esprit
Ecole Supérieure Privée d'Ingénierie et de Technologies

Examen

Semestre : 1 ☐ 2 ■

Module : **Système et scripting**
Enseignant(s) : **UP Système**
Classe(s) 2A2-->2A39
Documents autorisés : OUI ☐ NON ■ Nombre de pages : 7
Date : **29/05/2023** Heure: 11H00 Durée : 1h30

N° Carte : ……..... Classe : ……..
Nom et Prénom : ……

Exercice 1 : (8pts)

Choisir la bonne réponse (QCM)

1. Comment est-il possible de faire référence à une variable ***mavar*** sous shell?
A. (mavar)
B. Mavar
C. ^mavar
D. $mavar

2. On souhaite lancer un script nommé "calcul.sh" avec les paramètres 4 et 12. Quelle est la bonne syntaxe à utiliser ?
A. ./calcul.sh $1=4 $2=12
B. ./calcul.sh --params 4 12
C. ./calcul.sh 4 12
D. PARAMS=4,12 ./calcul.sh

3. Pour ajouter une ligne supplémentaire dans un fichier déjà existant, on utilise la commande :
A. echo "text" > file
B. cat "text" > file
C. echo "text" >> file
D. "text" >> file

4. Comment afficher la longueur d'un tableau ?
A. echo ${#tableau.length()}
B. echo ${#tableau[*]}
C. echo length(tableau)
D. echo count(tableau)

5. Comment vérifier si une variable est vide ?
A. if [variable == ""]
B. if [[variable = ""]]
C. if [-z variable]
D. if [! variable]

6. Quelle instruction permet de vérifier si un fichier existe ?
A. if [-e fichier]
B. if [[-d fichier]]
C. if (-e fichier)
D. if { -f fichier }

7. Pour saisir une valeur au clavier et l'affecter à une variable en Shell, on utilise la syntaxe :
A. get input
B. user_input = read
C. input == read
D. read input

8. Que contient la variable ***$?*** en Shell?
A. Le nom du script
B. Le nombre de paramètres
C. La liste de tous les paramètres
D. Le code de retour de la dernière ligne de commande

NE RIEN ECRIRE ICI

9. La commande « ***grep '^r' /etc/passwd*** » :

A. Affiche toutes les lignes du fichier /etc/passwd
B. N'affiche que les lignes du fichier /etc/passwd qui commencent par la lettre r
C. N'affiche que les lignes du fichier /etc/passwd qui contiennent la lettre r
D. N'affiche que les lignes du fichier /etc/passwd qui se terminent par la lettre r

10. Quel est le résultat de la commande suivante :

[[10 -lt 5 || 20 -gt 15]] && echo "Vrai" || echo "Faux"

A. Vrai
B. Faux
C. L'opérateur || est invalide dans cette syntaxe
D. La commande entraînera une erreur de syntaxe.

11. Pour afficher le nombre d'arguments passés à un script, on exécute la commande :

A. echo #
B. echo $#
C. echo ${#}
D. echo $$#

12. Quel est le résultat d'exécution de la commande suivante sachant que stats.txt est un fichier contenant 500 lignes et chacune des lignes est formée par 10 colonnes?

head -n 400 stats.txt | tail -n 10 | cut -f 2-5,8 >> new_stats.txt

A. Si new_stats.txt existe, les colonnes 2,3,4,5 et 8 des lignes 391 à 400 du fichier stats.txt sont copiées à la fin de new_stats.txt
B. Si new_stats.txt n'existe pas, aucune action n'est exécutée
C. Si new_stats.txt n'existe pas, il est créé, et les 400 premières lignes du fichier stats.txt y sont copiées
D. Les colonnes 2,5 et 8 des lignes 391 à 400 du fichier stats.txt sont copiées à la fin du fichier new_stats.txt

13. Que fait le script ci-dessous ?

14. Le lancement du script **myscript.sh** ci-dessous par la commande « *bash myscript A B C D E F G H I J K L* » affiche:

```
#!/bin/sh

var=Ali
w=`who | grep $var`
if [ -z "$w" ]; then
 echo $var
fi
```

A. Tester si la taille du « var » est nulle puis l'affiche sur l'écran
B. Affiche le message « Ali » s'il est connecté à la machine
C. Tester si la taille du « var » est non nulle puis l'affiche sur l'écran
D. Affiche le message « Ali » s'il n'est pas connecté à la machine

```
#!/bin/bash
if [ $# -ge 10 ]
then
echo ${10}
fi
```

A. J
B. A0
C. JA
D. A

15. Soit le script ci-dessous, laquelle des propositions suivantes est correcte ?

```
#!/bin/bash

for V in $(seq 0 $2)
do
  echo $V "x" $1 "=" $(expr $V "*" $1 )
done
```

A. Le script permet de sommer $1 avec les chiffres inférieurs ou égal $2
B. Le script affiche la multiplication de $2 par tous les chiffres inférieurs ou égaux à $2
C. Le script permet de multiplier $1 par tous les chiffres compris entre 0 et $2
D. Le script affiche le message « $V x $1 = $V * $1 » $2 fois

16. Parmi les options suivantes, quelle est la première option du menu déroulant généré par ce script?

```
#!/bin/bash
options=("Option 1" "Option 2" "Option 3" "Option 4" "Option 5")
correct_answer="Option 4"
echo "Choose the correct answer:"
select answer in "${options[@]}"; do
    case $answer in
        "$correct_answer")
            echo "Congratulations!"
            break
            ;;
        *)
            echo "Sorry, Try again."
            ;;
    esac
done
```

A. Option 1
B. Option 2
C. Option 3
D. Option 4

Exercice 2 : (6 pts)

