**453---SYSTEME D'EXPLOITATION - e-learning - Séquence 14**

**Programmation de scripts – 3**

**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

En programmation de scripts shell, certains paramètres sont utilisables :

$0 Le nom du script (s’il se trouve dans le répertoire de travail)

ou le chemin d’accès absolu du script

$1 ; $2 ; … Le premier paramètre du script ; le deuxième paramètre du script ; …

$# Le nombre de paramètres du script

$\* ou $@ L’ensemble des paramètres du script

Une commande ou un script, dont l’exécution se termine, émet toujours un « code numérique de retour », appelé « exit status », reçu par le shell qui a déclenché l’exécution (valeurs possibles pour ce code : de 0 à 255 – la valeur 0 représente une exécution réussie, ou une valeur logique Vrai, tandis qu’une valeur différente de 0 représente une erreur d’exécution).

Ce code est utilisable dans la programmation de scripts shell ; en effet, « ? » est le nom de la variable prédéfinie du shell qui est automatiquement initialisée avec le code de retour de la dernière exécution.

**Quelques commandes :**

La commande « read » permet de créer une variable et lui affecter une valeur lue à partir de l’entrée standard (c’est-à-dire le clavier).

READ(2)

NAME

read - read from a file descriptor

Les trois premiers « file descriptors » (variable système fd), dans le système d’exploitation, [fd = 0, 1, 2] concernent les trois fichiers spéciaux de périphérique caractère suivants :

entrée standard (STDIN) [fd = 0] [lecture d’information à partir du clavier]

sortie standard (STDOUT) [fd = 1] [écriture d’information sur le terminal]

sortie d’erreur standard (STDERR) [fd = 2] [écriture de message d’erreur sur le terminal]

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

TEST(1)

NOM

test - Vérifier le type d'un fichier, et comparer des valeurs

SYNOPSIS

test EXPRESSION

test

[ EXPRESSION ]

[ ]

[ OPTION ]

DESCRIPTION

Quitter avec un code de retour déterminé par EXPRESSION.

Une expression omise est fausse par défaut. Sinon, EXPRESSION est évaluée

comme vraie ou fausse, et détermine la valeur de retour.

Syntaxe de la commande test (utilisable avec if ou while) :

if test liste\_d\_options ; then … ; elif … ; then … ; else … ; fi

if [ liste\_d\_options ] ; then … ; elif … ; then … ; else … ; fi

Quelques options :

-e fichier Vrai si fichier existe

-f fichier Vrai si fichier est un Fichier ordinaire

-d fichier Vrai si fichier est un Répertoire

chaine1 = chaine2 Vrai si chaine1 et chaine2 sont identiques

chaine1 != chaine2 Vrai si chaine1 et chaine2 sont différentes

Nombre1 -eq Nombre2 Vrai si Nombre1 = Nombre2

Nombre1 -ne Nombre2 Vrai si Nombre1 ≠ Nombre2

Nombre1 -lt Nombre2 Vrai si Nombre1 < Nombre2

Nombre1 -le Nombre2 Vrai si Nombre1 ≤ Nombre2

Nombre1 -gt Nombre2 Vrai si Nombre1 > Nombre2

Nombre1 -ge Nombre2 Vrai si Nombre1 ≥ Nombre2

Quelques opérateurs booléens :

! option Vrai si option est fausse

option1 -a option2 Vrai si option1 et option2 sont vraies

option1 -o option2 Vrai si option1 ou option2 est vraie

Syntaxe des opérateurs && et || :

Commande1 && Commande2 if Commande1 ; then Commande2 ; fi

Commande1 || Commande2 if Commande1 ; then : ; else Commande2 ; fi

: est une commande qui renvoie toujours la valeur logique Vrai.

**Un fichier :**

« /dev/null » est un fichier spécial utilisé comme une poubelle : on y redirige les messages d’erreurs que l’on veut ignorer.

**LABORATOIRE – Programmation de scripts – 3**

**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

Paramètres d’un script :

$0 Le nom du script (s’il se trouve dans le répertoire de travail)

ou le chemin d’accès absolu du script

$1 ; $2 ; … Le premier paramètre du script ; le deuxième paramètre du script ; …

$# Le nombre de paramètres du script

$\* ou $@ L’ensemble des paramètres du script

Dans ~, enregistrez sous le nom « scriptparam » un fichier de script qui affiche le nom (ou le chemin) du script, puis le premier paramètre, puis le deuxième paramètre, puis tous les paramètres et enfin le nombre de paramètres

gedit scriptparam

#!/bin/bash

# Un script qui affiche des informations sur les paramètres du script

echo

echo "Voici le nom (ou le chemin) du script : $0"

echo

echo "Voici le premier parametre : $1"

echo "Voici le deuxieme parametre : $2"

echo

echo "Voici tous les parametres : $\*"

echo "Voici le nombre de parametres : $#"

echo

Exécutez un dump du fichier scriptparam sur le terminal

cat scriptparam

Lancez un shell bash qui exécute scriptparam en lui passant cinq paramètres

bash scriptparam A B C D E

Essayez d’afficher des pages de manuel concernant les commandes if, while, for et case

man if

man while

man for

man case

Lancez un navigateur et, dans un moteur de recherche sur Internet, recherchez des pages de manuel concernant les commandes if, while, for et case

Une commande UNIX, un script, dont l’exécution se termine, émet un code numérique de retour, appelé « exit status », reçu par le shell qui a déclenché l’exécution (valeurs : de 0 à 255 – valeur 0 : exécution réussie ou valeur logique Vrai).

? est la variable prédéfinie du shell qui est automatiquement initialisée avec le code de retour de la dernière exécution.

/dev/null est un fichier spécial utilisé comme une poubelle : on y redirige les messages d’erreurs que l’on veut ignorer.

