LO03-Le Réseau Virtuel de Machine Linux

Du TianRen 15124842 & Yang ZheYu 16124720

01/03/2019, UTSEUS

Sommaire

1 Introduction

Nous avons créé un réseau Virtuel de Machine Linux qui nous permet de réaliser les fonctions dans le document.

1.1 Plan du projet

Selon le document, nous avons divisé le projet en 4 parties : la partie cadre, la partie fonction connect, la partie fonction admin et la partie fichier. Nous vous les préciserons à la suivante.

---

2 Description et le Codage

2.1 L'introduction des fichiers (log)

Au total, 5 fichiers sont utilisés en fonction des informations stockées et des commandes différentes, ils sont en dessous :

`Utmp.log` : Un fichier pour enregistrer des informations sur l'utilisateur actuellement connecté. Il record le nom de la machine, le nom de l'utilisateur, le data, le temps et tty/pts (le numéro de terminal virtuel).

`Password` : Un fichier pour stocker les mots de passe avec le format : user_name, mot de passe, machine_name. Chaque combinaison d' utilisateur et de la machine a un mot de passe unique.

`Information.log` : Un fichier pour stocker les serveurs et les remarque avec le format : machine_name, user_name, server, remarque. Ce fichier est associé à la commande « `afinger` ».

`login.log` : Un fichier pour stocker les informations les plus détaillées avec le format : Login, Nom, Tty, Port, Login Temps, Serveur, Remarques. Ce fichier est associé à la fichier `utmp.log` et est modifié par la fonction « `login_log` ».

`machine_name_user_name.txt` : Un fichier pour stocker les messages associés à la commande « `write` ». Il est un fichier temporaire qui est supprimé après avoir été publié.

2.2 La partie cadre

Les cinq fichiers concernant la partie cadre sont en dessous :

`rvsh.sh` : Un fichier qui fait la distinction entre le mode utilisateur et le mode administrateur et qui appelle les autres fonctions shells dans le mode correspondant. Après avoir entré le mode spécifique, vous devez entrer votre mot de passe, en ce moment-là, `login.log` enregistrera vos informations de connexion, `utmp.log` enregistrera vos informations en ligne. `Utmp.log` supprimera vos informations en ligne lorsque vous entrez « `exit` ». Enfin vous pouvez choisir la commande.

`login_log{...}` : Une fonction pour lire, écrire et vérifier les informations (en ligne) dans le fichier `utmp.log` et dans le fichier `infomation.log` , écrire les informations dans `login.log` . Cette fonction est appelée chaque fois quand l'utilisateur ou la machine est changé dans le réseau. Par example, la commande « `connect` », la commande « `su` » et etc...

Le code pour cette fonction est en dessous :

```
function login_log
{
    if ! [ -e './utmp.log' ]
    then
        : > './utmp.log'
    fi
    #Extraire les informations de « machine_name » et de « user_name » de utmp.log.
    rv_utmp_log_line=($(grep -Fs "${rv_machine_name} ${rv_user_name}" './utmp.log'))
    if [ ${rv_utmp_log_line} ]
    then
        rv_utmp_log_tty=${rv_utmp_log_line[4]}
        rv_tty='pts/'$((${rv_utmp_log_tty#*/}+1))
    else
        rv_tty='pts/0'
    fi
    echo "${rv_machine_name} ${rv_user_name} $(date +%F) $(date +%T) ${rv_tty}" >> './utmp.log'

    #Extraire les informations de « host » et de « commentaire » de Information.log
    rv_information_log=$(grep ^${rv_machine_name}$'\t'${rv_user_name} './information.log'
    rv_information_log=${rv_information_log#*$'\t'};rv_information_log=${rv_information_log#*$'\t'}
    rv_server=${rv_information_log%$'\t'*}
    rv_remark=${rv_information_log#*$'\t'}
    rv_login_log=$(printf "%-8s %-8s %-8s %-8s %-19s %-30s %-15s\n" ${rv_user_name} ${rv_user_name} ${rv_tty}
$(($RANDOM % 10)) "$(date +%F) $(date +%T)" "${rv_server}" ${rv_remark})
    rv_insert_command='1a\'
    sed -i "${rv_insert_command}${rv_login_log}" './login.log'
}
```

