LO14: RÉSEAU VIRTUEL DE MACHINES

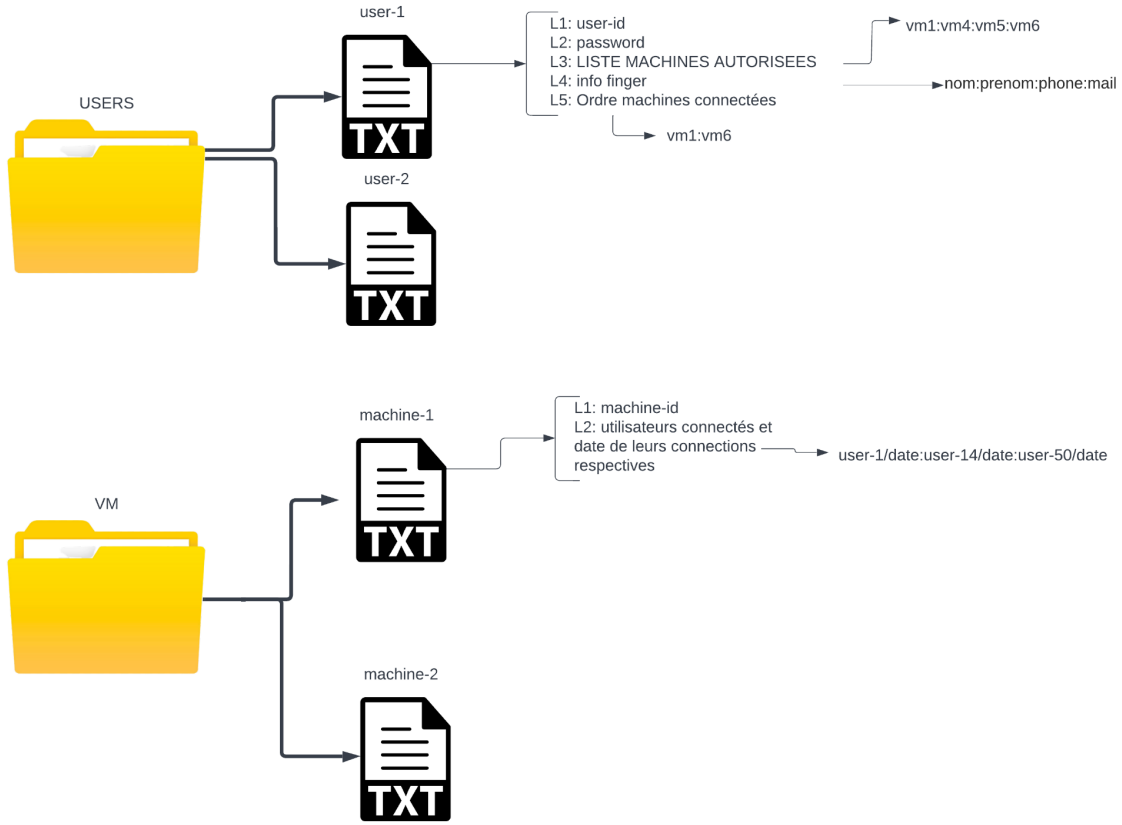
INTRODUCTION

xde la commande rvsh que nous allons créer en script bash. Cette commande aura deux modes de fonctionnement : Administrateur, pour gérer les machines existantes, en ajouter, en supprimer etc... L'autre mode sera le mode connect et sera destiné aux utilisateurs pour pouvoir accéder à leurs machines et exécuter les fonctions auxquels ils auront accès.

Ces machines étant virtuelles, nous pouvons donc les représenter sous la forme de notre choix, sous forme de variables par exemple. Nous avons décidé de les représenter sous forme de fichiers texte ce qui est selon nous le plus adéquat pour répondre à l'énoncé de projet. Dans ces fichiers textes, nous placerons une entête qui stockera les différents paramètres de la machine (nom de la machine, utilisateurs connectés, etc…). Nous aurons donc un fichier par machine.

Pour les utilisateurs, nous aurons également besoin de stocker leurs données (id, mot de passe, informations finger etc…). Pour cela, nous utiliserons également des fichiers texte respectant une syntaxe simple à utiliser au sein de notre programme. Nous aurons donc un fichier par utilisateur.

Afin d'obtenir un rendu cohérent nous aurons deux dossiers: users et vm contenant tous deux respectivement les fichiers de tous les utilisateurs existant et des machines existantes.



Le user-id est également le nom du fichier
Le vm-id est également le nom du fichier
Messagerie et messagerie admin ont ensuite été rajoutées lors du développement à la racine du projet.

MODE CONNECT

Pour accéder au mode connect l'utilisateur devra taper :
rvsh -connect nom_utilisateur nom_machine
Puis saisir son mot de passe.

Le mode connect permet à un utilisateur de se connecter à une machine grâce à son nom, son mot de passe et le nom de la machine. On utilisera une fonction afin de vérifier cela. Une fois connecté, il aura accès à une liste de 9 commandes qui seront notées en tant que fonctions dans le fichier "rvsh.bash".

