TP - ASR7 Programmation Concurrente

—

Administration Système – GNU/Linux

Matthieu Moy, Fabien Rico, Adil Khalfa

Printemps 2018

En cours, on vous a demandé de lire les URLs suivantes en préparation de ce TP :
— `https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.fr.html`
— `https://www.debian.org/doc/manuals/debian-faq/ch-pkgtools.fr.html`
— `https://www.debian.org/doc/manuals/debian-faq/ch-uptodate.fr.html`
Elles contiennent les principales informations *théoriques* qu'il vaut mieux retenir... ;)

I Administration Système

Où il est « intéressant » de connaître les commandes `top source alias less tail head ps grep whoami chown chgrp passwd adduser addgroup sudo su lsmod modprobe apt-get aptitude dpkg man` et certaines de leurs options.

I.1 Les utilisateurs et root

Q.I.1) - Avant tout, connectez-vous (ssh) sur la VM OpenStack que vous avez créée la dernière fois, en tant qu'utilisateur `chaprot` (reprenez l'énoncé du TP4 si besoin).

Q.I.2) - Sur votre VM, créez un autre utilisateur de login `lurong` (nom complet : Gai Luron), et donnez-lui un mot de passe (rappel : la commande `adduser` est plus pratique que `useradd` ...).

```
$ sudo adduser lurong
[sudo] password for chaprot:
Adding user ‘lurong’ ...
Adding new group ‘lurong’ (1000) ...
Adding new user ‘lurong’ (1000) with group ‘lurong’ ...
Creating home directory ‘/home/lurong’ ...
Copying files from ‘/etc/skel’ ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for lurong
Enter the new value, or press ENTER for the default
```

```
            Full Name []: Gai Luron
            Room Number []:
            Work Phone []:
            Home Phone []:
            Other []:
Is the information correct? [Y/n] y
```

Q.I.3) - Depuis un shell appartenant à l'utilisateur `chaprot`, ouvrez un shell en tant qu'utilisateur `lurong`. Fermez ce shell pour revenir au shell de `chaprot`.

Depuis un shell existant, on peut entrer :

```
$ su - lurong
Password:
```

Pour sortir du shell, Control-D, ou `exit`.

Q.I.4) - Depuis votre PC physique, ouvrez un nouveau terminal (pour conserver le shell `chaprot` ouvert), et dans ce nouveau terminal ouvrez une connexion sur votre VM en vous connectant directement en tant qu'utilisateur `lurong`.

Au choix, `ssh lurong@IP_VM` ou bien `ssh IP_VM -l lurong`.

Q.I.5) - Regardez le contenu du fichier `/etc/passwd` (il est accessible en lecture pour tout le monde, vous pouvez voir son contenu avec une commande comme `less` ou `cat`, ou l'ouvrir dans votre éditeur de texte). Vous devriez trouver une ligne pour l'utilisateur `lurong` contenant entre autres le nom complet que vous avez entré à la création, et d'autres informations qui ont été ajoutées automatiquement comme le répertoire personnel `/home/lurong` et le shell par défaut `/bin/bash`. Notez que le mot de passe ne se trouve pas dans ce fichier (il y a un `x` dans la colonne où il aurait pu se trouver, qui signifie qu'il est stocké ailleurs).

Q.I.6) - Regardez maintenant le contenu du fichier `/etc/shadow`. Celui-ci n'est accessible que par `root`. Dans ce fichier, il y a bien une information sur le mot de passe, mais pas le mot de passe lui-même. Pourquoi ?

Le fichier contient un hash du mot de passe. Une explication sur le format de stockage est disponible ici : `https://www.aychedee.com/2012/03/14/etc_shadow-password-hash-formats/`.

Q.I.7) - Quel est l'`UID` (User IDentifier, le nombre qui identifie l'utilisateur de manière unique) de l'utilisateur `lurong` ?

Soit avec `id lurong`, soit la 3ème colonne de `/etc/passwd`.

Q.I.8) - Comment connaître les groupes auxquels il appartient ?

Soit avec `groups lurong` soit dans le fichier `/etc/group`.

