TP d'Administration

TP2 : Terminal distant et premiers scripts BASH

Benoit Da Mota, Fabien Garreau

Janvier 2019

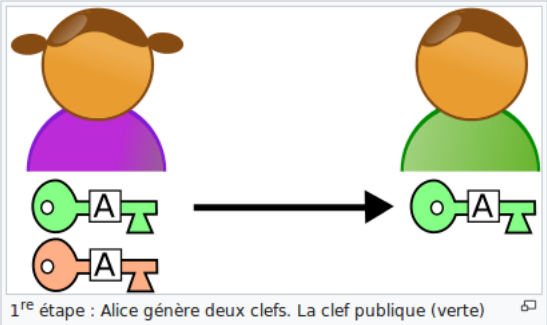
1 SSH : Secure SHELL

Dans de nombreuses situations, il est indispensable de se connecter à distance (réseau local ou internet) à une autre machine, un serveur web ou une machine de développement par exemple. Au département d'informatique de l'Université d'Angers, nous avons plusieurs machines dont certaines sont destinées à la formation des étudiants et au développement. Ainsi, vous possédez un compte (login + password) sur les machines `janus` et `forge`. Pour vous y connecter, il va falloir utiliser un SHELL sécurisé, `ssh`. Regardons la page du manuel :

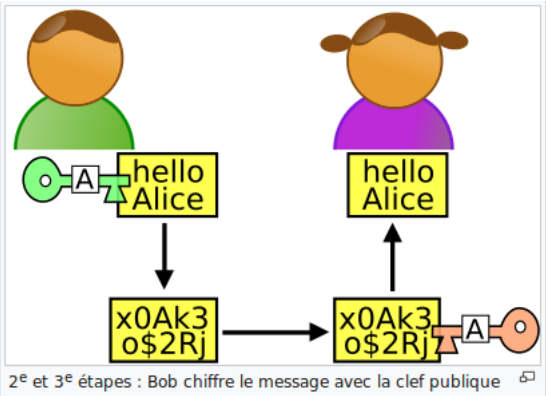
```
etudiant@25B:~$ man ssh
NAME
ssh - OpenSSH SSH client (remote login program)
SYNOPSIS
ssh [-1246AaCfgkMNnqsTtVvXxY ] [-b bind_address ] [-c cipher_spec ] -words [-D port ] [-e escape_char ] [-F configfile ]
     [-i identity_file ] [-L port host hostport ] [-l login_name ] [-m mac_spec ] [-o option ] -words [-p port ] [-R port
     host hostport ] [-S ctl ] [user @ hostname ] [command ]
DESCRIPTION
ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is
     intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an
     insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.
...
```

Le manuel fait quelques centaines de lignes et nous renseigne sur de nombreuses options et fonctionnalités. Comme pour de nombreuses commandes, il nous faudrait beaucoup de temps pour couvrir tous les aspects de SSH. La plupart du temps vous consulterez le manuel avec un objectif en tête. Vous pouvez également trouver des tutoriels ou blog posts sur internet, de l'aide sur les forums ou IRC. Les objectifs de cet exercice couvrent des besoins très courants : se connecter à une machine distante, lancer une commande sur une machine distante, transférer des fichiers entre deux machines, connexion en mode graphique (forward X11) et enfin, connexion sans mot de passe. Introduction à la cryptographie par clé publique, cf. Wikipedia :

`https://fr.wikipedia.org/wiki/Cryptographie_asym%C3%A9trique`



1re étape : Alice génère deux clefs. La clef publique (verte) qu'elle envoie à Bob et la clef privée (rouge) qu'elle conserve précieusement sans la divulguer à quiconque.



2e et 3e étapes : Bob chiffre le message avec la clef publique d'Alice et envoie le texte chiffré. Alice déchiffre le message grâce à sa clef privée.

Question 1.1 Connectez-vous à `janus`, lancez quelques commandes et déconnectez-vous.

Question 1.2 Vérifiez l'espace libre de votre home sur `janus` en une seule commande

Question 1.3 Copiez un fichier depuis votre machine vers votre home sur `janus`.

Question 1.4 Copiez un répertoire depuis votre machine vers votre home sur `janus`.

Question 1.5 Connecter-vous à `janus` en mode graphique et lancez une application en mode fenêtré.

Question 1.6 Générez une paire de clés cryptographiques RSA 2048 bits (par défaut) sans mot de passe.

Question 1.7 Configurez votre compte sur `janus` pour autoriser la connexion avec votre paire de clés. Il faut donc que `janus` « connaisse » votre clé publique.

Question 1.8 Pour vérifier l'absence de mot de passe, faites de nouveau les questions 1.1 à 1.3 (rappel : `history` ou `CTRL+R` pour `reverse-i-search`).

