**Linux**

**--------------------------------------------------------------------**

**Les boucles**

**--------------------------------------------------------------------**

* boucle if :

if [ ] ;

then

commands;

elif [ ] ;

then

commands;

else

commands;

fi ;

* boucle case :

case <var> in

11)

#les différentes valeurs de la variable

commands;

;;

12)

commands;

;;

13)

commands;

;;

14)

commands;

;;

\*) #defaults

commands;

;;

esac ;

case <var> in

0 | 1 | 2 | 3 | 4 | 5 | 6 | S )

commands ;

;;

\* )

commands;

;;

esac ;

* boucle while :

while

commands;

[ commande d'arrêt ] ;

do

commands;

done ;

**--------------------------------------------------------------------**

**Autres**

**--------------------------------------------------------------------**

* lecture d’une variable:

**read** <var> ;

* pour mettre une execution dans une variable :

var=**$(commands )**;

* ecriture avec printf:pour préciser le format :

**printf** "%d %s\n" **$var $var ;**

* sauvegarder dans un fichier:
  + écraser le contenu du fichier : **>**
  + ajouter a la fin du fichier : **>>**
* lire des données d’un fichier :

**read** [ensemble des var] **<** fichier

* afficher contenu fichier :

**cat** fichier;

exemple:

**while** **read** num nom

**do**

**echo** **$num $nom** ;

**if [ $num** -eq **$numAj** **]** ;

**then**

**modprobe -v** **$nom** ;

**sed -i** '/'**$numAj**'/d' SuppHis ;

**fi ;**

**done** < SuppHis

* supprimer une ligne d’un fichier :

lien:https://www.folkstalk.com/2013/03/sed-remove-lines-file-unix-examples.html

**sed -i** '/'**<contenant la var>**'/d' fichier ;

* **awk options 'selection \_criteria {action }' input-file**

exemple: awk -F: **'{**printf("%s\t%d\n",$1,$3);**}'** fichier

* **column -t [FileName]:**crée des tables visuellement attrayantes. Par défaut ,chaque ligne est interprétée comme une case de tableau
* **PATH=$PATH:.** : On a ajouté '.' à la variable PATH pour pouvoir exécuter le script sans './'
* **/dev/pts/[ID] :** contains the terminals and **[ID]** correspond to the **terminal ID**. To get it , execute the **ps** command and you’re going to find it in the **TTY** column
* **sudo p2 > /dev/pts/1 :** redirect the p2 script’s execution in the terminal with process number 1
* [**pidof**](https://man7.org/linux/man-pages/man1/pidof.1.html) **<process\_name> --** find the process ID of a running program.

**--------------------------------------------------------------------**

**gestion des modules**

**--------------------------------------------------------------------**

MODULE :

Modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system. Without modules, we would have to build monolithic kernels and add new functionality directly into the kernel image. Besides having larger kernels, this has the disadvantage of requiring us to rebuild and reboot the kernel every time we want new functionality.

(adding functionality without rebooting)

* **lsmod** : lists the loaded kernel modules
* **uname** : displays the information about the system
  + - **uname -a :** Shows all information in the following order: -s, --kernel-name Print the kernel name
    - -n, --nodename Print the network node hostname
    - -r Print the current kernel release.
    - -v, --kernel-version Print the kernel version (Build date)
    - -m, --machine Print the hardware machine name
    - -p, --processor Print the processor type
    - -i, --hardware-platform Print the hardware platform
    - -o, --operating-system Print the operating system
* **nl :** permet de numériser l’affichage

lsmod | nl

* Loading Modules : 2 ways (manually , automatically)
  + automatically:
    - * The hotplug infrastructure (see LPI-101 Hardware section) for filesystems etc. The devfsd daemon and an alias entry in /etc/modules.conf. devfsd will load the module each time the device is accessed
  + manually:
    - * **insmod** **<moduleName>** (loads modules without checking for directories)
      * **modprobe** **<moduleName>** (Checks the module's dependencies. Loads all the dependencies if needed and then loads the module)
* **rmmod <moduleName>** or **modprobe -r <moduleName>** : unload the module
* **rmmod -R <moduleName>**: remove recursively:
* **modinfo <NomModule>** fournit une description du module en question ainsi que d'autres informations. On peut fournir une description sur un certain paramètre
  + author **(-a**) description (-d) license (-l) depends alias param (-p) ( -n = /path/filename of module's file) (-F --field Only print this field value, one per line)