I.2 Configuration du shell bash

Avant de commencer, un peu de vocabulaire :

— Un *shell* est un programme qui vous permet de lancer d'autres programmes. Dans ce cours on considère les shells Unix textuels, donc le rôle se résume à exécuter, en boucle, les étapes :

1. Afficher *l'invite de commande* (« *prompt* »)
2. Lire une ligne de commande au clavier
3. Exécuter la commande

— `bash` est le nom du shell Unix le plus utilisé aujourd'hui. D'autres shells sont `zsh` (qui a plus de fonctionalités), `ksh` qui est un ancêtre de `bash`, `tcsh`, ...

— Un *terminal* est un dispositif permettant d'accéder à un ordinateur. Historiquement, un terminal a d'abord été un ensemble clavier + écran (voire machine à écrire). Dans ce document, nous utilisons le mot « terminal » pour « terminal virtuel », c'est à dire une fenêtre graphique dans laquelle l'ordinateur lit et affiche du texte. En général, on ouvre un terminal pour exécuter un shell dedans (c'est fait par défaut).

— `xterm`, `gnome-terminal`, `konsole` ou `rxvt` sont des noms de terminaux classiques sous Linux.

I.2.1 Les variables d'environnement

Q.I.9) - En tant qu'utilisateur `chaprot`, exécutez les commandes :

```
TOTO=tutu
echo "$TOTO"
```

La première ligne crée une variable du shell nomée `TOTO` et de valeur tutu. La seconde affiche la valeur de la variable `TOTO` (donc la chaîne de caractères « tutu »).

Q.I.10) - Pour vous convaincre que les espaces sont significatifs en shell, essayez aussi :

```
TOTO = tutu
```

Ça ne marche pas (tentative d'exécuter la commande `TOTO` avec les arguments `=` et `tutu`).

Q.I.11) - Lancez un nouveau shell avec la commande `bash`. Vous aurez peut-être l'impression que rien ne s'est passé, en réalité :

— Le premier shell a créé un nouveau processus (primitive `fork()`)
— Le nouveau processus exécute la commande `bash` (primitive `exec()`)
— Le premier shell attend que le second termine (primitive `waitpid()`)

Les commandes que vous entrez maintenant sont donc exécutées par le nouveau shell.

Q.I.12) - Exécutez `echo "$TOTO"`. La commande devrait afficher une chaîne vide. Pourquoi ?

La variable `TOTO` est définie dans le premier shell, mais elle n'est pas héritée par le second (techniquement, l'appel à `exec()` réinitialise les variables).

Q.I.13) - Quittez le second shell (`exit`), et vérifiez que `echo "$TOTO"` affiche bien `tutu` maintenant que vous êtes revenu au premier shell.

Q.I.14) - Exécutez la commande `export TOTO` pour transformer `TOTO` en variable d'environnement. Une variable d'environnement est transmise aux processus fils (contrairement à ce qui vient de se passer quand nous avons affiché la valeur de `TOTO` dans le deuxième shell ci-dessus). Trouvez un moyen de vérifier que c'est bien le cas.

Il suffit de refaire la même manipulation que ci-dessus :

```
bash
echo "$TOTO"
```

Cette fois-ci la valeur tutu devrait être affichée.

Q.I.15) - La commande `env` permet d'afficher les variables d'environnement du shell courant. Vérifiez que vous retrouvez bien la définition de `TOTO`.

En résumé :
- `VARIABLE=valeur` : affectation de valeur à la `VARIABLE`.
- `echo "$VARIABLE"` : Affichage de la valeur de la variable.
- `export VARIABLE` : faire de `VARIABLE` une variable d'environnement.
- `export VARIABLE=valeur` : raccourcis pour `VARIABLE=valeur; export VARIABLE`.

I.2.2 La variable d'environnement `$PATH`

La variable d'environnement `$PATH` définit les répertoires dans lesquels le système va chercher les exécutables quand on lance une commande. C'est une liste de répertoires séparés par des « deux points » (`:`).

Q.I.16) - Affichez le contenu de cette variable. Vérifiez que les répertoires classiques `/bin/`, `/usr/bin/` s'y trouvent.

Q.I.17) - Exécutez la commande `PATH="$HOME"/test1:"$PATH":"$HOME"/test2` puis affichez le nouveau contenu de la variable `$PATH`.

Q.I.18) - Créez les répertoires `"$HOME"/test1` et `"$HOME"/test2`.

