POLYTECH DIJON

TRAVAUX DIRIGES

SHELL

Scripts, scripts, scripts !

OBJECTIFS

Manipuler les différentes notions utiles dans la rédaction de scripts Shell :

- Paramètres et variables
- Alternatives et répétitives
- Expressions rationnelles

> **Note**
> Chaque question est accompagnée d'un temps indicatif pour sa réalisation. Si vous êtes hors délai, c'est qu'il vous faut travailler davantage ce module à la maison.

ECHAUFFEMENT

2 Paramètres [10 minutes]

➢ Créez un script qui prendra 2 paramètres en entrée et qui en affichera la valeur :

```
> ./script_1.sh param1 param2
Le parametre 1 = param1
Le parametre 2 = param2
```

➢ Modifiez le script précédent afin d'afficher une erreur si moins ou plus de deux paramètres sont fournis au script :

```
> ./script_1.sh param1 param2 param3
ERREUR : 3 parametres fournis, 2 attendus
```

N paramètres [10 minutes]

Quelque chose me dit que vous n'avez pas utilisez de boucle dans l'exercice précédent…

➢ Créez un nouveau script qui affichera la liste de tous les paramètres fournis, quel qu'en soit le nombre :

```
> ./script_2.sh param1 param2 param3 … paramN
N parametres recus :
Le parametre 1 = param1
Le parametre 2 = param2
…
Le parametre n = paramN
```

C'est plus ! C'est moins ! [10 minutes]

Un grand classique des exercices pour la prise en main d'un langage est le jeu du juste prix.

- Créez un script qui :
  - Génère un nombre aléatoire
  - Demande à l'utilisateur de saisir une valeur
  - Indique à l'utilisateur si la valeur saisie est supérieure ou inférieure au nombre généré
  - Si l'utilisateur trouve le nombre, le script s'arrête après un message de victoire, sinon le script demande la saisie d'une nouvelle valeur et on recommence jusqu'à ce que l'utilisateur trouve la solution.

*Astuce : la commande **read** demande à l'utilisateur de saisir une valeur et la stocke dans une variable :*

```
> read -p "Quel est votre nom ? " nomVariable
Quel est votre nom ? Michel
> echo $nomVariable
Michel
```

RegEx [15 minutes]

- Créez un script qui vérifie que le paramètre fourni est un numéro de téléphone respectant le format suivant : 5 groupes de 2 chiffres séparés par un espace.

Astuce : il est possible de vérifier qu'une chaine de caractères corresponde à une expression rationnelle grâce à l'opérateur =~

```
> [[ "1234" =~ ^[0-9]{4}$ ]] && echo "OK"
OK
```

- Créez un script qui vérifie que le paramètre fourni est une date valide au format « jj/mm/yyyy ».

SAUVEGARDE AUTOMATIQUE

Préparation [1 minute]

- Dans votre dossier personnel, créez un dossier « data » qui contiendra un fichier portant votre nom.

Archivage [15 minutes]

- Créez un script qui réalisera les opérations suivantes :
  1. Le script prendra en paramètre le chemin d'un dossier à archiver :

```
> ./nom_du_script chemin_du_dossier
```

  2. Le script testera si le chemin fourni en paramètre correspond bien à un dossier. Dans le cas contraire, il affichera une erreur et prendra fin.
  3. Puis le script créera une archive compressée au format « gz » du dossier fourni. L'archive sera enregistrée dans le dossier « /tmp ». Le nom de l'archive sera construit sur le modèle suivant :

backup_[votre_ip]_[année]-[mois]-[jour]_[heure]-[minute]-[seconde].tar.gz

- Testez votre script avec le dossier « data » créé précédemment et vérifiez que l'archive est convenablement créée (décompressez-la pour contrôler son contenu).

SCP [8 minutes]

SCP (Secure CoPy) est une commande permettant de transférer des documents entre deux ordinateurs en utilisant les protocoles RCP (Remote Copy Protocol) et SSH (Secure SHell).

