TP Scripting SHELL

Votre entreprise a besoin d'une solution sécurisée permettant aux personnels de la supervision d'intervenir dans un cadre maîtrisé sur les serveurs.

1. Etude du besoin

Votre responsable vous demande de développer un outil destiné aux superviseurs. Ils pourront effectuer quelques actions d'administration ainsi que les premiers diagnostics avant de faire intervenir le personnel d'astreinte.

Le personnel doit pouvoir se connecter aux serveurs via un compte générique : **supervision**.

Lorsque l'utilisateur se connecte, un menu est proposé, lui permettant :

- De gérer les utilisateurs :
  - afficher le nombre d'utilisateurs du serveur et les afficher sous formes de 2 listes :
    - les utilisateurs systèmes,
    - les utilisateurs standards ;
  - afficher les groupes du serveur et les afficher sous forme de 2 listes :
    - les groupes systèmes,
    - les groupes standards ;
  - créer un groupe : le superviseur devra fournir le GID ;
  - créer un utilisateur : le superviseur devra fournir l'UID, le GID, etc. ;
  - changer le mot de passe d'un utilisateur ; l'utilisateur sera forcé de changer son mot de passe lors de sa prochaine connexion.
- De gérer les services :
  - relancer le serveur apache ;
  - relancer le serveur postfix.
- De tester le réseau :

    - Afficher les informations du réseau (Adresse IP, masque, passerelle, serveurs DNS) ;
    - Tester le réseau (localhost, ip, passerelle, serveur distant, résolution DNS).
- Actions diverses :
    - redémarrer le serveur ;
    - quitter le script (l'utilisateur est déconnecté).

Les actions du superviseur devront être renseignées dans les journaux systèmes.

2. Consignes

- Les scripts sont stockés dans /opt/supervision/scripts/ ;
- Effectuer tous les tests que vous jugerez nécessaires ;
- Découper le code en plusieurs scripts ;
- Utiliser des fonctions pour organiser le code ;
- Commenter le code.

3. Pistes de travail

- L'utilisateur supervision aura besoin des droits sudo pour les commandes réservées à root.
- Le système attribue le shell /bin/bash à un utilisateur standard, tentez d'attribuer votre script à la place !
- Utilisez la commande logger pour suivre les actions des superviseurs.
- Visitez le site : https://www.shellcheck.net/

4. Proposition de correction

Le code présenté ci-dessous n'est qu'une ébauche effectuée en TP par des stagiaires après 12 heures de cours de script. Il n'est pas parfait mais peut servir de base de correction ou de départ pour l'élaboration d'un travail plus complet.

4.1. Création de l'utilisateur

L'utilisateur doit être créé en remplaçant son shell (option -s) par le script que nous allons créer :

```
useradd -s /opt/supervision/scripts/supervision.sh -g users supervision
```

Il faut autoriser l'utilisateur supervision à utiliser sudo mais seulement pour les commandes autorisées. Pour cela, nous allons créer un fichier /etc/sudoers.d/supervision contenant les directives nécessaires :

```
# Liste les commandes autorisees aux superviseurs
Cmnd_Alias SUPERVISION = /sbin/reboot, /sbin/ip

# Autorise le superviseur a lancer les commandes precedentes sans saisir
 de mot de passe
supervision    ALL=NOPASSWD:SUPERVISION
```

4.2. Menu

Créer le fichier /opt/supervision/scripts/supervision.sh et lui donner les droits en exécution :

```
mkdir -p /opt/supervision/scripts
touch /opt/supervision/scripts/supervision.sh
chown supervision /opt/supervision/scripts/*
chmod u+x /opt/supervision/scripts/*
```

La même opération sera effectuée pour chaque script créé.

```
#!/bin/bash

# Base des scripts

BASE=$(dirname "$0")
readonly BASE

. $BASE/utils.sh

function print_menu {
  while (true)
```

```
  do
    clear
    banner
    warning
    echo "Vous pouvez : "
    echo ""
    echo "  => 1) Relancer le serveur"
    echo "  => 2) Afficher la conf IP"
    echo "  => 3) Tester le reseau  "
    echo "  => 4) Afficher les utilisateurs"
    echo "  => 5) Relancer le service apache"
    echo "  => 6) Relancer le service postfix"
    echo "  => Q) Quitter ce super programme "
    echo ""
    read -p "Que voulez vous faire : " choix
    echo ""
    case $choix in
      "1")
        sudo reboot
        ;;
      "2")
        $BASE/print-net.sh
        ;;
      "3")
        $BASE/check-net.sh
        ;;
      "4")
        $BASE/affiche-utilisateurs.sh
        ;;
      "5")
        $BASE/gestion-services.sh "httpd"
        ;;
      "6")
        $BASE/gestion-services.sh "postfix"
        ;;
      "q" | "Q" | "quitter" | "quit")
        exit 0
        ;;
      *)
        echo "Cette fonction n'est pas encore developpee"

    esac
    pause
  done
}
```

```
banner

echo "Bienvenue sur votre console d'administration"
echo ""
echo "Vous pouvez effectuer quelques diagnostics avantd'appeler le
 personnel d'astreinte"
echo ""
warning

pause

print_menu

exit 0
```

