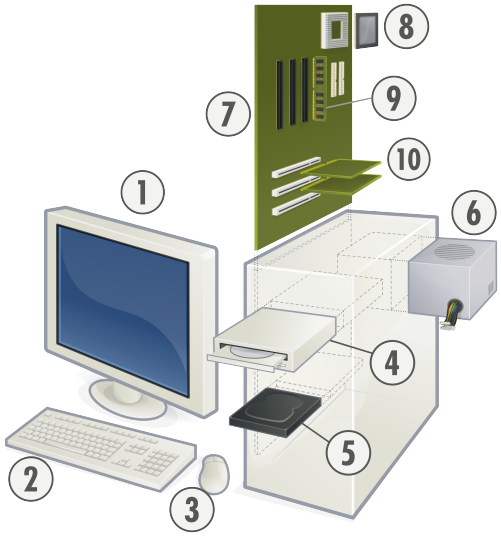
Pratique, Installation et Utilisation des machines
TP 10 - Révisions (Linux/Bash)

02 décembre 2016

Complétez les lignes en indiquant chaque fois la commande à taper ou la réponse appropriée.

1 - Connaitre sa machine

Les différents éléments visibles sur ce schéma sont :



1. écran
2. clavier
3. souris
4. lecteur/graveur cd/dvd
5. disque dur
6. alimentation
7. carte mère
8. processeur
9. barrette mémoire RAM
10. cartes PCI (réseau, graphique, son)

Quand je veux installer Linux je fais les opérations suivantes (complétez et numérotez dans l'ordre) :

(5) Je démarre le PC avec la clé USB

(3) Je crée une clef bootable/liveUSB d'installation GNU/Linux.

(6) J'installe GNU/Linux.

(4) Je configure le BIOS pour démarrer sur une clef USB

(1) Je sauvegarde mes données sous Windows.

(2) Je télécharge une distribution Linux

Linux est un système d'exploitation multi-utilisateur.

Quand la mémoire est utilisée à 100%, pour pouvoir continuer à fournir de la mémoire aux nouvelles applications,

Linux recopie une partie de la mémoire dans le SWAP, c'est une partition spéciale située à la fin du disque dur.

2 - Se déplacer, manipuler les fichiers

Pour me déplacer dans mon répertoire personnel : $ cd ~

Pour créer un dossier nommé `tp10` : $ mkdir tp10

Pour me déplacer dans le répertoire `tp10` : $ cd tp10

Pour créer un chemin `tp10/script/data/tmp` : $ mkdir -p tp10/script/data/tmp

Pour me déplacer dans `tp10/script/data/tmp` : $ cd tp10/script/data/tmp

Pour créer les fichiers `un`, `deux` et `trois` : $ touch un deux trois > un ; > deux ; > trois

Pour supprimer le fichier `trois` : $ rm trois

Pour revenir au niveau précédent : $ cd ..

Pour savoir où je suis : $ pwd

Pour lister les fichiers présents dans ce répertoire : $ ls

Pour lister les fichiers en affichant les fichiers cachés : $ ls -a

Pour supprimer le répertoire `tmp` qui contient des fichiers : $ rm -r tmp

Pour remonter de deux niveaux d'un coup : $ cd ../..

Pour effacer le répertoire vide `script/data` : $ rmdir script/data

Pour créer le fichier `"mon texte"` : $ touch "mon texte"

Pour renommer ce fichier `mon_texte` : $ mv "mon texte" mon_texte

Pour visualiser le contenu du fichier texte `~/.bashrc` : $ cat ~/.bashrc

Pour connaître le type du fichier `/bin/date` : $ file /bin/date

3 - Permissions, propriété, chemins d'accès

Pour lister les fichiers en affichant le détail des droits : $ ls -l

Sous Linux, le superutilisateur ou administrateur se nomme root et son groupe est root.

Cette ligne : `lrwxrwxrwx 1 guss users 10 jan. 1 12 :24 cours -> Documents/fac/python/`

indique que le fichier nommé cours est un lien symbolique qui appartient à guss

et au groupe users.

Cette ligne : `-rw-r r  1 usr1 usr1 19554 dec. 30 14 :22 exercices.odt`

Indique que le fichier est lisible par tous modifiable par usr1 et exécutable par root.

Les informations concernant les différents groupe sont stockées dans le fichier /etc/group, les informations concernant les utilisateurs dans le fichier /etc/passwd et les mots de passe dans le fichier /etc/shadow.

