

# LO03 Projet Linux 2019

*rvsh*

« Ce projet a pour objectif de créer une réseau virtuel de machine linux. »

Ziyi XU [REDACTED] & [REDACTED] YUAN [REDACTED] - janvier et février 2019<sup>1</sup>

```
== MODE CONNECT ==
Bienvenu ! Ce mode vous permet de vous connecter à une machine virtuelle.

-- Les commandes disponibles --
who      rusers      rhost      connect
su      passwd      finger      write

MacBookPro@ZiyiXU > who
armandxu console  Feb  2 20:46
terry    console  Feb  1 15:55
terry    ttys000  Feb  3 13:06
armandxu ttys001  Feb  2 21:04
MacBookPro@ZiyiXU > █
```

<sup>1</sup> Toutes les présentations sont réalisé sur macOS.

---

# SOMMAIRE

## LO03 Projet Linux 2019

Introduction .....	1
Configuration.....	2
Réalisation .....	4
Sécurité .....	10
Programmation multiThread .....	12
Référence .....	13

# Introduction

Ce projet a pour objectif de créer un réseau virtuel de machine Linux par créer une nouvelle commande Shell, qui s'appelle « rvsh ». Elle fonctionne selon deux modes :

## 1. Le Mode CONNECT

```
rvsh -connect nom_machine nom_utilisateur
```

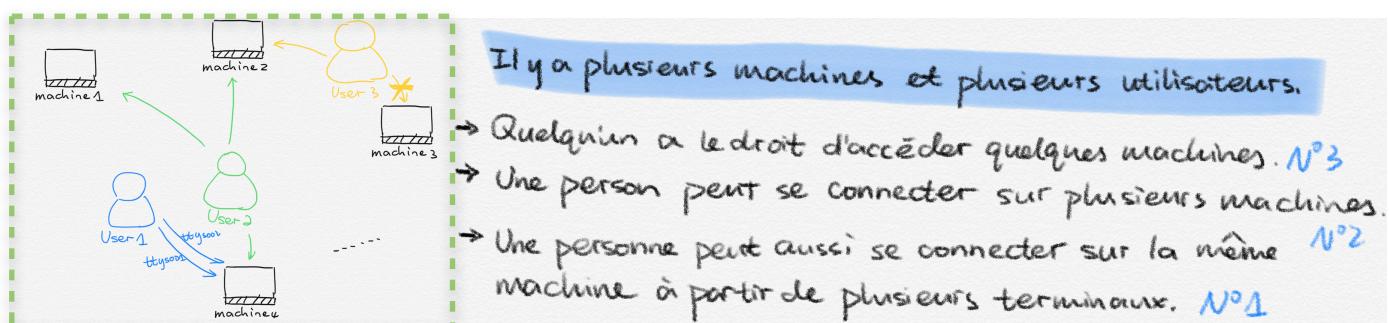
Le mode connect permet à un utilisateur de se connecter à une machine virtuelle.

Si le nom de l'utilisateur et le nom de la machine virtuelle sont corrects, la connexion est acceptée (c'est-à-dire l'utilisateur a le droit de se connecter sur cette machine et son mot de passe est correct) et l'utilisateur arrive sur le prompt :  
`nom_utilisateur@nom_machine >`

## 2. Le Mode ADMIN

```
rvsh -admin
```

Seul l'administrateur du réseau virtuel doit pouvoir utiliser ce mode. Donc l'accès à cette commande doit être géré par un mot de passe (mot de passe de l'administrateur). Une fois la commande lancée et le mot de passe validé, l'administrateur arrive sur le prompt suivant : `rvsh >`



Ce projet a été développé sur macOS 10.14.2, GNU bash, version 3.2.57(1)-release (x86\_64-apple-darwin18) Copyright (C) 2007 Free Software Foundation, Inc., en coopération entre Ziyi XU et ██████ YUAN, avec l'aide d'internet et des diapos de LO03.

La fonction est réalisée sous la forme du script Shell. Nous avons crée une interface approximativement complète, avec l'interaction entre la console et l'utilisateur, le prompt modifiable, les conseils d'utilisation, etc.

