1. Introduction

Après avoir préparé la carte micro SD, connecté tous les câbles sur le Raspberry Pi, mis sous tension et configuré le système, l'écran affiche enfin ses premiers mots.

Mais pourquoi utiliser ce mode texte archaïque alors qu'il est possible de travailler sur de superbes écrans graphiques avec la souris ? Plusieurs raisons expliquent ce choix.

Le mode texte est disponible sur toutes les distributions

Historiquement, le mode texte est le mode natif de Linux. Les interfaces graphiques sont venues se greffer par-dessus. Sur toutes les distributions, le mode texte est accessible à travers des consoles ou *tty* (*TeleType* = machine à écrire pilotée à distance). Les combinaisons de touches [Ctrl][Alt][F1] à [Ctrl][Alt][F6] donnent accès à six consoles dans lesquelles des utilisateurs différents peuvent lancer des tâches différentes. La combinaison [Ctrl][Alt][F7] permet de passer au mode graphique (s'il existe).

Sur tout ordinateur, le microprocesseur partage son temps entre les différentes tâches qu'il doit exécuter. La création, la gestion d'un ensemble de fenêtres et de leur contenu consomme de la puissance de calcul. Sur certaines machines, il y a parfois une dizaine de fenêtres ouvertes sur autant d'applications, un navigateur web dans lequel des onglets sont restés ouverts… Tout cela est géré par un unique processeur et chaque tâche occupe de l'espace mémoire.

En mode texte, une page ne représente que quelques kilo-octets et le microprocesseur n'envoie du texte sur la console que lorsque cela est nécessaire. L'occupation mémoire est réduite au strict minimum. Le processeur peut se consacrer entièrement à l'exécution du travail qui lui est confié.

Le processeur ARM du Raspberry Pi 3 n'est pas de la dernière génération. Il fonctionne à 1200 MHz et dispose 1 Go de mémoire qu'il partage avec le GPU. Sur le Raspberry Pi Zero la fréquence de fonctionnement est de 1 GHz et la mémoire limitée à 512 Mo. L'utilisation du mode texte permet d'optimiser les ressources disponibles.

Le mode texte est souvent plus rapide

Avec l'habitude, l'utilisation du mode texte est souvent plus rapide que le mode graphique. Alors qu'en mode graphique il faut parfois naviguer sur plusieurs niveaux de fenêtre pour aboutir au paramètre à modifier, la même modification en mode texte se résume à une ligne de commande. De plus si on doit renouveler l'opération, le rappel de commandes précédentes accélère encore l'opération.

Le mode texte pour se faire aider par la communauté

En général sur les forums, l'assistance apportée par les participants se fait en mode texte. Les nombreuses interfaces graphiques disponibles sous Linux rendent difficile l'assistance en mode graphique. Le mode texte est valable sur toutes les distributions, quel que soit l'environnement graphique. De plus, pour envoyer le résultat des commandes sur le forum, un copié-collé suffit, pas besoin de faire des copies d'écran et de joindre des images volumineuses. Ce chapitre fournit les informations sur les commandes nécessaires pour démarrer avec Linux en mode texte. Ce kit d'une quarantaine de commandes est le minimum de base permettant à un débutant de :

– Démarrer et arrêter le système d'exploitation.

– Naviguer dans le système de fichiers et le modifier.

– Administrer le système.

Conventions utilisées pour la syntaxe

```
commande [options facultatives] paramètre_obligatoire
```

Le premier mot de la ligne est la commande qui fait l'objet de la description.

Entre crochets figurent des options ou paramètres facultatifs. Leur absence n'empêche pas la commande de s'exécuter. Dans ce cas certaines commandes utilisent des paramètres par défaut.

Le ou les paramètres obligatoires doivent figurer dans la ligne de commande. Leur absence entraîne un message d'erreur.

Pour en savoir plus sur les commandes

Chaque commande peut avoir une grande quantité d'options. Seules les plus fréquemment utilisées sont listées ci-dessous. Pour en savoir plus sur chaque commande, il existe une commande universelle sur toutes les distributions Linux : `man` !

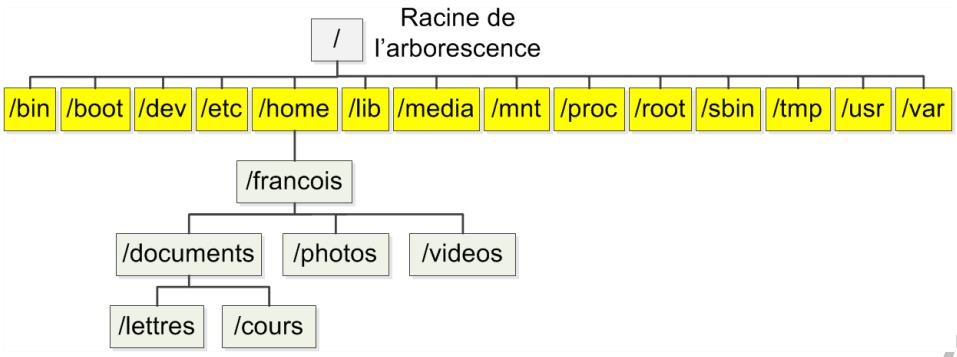
Syntaxe

```
man [nom_de_commande]
```

Cette commande affiche une page d'aide sur chaque commande. Appuyez sur [Espace] ou [Entrée] pour faire défiler, [q] pour revenir à l'interpréteur de commande, [h] permet d'afficher l'aide et toutes les options, [/] permet de faire des recherches dans le texte. Parfois un peu rébarbatives pour les débutants, ces pages d'aide deviennent vite des ressources indispensables pour tout utilisateur de Linux.

2. L'arborescence de Linux

Le système de fichiers de Linux est organisé à partir d'un point de départ appelé root, racine ou encore /. Sous cette racine se déploient des répertoires contenant les fichiers et programmes nécessaires au système d'exploitation. La forme d'arbre inversé de cette structure fait qu'elle a été baptisée arborescence.