**--------------------------------------------------------------------**

**gestion des utilisateurs / groupes**

**--------------------------------------------------------------------**

* afficher la liste des utilisateurs :

cat **/etc/passwd**

/etc/passwd contient la liste des utilisateurs

* ajouter un utilisateur:

**useradd -m** $NomUtilA ;

-m :creates the user’s home directory if it does not exist

* supprimer un utilisateur:

**userdel -r** $NomUtilA ;

-r : supprime les fichiers présents dans le répertoire personnel de l’utilisateur seront supprimés en même temps que le répertoire lui-même ainsi que le répertoire d’attente des courriels

* modifier un utilisateur :
  + modifier le nom:

**usermod -l** $NouNom $NomUtilM ;

* + modifier password :
    - changer le mdp de l’utilisateur:

**sudo passwd <nom\_utilisateur>**

* + - supprimer le mdp de l’utilisateur:

**passwd -d <nom\_utilisateur>**

* Gérer la liste des groupes :
  + Afficher la liste de tous les groupes :

**cat /etc/group | cut -d':' -f1 ;**

#afficher la liste des noms des groupes

* + lister les groupes d’un utilisateur:

**groups <NomUtilM> ;**

* + Ajouter un groupe à un utilisateur:
    - **usermod -aG <GroupA><NomUtilM>;(comme groupe normal)**
    - **usermod -g <GroupA><NomUtilM>;(comme grp primaire)**
  + Supprimer un groupe d’un utilisateur:

**gpasswd -d <NomUtilM> <GroupS> ;**

* + Afficher les informations d’un utilisateur:

**getent passwd <NomUtilM>**| cut -d':' -f5 ;

**\*getent passwd [NomUtil]:** cette commande nous a permis d’afficher la ligne du fichier /etc/passwd ou le nom d’utilisateur est $NomUtilM

* + Changer les informations d’un utilisateur:

**usermod -c "Info" <NomUtil> ;**

**--------------------------------------------------------------------**

**Boot, Initialization, Shutdown and Runlevels**

**--------------------------------------------------------------------**

* Afficher le runlevel :

**who -r**

**runlevel (displays previous and present runlevel)**

exp: N(none) 3

* afficher les processus lancés au démarrage :
  + afficher tous les processus:

**ps -ef | more;(pour afficher les processus en page )**

-e : afficher tous les processus

-f : pour formater l’affichage

* + les processus du démarrage ont un ppid 1 donc pour afficher les processus de démarrages on écrit :

while read **uid pid ppid c stime tty time cmd**

do

if [ $ppid == '1' ]

then

echo $cmd ;

fi ;

done < ListPro(contient le résultat de la commande **ps -ef** )

* il existe 0-6 runlevels :
  + **Runlevel 0 :** Arrêt de la machine
  + **Runlevel 1 :** Mode maintenance (mono-utilisateur en ligne de commande)
  + **Runlevel 2 à 5 :** Mode multi-utilisateur complet avec serveur graphique si installé
  + **Runlevel 6 :** Redémarrage de la machine
* Afficher un certain runlevel:

**ls /etc/rc<runlevel>.d ;**

* Changer le runlevel:

**telinit <newrunlevel>**

* **ls \*.conf :** les fichiers de configuration
* **L'outil de gestion des** [**services**](https://doc.ubuntu-fr.org/services) **(et des autres unités d'ailleurs)** dans **systemd** s'appelle **systemctl**.
* **commande gestion des services :**

**systemctl** **ACTION <Nom\_du\_service>.service**

Où

* **ACTION** sera la commande que l'on souhaite appliquer à la dite unité:
  + **start :** démarrer le service
  + **stop :** arrêter le service
  + **restart :** relancer le service
  + **reload :** recharger le service
  + **status :** connaître l'état du service
* **<Nom\_du\_service>** est le nom du service visé.

=>Quelle que soit l'action menée sur un service, au prochain démarrage de la machine celui-ci devrait retrouver le status qui lui a été [défini par défaut](https://doc.ubuntu-fr.org/systemd#modifier_l_execution_d_un_service).