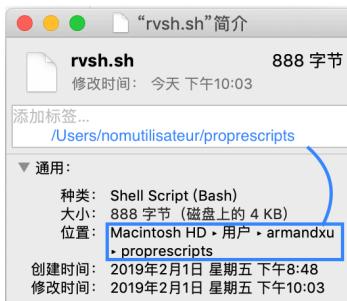
# Configuration

« rvsh », qui est créée par notre groupe est développé sous la forme du script, c'est donc facile pour équiper cette commande :

## Installation - Manuel

1<sup>er</sup> étape :

- Copier le script (nommé « rvsh.sh » par exemple) dans `~/proprescripts`.



2<sup>ème</sup> étape :

- Dans terminal, utiliser la commande « alias » pour définir le synonyme « rvsh » de la commande « `bash ~/proprescripts/rvsh.sh` ».

```
xuziyideMacBook-Pro:~ armandxu$ alias  
alias rvsh='bash ~/proprescripts/rvsh.sh'
```

- Mais en effet la commande ci-dessus ne prends effet que dans la session en cours, donc nous recommandons d'utiliser une autre méthode : ajouter la commande à la fin du fichier `~/.bashrc`, puis ajouter les scripts dans `~/.bash_profile`

```
if [ "${BASH-no}" != "no" ]; then  
    [ -r ~/.bashrc ] && . ~/.bashrc  
fi
```

(dans `~/.bash_profile`)

```
alias rvsh='bash ~/proprescripts/rvsh.sh'
```

(dans `~/.bashrc`)

3<sup>ème</sup> étape :

- Créer 6 fichiers vides dans `~/proprescripts/rvsh` : « droit », « passwd », « who\_liste », « machine\_list », « users » et « rhost ».

## Installation - Automatique

- Exécuter « `initialisation.sh` » : `bash ./initialisation.sh` qui initiale et se prépare automatiquement. Et puis `source ~/.bashrc`.

## Désinstallation

- Exécuter « desinstallation.sh » : `bash ./desinstallation.sh` qui restaure toutes les modifications automatiquement.

### DEMO

Monsieur le professeur, vous pouvez utiliser le répertoire créé préalablement : « DEMO ».

Il y a 3 utilisateurs en total :

	Login	Mot de passe	UID	GID	GECOS	Shell
1	root	mot de passe de l'administrateur	1	1	gecos	shell
2	terry	terry	501	100	gecos_terry	shell
3	maxime	maxime	502	100	gecos_maxime	shell

Avec ces informations, vous pouvez réaliser toutes les fonctions de la commande.

Par exemple :

```
Last login: Tue Feb 12 17:58:48 on ttys003
[xuziyideMacBook-Pro:~ armandxu$ rvsh -admin
==== MODE ADMIN ====
[Saisissez le code S.V.P.   ↗mot de passe de l'administrateur
Bonjour !

          -- Les commandes disponibles --
who      rusers      rhost      connect
su      passwd      finger      write
host -a host -d      users -a      users -d
afinger

rvsh > afinger
Renseigner les informations complémentaires à QUI : ]
```

# Réalisation

## Données et Fichiers



- ❖ **~/proprescripts/rvsh.sh** contient toutes les commandes qui marchent pour réaliser la fonction.
  - ❖ **~/proprescripts/installation.sh** est un script qui crée les données nécessaires.
  - ❖ **~/proprescripts/desinstallation.sh** est un script qui restaure toutes les modifications automatiquement.
  - ❖ **~/proprescripts/rvsh/passwd** est semblable à /etc/passwd dans Linux :  
login:motdepasse:UID:GID:Gecos:homdir:shell - 7 champs pour les attributs qui caractérisent un utilisateur du réseau virtuel. Mais le 2<sup>ème</sup> champ pour le mot de passe n'est pas « \* » comme « passwd » dans Linux, l'algorithme d'encodage MD5 est utilisé ici.
  - ❖ **~/proprescripts/rvsh/machine\_liste** est une liste qui contient les machines existantes :  
MacBook Pro  
Mac  
mac  
windows
- Il peut facilement modifié par la commande « host » du mode ADMIN.
- ❖ **~/proprescripts/rvsh/droit** contient UID et les machines qu'il est accessible :  
501:Macbook Pro:windows
  - Modifiable avec la commande « users » du mode ADMIN.
  - ❖ **~/proprescripts/rvsh/who\_liste** contient le nom de l'utilisateur, à partir de laquelle terminal il a utilisé, la date et l'heure de connexion et le nom de la machine. Ce fichier garantit le fonctionnement normal de la commande « who » et « users ».