Q.I.19) - Créez un fichier `"$HOME"/test1/less` contenant :

```
#!/bin/sh
echo "Je suis test1"
```

Q.I.20) - Créez un fichier `"$HOME"/test2/less` contenant :

```
#!/bin/sh
echo "Je suis test2"
```

Q.I.21) - Lancez la commande `command -v less` pour voir quelle commande sera exécutée quand vous entrerez `less` en ligne de commande. Vérifiez que la commande se comporte comme prévu. Pour l'instant, les deux fichiers que nous avons créé ne sont pas considérés comme des exécutables car les fichiers n'ont pas le droit `x` (eXecutable).

Q.I.22) - Lancez la commande `chmod +x "$HOME"/test1/less "$HOME"/test2/less`. Ré-essayez les commandes `command -v less` et `less`. Si vous ne voyez pas de changement par rapport à la question précédente, exécutez la commande `hash -r`. Que remarquez-vous ?

Cette fois-ci, les fichiers sont exécutables. La commande `less` sera cherchée dans `"$HOME"/test1/less` puis dans les répertoires du système dont `/usr/bin` où se trouve la commande `less` par défaut, et enfin dans `"$HOME"/test2/less`. Comme la commande est trouvée dans `"$HOME"/test1/less`, la recherche s'arrête et notre premier script est exécuté. Vous devez donc voir :

```
$ command -v less
/home/chaprot/test1/less
$ less
Je suis test1
```

Q.I.23) - Fermez votre shell et ouvrez-en un nouveau. Ré-essayez les commandes `command -v less` et `less`. Que remarquez-vous ?

La variable `$PATH` que nous avions positionné n'est pas persistante d'un shell à l'autre. Notre nouveau shell a donc repris la valeur par défaut qui n'inclue pas les répertoires `"$HOME"/test*/less`.

I.2.3 Les fichiers de configuration : `.bashrc`, `.bash_profile`

Les manipulations que nous avons faites jusqu'ici étaient temporaires : leur effet était limité au shell courant. La plupart du temps, quand on modifie une variable d'environnement (comme `$PATH`), on souhaite que la modification soit persistante et que tous les shells ouverts dans le futur aient la nouvelle configuration automatiquement. Pour cela, nous allons entrer les commandes comme `PATH=...` non pas dans la ligne de commande interactive, mais dans des scripts qui seront exécutés à chaque démarrages du shell. Pour bash, ces fichiers sont :

- `~/.bashrc` : fichier exécuté au démarrage du shell
- `~/.bash_profile` : fichier exécuté au démarrage des shells « login » (c'est à dire quand un shell est ouvert suite à une entrée de nom d'utilisateur et mot de passe, par exemple suite à une connexion SSH). Sur certains systèmes (comme Mac OS X), ouvrir un terminal lance un shell « login ».

En pratique, la plupart des éléments de configurations doivent être les mêmes pour les deux types de shell, donc nous allons nous arranger pour que `~/.bash_profile` inclue automatiquemnt `~/.bashrc`.

Les manipulations ci-dessous sont potentiellement dangereuses. Faites les sur le compte `lurong` sur votre VM. Si vous voulez travailler sur votre machine personnelle, assurez-vous que vous savez ce que vous faites à chaque étape.

Q.I.24) - Connectez-vous sur le compte `lurong`. Faites toutes les manipulations ci-dessous sur ce compte.

Q.I.25) - Pour repartir à zéro, supprimez les fichiers `.bashrc` et `.bash_profile` (ATTENTION : à ne pas faire sur votre compte).

Q.I.26) - Créez le fichier `.bashrc` avec le contenu :

```
echo "chargement de .bashrc"
```

Q.I.27) - Créez le fichier `.bash_profile` avec le contenu :

```
echo "chargement de .bash_profile"
```

Q.I.28) - Ouvrez une nouvelle connexion en tant que `lurong` via SSH. Quel fichier est chargé ?

Q.I.29) - Depuis un shell existant, relancez un shell avec la commande `bash`. Quel fichier est chargé ?