`logout_utmp_del_record` : Une fonction pour supprimer des informations dans le fichier `utmp.log` quand l’ utilisateur entre « `exit` » (hors ligne). Le code est en dessous :

```
function logout_utmp_del_record
{
    sed -i "/${rv_machine_name} ${rv_user_name} .* pts\/${rv_tty:4}/d" './utmp.log'
}
```

2.2.1 Le mode de connect et d'administrateur

Pour distinguer le mode et vérifier le nom d'utilisateur et le mot de passe, il faut obtenir la commande d'entrée, faire une connexion avec `password` , publier les messages (`machine_name_user_name.txt`) et enregistrer les informations de « `login` » de l'utilisateur dans le fichier `login.log` . Les codes sont en dessous :

```
#Obtenir la commande
#Utiliser le tableau
rv_argument_line=(${*})
#Si le mode d'administrateur
if [[ ${rv_argument_line} = '-admin' ]]
then
    #Le machine et le utilisateur sont par défaut
    rv_machine_name='rvsh'
    rv_user_name='root'
    echo 'Bienvenue à vous connecter "administrateur"! Veuillez entrer votre
    mot de passe (le mot de passe par défaut est "admin"):'
    #Le mot de passe et la vérification
    read rv_password
    grep -Fxsq "${rv_user_name} ${rv_password} ${rv_machine_name}" './password' #Extraire
    if [ ${?} = 0 ]
    then
        #Enregistrer les informations de «login» de l'utilisateur dans le fichier login.log
        login_log
        while true
        do
            #S'il y a une message (shell fonction «write»)
            rv_message_file="${rv_machine_name}_${rv_user_name}.txt"
            if [ -e ${rv_message_file} ]
            then
                echo 'rvsh>'
                #Publier des messages et supprimer le fichier (message txt)
                cat ${rv_message_file}
                rm -f ${rv_message_file}
            fi
        #commande choix...
        done
    else
        echo "C'est le mauvais mot de passe!"
    fi
```

```
#Si le mode de connect
#Utiliser le tableau
elif [[ ${rv_argument_line[0]} = '-connect' ]]
then
    rv_machine_name=${rv_argument_line[1]}
    grep -Esq "^[^[:space:]]+ [^[:space:]]+ ${rv_machine_name}$" './password'
    #S'il existe la machine
    if [ ${?} = 0 ]
    then
        rv_user_name=${rv_argument_line[2]}
        echo "Veuillez entrer le mot de passe pour ${rv_user_name}"
        read rv_password
        grep -Esq "${rv_user_name} [^[:space:]]+ ${rv_machine_name}" './password'
        #S'il le nom est correct
        if [ ${?} = 1 ]
        then
            echo "C'est le mauvais nom d'utilisateur!"
            exit
        fi
        grep -Fxsq "${rv_user_name} ${rv_password} ${rv_machine_name}" './password'
        #Vérification du mot de passe
        if [ ${?} = 0 ]
        then
            #Enregistrer les informations de « login » de l'utilisateur dans le fichier login.log
            login_log
            echo "Bienvenue ${rv_user_name}!"
            while true
            do
              rv_message_file="${rv_machine_name}_${rv_user_name}.txt"
                if [ -e ${rv_message_file} ]
                then
                    #Publier des messages et supprimer le fichier (message txt)
                    echo "${rv_user_name}@${rv_machine_name}>"
                    cat ${rv_message_file}
                    rm -f ${rv_message_file}
                fi
    #commande choix...
            done
        else
            echo "C'est le mauvais mot de passe!"
        fi
    else
        echo "C'est le mauvais nom de machine!"
    fi
#./rvsh.sh paramètre
else
    echo "C'est le mauvais paramètre!"
fi
```