* Shutdown the system:

**init0**

**shutdown -h now**

**halt**

**poweroff**

* Reboot the system:

**init6**

**shutdown -r now**

**reboot**

* Cancels the already scheduled shutdown:

**shutdown -c**

* Note: The reboot, poweroff and suspend commands are symbolic links to halt.
* Syntax: **shutdown [options] time**

fsck checks and repairs a Linux filesystem

* + Options:
    - **-c** Cancels a shutdown
    - **-f** Will not run fsck on the reboot
    - **-F** This WILL run fsck on reboot
    - **-h** Halts system after shutdown
    - **-k** Sends warning / does not shutdown
    - **-n** Shuts down without calling init -r Reboots, does not halt
    - **-t** {Seconds} Delay time after killing process (before init)
  + Time format:
    - **now** Well...NOW!
    - **+2m** In 2 minutes 4 In 4 minutes
    - **hh:mm** At the specified time
  + Command access rights:
    - The file /etc/shutdown.allow may contain user names (one per line) who have permission to run the shutdown command

**-------------------------------------------------------------------**

**Gestion de Processus**

**--------------------------------------------------------------------**

* Afficher la liste de process ( tous les processus ) :

**ps -ef**

* Tuer un process :

**kill -9 <pid>**

* Stopper un processus :

**kill -19 <pid>**

* Continuer un processus :

**kill -18 <pid>**

* **kill -l :** afficher les options de la commande kill
* **kill -i <pid>**
  + **1 (HUP)** - Reload a process.
  + **9 (KILL)** - Kill a process.
  + **15 (TERM)** - Gracefully stop a process
* Afficher les parents d’un processus :

**do**

**echo "Les parents du fils dont le pid est :" ;**

**pidP=$(sudo ps -ef | awk '{printf("%d\t%d\t%s\n",$2,$3,$8)}'| grep "^$pidf"| awk '{printf("%d",$2)}' );**

**sudo ps -ef | awk '{printf("%d\t%d\t%s\n",$2,$8)}'| grep "^$pidP"| more ;**

**done ;**

* Afficher le propriétaire d'un process:

**do**

**echo "Le propriétaire du process dont le pid est :" ;**

**sudo ps -ef | awk '{printf("%d\t%s\n",$2,$1)}'| grep "^$pidp" | more ;**

**done ;**

//le propriétaire est le premier champ

* Afficher les fils d'un process:

**do**

**echo "Les fils du parent dont le pid est :" ;**

**sudo ps -ef | awk '{printf("%d\t%d\t%s\n",$3,$2,$8)}'| grep "^$pidp" | more ;**

**done ;**

* Ouvrir deux terminaux ---> **ps -ef > f1** (dans le terminal 1)

---> fermer le terminal 2 ---> **ps -ef > f2**(dans le terminal 1)

---> **diff f1 f2** (dans le terminal 1) : ça nous permet de déterminer le PID du process du terminal 2

**-------------------------------------------------------------------**

**Archivage:Sauvegarde et restauration**

**--------------------------------------------------------------------**

* Sauvegarde:
  + tar : **tar -cvf <archive> <fichier(s)>** 
    - **Afficher le contenu de l’archive:**

**tar -tvf $Arch ;**

* + cpio : **find <fichier> -depth | cpio -ov > <archive>**
    - **Afficher le contenu de l’archive:**

**cpio -tv < $Arch;**

* + dd : **dd if=<fichier(s)> of=<archive>**
  + dump : **dump -f <archive> <fichier(s)>** 
    - dump allows you to backup a whole partition or a full directory. You can't use DUMP to backup a subdirectory, **DUMP work only on partition.**
* Restaurer:
  + tar : **tar -xvf <archive> <fichier(s)>** 
    - restaurer dans l’emplacement initial:

**tar -xvf $Arch $fichiers ;**

* + - restaurer dans l’emplacement choisi:

**tar -xvf $Arch $fichiers -C $emp ;**

* + cpio : **cpio -idvE tmp < <archive>** avec **tmp** un fichier contenant le nom des fichiers à restaurer
    - pour afficher tout le contenu de l’archive:

**cpio -iudv < $Arch ;**

* + dd : **dd if=<emplacement> of=<archive>**
  + dump : **Sous menu de l’option dump :**
    - Comparer le contenu de l'archive avec l'original :

**restore -C -f <archive>**

* + - Entrer en mode interactive

**restore -i -f <archive>**

* Quelques commandes nécessite les droits de root ( les permissions ) en ajoutant la commande sudo notamment : **dump , restore**

**-------------------------------------------------------------------**

**Planification des exécutions**

**--------------------------------------------------------------------**

* le fichier contenant les commandes doit comporter des lignes **de format** :