terminal	user	date	time	machine
tty1	terry	2 12	4:09	windows
tty2	terry	2 12	4:31	windows
tty3	terry	2 12	4:31	windows
tty4	terry	2 12	4:35	windows
tty5	terry	2 12	4:36	windows
tty6	terry	2 12	4:37	windows
tty7	terry	2 12	4:37	windows

---

## Préparation

- ❖ **La fonction « densifier()** pour convertir le code en MD5.

Usage : `densifier "string"`.

```
mot_référence=`densifier ${motdepasse}`
```

- ❖ **La fonction « who\_shell()** simule la commande « who » définie en interne.

- ❖ **La fonction « users\_shell()** simule la commande « users » définie en interne.

- ❖ **La fonction « entrer\_verifier()** permet à l'utilisateur de saisir le mot de passe et vérifier avec le mot\_référence qui est obtenu de `~/proprescripts/rvsh/passwd`. Et elle va renvoyer une valeur entière : 1 vrai et 0 faux pour montrer le résultat.

Usage : `entrer_verifier() nom_utilisateur`

- ❖ **La fonction « machine\_verifier()** permet de vérifier si la machine entrée existe ou pas. Sinon, la elle va afficher « La machine nom\_machine n'existe pas ! Vérifier de nouveau ! », si la machine existe déjà, elle ne ferait rien.

Usage : `machine_verifier() nom_machine`

- ❖ **La fonction « get\_uid()** » s'agit de renvoyer l'UID de l'utilisateur. Elle va lire le fichier « passwd » pour savoir l'UID.

Usage : `uid=`get_uid ${nom_utilisateur}``

- ❖ **La fonction « droit\_verifier()** pour vérifier tel que l'utilisateur a le droit d'accéder la machine.

Puisque le fichier « droit » mémorise toutes les machines qui sont accessibles par tel utilisateur, il suffit de la lire, et puis renvoie une valeur entière : 1 vrai et 0 faux pour montrer le résultat.

Usage : `droit_verifier nom_utilisateur nom_machine`

- ❖ **La fonction « login()** pour vérifier tel que l'utilisateur a saisi le correct mot de passe.

Usage : `login nom_utilisateur nom_machine`

Similaire au mécanisme de fonctionnement de Linux, après l'utilisateur a réussi à se connecter sur une machine, « login() » va écrire une ligne dans un fichier temporaire : `who_liste` ( Cf. `/var/run/utmp`), cette ligne de message contient le nom de l'utilisateur, à partir de laquelle terminal il a utilisé, la date et l'heure de connexion et le nom de la machine.

## Mode connect

- ❖ La commande « who » va afficher tous les utilisateurs connectés à cet époque là, avec les informations : login(le nom de chaque utilisateur), nom de tty, la date et l'heure de la connexion, et aussi le hostname si l'utilisateur n'est pas local.

Donc on utilise la fonction « who\_shell() » pour afficher des informations.

```
"who" )      #echo mode who
who_shell
;;
;
```

- ❖ La commande « rusers » doit renvoyer le nom de chaque utilisateur et (1) le nom de la machine où il est connectée, ainsi que (2) l'heure et la date de sa connexion.