2.2.2 Liens vers d'autres fonctionnalités

Dans le mode de connect, il y a 8 commandes qui correspondent à 8 fichiers (.sh). Chaque fichier est jugé avec une déclaration de « `case` » dans `rvsh.sh`. Le code est en dessous :

```
#Dans le mode d'administrateur
read -p 'rvsh>' rv_command rv_parameters
case ${rv_command} in
who)
    . ./who.sh
;;
rusers)
    . ./rusers.sh
;;
rhost)
    . ./rhost.sh
;;
connect)
    . ./connect.sh ${rv_parameters}
;;
su)
    if [ -z ${rv_parameters} ]
    then
        rv_parameters='root'
    fi
    . ./su.sh ${rv_parameters}
;;
passwd)
    . ./passwd.sh
;;
finger)
    . ./finger.sh
;;
write)
    . ./write.sh ${rv_parameters}
;;
host)
    . ./host.sh ${rv_parameters}
;;
users)
    . ./users.sh ${rv_parameters}
;;
#afinger machine_name user_name
afinger)
    . ./afinger.sh ${rv_parameters}
;;
#Supprimer l'information ( en ligne ) dans utmp.log
exit)
    logout_utmp_del_record
    exit
;;
'')
    :
;;
*)
    echo "${command} C'est la mauvaise commande!"
;;
esac
================================================
#Dans le mode de connect
read -p "${rv_user_name}@${rv_machine_name}>" rv_command rv_parameters
case ${rv_command} in
who)
    . ./who.sh
;;
rusers)
    . ./rusers.sh
;;
rhost)
    . ./rhost.sh
;;
connect)
    . ./connect.sh ${rv_parameters}
;;
su)
    if [ -z ${rv_parameters} ]
    then
        rv_parameters='root'
    fi
    . ./su.sh ${rv_parameters}
;;
passwd)
    . ./passwd.sh
```

```
;;
finger)
    . ./finger.sh
;;
write)
    . ./write.sh ${rv_parameters}
;;
exit)
    #Supprimer l'information ( en ligne ) dans utmp.log
    logout_utmp_del_record
    exit
;;
'')
    :
;;
*)
    echo "${command} C'est la mauvaise commande!"
;;
esac
```

3 Les commandes

3.1 Les commandes pour le mode d'administrateur

La commande `host` : Elle permet à l' administrateur d' ajouter ou d' enlever une machine au réseau virtuel. À la première fois, la machine ajoutée doit connecter avec l' administrateur (root). Le code est en dessous :

```
#Dans rvsh.sh, le choix host sur le mode d'administrateur
host)
    . ./host.sh ${rv_parameters}

#--------------------
#Dans le fichier host.sh
#Deux choix add/del
rv_host_option=${1}
#ADD
if [ ${rv_host_option} = add ]
then
    rv_host_machine_name=${2}
    grep -Esq "^[^[:space:]]+ [^[:space:]]+ ${rv_host_machine_name}$" './password'
    if [ ${?} = 1 ]
    then
        rv_host_username=root
        echo "Entrez le mot de passe pour root"
        read rv_host_password
        echo "${rv_host_username} ${rv_host_password} ${rv_host_machine_name}" >> './password'
        echo "Fini"
    elif [ ${?} = 0 ]
    then
        echo "Le nom de la machine à ajouter existe déjà."
    fi
#DEL
elif [ ${rv_host_option} = del ]
then
    rv_host_machine_name=${2}
    grep -Esq "^[^[:space:]]+ [^[:space:]]+ ${rv_host_machine_name}$" './password'
    if [ ${?} = 0 ]
    then
        #Supprimer tout le connection avec la machine
        sed -Ei "/^[^[:space:]]+ [^[:space:]]+ ${rv_host_machine_name}$/d" './password'
    elif [ ${?} = 1 ]
    then
        echo "Le nom de la machine à supprimer n'existe pas."
    fi
else
    echo "Il n'y a pas d'option ${rv_host_option}."
fi
```