Chaque utilisateur peut créer dans son répertoire une arborescence à sa convenance. La connaissance complète de l'arborescence de Linux n'est pas nécessaire à l'utilisation de Linux. Le descriptif ci-dessous peut vous aider à trouver le fichier qui vous intéresse.

Principaux répertoires de l'arborescence Linux	
/	Racine ou root, contient les répertoires de l'arborescence Linux.
bin	Exécutables binaires du système *cp*, *ls*, *mount*, *rm*…
boot	Fichiers de démarrage de Linux.
dev	Fichiers spéciaux assurant la liaison avec les périphériques.
etc	Fichiers de configuration du système, des services…
home	Répertoire personnel des utilisateurs.
lib	Bibliothèques système partagées.
media	Point de montage des clés USB, CD-ROM…
mnt	Point de montage temporaire de partitions et périphériques.
proc	Informations sur les processus et le noyau Linux.
root	Répertoire personnel du super-utilisateur.
sbin	Binaires système et outils comme *fsck*.
tmp	Fichiers temporaires.
usr	Fichiers binaires et commandes utilisateurs.
var	Système de fichiers "variables" (modifiables) ; on y trouve le contenu web (répertoire www), mais aussi les logs (journaux).

Ces répertoires contiennent eux-mêmes d'autres répertoires qui constituent l'arborescence de Linux. Quelques différences peuvent exister entre les distributions, mais la plus grande partie de l'arborescence est commune.

3. La ligne de commande

Après le démarrage, Linux vous accueille avec une demande de *login*. C'est le premier contact avec l'interface en ligne de commande.

3.1 Connexion à Raspbian

Sur les systèmes Linux, il y a deux types d'utilisateurs. L'utilisateur normal a des droits limités et c'est ce qui empêche de faire des erreurs de configuration, de détruire par erreur le système, et limite la propagation de virus sous GNU/Linux. Il peut exécuter des applications et modifier les fichiers auxquels il a le droit d'accéder mais ne peut absolument pas intervenir sur le système lui-même.

Le super-utilisateur a les droits complets sur le système. Il peut effectuer les mises à jour, supprimer ou modifier n'importe quel fichier. Le super-utilisateur s'appelle également *root*. Il ne faut pas confondre *root*, racine du système de fichiers avec *root* le super-utilisateur ou encore */root* le répertoire du super-utilisateur.

Sur certaines distributions, *root* n'a pas le droit de se connecter au système pour des raisons de sécurité. Ceci évite qu'un intrus ou un virus puisse utiliser les privilèges de *root* pour saboter le système ou y introduire des programmes nuisibles. Dans Raspbian *root* n'a pas le droit de se connecter par défaut.

Lors du démarrage en mode texte, le logo du Raspberry Pi est affiché en haut et à gauche de l'écran (4 framboises pour le Raspberry Pi 3, une seule pour le Raspberry Pi Zero). Les messages envoyés par le système pendant sa phase de démarrage défilent sous le logo. Lorsque le système a démarré, il indique l'adresse IP qu'il utilise, informe l'utilisateur qu'il travaille sur la console tty1 et se met en attente de la connexion d'un utilisateur.

3.1.1 Connexion en utilisateur normal

Attention : contrairement à d'autres systèmes d'exploitation, Linux est sensible à la casse des caractères. Il considère que les majuscules et les minuscules sont des caractères différents ! C'est d'ailleurs le cas puisque leurs codes ASCII sont différents. Il convient de respecter les majuscules/minuscules dans la saisie des commandes.

Au démarrage, le Raspberry Pi affiche le nom de l'ordinateur *raspberrypi* et le mot *login:* qui invite l'utilisateur à s'identifier en saisissant son nom.

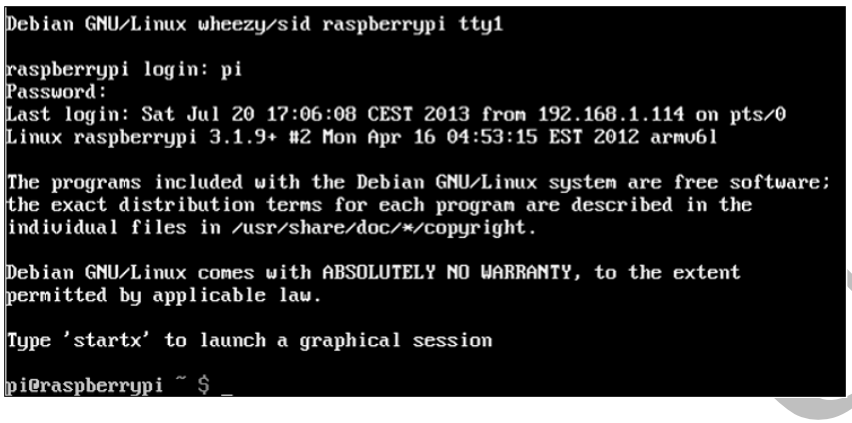
Un utilisateur est déjà créé par défaut dans la distribution Raspbian. Il s'agit de l'utilisateur *pi* et son mot de passe est *raspberry*.

d Saisissez le login `pi` et validez avec la touche [Entrée].

Le système demande un mot de passe pour authentifier l'utilisateur.

d Saisissez `raspberry` et validez avec la touche [Entrée].

Le mot de passe saisi par l'utilisateur ne provoque aucune action sur l'écran. C'est une mesure de sécurité pour éviter qu'un observateur puisse deviner ne serait-ce que le nombre de caractères du mot de passe.