Dans ~, enregistrez sous le nom « scriptcreerep » un fichier de script qui demande à l’utilisateur de saisir un nom de répertoire à créer, puis qui utilise une alternative pour soit le créer et afficher : le code de retour, puis « Création réussie », soit afficher : le code de retour, puis « Echec de l’opération », dans le cas où le répertoire existe déjà (sans affichage d’un message d’erreur système)

gedit scriptcreerep

#!/bin/bash

# Un script qui crée un répertoire s’il le peut

echo

echo -n "Quel est le nom du repertoire a creer ? "

read nom\_rep

echo

if mkdir $nom\_rep 2> /dev/null # Création du répertoire, s’il n’existe pas encore

# Mais s’il existe déjà, alors le message d’erreur

# ne s’affiche pas, car on a programmé ici

# la redirection de la sortie d’erreur standard

# [fd = 2] vers /dev/null

then

echo $?

echo "Creation reussie"

else

echo $?

echo "Echec de l’operation"

fi

echo

Exécutez un dump du fichier scriptcreerep sur le terminal

cat scriptcreerep

Vérifiez qu’aucun répertoire « repessai » n’existe

ls -l repessai

Si un répertoire « repessai » existe, supprimez-le (récursivement et de manière forcée)

rm -rf repessai

Lancez un shell bash qui exécute scriptcreerep, pour créer le répertoire « repessai »

bash scriptcreerep

repessai

Recommencez l’exécution, le répertoire « repessai » étant déjà créé

Syntaxe des opérateurs && et || :

Commande1 && Commande2 if Commande1 ; then Commande2 ; fi

Commande1 || Commande2 if Commande1 ; then : ; else Commande2 ; fi

: est une commande qui renvoie toujours la valeur logique Vrai.

Syntaxe de la commande test (utilisable avec if ou while) :

if test liste\_d\_options ; then … ; elif … ; then … ; else … ; fi

if [ liste\_d\_options ] ; then … ; elif … ; then … ; else … ; fi

Quelques options :

-e fichier Vrai si fichier existe

-f fichier Vrai si fichier est un Fichier ordinaire

-d fichier Vrai si fichier est un Répertoire

chaine1 = chaine2 Vrai si chaine1 et chaine2 sont identiques

chaine1 != chaine2 Vrai si chaine1 et chaine2 sont différentes

Nombre1 -eq Nombre2 Vrai si Nombre1 = Nombre2

Nombre1 -ne Nombre2 Vrai si Nombre1 ≠ Nombre2

Nombre1 -lt Nombre2 Vrai si Nombre1 < Nombre2

Nombre1 -le Nombre2 Vrai si Nombre1 ≤ Nombre2

Nombre1 -gt Nombre2 Vrai si Nombre1 > Nombre2

Nombre1 -ge Nombre2 Vrai si Nombre1 ≥ Nombre2

Quelques opérateurs booléens :

! option Vrai si option est fausse

option1 -a option2 Vrai si option1 et option2 sont vraies

option1 -o option2 Vrai si option1 ou option2 est vraie

Affichez le manuel de la commande test

man test

Dans ~, enregistrez sous le nom « scriptdialogue » un fichier de script qui demande à l’utilisateur s’il (si elle) est étudiant(e) en informatique, puis qui répond : « Parfait, étudiez l’administration des systèmes UNIX » si la réponse est « oui », « Très bien, utilisez les logiciels libres, moins onéreux » si la réponse est « non », et « Si vous n’en avez pas, équipez-vous d’un ordinateur » dans le cas d’une autre réponse

gedit scriptdialogue

#!/bin/bash

# Un script qui dialogue

echo

echo -n "Hello ! Etes-vous etudiant(e) en informatique ? "

read reponse

echo

if [ "$reponse" = "oui" ]

then

echo "Parfait, etudiez l’administration des systemes UNIX"

elif [ "$reponse" = "non" ]

then

echo "Tres bien, utilisez les logiciels libres, moins onereux"

else

echo "Si vous n’en avez pas, equipez-vous d’un ordinateur"

fi

echo

Exécutez un dump du fichier scriptdialogue sur le terminal

cat scriptdialogue

Lancez un shell bash qui exécute scriptdialogue

bash scriptdialogue

oui

bash scriptdialogue

non

bash scriptdialogue

xxx

Dans ~, enregistrez sous le nom « scriptdialogue2 » un fichier de script qui contient une version améliorée du script précédent, acceptant les réponses : « oui », « o », « yes », « y », d’une part, « non », « no », « n », d’autre part, ou toute autre réponse

gedit scriptdialogue2

#!/bin/bash

# Un second script qui dialogue

echo

echo -n "Hello ! Etes-vous etudiant(e) en informatique ? "

read reponse

echo

if [ "$reponse" = "oui" -o "$reponse" = "o" -o "$reponse" = "yes" -o "$reponse" = "y" ]

then

echo "Parfait, etudiez l’administration des systemes UNIX"

elif [ "$reponse" = "non" -o "$reponse" = "no" -o "$reponse" = "n" ]

then

echo "Tres bien, utilisez les logiciels libres, moins onereux"

else

echo "Si vous n’en avez pas, equipez-vous d’un ordinateur"

fi

echo

Exécutez un dump du fichier scriptdialogue2 sur le terminal

cat scriptdialogue2

Lancez un shell bash qui exécute scriptdialogue2

bash scriptdialogue2

oui

bash scriptdialogue2

o

bash scriptdialogue2

yes

bash scriptdialogue2

y

bash scriptdialogue2

non

bash scriptdialogue2

no

bash scriptdialogue2

n

bash scriptdialogue2

yyy

Déconnectez-vous

exit