**453---SYSTEME D'EXPLOITATION - e-learning - Séquence 16**

**Programmation de scripts – 5**

**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

On nomme « sous-programme » un bloc d’instructions, qui peut être appelé ensuite dans un programme. Dans un script, un sous-programme constitue en quelque sorte une nouvelle commande pouvant être exécutée.

En programmation de scripts shell, il y a deux catégories de sous-programmes : les scripts (qui peuvent être activés par d’autres scripts) et les « fonctions ». Mais, l’exécution d’une fonction se fait sans activation d’un nouveau processus shell, au contraire de l’exécution (par défaut) d’un script. Ainsi, les variables d’un script sont visibles et modifiables par ses fonctions, mais pas par les scripts qu’il active.

Syntaxe de la commande function :

function nom\_de\_la\_fonction

{

liste\_de\_commandes

}

nom\_de\_la\_fonction ()

{

liste\_de\_commandes

}

Dans le corps d’une fonction, on peut utiliser les variables : $0 ; $1 ; $2 ; … ; $\* ; $#

**Quelques commandes :**

FUNCTION

NAME

function - Declare a shell function

SYNTAX

function name

{ command-list }

name ()

{ command-list }

DESCRIPTION

Shell functions are a way to group commands for later execution using a single name

for the group. They are executed just like a regular command. When the name of a

shell function is used as a simple command name, the list of commands associated

with that function name will be executed.

La commande « return » permet de sortir d’une fonction et de générer un code retour.

RETURN

NAME

return - Cause a shell function to exit with the return value n

SYNTAX

return [n]

La commande « typeset » permet de déclarer une variable locale dans une fonction.

TYPESET

NAME

typeset, whence - shell built-in functions to set/get attributes and values for shell

variables and functions

SYNOPSIS

typeset [ \_ HLRZfilrtux [n]] [ name [ = value]]...

whence [-pv] name...

DESCRIPTION

typeset sets attributes and values for shell variables and functions. When typeset

is invoked inside a function, a new instance of the variables name is created.

The variables value and type are restored when the function completes.

FIND(1)

NOM

find - Rechercher des fichiers dans une hiérarchie de répertoires

SYNOPSIS

find [-H] [-L] [-P] [-D option-debogage] [-Oniveau] [chemin...] [expression]

DESCRIPTION

GNU find parcourt les arborescences des répertoires de chacun des chemins

mentionnés, en évaluant les expressions fournies pour chaque fichier

rencontré. L'évaluation de l'expression se fait de gauche à droite, en suivant

les règles de priorité décrites dans la section OPÉRATEURS, jusqu'à l'obtention

du résultat (par exemple la partie gauche est fausse pour un opérateur et, vraie pour

un opérateur ou), puis find passe au nom de fichier suivant.

UNSET

NAME

unset - unset values and attributes of variables and functions

SYNOPSIS

unset [-fv] name ...

DESCRIPTION

Each variable or function specified by name shall be unset.

If -v is specified, name refers to a variable name and the shell shall unset it and remove it

from the environment. Read-only variables cannot be unset.

If -f is specified, name refers to a function and the shell shall unset the function

definition.

If neither -f nor -v is specified, name refers to a variable; if a variable by that name

does not exist, it is unspecified whether a function by that name, if any, shall be unset.

La commande « break » permet de sortir d’une boucle, c’est-à-dire de mettre fin à un traitement en boucle programmé au moyen d’une instruction de contrôle telle que « for ».

BREAK

NAME

break - exit from a for, while, until, or select loop

La commande « continue » permet de passer à l’itération suivante dans une boucle.

CONTINUE

NAME

continue - resume the next iteration of an enclosing for, while, until, or select loop

La commande « exit » permet de terminer l’exécution d’un script et de renvoyer un code retour.

EXIT(3)

NAME

exit - cause normal process termination

**LABORATOIRE – Programmation de scripts – 5**

**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

Dans ~, enregistrez sous le nom « Bienvenue » un fichier de script qui affiche en boucle toutes les secondes le message : « BIENVENUE DANS LE SYSTEME LINUX ! », tant que la durée de l’exécution du script reste inférieure à 10 secondes, puis qui affiche un message d’adieu à l’utilisateur

gedit Bienvenue

#!/bin/bash

# Un script qui affiche en boucle

echo

t=0

while ((t<10))

do

echo "BIENVENUE DANS LE SYSTEME LINUX !"

sleep 1

let t=t+1

done

echo

echo "Le script a termine son execution. Au revoir."

echo

Exécutez un dump du fichier Bienvenue sur le terminal

cat Bienvenue

Lancez un shell bash qui exécute Bienvenue

bash Bienvenue

En programmation de scripts shell, il y a deux catégories de sous-programmes : les scripts (qui peuvent être activés par d’autres scripts) et les fonctions. Mais, l’exécution d’une fonction se fait sans activation d’un nouveau processus shell, au contraire de l’exécution (par défaut) d’un script. Ainsi, les variables d’un script sont visibles et modifiables par ses fonctions, mais pas par les scripts qu’il active.