Q.I.30) - Ajoutez les lignes suivantes au fichier `.bash_profile` :

```
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

La signification de ces lignes est : « si le fichier `~/.bashrc` existe, alors le charger ». C'est ce qui nous permettra d'écrire la configuration de l'utilisateur dans `~/.bashrc`, qui sera chargé quoi qu'il arrive.

Q.I.31) - Une proposition de configuration commune à tous les utilisateurs est disponible dans `/etc/profile`. Ajoutez les lignes suivantes en début de `~/.bashrc` pour en bénéficier :

```
if [ -f /etc/profile ]; then
    . /etc/profile
fi
```

Q.I.32) - Refaites les manipulations ci-dessus pour ouvrir des shells.

Q.I.33) - Ajoutez la ligne suivante au fichier `~/.bashrc` :

```
PATH="$HOME"/bin:"$PATH"
```

Q.I.34) - Créez le répertoire `~/bin` et placez-y un ou plusieurs fichiers exécutables (par exemple un script, comme nous l'avons fait ci-dessus pour `~/test*/less`).

Q.I.35) - Exécutez ces exécutables en entrant simplement leur nom. Vous aurez probablement besoin de relancer un shell pour que la modification du `$PATH` soit prise en compte.

> Petite astuce pour relancer un shell : `exec bash`.

Q.I.36) - Supprimez les lignes `echo "..."` des deux fichiers : ces lignes étaient là pour nous aider à comprendre mais c'est une très mauvaise idée de produire du texte sur la sortie standard dans un de ces fichiers.

I.2.4 Un peu de confort et de couleurs !

La variable PS1 contient une chaîne de caractère qui est affichée comme invite de commande (prompt).

Q.I.37) - Essayez par exemple :

```
PS1="Bonjour maitre, que dois-je faire ? "
```

Q.I.38) - Essayez d'autres valeurs pour `$PS1` comme :

```
PS1="\h:\w\\\$ "
PS1="\[\e[31m\]rouge\[\e[m\] \[\e[32m\]vert\[\e[m\] "
PS1="\[\033]0;\u@\h:\w\007\]\[\033[01;32m\]\u@\h\[\033[01;34m\] \w \$\[\033[00m\] "
```

Q.I.39) - N'y passez pas de temps maintenant mais vous pourrez plus tard générer un joli prompt personnalisé avec un outil comme `http://ezprompt.net/`.

Q.I.40) - Faites en sorte que l'invite de commande soit différente sur les utilisateurs `chaprot`, `lurong` et sur votre compte habituel Lyon 1. Vous pourrez ainsi voir d'un coup d'œil quel terminal correspond à quel utilisateur. Faites en sorte et vérifiez que ces changements sont persistants d'une session à l'autre en ouvrant un nouveau shell sur chaque utilisateur.

La complétion dite « intelligente » permet, lorsqu'on appuie sur la touche tabulation (TAB) pendant qu'on entre une commande, de fournir une complétion dépendante de la commande et du contexte. Par exemple, `ls` *[TAB]* (si besoin, répétez *[TAB]* une deuxième fois) proposera des noms de fichiers, alors que `ls --`*[TAB]* proposera les options de la commande `ls`. Pour l'activer il suffit d'inclure le fichier `/etc/bash_completion` dans un fichier d'initialisation du shell. C'est souvent le cas par défaut, mais si ce n'est pas le cas il suffit d'ajouter ces lignes au `~/.bashrc` :