Le système affiche un certain nombre d'informations : la date de la dernière connexion de l'utilisateur *pi*, la version du noyau Linux... puis, sur la dernière ligne, l'invite de commande composée ici de 16 caractères (sans compter les espaces) : `pi@raspberrypi ~ $`

Cette invite de commande s'interprète comme suit : l'utilisateur *pi* est connecté à l'ordinateur *raspberrypi*. Il travaille dans son répertoire personnel ~ (*/home/pi*). Il ne peut exécuter qu'un nombre limité d'actions car il n'est qu'un utilisateur normal. C'est ce qu'indique le $ à la fin de l'invite de commande. Tout cela est exprimé avec 16 caractères, ce qui montre bien la concision du mode texte.

Si l'utilisateur *pi* essaye d'arrêter le système :

```
pi@raspberrypi ~ $ shutdown -h now
shutdown: you must be root to do that!
Usage:    shutdown [-akrhPHfFnc] [-t sec] time [warning message]
```

Un message l'avertit que seul *root* peut arrêter le système.

Deux possibilités se présentent à l'utilisateur :

– Se déconnecter puis se reconnecter en tant que *root*.

– Devenir momentanément super-utilisateur.

Déconnexion d'un utilisateur

Syntaxe

```
logout
```

Pour mettre fin à une session Linux saisissez `logout` suivi de la touche [Entrée].

La session se termine, le système fonctionne toujours. La console affiche de nouveau `raspberrypi login:` dans l'attente de la connexion d'un nouvel utilisateur.

L'utilisateur peut maintenant se reconnecter en tant que *root* pour arrêter la machine.

Devenir momentanément super-utilisateur

Un utilisateur normal peut devenir momentanément super-utilisateur, le temps d'exécuter une commande que seul *root* peut lancer. Il existe deux façons de parvenir à ce résultat.

La première est `sudo` (*Substitute User Do* = exécuter en tant qu'autre utilisateur). Pour utiliser `sudo`, cet utilisateur doit cependant figurer dans un fichier appelé *sudoers* qui recense les utilisateurs autorisés à effectuer cette opération. C'est l'administrateur qui décide des membres de la liste des *sudoers*.

Syntaxe

```
sudo [commande]
```

Pour tester `sudo`, l'utilisation de `ls` qui ne fait que lister le contenu du répertoire (voir la section Se déplacer dans l'arborescence dans ce chapitre) est sans danger.

```
pi@raspberrypi ~ $ sudo ls
Desktop  python_games
```

Le mot de passe demandé par `sudo` est celui de l'utilisateur qui souhaite devenir super-utilisateur. Par défaut, il est mémorisé pendant 15 minutes. L'utilisateur peut réutiliser la commande `sudo` sans avoir besoin de saisir à nouveau le mot de passe pendant ces 15 minutes.

La commande `ls` a été exécutée en tant que super-utilisateur. Si un utilisateur non autorisé (*user*) essaye d'effectuer la même opération :

```
user@raspberrypi ~ $ sudo ls

We trust you have received the usual lecture from the local
System Administrator.
It usually boils down to these three things:

     #1) Respect the privacy of others.
     #2) Think before you type.
     #3) With great power comes great responsibility.

[sudo] password for user:
user is not in the sudoers file.  This incident will be reported.
user@raspberrypi ~ $
```

L'utilisateur *user* ne fait pas partie des utilisateurs autorisés à utiliser la commande `sudo`. Sa tentative est rejetée.

`sudo` permettra à l'utilisateur de procéder à l'arrêt du système de la même manière :

```
pi@raspberrypi ~ $ sudo shutdown -h now

Broadcast message from root@raspberrypi (pts/0) (Sat Jul 20
17:43:56 2013):
The system is going down for system halt NOW!
pi@raspberrypi ~ $
```

L'autre possibilité pour devenir super-utilisateur est d'ouvrir une session momentanée sous une autre identité. C'est ce qu'autorise la commande `su`. Si aucun nom d'utilisateur ne suit la com-

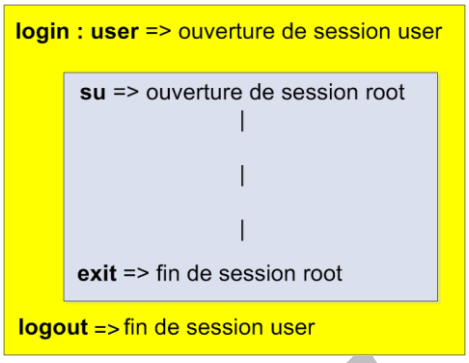
mande `su`, c'est le super-utilisateur *root* qui est utilisé par défaut.

Syntaxe

```
su [nom_d_utilisateur]
```

`nom_d_utilisateur`	Paramètre facultatif. Si `nom_d_utilisateur` est omis, c'est l'utilisateur root qui est choisi par défaut.
`- nom_d_utilisateur`	La présence du tiret après la commande `su` permet de lancer un shell en récupérant les variables de l'utilisateur cible.
`-`	Le tiret seul lance un shell avec l'environnement par défaut de *root*. `su -` est donc préférable à `su`.

Le fonctionnement de `su` est schématisé ci-dessous. Un utilisateur (*user*) ouvre une session. Au cours de cette session, il se connecte momentanément comme *root* en utilisant la commande `su`. Lorsqu'il sort de cette session *root*, il revient à sa session précédente.



Le mot de passe demandé par `su` est celui de l'utilisateur sous l'identité duquel l'utilisateur actuel souhaite se connecter. Dans l'exemple ci-dessous, c'est le mot de passe de *root* qui est attendu.

```
user@raspberrypi ~ $ su
Mot de passe :
root@raspberrypi:/home/user# ls
fichier1.txt
root@raspberrypi:/home/user# exit
exit
user@raspberrypi ~ $
```

Ceci implique que l'administrateur communique son mot de passe à au moins une personne. En matière de sécurité informatique, c'est absolument proscrit. Par contre sur votre propre Raspberry Pi, rien n'empêche d'utiliser `su` pour devenir momentanément super-utilisateur.