Conseils :

- regardez les manuels de `ssh`, `scp`, `ssh-keygen`
- cherchez à quoi correspond le fichier `$HOME/.ssh/authorized_keys`
- pensez à la communauté, par exemple : `http://doc.ubuntu-fr.org/ssh`

Question 1.9 Comment peut-on revenir à l'état initial, c'est-à-dire à la situation où la saisie d'un mot de passe est nécessaire pour se connecter à `janus` ?

2 Screen : un multiplexeur de terminaux

Nous avons pu voir en cours qu'un processus possède un `PID` et un `PPID`. Les commandes lancées depuis un Secure SHELL n'échappent pas à cette règle. Aussi, si le processus SSH se termine (déconnexion par exemple), les processus enfants seront aussi tués. Un multiplexeur de terminaux permet d'ouvrir plusieurs terminaux dans une même console, de passer de l'un à l'autre et de les récupérer plus tard, voire de les récupérer depuis une autre machine. Screen permet aussi de partager un terminal pour par exemple regarder ce que nous montre un autre utilisateur. Il existe d'autres multiplexeurs de terminaux comme `tmux`. Pour plus d'aide pour l'exercice `man screen` ou `http://doc.ubuntu-fr.org/screen`.

2.1 Exercice/démonstration

Question 2.1 Connectez-vous à `janus` et lancez `screen`.

Question 2.2 Tapez quelques commandes et fermez votre terminal.

Question 2.3 Reconnectez-vous à `janus`, listez les screens existants et reconnectez-vous au screen de la question 2.1. Que constatez-vous ?

Question 2.4 Quittez le screen, puis quittez `janus`.

3 Mes premiers scripts

Quelques exercices simples avec des variables, des tests et des boucles.

3.1 Exercice : saisie de valeur

```
#!/bin/bash

echo -e "\\n   \\t   On déclare la variable locale v1 \\n "
v1=quelque chose
echo -e "\\t La valeur de v1 est : $v1 \\n"

echo -e "\\t  Entrez la nouvelle valeur de v1 : "
read v1
if [[ -z $v1 ]]; then
    echo "La valeur de v1 est vide"
else
echo "Nouvelle valeur de v1 : $v1"
fi
```

Question 3.1.1 Exécutez le fichier `SC1.sh` ci-dessus qui comprend un exemple de script bash.

Question 3.1.2 Indiquez les variables utilisées dans ce script.

Question 3.1.3 Testez ce script.

Question 3.1.4 Modifiez ce script pour répéter la saisie tant que la variable `v1` est non vide.

3.2 Exercice : changement des permissions

Question 3.2.1 Écrivez un script bash rendant un fichier exécutable pour son propriétaire, où le nom du fichier est passé en paramètre au script.

Question 3.2.2 Améliorez le script précédent pour que l'utilisateur puisse préciser s'il veut changer le droit pour le propriétaire uniquement ou pour tout le monde, à l'aide d'un second argument.

3.3 Exercice : calcul du carré d'un entier

Question 3.3.1 Écrivez un script bash qui affiche le carré d'un entier passé en paramètre.

Question 3.3.2 Écrivez un script bash qui affiche la liste des carrés des entiers compris dans un intervalle dont les bornes sont passées en paramètre.

3.4 Exercice : affichage d'informations sur les fichiers

Question 3.4.1 Écrivez un script bash qui teste la nature de l'argument passé en paramètre, affiche son contenu et son nombre de lignes si c'est un fichier et affiche son contenu si c'est un répertoire.

3.5 Exercice : l'instruction case de bash

Question 3.5.1 Tapez et exécutez le script suivant.

```
#!/bin/bash

clear
essai=1 ;
while [[ 1 ]] ; do
  echo -e " \\n\\n\\n  Menu - Essai : $essai"
  echo " Afficher le répertoire courant          1 "
  echo " Lister les fichiers                     2 "
  echo " Informations sur un fichier             3 "
  echo " Changement de répertoire                4 "
  echo " n premières lignes d'un fichier         5 "
  echo " Sortie                                  0 "
  echo -n " Choix : "

  read choix;
  echo " "

  case $choix in
    1) pwd ;;
    2) ls ;;
    3) ;; # à compléter
    4) ;; # à compléter
    5) ;; # à compléter
    0) exit ;;
    *) echo "Merci de saisir un entier entre 0 et 5" ;;
  esac
  essai=$(( essai + 1 )) ;
done
```

Question 3.5.2 Expliquez le rôle de la variable `essai` en précisant son type.

Question 3.5.3 Programmez les choix 3, 4 et 5.