La commande « scp » s'utilise de la manière suivante :

```
> scp source username@hostname:target
```

où :

- **source** est le chemin du fichier ou du dossier à transférer (pensez à l'option -r pour un dossier)
- **username** est le nom de l'utilisateur avec lequel se connecter sur la machine distante
- **hostname** est l'adresse IP ou le nom de la machine distante sur le réseau
- **target** est le chemin du dossier distant dans lequel copier la source

- L'adresse IP de la machine de stockage des sauvegardes est **51.68.94.83**
- Copiez l'archive créée précédemment sur la machine virtuelle de stockage à l'aide de la commande scp.
  - L'utilisateur à utiliser sur la machine distante est :
    - Utilisateur : esistock
    - Mot de passe : esistock

- Le dossier cible est « backups »

➢ Connectez-vous en SSH à la machine de stockage avec les identifiants précédents et vérifiez que votre archive est bien présente dans le dossier « /home/esistock/backups »

Envoi par script [5 minutes]

➢ Modifiez votre script précédent afin que l'archive générée soit transférée sur la machine de stockage puis supprimée du dossier « /temp » de la machine locale.
➢ Testez le bon fonctionnement de votre script.
➢ Peut-on exécuter ce script avec une tâche planifiée ? Justifiez votre réponse.

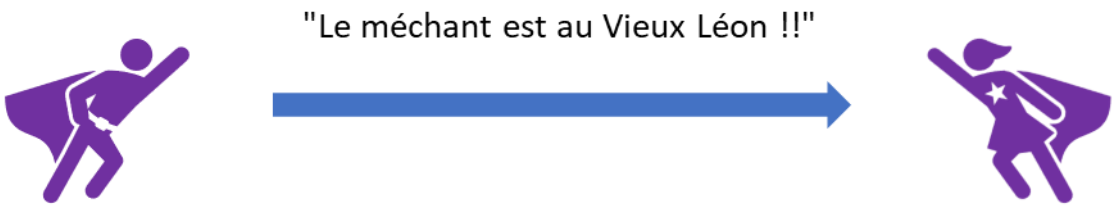
Chiffrement asymétrique [5 minutes]

Jusque-là, pour vous connecter à une machine distante en SSH, vous avez utilisé une authentification par login / mot de passe. C'est une méthode classique et tout à fait fonctionnelle… pour un humain. Mais pour un script, c'est plus compliqué, étant donné qu'il n'a pas de petits doigts pour saisir son mot de passe au clavier.

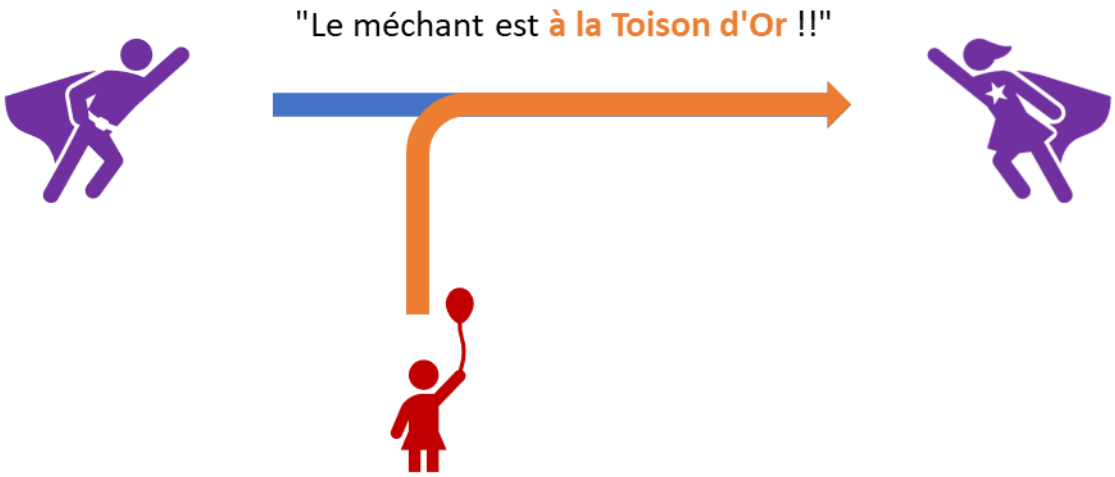
Heureusement, il existe une autre méthode qui consiste à utiliser une clé SSH qui repose sur le principe des clés privées / clés publiques, autrement appelé cryptographie asymétrique. Peut-être que quelques explications s'imposent. Prenons un exemple :

Mohammed, dit Super Momo, doit échanger des informations avec Lise, dite Super Lili, via un canal de communication numérique.