La fin d'une session super-administrateur ouverte avec `su` se fait par la commande `exit` qui provoque un retour à l'invite de commande de départ (voir la copie d'écran précédente).

La commande `su` peut également servir à devenir momentanément un autre utilisateur :

```
pi@raspberrypi ~ $ su user
Mot de passe :
user@raspberrypi /home/pi $ ls
Desktop  python_games
user@raspberrypi /home/pi $ exit
```

```
exit
pi@raspberrypi ~ $
```

Cette fois l'utilisateur *pi* endosse momentanément l'identité de l'utilisateur *user* dont il connaît le mot de passe. Cependant même lorsqu'il a l'identité de *user*, il reste dans son dossier personnel propre : */home/pi* sauf s'il utilise la commande `su - user`. La commande `exit` termine la session ouverte en tant que *user* et l'invite de commande redevient `pi@raspberrypi ~ $`.

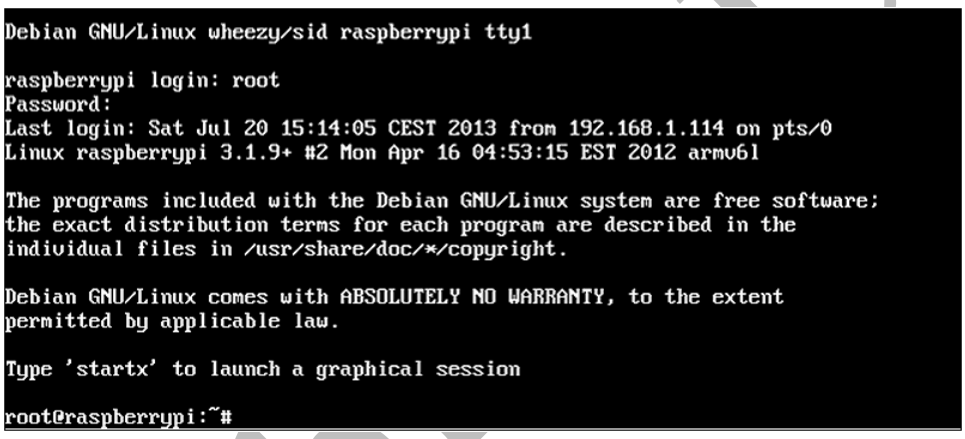
3.1.2 Connexion en root

Pour permettre la connexion du super-utilisateur *root* sur Raspbian, il faut d'abord activer le compte du super-utilisateur. Avec le compte de l'utilisateur *pi*, il faut passer par la commande `sudo` et la commande `passwd` :

```
pi@raspberrypi ~ $ sudo passwd root
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd : le mot de passe a été mis à jour avec succès
pi@raspberrypi ~ $
```

Notez bien le mot de passe que vous venez d'attribuer à *root*. Fermez la session avec la commande `logout` et connectez-vous de nouveau.

d Saisissez le login *root* et validez avec la touche [Entrée].



Le système demande alors le mot de passe de l'utilisateur pour l'authentifier.

Après saisie du mot de passe et validation, le système envoie un certain nombre d'informations à l'écran dont la console utilisée (tty1) avant d'afficher : `root@raspberrypi:~#`.

Cette ligne signifie que l'utilisateur *root* est connecté sur la machine *raspberrypi* et que son répertoire de travail est ~ (abréviation de « répertoire personnel », ici */root*). Le # à la fin de la ligne signifie que les commandes sont exécutées au niveau super-utilisateur, donc sans restriction.

Une fois connecté, *root* a accès à l'ensemble du système. Il peut lancer toutes les commandes sans essuyer de refus de la part du système. La dangerosité de certaines commandes fait que le compte *root* est utilisé le moins longtemps possible sur les machines Linux.

Arrêter proprement le système

Seul le super-utilisateur est autorisé à arrêter le système.

Linux utilise un système de cache pour l'écriture sur les mémoires de masse (carte micro SD, disque dur, clé USB…). Il stocke le plus longtemps possible les données en mémoire. L'écriture de

données à la fin d'un fichier journal modifie très fréquemment la taille du fichier. Si le fichier reste en mémoire, sa modification est très rapide, et ce n'est que lorsque le système s'arrête que le fichier définitif ainsi que sa taille sont écrits sur la carte micro SD ou le disque dur. En cas d'arrêt brutal, des fichiers restent ouverts, des données sont perdues, les tailles de fichiers ne correspondent plus… Un reset ou une coupure de l'alimentation peuvent laisser le système de fichiers dans un état tel que le système soit dans l'impossibilité de redémarrer. C'est l'une des raisons de l'utilisation d'un onduleur chargé de maintenir une alimentation permanente sur les postes de travail ou serveurs sensibles.

Lors d'un arrêt déclenché par l'administrateur avec la commande `shutdown`, le système demande d'abord aux processus de mettre fin à leur activité. Il force l'arrêt de ceux qui seraient encore en fonctionnement, les caches sont écrits sur les mémoires de masse et les partitions sont démontées proprement. Ce n'est qu'ensuite que le système s'arrête réellement.

Syntaxe

```
shutdown [-t secondes] [-hfFr] heure [message aux utilisateurs]
```

`-t secondes`	Paramètre facultatif. Indique le temps d'attente en secondes entre l'envoi du message d'avertissement aux utilisateurs et l'arrêt des processus.
`-h`	Paramètre facultatif. Indique que la machine doit être arrêtée après l'arrêt du système.
`-f`	Paramètre facultatif. Indique qu'il ne faut pas effectuer de vérification du système de fichiers après le redémarrage.
`-F`	Paramètre facultatif. Indique qu'il faut obligatoirement effectuer une vérification du système de fichiers après le redémarrage.
`-r`	Paramètre facultatif. Indique qu'il faut redémarrer la machine après son arrêt (reboot).
`heure`	Paramètre obligatoire. Indique l'heure à laquelle le système doit s'arrêter. `hh:mm` : heure sous la forme heure:minute. `+m` : nombre de minutes avant l'arrêt. `now` : arrête maintenant, équivaut à +0.
`message aux utilisateurs`	Message envoyé à tous les utilisateurs connectés pour signifier l'heure d'arrêt de la machine et permettre leur déconnexion.