```
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
fi
```

Q.I.41) - Mettez en place également la complétion "intelligente". Vérifiez avec l'exemple ci-dessus (ls) qu'elle fonctionne.

> Il faut s'assurer que le package `bash-completion` est installé, et faire `source /etc/bash_completion`. Attention la façon de procéder peut être différente sur d'autres distributions (Gentoo..)

Q.I.42) - Avec `git` *[TAB]*, regardez la liste des commandes git disponibles. Comment entrer la commande `git commit --amend` en appuyant seulement sur 13 touches (pour 18 caractères) ?

> `git com[TAB]--am[TAB]`

I.3 Utilisateurs et gestion des droits

Q.I.43) - Arrangez-vous pour avoir deux terminaux ouverts, l'un avec un shell appartenant à `chaprot`, l'autre avec un shell `lurong`. Si vous avez bien suivi les consignes ci-dessus, vous l'avez déjà fait. Vous devriez aussi avoir des invites de commandes différentes dans ces deux terminaux.

Q.I.44) - Dans les deux terminaux, vérifiez que vous êtes bien celui que vous pensez à l'aide de la commande `whoami`.

Q.I.45) - Vérifiez que chaque utilisateur se trouve, pour l'instant, dans son propre répertoire personnel (commande `pwd`).

Q.I.46) - Dans le shell sur la VM, sous l'identité `chaprot`, créez un fichier avec la commande `touch poi` et exécutez `ls -l`. Modifiez ce fichier avec la commande de votre choix.

Q.I.47) - Dans le shell sous l'identité `lurong`, retrouvez le fichier `poi`. Faites un `ls -l` dessus et regardez son contenu. Essayez de le modifier. Que se passe-t-il ?

Le fichier n'est pas accessible en écriture, donc on peut faire `ls -l`, regarder le contenu mais pas l'éditer.

Q.I.48) - Revenez sous le shell de `chaprot` et exécutez la commande `chmod go+w poi`. Ré-essayez de modifier le fichier en tant que `lurong` (cela devrait marcher maintenant). Faites un `chmod go-w poi` en tant que `chaprot`, et vérifier que la commande a bien coupé les droits en écriture.

Q.I.49) - Exécutez `chmod go-r poi` en tant que `chaprot` et vérifiez que les droits en lecture ont été coupés. Exécutez `chmod 644 poi` en tant que `chaprot` pour revenir à la situation précédente.

Explication : `chmod go+w` = pour Group et Other, ajouter (+) le droit Write. `chmod go-w` = idem, mais supprime (-). `chmod 644` : notation octale (6 = 4 + 2 = read + write, 4 = read).

Q.I.50) - en tant que `lurong`, exécutez `chmod go+rw poi`. Vous devriez obtenir une erreur : vous n'êtes pas propriétaire du fichier, vous n'avez pas le droit de modifier ses permissions.

Q.I.51) - Quels sont les utilisateurs actuellement connectés sur la machine (locale, distante) ?

Il faut utiliser la commande `w`, ou la commande `who` qui font presque la même chose

I.4 Les packages et leur gestion

Q.I.52) - Quelle est la liste complètes des packages installés ?

`dpkg --list` (ou `dpkg --get-selection`)

Q.I.53) - Faire une mise-à-jour de la base des packages.

```
sudo apt update
```

Q.I.54) - Démarrer une mise-à-jour du système (pour ne pas perdre de temps vous pouvez annuler la mise à jour en répondant « n » à la dernière question).

```
sudo apt upgrade
```

Nous allons maintenant installer un serveur web sur votre VM. Un serveur web très connu est apache2, mais nous allons jouer dans un premier temps avec Nginx (prononcer « haine-gi-nex », comme « engine-X » en anglais).

Q.I.55) - Comment savoir si Nginx est installé ?

`apt policy nginx` ou bien `dpkg -l nginx` qui doit contenir « ii ». Mais nécessite de répondre à question suivante d'abord. **Ne pas oublier d' utiliser la tabulation intelligente quand c'est possible !**

Q.I.56) - Quel est le package concernant nginx ?

```
apt search nginx
```

Q.I.57) - Voir si nginx est installé.
Quelle version serait installée si on décidait d'installer nginx ?

```
apt show nginx | grep Version
```

Q.I.58) - Installer le package `nginx`

```
sudo apt install nginx
```

Q.I.59) - Quels sont les fichiers installés par nginx ?

`dpkg -L nginx` ou l'option longue : `dpkg --listfiles nginx`, mais en réalité la majorité des fichiers sont installés par les dépendances du package, surtout `nginx-common`

Q.I.60) - Voir les fichiers de configuration de nginx.

`dpkg --listfiles nginx-common | grep etc`, principalement dans `/etc/nginx/`

Q.I.61) - Où sont les fichiers de log de nginx

```
/var/log/nginx/
```

I.5 Gestion des services avec systemd

Nginx est un logiciel prévu pour tourner en tâche de fond sur la machine. Il est lancé au démarrage et tourne en permanence même si personne n'est connecté à la machine. On appelle cela un *démon* (daemon en anglais). Son rôle est de répondre aux requêtes HTTP reçues, donc c'est aussi un *serveur*. Ce serveur fournit un *service* aux clients.

Q.I.62) - Vérifiez que votre machine répond bien quand un navigateur web l'interroge sur le port 80. Plusieurs solutions :

- Lancer votre navigateur web habituel sur `http://IP_VM/` (cela ne marchera que si vous avez un accès direct à la VM).
- Depuis un shell qui tourne sur votre VM, lancez la commande `links http://localhost/` (si besoin, installez `links` au préalable). Links est un navigateur en mode texte, peu convivial mais cela présente entre autres l'avantage de pouvoir être lancé facilement via une connexion SSH.
- Si vous voulez vous amuser : faire un tunnel SSH pour accéder à `IP_VM:80` depuis votre PC physique.

Vous devriez voir apparaître la page d'accueil par défaut de nginx (« Welcome to nginx ! »). Si vous le voulez, vous pouvez aussi voir et modifier le contenu de cette page dans le fichier `/var/www/html/index.nginx-debian.html`.

Q.I.63) - Interrogez systemd pour avoir le statut du service nginx avec la commande :