Solution n°1 : Super Momo peut envoyer un simple message à Super Lili



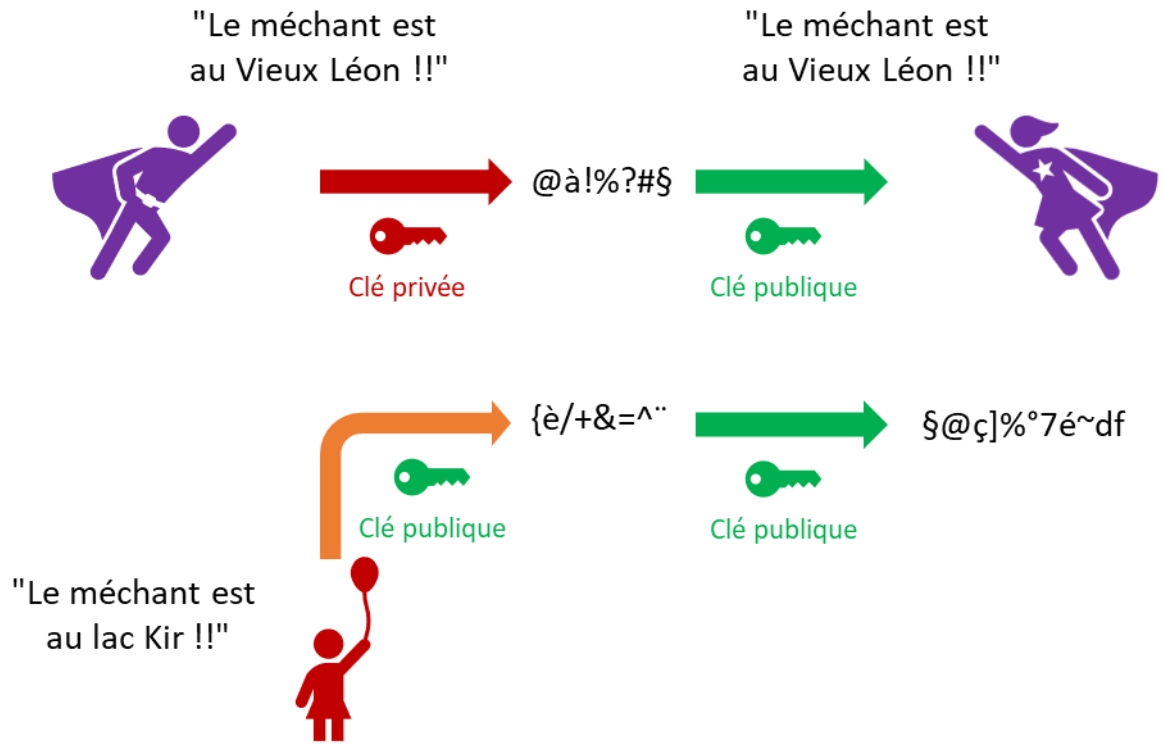
Problème : Un imitateur peu facilement intercepter le message de Super Momo et en modifier le contenu sans que Super Lili puisse s'en rendre compte.



Solution n°2 : utiliser une clé privée et une clé publique.