Pour arrêter le Raspberry Pi proprement et immédiatement, utilisez la commande suivante :

```
root@raspberrypi:~# shutdown -h now
```

Attendez l'arrêt complet du système. L'extinction de l'écran n'est pas suffisante pour garantir l'arrêt du système. Il faut observer les LED, en particulier la LED ACT qui indique l'accès du système à la carte micro SD. Sur le Raspberry Pi 3 l'arrêt du système se traduit par plusieurs clignotements de la LED ACT suivis par son extinction. La LED d'alimentation PWR reste allumée. Sur le Raspberry Pi Zero, la seule LED ACT s'éteint.

Le système s'arrête, mais le Raspberry Pi ne possède pas de système d'extinction. À la fin de la séquence d'arrêt (LED rouge PWR allumée seule sur le Raspberry Pi 3, LED rouge ACT éteinte sur le Raspberry Pi Zero), il faut débrancher manuellement l'alimentation.

3.2 Se déplacer dans l'arborescence

Une fois connecté sur un système d'exploitation, il faut prendre ses repères, c'est-à-dire connaître le répertoire de travail (ou répertoire courant) et être capable de se déplacer dans l'arborescence.

3.2.1 Identifier le répertoire courant

Lorsqu'un utilisateur se connecte, le système le place automatiquement dans son répertoire personnel, ce qu'il indique en affichant **~** qui est l'abréviation correspondante :

```
user@raspberrypi ~ $
```

La commande `pwd` (*print name of working directory* = afficher le nom du répertoire de travail) permet de connaître le répertoire courant de façon plus détaillée :

```
user@raspberrypi ~ $ pwd
/home/user
```

Cette commande confirme ici que le répertoire de travail est */home/user* qui est le répertoire personnel de l'utilisateur *user*.

3.2.2 Lister le contenu d'un répertoire

Pour pouvoir se déplacer dans l'arborescence, il faut connaître le nom des répertoires à atteindre. La commande `ls` (*list sort* = liste triée) affiche le contenu d'un répertoire : fichiers et sous-répertoires. Cette commande supporte de nombreuses options, en particulier des options de tri pour la présentation du résultat. Seules les plus utiles sont listées ci-dessous.

Syntaxe

```
ls [options] [nom_de_fichier]
```

Le tableau ci-dessous présente quelques options utiles pour l'utilisation de la commande `ls`. Consultez le manuel de cette commande pour découvrir toutes ses possibilités (`man ls`).

`-F`	Ajoute un **/** à la fin du nom des répertoires.
`-R`	Affiche récursivement le contenu des sous-répertoires. Si un sous répertoire existe, son contenu est affiché.
`-l`	Affiche des informations très complètes sur les fichiers et répertoires (type, droits d'accès, nom du propriétaire…).
`-a`	Affiche tous les fichiers, y compris ceux qui commencent par un . et qui sont des fichiers cachés.
`nom_de_fichier`	Nom de fichier à afficher, accepte les "jokers" * et **?**.

Exemple de commande ls

```
user@raspberrypi /etc $ ls -al n*
-rw-r--r-- 1 root root 8453 mars  30  2012 nanorc
-rw-r--r-- 1 root root  767 mai    2  2011 netconfig
-rw-r--r-- 1 root root   60 juil. 15  2012 networks
-rw-r--r-- 1 root root  475 août  28  2006 nsswitch.conf
-rw-r--r-- 1 root root 1988 mai   18  2012 ntp.conf

network:
total 28
drwxr-xr-x  6 root root 4096 juil. 15  2012 .
drwxr-xr-x 87 root root 4096 juil. 20 19:52 ..
drwxr-xr-x  2 root root 4096 juil. 15  2012 if-down.d
drwxr-xr-x  2 root root 4096 juil. 15  2012 if-post-down.d
drwxr-xr-x  2 root root 4096 juil. 15  2012 if-pre-up.d
drwxr-xr-x  2 root root 4096 juil. 15  2012 if-up.d
-rw-r--r--  1 root root   53 juil. 15  2012 interfaces
lrwxrwxrwx  1 root root   12 juil. 15  2012 run -> /run/network
user@raspberrypi /etc $
```

Le répertoire courant est */etc*. L'utilisateur saisit la commande `ls -al n*` ce qui signifie :

`ls` = afficher la liste des fichiers et dossiers.

`-al` = même les fichiers cachés, avec tous les détails.

`n*` = seulement les fichiers commençant par la lettre n.

Le système affiche cinq fichiers et le contenu du répertoire *network* avant de revenir à l'invite de commande.

La commande `ls` permet de repérer les répertoires : la première lettre de la ligne est un `d` si le

nom correspond à celui d'un répertoire. Un – indique que le nom correspond à un fichier normal.

Les informations rwx… qui suivent le – ou le d sont expliquées plus loin dans ce chapitre : voir le paragraphe Gérer les droits

```
user@raspberrypi /etc $ ls -al | more
total 804
drwxr-xr-x 87 root root    4096 juil. 20 19:52 .
drwxr-xr-x 22 root root    4096 juil. 15  2012 ..
-rw-r--r--  1 root root    2981 juil. 15  2012 adduser.conf
drwxr-xr-x  2 root root    4096 juil. 15  2012 alternatives
drwxr-xr-x  7 root root    4096 juil. 15  2012 apm
drwxr-xr-x  2 root root    4096 juil. 15  2012 apparmor.d
drwxr-xr-x  6 root root    4096 juil. 15  2012 apt
drwxr-xr-x  3 root root    4096 juil. 15  2012 avahi
-rw-r--r--  1 root root    1762 avril 30  2012 bash.bashrc
```

La barre après la commande `ls -al` est obtenue en appuyant simultanément sur les touches [Alt Gr] 6. Cette barre s'appelle un *pipe* ou tuyau. La sortie de la commande `ls` qui aurait dû être envoyée sur l'écran est envoyée vers la commande `more`, chargée de gérer l'affichage. Pour avancer, appuyer sur [Espace] (avance d'une page) ou [Entrée] (avance d'une ligne), pour quitter appuyer sur [q].