```
systemctl status nginx.service
```

Q.I.64) - Vérifiez « à la main » que le processus nginx tourne : `ps aux | grep nginx`

Q.I.65) - Coupez le service nginx avec la commande :

```
sudo systemctl stop nginx.service
```

Q.I.66) - Ré-essayez de charger la page web : vous aurez une erreur du type « connection refused ».

Q.I.67) - Interrogez systemd pour avoir le status du service nginx avec la même commande que ci-dessus. La ligne « Active » doit être passée de « active (running) » à « inactive (dead) ».

Q.I.68) - Vérifiez « à la main » que le processus nginx ne tourne plus : `ps aux | grep nginx`

Q.I.69) - Redémarrez le service :

```
sudo systemctl start nginx.service
```

Q.I.70) - Regardez à quoi ressemble le fichier de description du service pour systemd : `/etc/systemd/system/multi-user.target.wants/nginx.service`
Vous n'avez pas besoin de comprendre les détails, mais ce fichier contient au moins :
— La commande pour démarrer le démon (`ExecStart`)
— La commande pour arrêter le démon (`ExecStop`)
— Un descriptif (`Description`)
— Les dépendances (`After`, `WantedBy`)

Q.I.71) - Pour lister les services disponibles, lancez la commande `systemctl`. Retrouvez la ligne concernant nginx avec `systemctl | grep nginx`.

I.6 Les logs !

Q.I.72) - Naviguez dans `/var/log`. Quels sont tous ces fichiers, leurs droits et que contiennent-ils ?

Insister sur messages, daemon, X et... auth.log pour les connexions ssh !

Q.I.73) - Lancez la commande `tail -f /var/log/nginx/access.log`, puis, pendant que cette commande tourne, rechargez la page d'accueil de nginx. Vous devriez voir apparaître une ligne par requête dans le fichier de log (`tail -f` les affiche au fur et à mesure).

Q.I.74) - Chargez la page `http://IP_VM/404/`. Comme cette page n'existe pas, vous verrez apparaître la page d'erreur 404 (Not Found) de nginx.

Q.I.75) - Installez maintenant le paquet `apache2`. Ce paquet correspond à la version 2 du serveur web Apache httpd. C'est un concurrent direct de nginx, qui va essayez d'écouter sur le port 80 (Ce qui est impossible pour l'instant car nginx est déjà en écoute dessus, mais faisons semblant de ne pas savoir pourquoi un instant ...).

Q.I.76) - Forcez un lancement d'apache :