4.3. Quelques fonctions utilitaires

Le fichier utils.sh contient des fonctions que nous utiliserons dans chaque script :

```
#!/bin/bash
#
# Fonctions utilitaires et variables globales
# Version 1
#
ok="                                                [OK]"
nok="                                               [NOK]"

# Affiche ok ou nok
# Arguments :
# $1 = 0 ou 1
# $2 = message a imprimer
function printOK {
  echo "$1"
  if test "$2" = "0"
  then
    echo "$ok"
  else
    echo "$nok"
  fi
}

function banner {
echo "*            Bienvenue dans l'outil de la           *"
echo "                S U P E R V I S I O N                "
```

```
echo "                                  _     _                "
echo " ___ _   _ _ __   ___ _ ____   _(_)___(_) ___  _ __   "
echo "/ __| | | | '_ \ / _ \ '__\ \ / / / __| |/ _ \| '_ \ "
echo "\__ \ |_| | |_) |  __/ |   \ V /| \__ \ | (_) | | | |"
echo "|___/\__,_| .__/ \___|_|    \_/ |_|___/_|\___/|_| |_|"
echo "          |_|                                        "
}



function pause {
  echo "Appuyer sur Entree pour continuer..."
  read
}

function warning {
  echo "ATTENTION !!!"
  echo "Toutes les actions entreprises sont renseignees dans le journal
 d'evenement !"
  echo ""
}
```

Le fichier net-utils.sh contient les fonctions liées au réseau :

```
#!/bin/bash

#
# Fonction utilitaires du reseau
# Version 1
# Depends de utils.sh

function getGateway {
  gateway=$(sudo ip route | grep default | cut -d" " -f3)
  echo $gateway
}

function getDNS {
  DNS=$(grep "nameserver" /etc/resolv.conf | tail -1 | cut -d" " -f2)
  echo $DNS
}

# Test une adresse IP
function checkIP {
  ip=$1
```

```
  msg="Test de l'adresse ip : $ip"
  ping -c 1 $ip 1> /dev/null 2>&1
  printOK "$msg" "$?"
}

# test une resolution DNS
function checkDNS {
  res=$(dig +short www.free.fr | wc -l)
  if test "$res" -gt "0"
  then
    printOK "La resolution DNS fonctionne" "0"
  else
    printOK "La resolution DNS ne fonctionne pas" "1"
  fi
}

function getPrefix {
  sudo ip add sh | grep " inet " | grep -v "127.0.0.1" | tr -s ' ' | cut
 -d" " -f 3 | cut -d "/" -f2
}
```

4.4. La gestion du réseau

Le fichier print-net.sh :

```
#!/bin/bash

#
# Test du reseau
# Version 1
#
# Arguments :
#
ici=$(dirname "$0")
. $ici/utils.sh
. $ici/net-utils.sh

echo "L'adresse IP de votre serveur est     : $(hostname -i)"
echo "L'adresse IP de votre gateway est     : $(getGateway)"
echo "L'adresse IP de votre serveur DNS est : $(getDNS)"
echo -n "Votre prefix est : "
getPrefix
```

```
echo ""
```

Le fichier check-net.sh :

```
#!/bin/bash

#
# Test du reseau
# Version 1
#
# Arguments :
#
ici=$(dirname "$0")
. $ici/utils.sh
. $ici/net-utils.sh

# Gestion du service fourni en argument
checkIP 127.0.0.1
checkIP $(hostname -i)
checkIP $(getGateway)
checkIP $(getDNS)
checkDNS
```

4.5. La gestion des services

```
#!/bin/bash

#
# Gestion des services
# Version 1
#
# Arguments :
# $1 : le nom du service a relancer
#
. ./utils.sh

# Test l'etat du service
# Si le service est demarre, il propose de le relancer
# Sinon le service est demarre
function startService {
  service=$1
  service $service status 1> /dev/null 2>&1
  status=$?
```

```
  if test "$status" = "0"
  then
    # Le service fonctionne deja
    # Faut-il le relancer ?
    echo "Le service $service fonctionne..."
    read -p "Voulez vous le relancer ? O/N " rep
    if test "$rep" = "O" -o "$rep" = "o"
    then
      # L'utilisateur a demande a le relancer
      logger "SUPP -> Relance d'apache"
      msg="Relance du serveur $service"
      service $service restart 1> /dev/null 2>&1
      printOK "$msg" "$?"
    else
      # L'utilisateur ne veut pas le relancer
      msg="Le service ne sera pas relance"
      printOK "$msg" "0"
    fi
  else
    # Le service ne fonctionne pas
    # Demarrage
    logger "SUPP -> Demarrage d'apache"
    msg="Lancement du serveur $service"
    service $service start 1> /dev/null 2>&1
    printOK "$msg" "$?"
  fi
}

# Gestion du service fourni en argument
service=$1
startService $service
```

4.6. L'affichage des utilisateurs

Le fichier affiche-utilisateur.sh :

```
#!/bin/bash

# Extrait du fichier /etc/passwd la liste :
# - des utilisateurs du systeme
# - des utilisateurs standards
# Chaque liste est affiche sur une ligne
#
# Version 1.0
```

```
# Date : 24/11/2016

# usersys : la liste des utilisateurs systemes
usersys="Voici la liste des utilisateurs systemes :\n"
# userstd : la liste des utilisateurs standards
userstd="Voici la liste des utilisateurs standard :\n"

# Stocker l'IFS dans une variable
OLDIFS='$IFS'
# Pour que la commande for fonctionne, il faut supprimer l'espace comme
 caractere de separation
IFS=$'\n'
# On boucle sur chaque ligne du fichier /etc/passwd
while read -r ligne
do
  # Isoler l'UID
  uid=$(echo $ligne | cut -d: -f3)
  # Isoler le Nom
  nom=$(echo $ligne | cut -d: -f1)
  # Si uid < 500 => Utilisateur systeme
  if test "$uid" -lt "500"
  then
    # Ajouter le nom a la liste
    usersys="${usersys}${nom}, "
  else
    # Ajouter le nom a la liste
    userstd="${userstd}${nom}, "
  fi
done < /etc/passwd


# Affichage de la liste
echo -e "$usersys"
echo ""
echo -e "$userstd"

IFS=$OLDIFS
```