Pour se donner le droit d'exécution sur le fichier `script.sh` : $ chmod u+x script.sh

Symbole pour donner un droit à l'utilisateur u, le groupe g, les autres o, tout le monde a

Symbole pour ajouter un droit +, pour supprimer un droit -, donner que le droit indiqué =

Symboles correspondants aux droits (en chiffre) : lecture r (4) écriture w (2) exécution x (1)

Équivalent de `chmod u=rwx script.sh ; chmod go=rx script.sh` en chiffres :

$ chmod 755 script.sh

Pour donner `script.sh` à l'utilisateur `martin` et au groupe `users` :

$ chown martin:users script.sh

Pour lancer `script.sh` qui est dans le répertoire courant : $ ./script.sh

Pour lancer `date` qui est dans le répertoire `/bin` : $ date /bin/date

Je n'ai pas besoin de préciser le chemin pour les commandes du système parce que les chemins sont indiqué dans le PATH

L'arborescence d'un système de fichier GNU/Linux débute par /

Exemple de chemin relatif : ../document/ ./document document/scripts/

Exemple de chemin absolu : ~/document /bin/ /home/martin/document

4 - Liens, redirections, pipes

Pour créer un lien vers `/home/arthur/Document` : $ ln -s /home/arthur/Document lien

Pour rediriger la sortie standard : `echo "bonjour"` > `fichier`

Pour rediriger la sortie d'erreur : `apt-get update` 2> `fichier`

Pour rediriger les deux sorties : `find / -name "*log*"` &> `fichier`

Pour rediriger les erreurs au même endroit : `./script.sh > fichier` 2>&1

Pour rediriger la sortie standard au même endroit : `date 2> fichier` 1>&2

Pour rediriger la sortie standard sans écraser le contenu du fichier : `echo "export PS1"` >> `fichier`

Pour rediriger la sortie d'erreur sans écraser le contenu du fichier : `./prog.py` 2>> `fichier`

Pour rediriger la sortie d'une commande vers la suivante : `sort listing.txt` | `uniq` | `wc -l`

5 - Réseau

Le service réseau qui attribue les adresses IP comme 192.168.3.129 se nomme DHCP.

Le service réseau qui permet de savoir que l'IP de Langoustine est 192.168.3.129 se nomme DNS.

Pour savoir si Langoustine est accessible sur le réseau : $ ping 192.168.3.129

Pour me connecter à distance sur langoustine avec le nom d'utilisateur usr1 :

$ ssh usr1@192.168.3.129

6 - Processus

Pour stopper proprement un processus qui demande une saisie de texte : Ctrl+D => EOF

Pour stopper en force un processus qui tourne en boucle : Ctrl+C

Pour mettre un processus en pause à l'arrière plan : Ctrl+Z

Pour récupérer le PID d'un processus : $ ps -a

Pour tuer ce processus (PID=23452) avec le signal SIGTERM : $ kill 23452

Pour tuer violemment ce processus (PID=23452) (perte de données, pas de nettoyage de la mémoire) :

$ kill -9 23452

Pour tuer tous les processus nommés `prog` : $ killall prog pkill prog (*prog*)

Le seul processus qui n'a pas de parents est init

Un processus dont le parent ne récupère pas la valeur de retour devient un processus zombie.

Pour examiner les ressources consommées par les processus : $ top atop htop

7 - Environnement

La variable PATH est une variable de l'environnement, je peux afficher sa valeur avec :

$ echo $PATH ou printenv PATH

La variable PS1 est une variable locale contenant le prompt, je peux afficher sa valeur avec :

$ echo $PS1

Les processus enfants reçoivent les variables de l'environnement du processus parent.

Pour créer une variable VAR qui contient 10 : $ VAR=10

Pour placer la variable VAR dans l'environnement : $ export VAR

Pour détruire la variable VAR : $ unset VAR

8 - Script Shell

Complétez les commentaires et les caractères manquants remplacés par des _. Ajoutez les instructions pour créer le répertoire de sauvegarde si il n'existe pas.

```
#!/bin/bash
# Basic script bash to backup files in a given directory
# Usage backup.sh [PATH of dir. you want backup] [PATH of dir. where to backup]
# John Doe september 1981

# Version : 0.1
DATE=`date +%d-%m-%y`

# ask for 2 arguments
if [ $# != 2 ]; then
    echo "Usage : backup.sh [origin PATH] [backup PATH]"
    exit 1
fi

#  check if first argument is a directory
if [ ! -d $1 ]; then
    echo "It seems $1 is not a directory"
    exit 1
else
    echo "Ok directory found ; we can go further"
fi

#  ask if we want to compress backup
read -n 1 -p "Do i have to compress : [y/n]? " COMPRESS ; echo
if [[ $COMPRESS =~ ^[Yy] ]]; then
    echo "Ok compress with name backup.$DATE.tar.gz "
elif [[ ! $COMPRESS != ^[Nn]$ ]]; then
    echo "I beg your pardon i did not understand, you must answer by y or n"
    exit 2
fi
```

```
# give few seconds to cancel backup
echo "You've got few seconds to cancel it by ctrl^C"
for i in {1..5}; do echo $i; sleep 1; done
echo "so let's go"


# create the backup directory if it does not exist

 if [ ! -d $2 ]; then
      mkdir $2
      if [ ! -e $2 ]; then
           echo "error : unable to create directory $2"
           exit 3
      fi
 fi



# create backup
if [[ $COMPRESS =~ ^[Yy]$ ]]; then
    archive="backup.$DATE.tar.gz"
    tar czf $archive $1
    mv $archive $2
else
    mkdir $2/$DATE
    cp -a $1  $2/$DATE
fi
exit 0
```