```
sudo systemctl start apache2
```

Q.I.77) - Essayez de recharger la page d'erreur 404 : c'est toujours nginx qui répond !

Attention, le paquet apache2 a installé un fichier HTML correspondant à sa page d'accueil par défaut. Si vous chargez la page d'accueil, vous aurez peut-être celle d'apache, servie par nginx.

Q.I.78) - Regardez le status du service apache2 avec `systemctl status`. Vous devriez obtenir « Active : inactive (dead) », et la commande vous donne quelques lignes de logs qui devraient vous mettre sur la voie de la raison de l'échec du lancement. Vous pouvez retrouver les détails avec la commande `journalctl`.

Q.I.79) - Devant l'échec de cette tentative de lancer deux serveurs web sur la même machine, nous décidons d'abandonner et de désinstaller apache.

II Si le temps le permet : copie de fichiers à distance

Vous devez apprendre à utiliser ssh pour copier des fichiers à distance. Une excellent alternative est `rsync`.

Si vous souhaitez copier votre `.bashrc` sur plusieurs machines, vous pouvez faire des cas particulier pour certaines machines en mettant des conditions sur le `HOSTNAME`.

Q.II.1) - Faites le en utilisant la ligne de commande, via `scp`, `pscp` (`PuTTY`) ou `rsync`.

Pour copier `machin` dans le repertoire `/tmp/` de la machine virtuelle : `scp -Cr machineacopier chaprot@192.168.77.3:/tmp/` l'option `-C` permet de compresser et l'option `-r` de copier de manière récursive un répertoire.

Attention : utilisation d'une autre clé ssh avec rsync `r`sync -e “ssh -i /.ssh/blabla” –progress root@IP /tmp/

Certains logiciels reposant sur la bibliothèque `FUSE` (File System in User Space) permettent de configurer un système de fichiers en réseau en tant que simple utilisateur. Vous pourrez en installer et utiliser un pour associer à un répertoire du compte chaprot de la VM un répertoire que vous utilisez à l'université pour les TPs de LIF12.

Q.II.2) - Qu'est-ce qu'un disque réseau ? Donner un exemple de logiciel utilisant ce type de bibliothèque.

Cela permet d'associer un disque (windows) ou un répertoire (unix) à un répertoire de fichiers situés sur un serveur réseaux. Une fois cette association faite, la synchronisation est automatique, les fichiers peuvent être utilisés comme s'ils étaient locaux à la machine. C'est ce qui est utilisé par vos comptes à l'université, ou des systèmes comme `DropBox`, ou mieux `Owncloud` ou `Pydio` que vous pouvez installer sur votre machine à la maison.

Q.II.3) - Installez le logiciel `sshfs`.

`sudo apt-get install sshfs` puis répondez oui aux questions.

Q.II.4) - Regardez le manuel de la commande `sshfs`, utilisez-la pour que le répertoire `votrevm:/home/chaprot/univ/` corresponde au répertoire `${HOME}/LIF12/VM` sur votre poste de travail de l'université.

YC-20150913 : non testé !
Utiliser la commande
`sshfs <votre login>@pedagolinux710.univ-lyon1.fr:<votre répertoire>/LIF12/ /home/chaprot/uni`
attention, pour le nom du repertoire. `~/` ne fonctionne pas, il faut mettre le nom complet : `/home/etu/?/p??????????/`.

Q.II.5) - Quel est l'intérêt de faire des systèmes de fichiers dans « l'espace utilisateur » ?

L'accès aux fichiers est une opération du système (espace noyau). Si un utilisateur veut avoir accès à un fichier qui se trouve ailleurs sur le réseau, il doit le télécharger ou utiliser un logiciel qui le télécharge à sa place. Un logiciel qui n'est pas « prévu pour » ne peut donc pas le faire. En configurant depuis l'espace utilisateur un système de fichiers, on permet à un utilisateur (non administrateur) de créer un système de fichiers réseau. Tous les logiciels seront donc capables de manipuler les fichiers de ce système, même s'il ne sont pas réellement présents sur la machine. Par exemple, sans avoir à faire de copie, vous pourrez compiler vos codes sur la VM et les modifier sur votre poste de travail. Pour cela, si le logiciel est installé, vous n'avez pas besoin de devenir administrateur : il vous suffit d'avoir le droit de lire et modifier les différents répertoires concernés.

III Pour aller plus loin (si vous êtes en avance)

III.1 Les "fichiers" du noyau Linux

Q.III.1) - Naviguez dans `/boot`, `/usr/src/`, `/lib/modules/`
Peut-on lire la même chose en local ?

S'assurer que les étudiants ont toujours bien 2 voire 3 consoles, et qu'ils les utilisent plutôt que se déconnectent et reconnectent.
Et on peut lire le même genre de choses en local, mais pas la même chose car machine différente...

Quelles sont les capacités du noyau chargé ?

`lsmod` pour lire les modules chargés, *i.e.*, ces capacités immédiates. On voit plus loin `modprobe`...

Q.III.2) - Faire `dmesg`, distant et local.
Repérer le type de processeur, et de disque dur.

Faites `cat /proc/cpuinfo` et `cat /proc/meminfo`
Quelles informations obtenez-vous ?
Peut-on les obtenir en local et/ou distant ?
Du coup, quel utilisateur peut exécuter la commande ?
Comment était-il possible de le savoir avant ?

Q.III.3) - Listez `/proc/` et commentez ce que vous y trouver.

> Parler des répertoire pid, et voir ce qu'on y trouve : c'est directement en lien avec le CM. Insister sur les droits d'accès, du type de système de fichiers utilisé pour accéder à ces informations.

Des remarques sur `/proc/sys/net/ipv4/ip_forward` ?

Q.III.4) - Quels systèmes de fichiers le noyau installé est-il capable de lire ?