Le fichier /etc/group contient la liste de tous les groupes utilisateurs créés sur la machine. Il est organisé sous la forme de quatre champs séparés par « : » de la manière suivante :

Nom du groupe : Indication sur le mot de passe : Identifiant du groupe(GID) : Membres de ce groupe

La figure suivante représente un extrait de l'affichage du contenu de ce fichier :

```
esprit:x:1000:
sambashare:x:135:esprit
fwupd-refresh:x:136:
user1:x:2500:
group1:x:2501:user1
```

On se propose d'extraire des informations à partir du fichier **/etc/group**. Pour cela, on vous demande d'écrire un script shell nommé « **script1.sh** » qui permet de :

1. Tester la présence d'un seul argument, sinon afficher le message « Le script nécessite un argument ». **(1.5pt)**

2. Compter le nombre de groupes dans ce fichier. **(1pt)**
3. Vérifier l'existence du groupe passé en argument. Si le groupe existe :
   - ✓ Afficher l'identifiant de ce groupe « GID ». **(1.75pt)**
   - ✓ Afficher les membres de ce groupe, sinon indiquer que le groupe ne contient pas d'utilisateurs. **(1.75pt)**

Ci-dessous un exemple d'exécution de ce script:

```
esprit@esprit-virtual-machine:~$ ./script1.sh
Le script nécessite un argument
esprit@esprit-virtual-machine:~$ ./script1.sh user1
Le nombre de groupes est : 78
Le GID de user1 est : 2500
Aucun membre n'est affecté à user1
esprit@esprit-virtual-machine:~$ ./script1.sh group1
Le nombre de groupes est : 78
Le GID de group1 est : 2501
Les membres du groupe group1 sont : user1
```

script1.sh

```
#! /bin/bash
#question1
if [ $# -ne 1 ];then
echo  Le script n\écessite un argument
exit 1
fi

#question2
echo Le nombre de groupes est : $(cat /etc/group|wc -l)

#question3 partie 1
if [ `grep ^$1 /etc/group|wc -l` -eq 0 ];then
echo le groupe $1 n\'existe pas
else
echo  Le GID de $1 est : $(grep "^$1" /etc/group| cut -d: -f3)
fi
#question3 partie 2
if [ -z `grep ^$1 /etc/group| cut -d: -f4`  ];then
echo  Aucun membre n\'est affecté à $1
else
echo Les membres du groupe $1 sont : $(grep "^$1" /etc/group| cut -d: -f4)
fi……
……
…
……
……
……
……
……
……
```

N° Carte : ……..... Nom et Prénom : ……	Classe : ……..

Exercice 3 : (6 pts)

On se propose d'écrire un script nommé « **GererFichier.sh** » qui effectue des opérations sur un fichier donné en argument et à travers des options tapées à la ligne de commande. Les options sont les suivantes:

-c : Copier le fichier dans un répertoire /tmp/backup

-s : Supprimer le fichier avec demande d'autorisation

-d : Déplacer le fichier exécutable dans un répertoire Trash créé dans le répertoire /var

-p : Afficher les permissions des fichiers

-h : Afficher le help du script en affichant le contenu du fichier /var/help_GererFichier.txt

1. Ecrire la fonction **affiche_usage** qui affiche dans la sortie standard l'usage du script comme suit **(1pt)**:

   USAGE : GererFichier : [-h] [-c] [-d] [-s] [-p] Fichier

```
#! /bin/bash
#afficher_usage pour afficher l'usage du script

affiche_usage() {
echo ' USAGE: GererFichier.sh : [-h] [-c] [-d] [-s] [-e] Fichier(s)..." '
}
```

NE RIEN ECRIRE ICI

2. Compléter la création des fonctions appelées pour chaque option tapée. **(2pt)**

```
#fn_tester pour tester l'existence des fichiers
fn_tester() {
if test ! -e $1
then
echo "le fichier est inexistant"
return 1
else
return 0
fi

}
#fn_copier pour copier les fichiers
fn_copier() {
if fn_tester $1 # si le fichier est existant
then
cp $1 /tmp/backup
fi
}
#fn_deplacer pour déplacer les fichiers
fn_deplacer() {
if fn_tester $1 # si le fichier est existant
then
mv $1 /var/trash
fi
}
#fn_supprimer pour déplacer les fichiers
fn_supprimer() {
if fn_tester $1 # si le fichier est existant
then
rm -i $1
fi
}

#fn_afficher_perm pour déplacer les fichiers
fn_afficher_perm() {
if fn_tester $1
then
ls -l $1
fi
}
```

3. Le script principal teste la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue. Compléter les champs manquants. **(2pt)**

```
#script principal
if [ $# -eq 0 ] || ! `echo $1 | grep -q "-"`
then
affiche_usage # juste pour afficher le syntaxe du script en cas de non utilisation des options et d'arguments
else
while
getopts "c:d:p:s:h" option # chaque option est suivi d'argument obligatoire sauf le h (pour le help)

do
case $option in
c) fn_copier $OPTARG;; #La variable réservée "$OPTARG" contient l'argument associé à l'option.
d) fn_deplacer $OPTARG;;
s) fn_supprimer $OPTARG;;
p) fn_afficher_perm $OPTARG;;
h) cat /var/help_GererFichier.txt;;
esac
done
fi
```

4. Donner la commande qui permet de déplacer et afficher les permissions du fichier **script1.sh** de l'exercice 2. **(0.5pt)**

…….../GererFichier –dp script1.sh

5. Quel est le résultat de l'exécution suivante sachant que **toto.txt** n'existe pas? **(0.5pt)**

./GererFichier -pc toto.txt

…Le fichier est inexistant……

Bon courage