Donc on utilise « who\_shell » pour obtenir (1) et (2), et quant au nom de l'utilisateur, utilisant « users\_shell » qui répertorie les noms des utilisateurs actuellement sur le système, dans un ordre trié, séparés par espace, sur une seule ligne. On a utilisé les commandes « grep », « tr », et « cut » afin de traiter le texte.

```
"rusers" ) #echo mode rusers
nombre_utilisateur=users_shell | wc -w` #Cette commande permet d'accéder à la liste des utilisateurs connectés sur le réseau.
                                         #Elle doit renvoyer 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.
for (( i = 1; i <= nombre_utilisateur; i++ )); do
    #statements
    utilis_current=users_shell | cut -f$i -d' '
    echo "___${utilis_current}___"
    who_shell | grep ${utilis_current} | tr '\t' ' ' | tr -s ' ' | cut -d ' ' -f2-5
done
;;
;
```

- ❖ La commande « rhost » doit renvoyer la liste des machines rattachées au réseau virtuel. En seulement affichant le 2<sup>ème</sup> champ de la résultat de la fonction « who\_shell() ».

```
"rhost" ) #echo mode rhost
echo "Des machines rattachées :"
who_shell | tr '\t' ' ' | tr -s ' ' | cut -d ' ' -f2
                                         #Cette commande doit renvoyer la liste des machines rattachées au réseau virtuel.
;;
;
```

- ❖ La commande « connect » <sup>2</sup> permet à l'utilisateur de se connecter à une autre machine du réseau avec vérifiant préalablement que l'utilisateur a le droit de faire ça. On a crée deux variables : « machine » et « utilisateur ». Quand une personne utilise la commande « connect », on vérifiera si la machine existe ou pas, et puis on essaie de connecter cette machine par l'utilisateur actuel. Si tout est bon, on va remplacer le prompt en modifiant la variable « machine ». Donc le module « droit\_verifier() » est utilisé.

- ❖ La commande « su » permet de changer d'utilisateur.

De la même façon, on a réalisé cette fonction par modifier « utilisateur », après avoir vérifié le compte. Donc le module « login() » est utilisé.

- ❖ La commande « passwd » permet de changer le mot de passeur l'ensemble du réseau virtuel.

<sup>2</sup> Les commandes avec cette icône () ont les vidéos de l'enregistrement de l'écran dans le répertoire « DEMO ».

Après avoir vérifié le mot de passe actuel, l'utilisateur peut changer son mot de passe, avec deux fois de confirmation.

```
"connect" ) #echo mode connect
    read -p "Sur quelle machine ? " nom_machine
    machine_verifier ${nom_machine}
    droit_verifier ${utilisateur} ${nom_machine}
    ((resultat=$?))

    if [[ ${resultat} == 0 ]]; then
        echo "L'utilisateur ${utilisateur} n'a pas de droit !"
        echo "Vérifier de nouveau !"
    else
        machine=${nom_machine}
        month=`date +%b` #(Jan..Dec)
        day=`date +%d` #(01..31)
        time=`date +%l:%M` #%l:(1..12) %M:(00..59)
        nb_tty=`cat ~/proprescripts/rvsh/who_liste | grep -w -c ${utilisateur}`
        ((nb_tty++))
        echo -e "${utilisateur}\t${nb_tty}\t$month\t$day\t$time\t$machine" >> ~/proprescripts/rvsh/who_liste
        echo Changé !
    fi

    if [[ `grep -c -w ${machine} ~/proprescripts/rvsh/rhost` == 0 ]]; then
        printf "%s\n" ${machine} >> ~/proprescripts/rvsh/rhost
    fi
;;
#Cette commande permet à l'utilisateur de se connecter à une autre machine
#(il faut préalablement ajouter l'entrée dans /etc/hosts)
    ♦ La commande « connect »
```

```
"su" ) #echo mode su

    while [[ 1 ]]; do
        read -p "À quel utilisateur voulez-vous passer ? " utilisateur_passer

        if [[ `grep -c -w ${utilisateur_passer} ~/proprescripts/rvsh/passwd` == 0 ]]; then
            echo "Utilisateur ${utilisateur_passer} n'existe pas !"
            echo "Vérifier de nouveau !"
        else
            break
        fi

    done

    login ${utilisateur_passer} ${machine}
    utilisateur=${utilisateur_passer}
    echo
;;
    ♦ La commande « su »
```

```
"passwd" ) #echo mode passwd
    read -p "Mot de passe actuel : " -s motdepasseactuel
    #Cette commande permet à l'utilisateur de changer de mot de passe sur l'ensemble du réseau virtuel
    #Cf. commande passwd de Linux
    # armandxu hai xia yao xiu gai !

    echo
    mot=`densifier ${motdepasseactuel}`
    mot_reference=`cat ~/proprescripts/rvsh/passwd | grep ${utilisateur} | cut -f2 -d ':'`
    if [[ ${mot} == ${mot_reference} ]]; then
        read -p "Nouveau mot de passe : " -s motdepassenouveau
        echo
        read -p "Retapez le nouveau mot de passe : " -s motdepassenouveau1
        echo
        if [[ ${motdepassenouveau} == ${motdepassenouveau1} ]]; then
            mot=`densifier ${motdepassenouveau1}`
            sed -i -e "/^${utilisateur}/s/\(\.*\):\(\.*\):\(\.*\):\(\.*\):\(\.*\)/\1:'${mot}'\:\:\:\:\:\5/g" ~/proprescripts/rvsh/passwd
            echo DONE !
        else
            echo "Vous devez saisir le même mot de passe deux fois pour le confirmer !"
        fi
    elif [[ ${mot} != ${mot_reference} ]]; then
        echo "Votre mot de passe est incorrect !"
    fi
;;
    ♦ La commande « passwd »
```