```
cat /proc/filesystems
```

Q.III.5) - Qu'est-ce que `/dev/` ?
Qu'est-ce que `/dev/shm/` ? Qu'est-ce que `/dev/disk/by-uuid/` ?

> `/dev/` contient les fichiers liés aux périphériques. `/dev/shm/` contient un système de fichier en mémoire RAM utilisé pour faire communiquer les applications en mémoire partagée (SHared Memory). `/dev/disk/by-uuid` contient la liste des disques, classés par UUID, c'est à dire par identifiant unique (en pratique ce répertoire contient des liens symboliques vers les fichiers `/dev/sd...`)

III.2 Les alias

Les alias permettent de gagner un peu de temps en définissant des raccourcis pour les commandes qu'on utilise le plus souvent.

Q.III.6) - Exécutez et commentez pour chaque **ligne** suivante :

```
alias
l
alias l="ls -l --color=auto"
l
alias
```

III.3 Jouons à casser notre environnement ;-)

Comme le titre l'indique, ne pas faire ces manipulations sur votre compte habituel pour ne pas prendre de risque.

— Exécutez la commande `PATH=` (équivalente à `PATH=''`, c'est à dire avec une chaîne vide à droite du `=`), puis essayez `ls` et `cd`. Expliquez ce qu'il s'est passé.

Avec un `PATH` vide, le shell ne trouve plus les commandes externes comme `ls`. On peut encore les exécuter en entrant leur chemin complet comme `/bin/ls`. Les commandes internes du shell comme `cd` sont encore disponibles.

III.4 Préserver une session de la déconnexion avec `screen`

Un scénario assez classique :
— Un utilisateur lance une commande sur son serveur
— L'heure tourne, l'utilisateur doit rentrer chez lui
— De chez lui, l'utilisateur se reconnecte à son serveur via SSH, et aimerait reprendre la main sur ce qu'il a démarré avant de partir.

La commande `screen` (ou un de ses petits frères comme `tmux` ou le couple `dvtm/dtach`) permet de répondre à ce problème (et bien d'autres comme la possibilité de découper votre terminal en sous-fenêtres).

— Dans un terminal lancé sur votre VM, entrez la commande `screen`, et validez le message d'accueil avec Entrée si besoin.
— Entrez quelques commandes : tout se passe normalement.
— Entrez Control-a puis d. Vous devriez revenir au shell initial, mais ce que vous avez démarré dans screen tourne toujours. Si un calcul est en cours, le calcul continue.
— Entrez la commande `screen -r` : votre environnement screen est de retour.
— Refaites Control-a puis d. Déconnectez-vous complètement de la VM.
— (Pour respecter strictement le sénario ci-dessus, il faudrait rentrer chez vous ici ...)
— Reconnectez-vous via SSH et faites `screen -r` : votre environnement est encore là.

Une alternative est la commande `nohup` qui permet de lancer une commande qui survit à la fin de la session de l'utilisateur courant (mais ne permet pas de reprendre la main facilement dessus).

III.5 Une commande less dopée aux stéroïdes

Q.III.7) - Aller plus loin :
— `lesspipe -h` et `https://www-zeuthen.desy.de/~friebel/unix/less/README`
Exécutez `less filename` où filename est un fichier C, une archive ou `/bin/bash`.
Puis : `export LESSCOLOR=yes; export LESS="-R -M --shift 5"; LESSOPEN="| lesspipe %s"`
et recommencez. Commentaire ?

Attention, il y a plusieurs variantes de `lesspipe`, celle décrit sur le README pointé gère la coloration syntaxique paramétrée par `$LESSCOLOR` mais pas celui fourni par défaut par Debian.

— Exécutez `man man` puis

```
1  man() {
2  env LESS_TERMCAP_mb=$'\E[01;31m' \
3      LESS_TERMCAP_md=$'\E[01;38;5;74m' \
4      LESS_TERMCAP_me=$'\E[0m' LESS_TERMCAP_se=$'\E[0m' \
```

```
5           LESS_TERMCAP_so=$'\E[38;5;246m' \
6           LESS_TERMCAP_ue=$'\E[0m' \
7           LESS_TERMCAP_us=$'\E[04;38;5;146m' \
8           man "$@"
9       }
10      man man
```