Toute commande suivie de `| more` verra sa sortie envoyée à l'écran page par page.

3.2.3 Changer de répertoire

La commande `cd` (*change working directory* = changer de répertoire de travail) permet de changer le répertoire actuel.

Syntaxe

```
cd [nouveau_répertoire]
```

Si `nouveau_répertoire` est omis, l'utilisateur est ramené dans son répertoire personnel. Il faut obligatoirement un espace entre la commande `cd` et le paramètre suivant.

Exemple d'utilisation de la commande cd

```
1 -  user@raspberrypi ~ $ cd /
2 -  user@raspberrypi / $ ls
3 -  bin   dev  home  lost+found  mnt  proc  run   selinux  sys
4 -  usr
5 -  boot  etc  lib   media       opt  root  sbin  srv      tmp
6 -  var
7 -  user@raspberrypi / $ cd usr
8 -  user@raspberrypi /usr $ ls -al
9 -  total 64
10 - drwxr-xr-x  10 root root   4096 juil. 15  2012 .
11 - drwxr-xr-x  22 root root   4096 juil. 15  2012 ..
12 - drwxr-xr-x   2 root root  20480 juil. 20 14:50 bin
13 - drwxr-xr-x   2 root root   4096 juin   2  2012 games
14 - drwxr-xr-x  56 root root   4096 juil. 15  2012 include
15 - drwxr-xr-x  64 root root  12288 juil. 20 14:50 lib
16 - drwxrwsr-x  10 root staff  4096 juil. 15  2012 local
17 - drwxr-xr-x   2 root root   4096 juil. 15  2012 sbin
18 - drwxr-xr-x 144 root root   4096 juil. 20 14:50 share
19 - drwxr-xr-x   2 root root   4096 juin   2  2012 src
20 - user@raspberrypi /usr $ cd bin
21 - user@raspberrypi /usr/bin $ cd
22 - user@raspberrypi ~ $
```

Ligne 1 : l'utilisateur est dans son dossier personnel, il demande à aller à la racine.

Ligne 2 : l'utilisateur est à la racine, il demande la liste des fichiers et dossiers.

Lignes 3 à 6 : le système affiche la liste.

Ligne 7 : l'utilisateur est à la racine, il demande à aller dans le répertoire *usr*.

Ligne 8 : l'utilisateur est dans le répertoire *usr*, il demande la liste des fichiers et dossiers avec détails.

Lignes 9 à 19 : le système affiche le résultat. Toutes les lignes commencent par un **d**, il n'y a donc que des répertoires.

Ligne 20 : l'utilisateur est dans le répertoire *usr*, il demande à aller dans le répertoire *bin*.

Ligne 21 : le répertoire de travail est le dossier */usr/bin*. L'utilisateur saisit **cd** sans paramètre.

Ligne 22 : le répertoire de travail est le dossier personnel de *user* : **~**.

On remarque aux lignes 10 et 11 deux répertoires qui portent des noms bizarres : **.** et **..**.

– Le premier, appelé **.** est le répertoire actuel. C'est ici que le système enregistre le répertoire de travail ou dossier actif.
– Le second appelé **..** est le répertoire parent, c'est-à-dire celui qui se trouve immédiatement au-dessus du dossier actuel dans l'arborescence.

Ces noms particuliers peuvent être utilisés dans les commandes.

Exemple d'utilisation de ..

```
1 - user@raspberrypi ~ $ cd /usr/bin
2 - user@raspberrypi /usr/bin $ cd ..
3 - user@raspberrypi /usr $
```

Ligne 1 : l'utilisateur travaille dans son répertoire personnel, il demande le changement de répertoire *vers /usr/bin*.

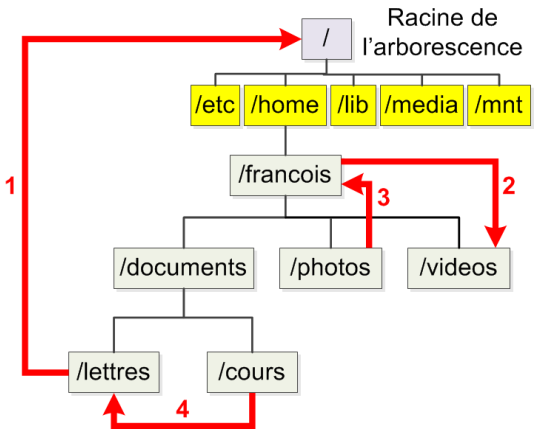
Ligne 2 : le répertoire actif est */usr/bin*. L'utilisateur demande au système de remonter vers le répertoire parent du répertoire de travail.

Ligne 3 : le répertoire de travail est maintenant */usr*.

3.2.4 Chemin relatif et chemin absolu

Il existe deux façons de se déplacer dans l'arborescence : le mode absolu, qui repart systématiquement de la racine, et le mode relatif, qui part du répertoire de travail actif.

Le schéma ci-dessous montre un exemple d'arborescence dans laquelle l'utilisateur souhaite se déplacer.



Le tableau ci-dessous donne les commandes à utiliser par l'utilisateur s'il utilise l'adressage absolu.

Utilisation de l'adressage absolu		
	Point de départ	**Adressage absolu**
1	/home/francois/documents/lettres	cd /
2	cd /home/francois/documents	cd /home/francois/videos
3	cd /home/francois/photos	cd /home/francois
4	cd /home/francois/documents/cours	cd /home/francois/documents/lettres

Le tableau ci-dessous donne les commandes à utiliser par l'utilisateur s'il utilise l'adressage relatif.

