# Shell script

**Remarques :**

**1)** On peut utiliser toutes les commandes fournies dans le système dans le code (*ls, rm*, etc.).

**2)** Quand on fait du scripting, on met le chemin absolu et non le chemin relatif car le script peut se lancer à partir de différents endroits. Ex : /home/debian/TP/fichier.txt et non fichier.txt

## Vérifier un script

• <https://www.shellcheck.net/>

## Créer et exécuter un script

• **Créer un script :** nano fichier.sh

• **Exécuter un script :** bash fichier.sh argument1 argument2

OU : chmod u+x fichier.sh

./fichier.sh argument1 argument2

## Rédiger le script (les bases)

• **Shebang qui permet au shell de savoir le programme qui doit interpréter le script :** #! /bin/bash

• **Faire un commentaire :** # commentaire

• **Afficher un message dans la console :** echo "message"

## Arguments

• **Utiliser un argument dans le script :** $numéro\_argument. Ex : argument1 -> $1

• **Récupérer le nom du script :** $0

• **Récupérer la chaine complète des arguments :** $\*

• **Récupérer la liste des arguments tels qu’ils ont été fournis en ligne de commande :** $@

• **Récupérer le nombre d’arguments :** $#

• **Récupérer les options en ligne de commande activées pour le script en cours d’exécution :** $-

• **Récupérer le code de retour de la dernière commande exécutée** (retourne 0 si elle s’est exécutée avec succès, sinon retourne 1)**:** $?

• **Récupérer le PID de processus du script en cours d’exécution :** $$

• **Récupérer le PID de processus en arrière-plan lancé dans le script :** $!

• **Dernier argument de la commande précédente exécutée :** $\_

## Variables

• **Déclarer une variable :** variable="valeur"

• **Afficher la valeur d’une variable :** echo "${variable}" # valeur

'${variable}' # $variable

La commande est exécutée et le résultat est affiché : "`commande`"

• **Permettre l’interprétation des caractères d’échappement spéciaux :** echo -e "texte\n"

• **Demander à l’utilisateur de saisir une valeur et l’enregistrer dans une variable :** read -p "demande" variable

• **Vérifier si une variable est définie** (renvoie 1 si elle est définie, sinon 0)**:** ${variable:+1}

• **Connaitre la longueur d’une variable :** ${#variable}

• **Récupérer une sous-chaine d’une variable :** ${variable:position:longueur}

• **Supprimer le plus petit préfixe de la variable qui correspond au pattern :** ${variable#pattern}

• **Supprimer le plus grand préfixe de la variable qui correspond au pattern :** ${variable##pattern}

• **Supprimer le plus petit suffixe de la variable qui correspond au pattern :** ${variable%pattern}

• **Supprimer le plus grand suffixe de la variable qui correspond au pattern :** ${variable%%pattern}

Variables locales

Variables d’environnement :

- définies dans le fichier ~/.bashrc

- on utilise la commande « env » pour les afficher

• Créer une variable d’environnement : export variable="valeur"

• Créer une variable d’environnement temporaire : set variable="valeur"

• Détruire une variable d’environnement : unset variable

## Tableaux

• **Créer un tableau indexé :**

declare -a tableau

tableau=("valeur1" "valeur2" "valeur3")

• **Créer un tableau associatif :**

declare -A tableau

tableau=([un]="valeur1" [deux]="valeur2" [trois]="valeur3")

• **Afficher un élément d’un tableau :** echo ${tableau[1]}, echo ${tableau[deux]}

• **Afficher tous les éléments d’un tableau :** echo ${tableau[\*]}

• **Afficher le nombre d’éléments contenus dans le tableau :** echo ${#[tableau][\*]}

## Conditions

• **if :**

if [[ condition ]]

then

instructions

elif [[ condition ]]

then

instructions

else

instructions

fi

• **Conditions possibles :**

**— -eq =** égal

— **-ne =** non-égal

— **-lt =** plus petit que

— **-le =** plus petit ou égal

— **-gt =** plus grand que

— **-ge =** plus grand ou égal

— **-z =** variable vide

— **-n** **=** variable non vide

Ex : $variable1 -eq $variable2

• **Opérateurs arithmétiques :** if (( variable1 opérateur\_arithmétique variable2 ))

• **Opérateurs logiques :**

— **ET :** if [[ condition1 ]] && [[ condition2 ]]

— **OU :** if [[ condition1 ]] || [[ condition2 ]]

## Boucles

• **Boucle for :**

for variable in $(seq 0 10) **OU :** for ((variable=0; variable<11 ; variable++))

do **OU :** for variable in ${tableau[\*]}

echo "$variable" **OU :** for variable in 1 2 3 4 5

instructions **OU :** for variable in ${seq 1 50}

done

• **Boucle while :**

$variable = 0

while [[ condition ]]

do

instructions

((variable++))

done

• **Boucle infinie :**

while [[ true ]]

do

instructions

done

• **Arrêter une boucle :** break

Ex : for i in $(seq 0 100)

do

if [[ $i -lt 5 ]]

then

echo $i

else

break

fi

done

## Switch case

• **Case :**

case $choix in

valeur1) instruction;;

valeur2) instruction;;

\*) instruction;;

esac

## Opérations arithmétiques

• variable1=1

variable2=1

résultat=$(($variable1 + $variable2))

• résultat=$((1+3))

## Code retour

• **Retourner le code de retour de la dernière commande exécutée** (pas dans le programme) (0 = tout s’est bien déroulé) **:** echo $?

• **Arrêter le programme :** exit

## Fonctions

function nom\_fonction() {

instructions

}

## Autre

• **Générer un nombre aléatoire entre 1 et 50 :** nombre=$(($RANDOM % 50 + 1))

• **Récupérer la date actuelle :** date\_actuelle=$(date +%d-%m-%Y)

• **Vérifier si un répertoire existe :** if [ -d "répertoire"]

• **Vérifier si un répertoire n’existe pas :** if [ ! -d "répertoire"]