La commande `users` : Elle permet à l' administrateur d' ajouter ou d' enlever un utilisateur , de lui donner les droits d' accès à une ou plusieurs machines du réseau et de lui fixer un mot de passe. À la première fois, l' utilisateur à ajouter doit connecter avec une machine existant.

```
#Dans rvsh.sh, le choix host sur le mode d'administrateur
users)
    . ./users.sh ${rv_parameters}

#--------------------

#Dans le fichier users.sh
rv_users_option=${1}
#Deux choix add/del
if [ ${rv_users_option} = add ]
then
    echo "Entrez le nom de la machine dans laquelle vous voulez mettre l'utilisateur"
    read rv_users_machine_name
    #Vérifier le fichier password  si le nom de la machine existe déjà ou non
    grep -Esq "^[^[:space:]]+ [^[:space:]]+ ${rv_users_machine_name}$" './password'
    if [ ${?} = 0 ]
    then
        rv_users_user_name=${2}
        #Vérifier le fichier password si le nom d'utilisateur existe déjà ou non.
        grep -Esq "^${rv_users_user_name} [^[:space:]]+ ${rv_users_machine_name}$" './password'
        if [ ${?} = 1 ]
        then
            echo "Tapez votre mot de passe"
            read rv_users_password
            echo "${rv_users_user_name} ${rv_users_password} ${rv_users_machine_name}" >> './password'
        elif [ ${?} = 0 ]
        then
            echo "Le nom d'utilisateur à ajouter existe déjà."
        fi
    elif [ ${?} = 1 ]
    then
        echo "Le nom de la machine ${rv_users_machine_name} n'existe pas."
    fi
elif [ ${rv_users_option} = del ]
then
    rv_users_user_name=${2}
    grep -Esq "^${rv_users_user_name} [^[:space:]]+ [^[:space:]]+$" './password'
    if [ ${?} = 0 ]
    then
        #Supprimer tout le connection avec la machine
        echo "Entrez le nom de la machine que vous voulez mettre dans l'utilisateur"
        read rv_users_machine_name
        grep -Esq "^[^[:space:]]+ [^[:space:]]+ ${rv_users_machine_name}$" './password'
        if [ ${?} = 0 ]
        then
            sed -Ei "/^${rv_users_user_name} [^[:space:]]+ ${rv_users_machine_name}$/d" './password'
        fi
    elif [ ${?} = 1 ]
    then
        echo "Le nom d'utilisateur à supprimer n'existe pas."
    fi
else
    echo "Il n'y a pas d'option ${rv_users_option}."
fi
```

La commande `afinger` : Elle permet à l'administrateur de renseigner les informations complémentaires sur l'utilisateur dans le fichier `infomation.log` (l'utilisateur aura l'accès aux informations avec la commande `finger` dans le mode connect). Après renseigner les informations, l'utilisateur correspondant doit se connecter pour mettre à jour les informations en utilisant le fonction `login_log`.

```
#Dans le fichier rvsh.sh, dans le mode d'administrateur
#afinger machine_name user_name
afinger)
    . ./afinger.sh ${rv_parameters}
;;

#-------------------------------------------
#Dans le fichier afinger.sh
#Un tableau de chaînes
rv_afinger_machine_name=${1}
rv_afinger_user_name=${2}
echo "S'il vous plaît entrer le mot de passe pour ${rv_afinger_user_name}"
#Vérifier le mot de passe
read rv_afinger_password
grep -Esq "${rv_afinger_user_name} [^[:space:]]+ ${rv_afinger_machine_name}" './password'
if [ ${?} = 0 ]
then
    grep -Fxsq "${rv_afinger_user_name} ${rv_afinger_password} ${rv_afinger_machine_name}" './password'
    if [ ${?} = 0 ]
    then
        echo "S'il vous plaît entrer le serveur"
        read rv_afinger_server
        echo "S'il vous plaît entrer les remarques"
        read rv_afinger_remark
        #Utiliser un tableau de chaînes pour stocker des informations

rv_afinger_information_log=${rv_afinger_machine_name}$'\t'${rv_afinger_user_name}$'\t'"${rv_afinger_server}"$'\t
'${rv_afinger_remark}
        #Écrire dans infomation.log
        if ! [ -e './information.log' ]
        then
            : > './information.log'
        fi
        grep ^${rv_afinger_machine_name}$'\t'${rv_afinger_user_name} './information.log'
        if [ ${?} = 0 ]
        then
            #Pour écraser les anciennes informations avec de nouvelles informations
            sed -i s/^${rv_afinger_machine_name}$'\t'${rv_afinger_user_name}.*$/"${rv_afinger_information_log}"/
'./information.log'
        else
            echo "${rv_afinger_information_log}" >> './information.log'
        fi

    else
        echo "C'est le mauvais mot de passe!"
    fi
else
    echo "Ceci est un nom de machine ou un nom d'utilisateur incorrect!"
```