Utilisation de l'adressage relatif		
	Point de départ	**Adressage relatif**
1	/home/francois/documents/lettres	cd ../../../..
2	cd /home/francois/documents	cd videos
3	cd /home/francois/photos	cd ..
4	cd /home/francois/documents/cours	cd ../lettres

Les deux méthodes de repérage dans l'arborescence ont des utilisations différentes. En absolu, on est sûr de toujours arriver dans le bon répertoire. L'adressage relatif est souvent utilisé dans les sites web, ce qui permet de déplacer une partie de l'arborescence dans laquelle on a utilisé l'adressage relatif, sans conséquence pour les liens entre les pages.

Pour bien appréhender le mécanisme des déplacements absolus et relatifs, construisez une arborescence (voir la section ci-dessous : Modifier l'arborescence) dans votre dossier personnel comme dans l'exemple précédent, et déplacez-vous dans l'arborescence en utilisant les deux méthodes.

3.2.5 Modifier l'arborescence

L'utilisateur peut rattacher à l'arborescence d'origine des mémoires de masse externes (disque dur USB, clé USB…). Ce "montage" de périphériques externes est traité dans le chapitre Utiliser une mémoire de masse externe.

L'utilisateur peut également avoir besoin de modifier l'arborescence d'origine de Linux. Ce peut être pour créer ou supprimer des sous-répertoires dans son répertoire personnel, afin de classer documents, photos, vidéos...

Créer un répertoire

La création de répertoire se fait avec la commande `mkdir` (*make directory* = créer un répertoire). Si un nom de répertoire est passé en paramètre, il est créé dans le répertoire de travail actuel. Pour le créer ailleurs que dans le répertoire actif, il faut donner le chemin en mode absolu.

Syntaxe

```
mkdir nom_du_répertoire
```

Exemple d'utilisation de la commande mkdir

```
1 - francois@raspberrypi ~ $ ls
2 - documents  photos  videos
3 - francois@raspberrypi ~ $ mkdir musique
4 - francois@raspberrypi ~ $ ls
5 - documents  musique  photos  videos
6 - francois@raspberrypi ~ $ mkdir musique
7 - mkdir: impossible de créer le répertoire « musique »:
8 - Le fichier existe
9 - francois@raspberrypi ~ $
```

Ligne 1 : l'utilisateur liste les fichiers et répertoires présents dans son répertoire personnel.

Ligne 2 : le système affiche les fichiers et répertoires.

Ligne 3 : l'utilisateur crée un répertoire *musique*.

Il n'y a pas de message d'erreur donc la commande a été exécutée sans problème.

Ligne 4 : l'utilisateur demande la liste des fichiers et répertoires.

Ligne 5 : le système affiche les fichiers et répertoires. Un répertoire *musique* a bien été créé.

Ligne 6 : l'utilisateur crée à nouveau un répertoire *musique*.

Lignes 7 et 8 : le système refuse la commande car un répertoire de ce nom existe déjà. Si un fichier *musique* avait existé, le système aurait également refusé d'exécuter la commande.

Pour créer les répertoires parents s'ils n'existent pas, il faut utiliser l'option `-p`. Par exemple, si le répertoire **/data** *n'existe pas, la commande `mkdir -p /data/sauvegarde` créera les deux répertoires.*

Supprimer un répertoire

La suppression de répertoire se fait avec la commande `rmdir` (*remove directory* = supprimer un répertoire) ou avec la commande `rm` si le répertoire n'est pas vide.

Syntaxe

```
rmdir nom_du_répertoire
```

Exemple d'utilisation de la commande rmdir

```
francois@raspberrypi ~ $ ls
```

```
documents  musique  photos  videos
francois@raspberrypi ~ $ rmdir musique
rmdir: échec de suppression de « musique »: Le dossier n'est pas
vide
```

Le répertoire *musique* contient un fichier. Il est impossible de l'effacer avec `rmdir`. Il faut utiliser `rm -r` :

```
francois@raspberrypi ~ $ rm -r musique
francois@raspberrypi ~ $ ls
documents  photos  videos
```

Cette fois le fichier a bien été effacé. La commande `rm` est normalement utilisée pour supprimer un fichier. Avec l'option `-r`, elle supprime un répertoire non vide.

```
francois@raspberrypi ~ $ rmdir -p musique/classique/mozart
francois@raspberrypi ~ $ ls
documents  photos  videos
```

L'utilisateur veut supprimer le répertoire *mozart* et ses parents *musique* et *classique*. L'option `-p` permet cette suppression à condition que le répertoire soit vide. De même si un répertoire parent n'est pas vide, la suppression des parents s'arrête à ce niveau.

3.3 Gérer les fichiers

L'intervention sur la structure de l'arborescence permet de préparer des répertoires pour accueillir les fichiers des utilisateurs. Les commandes de copie, déplacement, suppression… servent à peupler les répertoires.

3.3.1 Copier les fichiers

La copie de fichier et de répertoire sous Linux se fait avec la commande `cp`. Cette commande accepte plusieurs syntaxes pour :

– Copier un fichier vers un autre fichier. Si le fichier n'existe pas, le fichier est copié. Si la destination existe, le fichier est écrasé (voir l'option `-i` ci-dessous).
– Copier plusieurs fichiers vers un répertoire.
– Copier un répertoire et ses sous-répertoires vers un répertoire.

Syntaxe 1

```
cp [-iu] fichier destination
```

`fichier`	Nom du fichier à copier, appelé aussi fichier source.
`destination`	Si `destination` est un nom de fichier, le fichier source est recopié dans le fichier destination. Si `destination` est un nom de répertoire, le fichier source est copié dans ce répertoire.
`-i`	Cette option indique au système d'interroger l'utilisateur avant d'écraser un fichier existant.
`-u`	Le système ne fait pas la copie si le fichier destination a une date de dernière modification égale ou plus récente que celle du fichier source.