❖ **La commande « finger »** permet de renvoyer des éléments complémentaires sur l'utilisateur. Cf. la commande « afinger » dans le mode ADMIN

Des éléments complémentaires sont renseignés dans le 5<sup>ème</sup> champ dans « passwd », on utilise la commande « cut » pour afficher l'information.

```
"finger" ) #echo mode finger
    cat ~/proprescripts/rvsh/passwd | grep "${utilisateur}" | cut -f5 -d':'
;;

```

❖ **La commande « write »** 📢 permet d'envoyer un message à un utilisateur connecté sur une machine du réseau. Avec la syntaxe :write nom\_utilisateur@nom\_machine message.

## Mode admin

- ❖ À partir de prompt « rvsh > » l'administrateur doit pouvoir exécuter les commandes du **mode connecter** certaines commandes complémentaires.
- ❖ La commande « host » permet d'ajouter ou d'enlever une machine au réseau virtuel.  
« host -a » permet d'ajouter alors que « host -d » permet d'enlever. Une interface a été développée avec les instructions. Puisque les informations des machines sont mémorisées dans ~ / proprescripts / rvsh / machine \_ liste, il suffit de modifier le fichier.

```
"host -a" )                                #Cette commande permet à l'administrateur
    echo host -a mode
    read -p "Le nom de la machine : " nom_machine
    echo -e "${nom_machine}" >> ~/proprescripts/rvsh/machine_liste
;;
"host -d" )                                #Cette commande permet à l'administrateur d'enlever
    echo "La liste de machines :"
    echo "====="
    cat ~/proprescripts/rvsh/machine_liste
    echo "====="
    read -p "Vous voulez supprimer : " nom_machine
    echo "====="
    sed -e '/^"${nom_machine}"/d' ~/proprescripts/rvsh/machine_liste
    echo "====="
    read -p "Vous êtes sur Y/N ?" oui
    case "${oui}" in
        Y | y )
            sed -i -e '/^"${nom_machine}"/d' ~/proprescripts/rvsh/machine_liste
            echo DONE !;;
        N | n )
            echo ANNULÉ !
            break;;
    esac
;;
;
```

- ❖ La commande « users » permet 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.

```
"users -a" )
    read -p "Le nom de l'utilisateur : " nom_utilisateur
    #echo -e "${nom_utilisateur}" >> ~/proprescripts/rvsh/passwd
    read -p "Définir un mot de passe : " mot
    #densifier ${mot}
    motencry=`densifier ${mot}`
    read -p "Définir un UID : " uid
    #cunzaifou ?
    read -p "Définir un GID : " gid
    echo "${nom_utilisateur}:${motencry}::${uid}::${gid}:gecos:shell" >> ~/proprescripts/rvsh/passwd
    #echo "${uid}" >> ~/proprescripts/rvsh/droit
    read -p "Des machines accessibles : " nom_machine
    echo "${uid}: ${nom_machine}" >> ~/proprescripts/rvsh/droit
;;
"users -d" )
    read -p "Vous voulez supprimer : " nom_utilisateur
    echo "====="
    sed -e '/^"${nom_utilisateur}"/d' ~/proprescripts/rvsh/passwd
    echo "====="
    uid_utilisateur=`get_uid ${nom_utilisateur}`
    #echo ${uid_utilisateur}
    sed -e '/^"${uid_utilisateur}"/d' ~/proprescripts/rvsh/droit
    echo "====="
    read -p "Vous êtes sur Y/N ?" oui
    case "${oui}" in
        Y | y )
            sed -i -e '/^"${nom_utilisateur}"/d' ~/proprescripts/rvsh/passwd
            sed -i -e '/^"${uid_utilisateur}"/d' ~/proprescripts/rvsh/droit
            echo DONE !;;
        N | n )
            echo ANNULÉ !
            break;;
    esac
;;
;
```