3.2 Commandes dans les deux modes

Les 8 commandes ci-dessous peuvent être utilisées sous les deux modes :

La commande `who` : Elle permet d'accéder à l'ensemble des utilisateurs connectés sur la machine. Elle renvoie le nom de chaque utilisateur, l'heure et la date de sa connexion. Les informations proviennent du fichier `utmp.log`. S'il est sur mach1, la commande va seulement renvoyer les informations des utilisateurs qui sont connecté avec mach1. Le code est en dessous :

```
#Dans le fichier rvsh.sh
who)
    . ./who.sh
;;

#----------------------

#Dans le fichier who.sh
#Touts les utilisateurs connectés sur la machine maintenant

while read
do
    #Lire chaque ligne du fichier utmp.log et stocker la chaîne dans le tableau par défaut ${REPLY}
    rv_who_line=(${REPLY})
    #Si le nom de la machine correspond à la machine actuelle, echo le content
    if [ ${rv_who_line[0]} = ${rv_machine_name} ]
    then
        echo "${rv_who_line[@]:0:4}"
    fi
#Lire le fichier utmp.log
done < './utmp.log'
```

La commande `rusers` : Elle permet d' accéder à la liste des utilisateurs connectés dans le réseau. Elle renvoie le nom de chaque utilisateur et le nom de la machine où il est connecté, ainsi que l' heure et la date de sa connexion. Les informations proviennent du fichier `utmp.log`. Elle va renvoyer les informations de tous les utilisateurs connectés. Le code est en dessous :

```
#Dans le fichier rvsh.sh
rusers)
    . ./rusers.sh
;;

#------------------

#Dans le fichier rusers.sh

while read
do
    #Lire chaque ligne du fichier utmp.log et stocker la chaîne dans le tableau par défaut ${REPLY}
    rv_rusers_line=(${REPLY})
    #Echo le content
    echo "${rv_rusers_line[@]:0:4}"
#Lire le fichier utmp.log
done < './utmp.log'
```

La commande `rhost` : Elle renvoie la liste des machines rattachées au réseau virtuel. Le code est en dessous :

```
#Dans le fichier rvsh.sh
rhost)
    . ./rhost.sh
;;

#--------------------------

#Dan le fichier rhost.sh

while read
do
    #Lire chaque ligne du fichier utmp.log et stocker la chaîne dans le tableau par défaut ${REPLY}
    rv_rhost_line=(${REPLY})
    #Echo le content (machin_name)
    echo "${rv_rhost_line[0]}"
#Lire le fichier utmp.log
done < './utmp.log'
```

La commande `connect` : Elle permet à l' utilisateur de se connecter à une autre machine du réseau (elle va vérifier que si l'utilisateur a le droit de se connecter sur cette machine). Il faut entrer le nom de la machine et le nom de l' utilisateur. Par exemple, si « `user1@mach1` » souhait connecter avec « `mach2` », il faut entrer « `connect mach2 user1` ». Après elle va vérifier dans le fichier `password`. S' il existe l'information de user1@mach2, connecter avec le succès. Le code est en dessous :

```
#Dans le fichier rvsh.sh

connect)
    . ./connect.sh ${rv_parameters}
;;

#-------------------------

#Dans le fichier connect.sh
#${connect machine user}

rv_connect_machine_name=${1}
rv_connect_user_name=${2}
if ! [ ${rv_connect_user_name} ]
then
    rv_connect_user_name=${rv_user_name}
fi
#Obtenir le mot de password pour identifier
echo "Veuillez entrer le mot de passe pour ${rv_connect_user_name}"
read rv_connect_password
grep -Esq "${rv_connect_user_name} [^[:space:]]+ ${rv_connect_machine_name}" './password'
if [ ${?} = 0 ]
then
    grep -Fxsq "${rv_connect_user_name} ${rv_connect_password} ${rv_connect_machine_name}" './password'
    if [ ${?} = 0 ]
    then
        #Vérifier dans le fichier password. S'il exist l'information, il va connecter.
        rv_machine_name=${rv_connect_machine_name}
        rv_user_name=${rv_connect_user_name}
        rv_password=${rv_connect_password}
        #Enregistrer les informations de connexion
        login_log
        echo "Bienvenue ${rv_user_name}!"
    else
        echo "C'est le mauvais mot de passe!"
    fi
else
    echo "C'est le mauvais nom de machine ou le nom d'utilisateur!"
fi
```