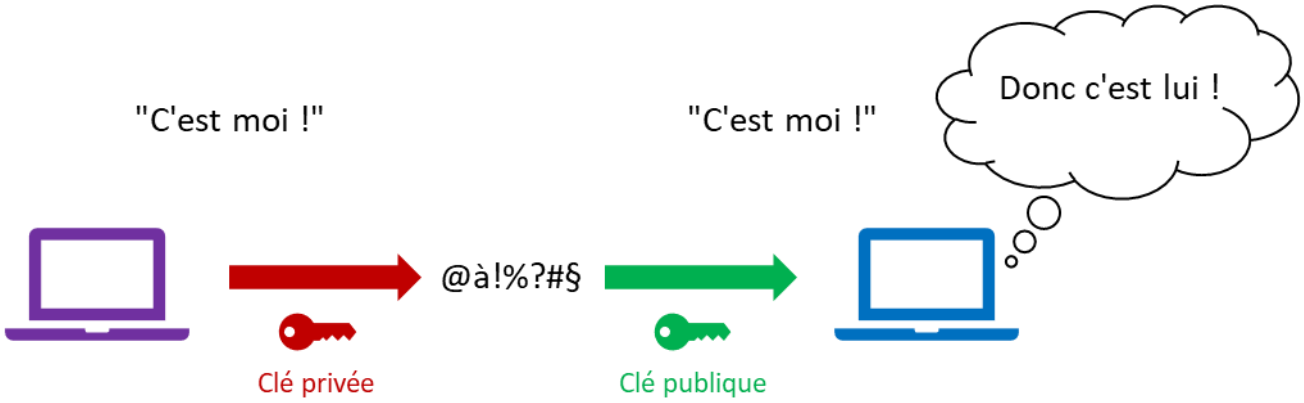
Super Momo conserve précieusement sa clé privée et l'utilise pour chiffrer ses messages. Par ailleurs, il transmet à Super Lili une clé publique qui permet de déchiffrer les messages chiffrés avec la clé privée de Super Momo. En revanche, et c'est là que l'astuce réside, la clé publique ne permet pas de rechiffrer le message.

Ainsi, même si un Super Vilain (ou un enfant, c'est sensiblement la même chose) parvenait à intercepter la clé publique de Super Momo, il ne pourrait pas envoyer d'information à Super Lili sans que celle-ci ne se doute de quelque chose.



Connexion par clé SSH [2 minutes]

Nous allons utiliser la même technique pour nous connecter à la machine distante. Au lieu de lui fournir un login et un mot de passe, nous allons lui transmettre une information chiffrée à l'aide d'une clé privée que nous serons les seuls à détenir. Nous aurons transmis préalablement le clé publique correspondante à la machine distante afin que cette dernière puisse vérifier que l'information transmise vient bien de nous.



Création de la clé SSH [10 minutes]

- Commencez par mettre à jour le gestionnaire de paquets « apt » :

```
> sudo apt update
```

- Installez l'application « openssh-client »

```
> sudo apt install openssh-client
```

- Créez un dossier « keys » dans votre dossier personnel
- Vous pouvez à présent générer une clé SSH avec l'algorithme RSA sur 2048 bits :

```
> ssh-keygen -t rsa -b 2048
```

La clé devra être stockée dans le fichier « backup_key » dans le dossier « keys » précédemment créé.
Lorsqu'il vous sera demandé de saisir une passphrase, laissez le champ vide.

- Vérifiez que la clé a bien été créée dans le dossier « keys ». Vous devez également y trouver un fichier « backup_key.pub » qui correspond à la clé publique générée à partir de la clé privée.

Transmettre la clé publique [5 minutes]

Afin que la machine distante puisse vous identifier, il faut que cette dernière ait en sa possession la clé publique correspondant à votre clé privée.

- Transmettez la clé publique à la machine distante :

```
> ssh-copy-id -i path/to/the/public_key.pub user@hostname
```

Configurer la méthode de connexion [5 minutes]

Il faut à présent indiquer à votre machine locale qu'elle doit utiliser votre clé privée pour se connecter à d'autres machines en SSH.

- Créez le fichier de configuration SSH de votre utilisateur :

```
> nano ~/.ssh/config
```

- Ajoutez les lignes suivantes :

```
Host remote_ip_address
IdentityFile path/to/the/private_key
```

- Sécurisez la configuration :
  - Le propriétaire des clés ne doit avoir que les droits de lecture sur ces fichiers. Les autres utilisateurs, groupe compris, ne doivent avoir aucun droit.
  - Le propriétaire du fichier config doit avoir les droits en lecture et en écriture seulement. Les autres utilisateurs, groupe compris, ne doivent avoir que les droits en lecture.

Tester la connexion [5 minutes]

- Essayez de vous connecter à la machine distante en SSH. Aucun login ou mot de passe ne devrait vous être demandé.
- Essayez à présent de relancer votre script.

Planification [10 minutes]

- Créez une tâche planifiée (cron) qui exécutera votre script toutes les minutes.
- Vérifiez sur la machine distante que vos backups sont bien transférés toutes les minutes.
- Modifiez la tâche cron pour qu'elle ne s'exécute qu'une fois par jour, à deux heures du matin.