Semblable à le fichier /etc/passwd dans Linux, les informations sont enregistrées dans « passwd », chaque ligne a 7 champs: login:motdepasse:UID:GID:Gecos:homdir:shell, en la modifiant, on peut ajouter un utilisateur, et puis fixer leur mot de passe, en md5. Et quant à donner les droits, il y a un fichier nommé « droit » qui contient ces informations. En formats : UID:machine1:machine2...

- ❖ La commande « afinger » permet de renseigner les informations complémentaires sur l'utilisateur (l'utilisateur aura accès à ces informations avec la commande finger dans le mode connect).

Le fichier « passwd » contient cette information : dans le 5<sup>ème</sup> champ. On peut le modifier en utilisant la commande « sed ».

```
"afinger" )  
    read -p "Renseigner les informations complémentaires à QUI : " nom_utilisateur  
    read -p "Saisissez des informations : " informations  
    echo ""-----"  
    sed -e '/^!'"${nom_utilisateur}"'/s/(.*\):(.*\):(.*\):(.*\):(.*)\1:2\3\4:'"${informations}"':\5/g' ~/proprescripts/rvsh/passwd | grep "${nom_utilisateur}"  
    echo -----  
    read -p "Vous êtes sur Y/N ?" oui  
    case "${oui}" in  
        Y | y )  
            sed -i -e '/^!'"${nom_utilisateur}"'/s/(.*\):(.*\):(.*\):(.*\):(.*)\1:2\3\4:'"${informations}"':\5/g' ~/proprescripts/rvsh/passwd  
            echo DONE !;  
        N | n )  
            echo ANNULÉ !  
            break;  
    esac;
```

L'utilisation de « sed » et l'expression régulière a augmenté la complexité, mais c'est plus visible.

Remarque :

- ❖ Une fois une personne s'est connecté au réseau comme l'administrateur, bien qu'il utilise la commande « su » pour changer utilisateur, il n'a pas besoin de saisir le mot de passe de l'administrateur de nouveau. Ce qu'il doit faire est saisir le mot de passe de l'utilisateur qu'il veut changer.
- ❖ Et une fois une personne a été changé d'utilisateur en utilisant la commande « su », il a encore le droit d'utiliser les trois commandes complémentaires en monde ADMIN. C'est-à-dire il n'a plus de besoin de saisir le mot de passe de l'administrateur de nouveau.
- ❖ Quand un utilisateur a quitté le mode CONNECT par « Q » ou « q », toutes les informations enregistrées dans les fichiers « who\_liste », « rhost » et « users » vont être supprimées. Et de la même façon, dès qu'un administrateur a quitté la console, les informations du compte « root » vont également être supprimées.

# Sécurité

- ❖ Quand un utilisateur vérifie son mot de passe, il aura 10 fois de tentative, on a créé un compteur pour calculer des fois restes pour lui, après 10 fois d'échec, `rvsh` va être désactivé, l'utilisateur doit attendre 5 minutes.
- ❖ Pour vérifier le correct mot de passe, on a comparé directement le string, mais plus en sécurité, on a décidé d'utiliser la valeur MD5, grâce au <http://blog.51cto.com>, qui on permet de savoir la méthode pour la calculer (sur macOS). Ainsi, la probabilité d'irruption est rabattue et en conséquence le système est plus sécurisé.

```
j=10
while [[ 1 ]]; do
    read -p "Saisi Compteur ."
    touch temp
    echo -n ${motdepasse} | md5 > temp
    if [[ `cat temp` != "799e43071e08b15df1ec49110184ab4c" && $j != 0 ]]; then
        echo Désolé ! WRONG !
        ((j--))
        echo "Vous avez encore ${j} fois de tentative"
    elif [[ `cat temp` == "799e43071e08b15df1ec49110184ab4c" && $j != 0 ]]; then
        break
    elif [[ $j == 0 ]]; then
        echo rvsh est désactivé
        echo Réssavyez dans 5 minute
        sleep 300
        ((j=10))
    fi
done
```