Syntaxe de la commande function :

function nom\_de\_la\_fonction

{

liste\_de\_commandes

}

nom\_de\_la\_fonction ()

{

liste\_de\_commandes

}

Dans le corps d’une fonction, on peut utiliser les variables : $0 ; $1 ; $2 ; … ; $\* ; $#

Essayez d’afficher des pages de manuel concernant les commandes function, return et typeset

man function

man return

man typeset

Lancez un navigateur et, dans un moteur de recherche sur Internet, recherchez des pages de manuel concernant les commandes function, return et typeset

Affichez le manuel de la commande find

man find

Créez une fonction nommée : « trouve », qui recherche à partir de la racine et affiche le chemin absolu d’un fichier, qu’on lui passe en paramètre à l’exécution

trouve () { find / -name "$1" -print 2> /dev/null ; }

Avec trouve, affichez le chemin absolu de l’exécutable ls

trouve ls

Avec trouve, affichez le chemin absolu de l’exécutable vi

trouve vi

Détruisez la fonction trouve

unset -f trouve

Dans ~, enregistrez sous le nom « Fonction\_affiche » un fichier de script qui, en utilisant une fonction de présentation d’affichage dans des lignes successives, affiche le message :

\*\*\*\*\* Bonjour,

\*\*\*\*\* maintenant

\*\*\*\*\* nous allons afficher

\*\*\*\*\* la date et l’heure système :

\*\*\*\*\* date\_système, heure\_système

\*\*\*\*\* Au revoir,

\*\*\*\*\* cher utilisateur.

date\_système, heure\_système étant obtenues par interprétation de la commande adéquate

gedit Fonction\_affiche

#!/bin/bash

# Un script qui affiche

# en utilisant une fonction de présentation d’affichage dans des lignes successives

# ----- Fonction de présentation d’affichage

presente ()

{

for input

do

echo "\*\*\*\*\* $input"

done

}

# ----- Programme principal : affichage dans des lignes successives

echo

presente "Bonjour," "maintenant" "nous allons afficher" "la date et l’heure systeme :" \

"`date`" "Au revoir," "cher utilisateur."

echo

Exécutez un dump du fichier Fonction\_affiche sur le terminal

cat Fonction\_affiche

Lancez un shell bash qui exécute Fonction\_affiche

bash Fonction\_affiche

Essayez d’afficher des pages de manuel concernant les commandes break et continue

man break

man continue

Lancez un navigateur et, dans un moteur de recherche sur Internet, recherchez des pages de manuel concernant les commandes break et continue

Dans ~, enregistrez sous le nom « Fonction\_menu » un fichier de script qui exécute en boucle, en utilisant une fonction, un menu permettant à l’utilisateur, soit 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, soit de terminer l’exécution du script, ce qui déclenche l’affichage du message : « AU REVOIR ! »

gedit Fonction\_menu

#!/bin/bash

# Un script qui exécute en boucle un menu en utilisant une fonction

# ----- Déclaration de variable globale

reponse="0"

# ----- Fonction d’exécution du menu

menu ()

{

typeset choix # Déclaration de variable locale

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 "7 >>> Terminer l’execution du script"

echo

echo -n "Quel est votre choix ? "

read choix

echo

reponse="$choix"

}

# ----- Programme principal

menu

while [ "$reponse" != "7" ]

do

case "$reponse" in

"1")

whoami ; read

;;

"2")

id ; read

;;

"3")

who ; read

;;

"4")

pwd ; read

;;

"5")

hostname ; read

;;

"6")

date ; read

;;

\*)

echo "Votre choix n’est pas prevu. Choisissez de 1 a 7." ; read

;;

esac

menu

done

if [ "$reponse" = "7" ]

then

echo "AU REVOIR !"

sleep 1

exit

fi

Exécutez un dump du fichier Fonction\_menu sur le terminal

cat Fonction\_menu

Lancez un shell bash qui exécute Fonction\_menu

bash Fonction\_menu

Déconnectez-vous

exit