple de script avec des variables numériques :

#!/bin/bash

read -p 'Valeur v1 ? ' v1

let "v2=10"

((v3=20))

echo "v2 = $v2"

echo "v3 = $v3"

if test $v1 -ge $v2

then echo 'v1 >= v2'

else echo 'v1 < v2'

fi

if [ $v1 -lt $v3 ]

then echo 'v1 < v3'

else echo 'v1 >= v3'

fi

v4=$(expr $v2 + $v3)

echo "v2 + v3 = $v4"

v5=`expr $v1 + $v2`

echo "v1 + v2 = $v5"

v6=$(expr $v1 \\* $v3)

echo "v1 \* v3 = $v6"

Réfléchissez à la tâche accomplie par l’exemple de script précédent

Déconnectez-vous

exit