Egale à :  
« mot de passe de l'administrateur »

(vérifier le mot de passe + compteur + la valeur md5)

Saisissez le code S.V.P.

(l'interface pour saisir le mot de passe)

```
== MODE ADMIN ==
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 9 fois de tentative
[Saisissez le code S.V.P.
Bienvenu ! Ce mode vous permet de vous connecter à une machine virtuelle.

-- Les commandes disponibles --
who      rusers      rhost      connect
su      passwd      finger      write
rvsh > ]
```

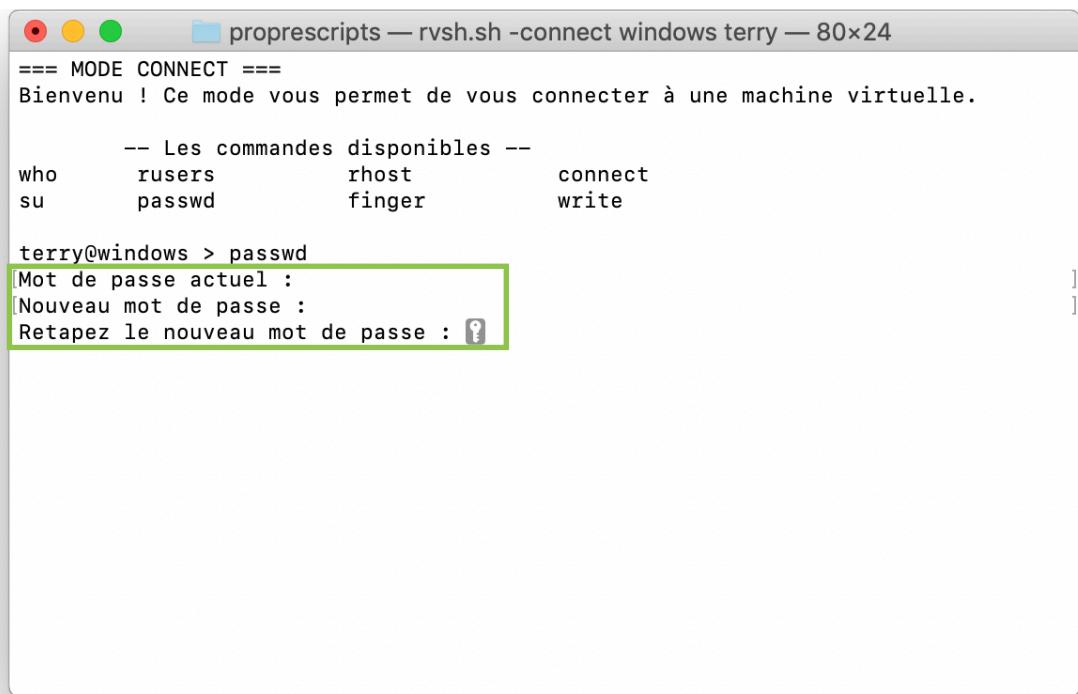
Gauche : Vérification du mot de passe réussie

```
== MODE ADMIN ==
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 9 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 8 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 7 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 6 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 5 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 4 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 3 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 2 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 1 fois de tentative
[Saisissez le code S.V.P.Désolé ! WRONG !
Vous avez encore 0 fois de tentative
[Saisissez le code S.V.P.rvsh est désactivé
Réssavyez dans 5 minute
```