La commande `su` : Elle permet de changer l'utilisateur. Si vous voulez changer dans le mode d' administrateur (root) , la commande `su` ne doit pas avoir de paramétres. Si la combinaison de l'utilisateur et de la machine n'est pas dans le fichier `password`, il ne va pas connecter. Le code est en dessous :

```
#Dans le fichier rvsh.sh

su)
    #S'il n'y a pas de paraméters=>root
    if [ -z ${rv_parameters} ]
    then
        rv_parameters='root'
    fi
    . ./su.sh ${rv_parameters}
;;

#-------------------------

#Dans le fichier su.sh

rv_su_user_name=${1}
echo "Veuillez entrer le mot de passe pour  ${rv_su_user_name}"
read rv_su_password
#Vérifier le mot de passe
#Vérifier la combinaison de l'utilisateur et de la machine
grep -Esq "^${rv_su_user_name} [^[:space:]]+ ${rv_machine_name}$" './password'
if [ ${?} = 0 ]
then
    grep -Fxsq "${rv_su_user_name} ${rv_su_password} ${rv_machine_name}" './password'
    if [ ${?} = 0 ]
    then
        #passer des paramètres
        rv_user_name=${rv_su_user_name}
        echo "Bienvenue ${rv_su_user_name}!"
    else
        echo "C'est le mauvais mot de passe!"
    fi
else
    echo "C'est le mauvais nom d'utilisateur!"
fi
```

La commande `passwd` : Elle permet à l’ utilisateur de changer le mot de passe sur l’ ensemble du réseau virtuel. « `Until[flag...]` » assure que les mots de passe entrés deux fois sont les mêmes. Le nouveau mot de passe écrase le mot de passe ancien et écrit dans le tableau d'origine. Le tableau (Le nom d'utilisateur, le mot de passe et le nom de la machine) met à jour dans le fichier `password`.

```
#Dans le fichier rvsh.sh

passwd)
    . ./passwd.sh
;;

#-------------------------

#Dans le fichier passwd.sh

#Déterminer l'utilisateur actuel en tant qu'objet modifié
#Afficher le mot de passe actuel
echo "Changer le mot de passe de ${rv_user_name}"
rv_passwd_passwordline=($(grep -Es "^${rv_user_name} [^[:space:]]+ ${rv_machine_name}$" './password'))
rv_passwd_password=${rv_passwd_passwordline[1]}
echo "(Maintenant) le mot de passe :${rv_passwd_password}"
rv_password_flag='false'
#rv_password_flag est pour assurer que les mots de passe entrés deux fois sont les mêmes
until [ ${rv_password_flag} = 'true' ]
do
    #Obtenir le nouveau mot de passe
    read -p 'Le nouveau mot de passe:' rv_passwd_password
    read -p 'Retaper le nouveau mot de passe:' rv_passwd_repassword
    #Confirmer le mot de passe à nouveau.
    if [ ${rv_passwd_password} = ${rv_passwd_repassword} ]
    then
        rv_password_flag='true'
        #Le nouveau mot de passe écrase l'ancien mot de passe et écrit dans le tableau d'origine
        rv_passwd_comm="s/${rv_passwd_passwordline[@]}/${rv_user_name} ${rv_passwd_password}
${rv_machine_name}/"
        #Le nom d'utilisateur, le mot de passe et le nom de la machine mis à jour dans le fichier password
        sed -i "${rv_passwd_comm}" './password'
    else
        echo 'Les mots de passe entrés deux fois sont incohérents. Veuillez ré-entrer.'
    fi
done
```