L'utilisateur pourra changer de machine et quitter sa machine en cours lorsqu'il souhaitera.

Commandes :

who : Affiche qui est connecté sur la machine. Renvoie le nom, l'heure et la date de connexion de tous les utilisateurs.
Prend les informations dans la ligne 2 du document texte "vm-id" (correspondant à l'utilisateur ayant entré la commande)

rusers : Affiche qui est connecté sur le réseau. Renvoie le nom de chaque utilisateur, le nom de la machine sur laquelle il est connecté, l'heure et la date de la connexion.
Prend les informations dans chaque document texte "vm-id" sur la ligne 2.

rhost : Affiche la liste des machines présentes sur le réseau virtuel.
Liste le contenu du dossier VM.

Les commandes rconnect et su ont les mêmes effets, elles appellent user_connect qui établit la connexion. La seule différence est l'élément à changer : machine ou utilisateur. Dans les deux cas, on demande à nouveau le mot de passe.

rconnect : rconnect nom_machine
Cette commande imite la commande connect, vérifie que l'utilisateur a bien accès à la machine et le connecte dessus.
Vérifie le droit d'accès à la machine dans "user-id". Puis inscrit la connexion dans la ligne 2 de "vm-id".

su : su nom_utilisateur
Cette commande permettra de changer d'utilisateur. Elle vérifiera si l'utilisateur existe bien puis demandera le mot de passe du nouvel utilisateur.
Vérifie l'existence de l'utilisateur sur "user-id", puis vérifie que le mot de passe soit le bon. Et enfin, inscrit la connexion dans la ligne 2 de "vm-id".

passwd : Demandera un nouveau mot de passe à l'utilisateur afin de le changer. Il demande l'ancien mot de passe pour effectuer l'action (3 essais).

Modifie le mot de passe sur le fichier “user-id”, ligne 2.

finger : Permet à l’utilisateur de voir les informations entrées à propos de lui par l’administrateur.
Lis la ligne 4 de “user-id”.

write : write nom_utilisateur@nom_machine message.
Collecte les informations de l’utilisateur qui envoie, de sa machine, de celui qui reçoit, de sa machine et du message dans le document “messagerie”. Et dès qu’un utilisateur se connecte ou bien tape une commande le message lui est affiché dans son terminal.

exit : Permet à l’utilisateur de se reconnecter à la machine sur laquelle il était précédemment.
Lis la ligne 5 de “user-id” et se connecte à l’avant dernière machine de la liste en supprimant la dernière, user_connect gère la connexion.

MODE ADMIN

Pour accéder au mode admin, l'utilisateur devra taper :

rvsh -admin

Puis saisir le mot de passe admin.

Le mode admin permet d'accéder à toutes les commandes du mode connect et 4 commandes de plus, notées de la même façon que celles du mode connect.
L'administrateur ne se connecte pas sur une machine choisie mais sur la machine "hostroot" qui permet l'ajout/retrait de machines et d'utilisateurs du réseau virtuel.

Commandes:

host:

2 options possibles:

REMOVE: host -r + machine-id : déconnecte les utilisateurs connectés à la machine en question et la supprime.

ADD: host -a : crée un nouveau fichier de machine après avoir vérifié que le nom/id de la machine n'est pas déjà utilisé.

user:

RIGHTS: user -r user-id : Modifie le contenu de la ligne 3 (liste des machines autorisées) de l'utilisateur correspondant.

PASSWORD: user -p user-id : Modifie le contenu de la ligne 2 (mot de passe) de l'utilisateur correspondant. (semblable à passwd)

REMOVE: user -s user-id : Supprime le fichier de l'utilisateur correspondant ainsi que ses connexions courantes aux machines.

ADD: user -a : Crée un fichier dont le nom n'a pas encore été utilisé, son id lui est attribué. On demande ensuite à l'administrateur d'entrer le mot de passe de l'utilisateur, les informations de son profil (finger) ainsi que les machines sur lesquelles il pourra se connecter.

wall:

GLOBAL: wall -n message :
On récupère la liste de tous les utilisateurs dans le dossier "USERS". Ensuite on crée une ligne par utilisateur dans le document messagerie_admin sous la forme "nom_utilisateur Admin : message". Ensuite même fonctionnement que pour le write.

CLASSIQUE: wall message: Même fonctionnement que pour le -n sauf qu'il filtre les utilisateurs connectés en cherchant ceux ayant leurs noms figurant sur la ligne 2 des machines virtuelles pour n'envoyer le message qu'a eux.

afinger:

afinger user-id: ouvre une interface permettant de modifier chaque information contenues à la ligne 4 de l'utilisateur concerné.

Gestion du projet

Afin de réaliser correctement ce projet, nous avons passé un certain temps à bien cerner les différents objectifs afin de répondre au mieux au sujet, et de préparer un plan adapté de chaque élément pour éviter d'être bloqué sur un point majeur.
Nous avons décidé de passer au maximum par des fonctions, des fonctions "fonctionnelles" et d'autres "utilisateurs".