Droite : Vérification du mot de passe échec + 11 tentatives

- 
- ❖ Avant qu'un utilisateur souhaite modifier son mot de passe, il doit d'abord confirmer que son mot de passe d'origine est correct, puis entrer le nouveau mot de passe et répéter pour confirmer.



```
proprescripts — rvsh.sh -connect windows terry — 80x24
==== MODE CONNECT ====
Bienvenu ! Ce mode vous permet de vous connecter à une machine virtuelle.

-- Les commandes disponibles --
who      rusers      rhost      connect
su      passwd      finger      write

terry@windows > passwd
Mot de passe actuel :
Nouveau mot de passe :
Retapez le nouveau mot de passe : ?
```

## Programmation multiThread

- ❖ Pour réaliser la commande « write », on a programmé en multiThread :

Dans « rvsh.sh » on appelle un script nommé `write.sh`, on le considère comme un processus différent. « `write.sh` » s'agit de lire et de traiter le fichier : `./rvsh/write`, qui contient des messages en forme de : `nom_utilisateur@nom_machine <texte>`.

The terminal window shows the content of the `rvsh.sh` script and its execution. The script uses `pidof` to find the process ID of `write.sh` and then runs it with the provided arguments. It also handles the case where the user wants to connect to another machine.

```
write.sh
1 pid_write=$$
2 #machine=$1
3 #utilisateur=$2
4
5 while [[ 1 ]]; do
6     cat ~/proprescripts/rvsh/write | grep "$2@$1 " | sed -e 's/^"$2@$1"' /Vous avez une message : /g'
7     sed -i -e '/^"$2@$1"/d' ~/proprescripts/rvsh/write
8     sleep 1
9 done
10

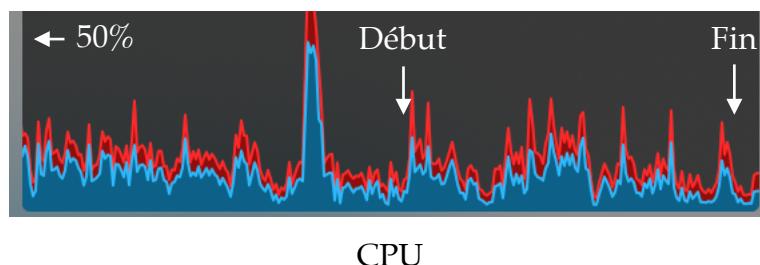
elif [[ $# == 3 && $1 == "-connect" ]]; then
    machine_verifier $2
    login $3 $2
    utilisateur=$3
    machine=$2
    bash ~/proprescripts/write.sh $2 $3 &
    pid_write=$!      # $! = le pid de "write.sh"
    clear
```

rvsh.sh

```
terry@mac ~ % write maxime@mac
Saisissez le message : Here is a message from terry@mac !
terry@mac ~ %
-----[redacted]-----
maxime@mac ~ % Vous avez une message : Here is a message from terry@mac !
```

- ❖ Et quant à l'occupation de la ressource et à la performance :

Bien que on ne peut pas le considérer comme parfait, on le trouve acceptable



---

## Référence

1. <http://igm.univ-mlv.fr/~masson/Teaching/PIM-INF3/shell.pdf>
2. <http://blog.51cto.com/ouziyou/1716945>
3. <https://www.cnblogs.com/lisqiong/p/5872804.html>
4. <https://www.cnblogs.com/i0ject/p/3658084.html>
5. <https://blog.csdn.net/gloriaied/article/details/80669390>
6. <https://www.frdic.com>
7. [https://www.google.com.hk/?gws\\_rd=ssl](https://www.google.com.hk/?gws_rd=ssl)
8. <http://www.mobibrw.com/2017/6029>
9. <https://blog.csdn.net/zly9923218/article/details/51855148>
10. <https://blog.csdn.net/weiyuefei/article/details/62216107>
11. <https://blog.csdn.net/lepton126/article/details/36374933>
12. <https://blog.csdn.net/classhao1/article/details/8182733>
13. <https://blog.csdn.net/jlds123/article/details/7427526>
14. <https://www.cnblogs.com/leezx/p/5589941.html>
15. <https://www.cnblogs.com/lovychen/p/6211209.html>
16. <http://www.mamicode.com/info-detail-2298188.html>
17. <https://blog.csdn.net/luo617/article/details/81221576>