**[Min] [H] [J] [Mois] [JourSem] [Util] [Command]**

* **[Min] :** champ de minutes **[0-59]** (**\***: désigne **chaque** minute)
* **[H] :** champ d’heure **[0-23]** (**\***: désigne **chaque** heure)
* **[J] :** Jour du mois **[1-31]** (**\***: désigne **chaque** jour du mois)
* **[Mois] :** champ du mois **[1-12]** (**\***: désigne **chaque** mois)
* **[JourSem] :** Jour de la semaine **[0-7]**(**\***: désigne **chaque** jour de la semaine et **0** est **le dimanche** )
* **[Util] :**  nom d'utilisateur avec lequel exécuter la tâche **(facultatif)**
* **[Command]:** commande à exécuter (Ou on peut même définir le chemin du script à exécuter)
* ajouter une commande:

if [ **$min** -eq 1 ] ;

then

printf "%s\t" "**\***" **>> /tmp/fcron** ;

else

printf "%s\t" "**$min**" **>> /tmp/fcron** ;

fi ;

if [ **$heu** -eq 1 ] ;

then

printf "%s\t" "**\***" **>> /tmp/fcron** ;

else

printf "%s\t" "**$heu**" **>> /tmp/fcron** ;

fi ;

if [ **$dayM** -eq 1 ] ;

then

printf "%s\t" "**\***" **>> /tmp/fcron** ;

else

printf "%s\t" "**$dayM**" **>> /tmp/fcron** ;

fi ;

if [ **$month** -eq 1 ] ;

then

printf "%s\t" "**\***" **>> /tmp/fcron** ;

else

printf "%s\t" "**$month**" **>> /tmp/fcron** ;

fi ;

if [ **$dayW** -eq 1 ] ;

then

printf "%s\t" "**\***" **>> /tmp/fcron** ;

else

printf "%s\t" "**$dayW**" **>> /tmp/fcron** ;

fi ;

**# Afin que la sortie de l'exécution du script sera redirigé de la**

**# console vers le fichier de log /tmp/Output , on ajoute à la fin de**

**# la ligne de planification ‘>> /tmp/Output’ ( ajout à la fin )**

printf "%s >> %s \n" "**${script}**" **"/tmp/Output" >> /tmp/fcron** ;

echo ;

**# cette commande ( crontab < [fichier] ) permet d’accéder au**

**# fichier [fichier] pour récupérer les commandes programmées et**

**# les ajouter au fichier crontab qui se charge de leur exécution**

**# selon la configuration attribuée à chacune**

**crontab < /tmp/fcron ;**

* La commande **crontab** a pour options :

**-l :** lister le contenu du fichier crontab

**-r :** supprimer toutes les actions du fichier crontab

**-e :** éditer les actions au niveau du fichier crontab

* Afficher la date:

**printf "%s" "date = " ; date +%D ;**

* Afficher l’heure:

**printf "%s" "heure = " ; date +%T ;**

**-------------------------------------------------------------------**

**File System & Log**

**--------------------------------------------------------------------**

* File System:
  + afficher l’usage de file system disk:

**df -h(human readable)**

* + Afficher l'espace du disque restant:

**sudo df -h | awk '{ print $4 , 100-$5"%" , $6 }' ;**

**# awk : afficher les colonnes 4 , 5 et 6**

* + Afficher les fichiers volumineux:

**find <rep> -exec wc -c {}\; 2> /dev/null | sort -r -n | head -n $nb ;**

* + retourner seulement les fichiers sous un certain répertoire:

**find <rep> -exec wc -c {}\; 2> /dev/null**

* + Compresser les fichiers volumineux :

**sudo tar -czvf $arch $fichiers ;**

**-z permet de faire la compression**

* + Supprimer les fichiers volumineux :

**sudo rm $fichiers ;**

* **awk :** permet d'appliquer un certain nombre d'actions sur un fichier. La syntaxe est inspirée du C

**awk [-Fs] [-v variable] [-f fichier de commandes] 'program' fichier**

**[-Fs] :** spécifie les séparateurs de champs

**[-v var] :** Définie une variable utilisée à l'intérieur du programme.

**[-f fichier] :** Les commandes sont lues à partir d'un fichier.

* **find :** une commande très pratique pour chercher **des fichiers** à partir de la ligne de commande