La commande `finger` : Elle permet de renvoyer des éléments complémentaires sur l' utilisateur. Les informations sont dans le fichier `login.log`. (Le fichier `login.log` est modifié par la fonction « `login_log` ». Le code est en dessous :

```
#Dans le fichier rvsh.sh
finger)
    . ./finger.sh
;;

#-----------

#Dans le fichier finger.sh
cat './login.log'

#Dans le fichier login.log
Login    Nom    Tty    Port    Login Temps      Serveur                Remarques
```

La commande `write` : Elle permet d' envoyer un message à un utilisateur connecté sur une machine du réseau. La syntaxe de la commande est : `write nom_utilisateur@nom_machine message`. Dans cette commande, la commande `trap` peut spécifier le signal Linux que le script shell doit surveiller et intercepter. Le code est en dessous :

```
#Dans le fichier rvsh.sh
write)
    . ./write.sh ${rv_parameters}
;;

#---------------------------
#Dans le fichier write.sh

#La commande trap peut spécifier le signal Linux que le script shell doit surveiller et intercepter.

#Surveiller s'il y a une entrée ^c

function onCtrlC

{

    rv_write_abort_flag='true'

    trap - INT

}



rv_write_user_name=${1%@*}

rv_write_machine_name=${1#*@}

#Le fichier est pour stocker le message

rv_write_message_file="${rv_write_machine_name}_${rv_write_user_name}.txt"

#write user2@mach2 01 02 03 04 05

#couper la string "user2@mach2 this is user1" => "this is user1"

rv_write_greeting=${@#*' '}

rv_write_greeting=${rv_write_greeting#*' '}

#Stocker les destinataires et le temps

echo "Message pour ${rv_write_user_name}@${rv_write_machine_name} sur pts/0, temps: $(date +%T), date: $(date
+%F)" >> ${rv_write_message_file}

echo "${rv_write_greeting}" >> ${rv_write_message_file}



echo "Taper ^ c pour finir"

trap onCtrlC INT

rv_write_abort_flag='false'

while [ ${rv_write_abort_flag} = 'false' ]

do

    #stocker le message dans le fichier temporaire

    read rv_write_message

    echo ${rv_write_message} >> ${rv_write_message_file}

done
```

4 Conclusion

Nous avons quelques difficultés lorsque nous travaillons sur le projet. Il nous a fallu beaucoup de temps pour bien comprendre chaque commande fournie.

1.Lors de la conception du cadre du programme, nous avons passé beaucoup de temps à effectuer des idées et des façons différentes. Enfin, le structure que nous avons sélectionné est un fichier principal (`rvsh.sh`) + plusieurs commandes fichiers (`.sh`) + plusieurs fichiers (`log`).

2.Le code écrit dans " `SublimeText3` " (Windows10-1803) ne fonctionne pas dans le système " `Linux-ubuntu-18.04` ". Résolution de problèmes est le changement du caractère de code en ' `unicode` '.

3.La commande « `write` » est la plus difficile de concevoir. Les fichiers temporaires nous ont aidé à résoudre le problème du stockage de l'information. Une autre difficulté consiste à afficher les informations lorsque l'utilisateur se connecte. Nous avons écrit le code permettant de lire les informations dans la fonction `login_log` .

4.Lorsque j'ai changé le mot de passe dans le fichier `password` , nous avons utilisé le tableau pour écraser le mot de passe ancien, mais au début, nous ne connaissions pas bien l'utilisation du tableau et il y avait un cas où le mot de passe n'avait pas été écrasé. Résolution de ce problème est utiliser `{@}` .

Enfin, merci à nos professeurs et à Internet de nous aider !

Codeing: Du TianRen

Debug: Yang ZheYu, Du TianRen

Editeur: Du TianRen