Les **fonctionnelles** avaient pour objectif de résoudre des tâches simples et réutilisables :

```
rmv_mot3 () {
  for i in ~/projet/USERS/*
  do
    sed -e 's/'$1'\://g' $i > temporair && mv temporair $i
  done
  rm -f temporair
}


# Ces deux fonctions servent à extraire les lignes passées en second argument :

user_ligne () {      # $1 = ID; $2 = ligne
  laligne=$(sed -n "$2 p" ~/projet/USERS/$1)
}


machine_ligne () {
  laligne=$(sed -n "$2 p" ~/projet/VM/$1)
}
```

Ici, ces fonctions n'ont pour seul objectif de lire les lignes d'un fichier ou encore d'en modifier le contenu.

Les fonctions **utilisateurs** quant à elles, servent à structurer le programme pour répondre aux requêtes des utilisateurs :

```
passwd (){

        echo "Nouveau mot de passe: "
        read new_mdp
        add_ligne $user 2 $new_mdp
        echo ok


}

finger (){

        user_ligne $user 4

        nom=$(echo $laligne | cut -d ':' -f1)
        prenom=$(echo $laligne | cut -d ':' -f2)
        phone=$(echo $laligne | cut -d ':' -f3)
        mail=$(echo $laligne | cut -d ':' -f4)
        message=$(echo $laligne | cut -d ':' -f5)


        echo " NOM : $nom  "
        echo " PRENOM : $prenom "
        echo " NUMERO DE TELEPHONE : $phone "
        echo " MAIL : $mail "
        echo " MESSAGE : $message  "

}
```

Ici, ces deux fonctions font appel à d'autres fonctions afin de changer un mot de passe ou lire des données utilisateurs.

Pour la répartition du travail, Hugo a travaillé sur la plupart des fonctions utilisateurs et la structuration du programme afin d'implémenter correctement les différents morceaux de code.
Thomas quant à lui, s'est occupé de la majorité des tâches fonctionnelles et d'une partie des fonctions utilisateurs. Son but était de simplifier et alléger les programmes en créant des fonctions pour éviter les répétitions.
Sur la fin, le binôme a travaillé ensemble dans le test et la résolution de problèmes du script.

Il était également important de bien communiquer afin de réaliser des fonctions qui fonctionnaient ensemble de manière cohérente, la suppression d'un utilisateur ou d'une machine devait impérativement être communiqué à travers le programme afin d'actualiser tous les fichiers nécessaires et garantir ainsi l'intégrité des données. Chacun des deux devait se mettre d'accord sur les arguments nécessaires en entrée et sortie de chaque fonction pour assurer un bon fonctionnement global.

CONCLUSION:

Hugo : Ce projet était intéressant car il a permis de mettre en pratique certains éléments étudiés en cours qui pouvait paraître plutôt abstrait. Le sujet de réseau virtuel m'a plu car nous avions la liberté de gérer nos machines et nos utilisateurs comme bon nous semble. Malgré quelques difficultés au début du développement du projet, il était d'un niveau plutôt abordable pour un étudiant ayant déjà fait un peu de programmation mais étant un débutant en bash. Ce qui est sure, c'est qu'en réalisant ce projet j'ai pu nettement progresser dans l'utilisation d'une machine Linux, nous avons été, je pense, confronter à la plupart des problèmes possibles (aussi bêtes soient-ils), comme par exemple un simple espace qui venait nous bloquer sur la progression. Mais au fur et à mesure de l'avancée du projet, nous étions de plus en plus efficaces afin de régler ces problèmes.
Pour conclure, ce projet m'a fait progresser et permis de mettre en application les notions de cours et de TD de LO14.

Thomas : Ce projet a été extrêmement intéressant à faire car il m'a permis de développer mes compétences en bash. Avoir un projet d'une telle ampleur et la liberté de l'exécuter comme bon me semble avec mon binôme était très instructif. Nous avons eu plusieurs méthodes de travail et nous avons vu nos compétences en programmation évoluer au fur et à mesure du projet, en découvrant de nouvelles façons de programmer certains éléments. Les moments les plus complexes de ce projet ont été son tout début où nous avons dû chercher les meilleures façons de réaliser le contenu demandé, façons auxquelles on s'est tenu tout du long de la programmation, ainsi que la toute fin où nous avons rencontré par mal de problèmes que nous avons dû résoudre. Le problème qui m'a le plus marqué était que lorsque l'input de l'utilisateur était vide, la fonction "su" se lançait sans raison avec des arguments au hasard, problème résolu en ayant mit un token.
Ce projet m'a beaucoup aidé et a été bien plus efficace que n'importe quelle autre méthode pour apprendre le bash, j'ai pu mettre en application tout ce que j'ai appris en LO14 et même en apprendre encore plus avec des recherches sur internet tout du long de la réalisation de ce projet.