Exemple d'utilisation de la syntaxe 1

Le fichier *bolero.wav* (fichier source) est copié vers le fichier *alcyone.wav* (fichier destination).

```
francois@raspberrypi ~/musique/classique $ ls
bolero.wav  miroirs.wav  ravel
francois@raspberrypi ~/musique/classique $ cp bolero.wav
alcyone.wav
francois@raspberrypi ~/musique/classique $ ls
alcyone.wav  bolero.wav  miroirs.wav  ravel
francois@raspberrypi ~/musique/classique $
```

Syntaxe 2

`cp [-iu] fichier_1 fichier_2… fichier_n répertoire`

`fichier_n`	Les fichiers 1 à n sont les fichiers à copier.
`répertoire`	Le répertoire dans lequel les fichiers seront copiés.
`-i` et `-u`	Ces options ont le même effet que dans la syntaxe 1.

Exemple d'utilisation de la syntaxe 2

Deux fichiers, *bolero.wav* et *miroirs.wav*, sont copiés dans le répertoire *ravel*.

```
francois@raspberrypi ~/musique/classique $ ls
bolero.wav  miroirs.wav  ravel
francois@raspberrypi ~/musique/classique $ cp bolero.wav
miroirs.wav ravel
francois@raspberrypi ~/musique/classique $ cd ravel
francois@raspberrypi ~/musique/classique/ravel $ ls
bolero.wav  miroirs.wav
```

Syntaxe 3

`cp [-iuR] répertoire source répertoire destination`

`répertoire_source`	Répertoire à copier.
`répertoire_destination`	Répertoire dans lequel le répertoire source et ses sous-répertoires seront copiés.
`-i` et `-u`	Ces options ont le même effet que dans la syntaxe 1.
`-R`	Copie récursivement les sous-répertoires du répertoire source. La présence de `-R` lors de la copie de répertoires évite l'affichage d'un message d'erreur.

Exemple d'utilisation de la syntaxe 3

Un répertoire est copié vers un autre répertoire contenant déjà les fichiers. Avec l'option `-i`, le système demande confirmation avant d'écraser les fichiers existants.

```
francois@raspberrypi ~ $ cp -R -i musique sauvegarde
cp : voulez-vous écraser
« sauvegarde/musique/classique/bolero.wav » ? o
cp : voulez-vous écraser
« sauvegarde/musique/classique/miroirs.wav » ? o
cp : voulez-vous écraser
« sauvegarde/musique/classique/alcyone.wav » ? o
cp : voulez-vous écraser
« sauvegarde/musique/classique/ravel/bolero.wav » ? o
```

3.3.2 Déplacer et renommer les fichiers

La commande `mv` (*move* = déplacer) permet de déplacer mais aussi de renommer fichiers et répertoires.

Syntaxe

```
mv [-fiuv] source destination
```

`source`	Nom du fichier ou du répertoire source.
`destination`	Nom du fichier ou du répertoire destination. Si `destination` est un nom de répertoire, `mv` déplace les fichiers sources dans ce répertoire. Si `source` et `destination` sont des fichiers dans le même système de fichiers, `mv` renomme le fichier. Si `source` et `destination` sont des fichiers dans des systèmes de fichiers différents, `mv` déplace le fichier. Si `source` est un nom de répertoire, `mv` renomme ce répertoire.
`-f`	Force l'écrasement des fichiers existants.
`-i`	Interroge l'utilisateur avant d'écraser un fichier. Si `-f` et `-i` existent sur la ligne de commande, c'est le dernier qui l'emporte.
`-u`	Le système ne fait pas la copie si le fichier destination a une date de dernière modification égale ou plus récente que celle du fichier source.
`-v`	Indique le nom des fichiers déplacés sur l'écran.

Exemple d'utilisation de la commande mv

L'utilisateur renomme le fichier *bolero.wav* qui devient *bolero_de_ravel.wav*.

```
francois@raspberrypi ~/musique/classique $ ls
alcyone.wav  bolero.wav  miroirs.wav  ravel
francois@raspberrypi ~/musique/classique $ mv bolero.wav
bolero_de_ravel.wav
francois@raspberrypi ~/musique/classique $ ls
alcyone.wav  bolero_de_ravel.wav  miroirs.wav  ravel
```

3.3.3 Supprimer les fichiers

La suppression de fichier se fait avec la commande `rm` (*remove* = supprimer).

Syntaxe

```
rm [-fiRv] nom_de_fichier
```

`nom_de_fichier`	Nom du fichier ou du répertoire à effacer.
`-f`	Ne pas avertir l'utilisateur si un fichier à effacer n'existe pas. Forcer l'effacement sans demander de confirmation si le fichier existe.
`-i`	Interroge l'utilisateur avant d'effacer un fichier. Si `-f` et `-i` existent sur la ligne de commande, c'est le dernier qui l'emporte.

`-R`	Efface récursivement les sous-répertoires du répertoire source et le répertoire lui-même.
`-v`	Indique le nom des fichiers effacés sur l'écran.

Exemple d'utilisation de la commande rm

L'utilisateur décide de supprimer tous les fichiers *.wav* du répertoire courant. L'astérisque * remplace le nom de tous les fichiers. L'option `-v` indique au système d'afficher les noms des fichiers effacés à l'écran.

```
francois@raspberrypi ~/musique/classique $ rm -v *.wav
« bolero_de_ravel.wav » supprimé
« miroirs.wav » supprimé
```

3.3.4 Afficher le contenu d'un fichier

L'affichage du contenu d'un fichier se fait avec la commande `cat`.

Syntaxe

```
cat [-n] nom_de_fichier_1 nom_de_fichier_2
```

`nom_de_fichier_1`	Nom du fichier dont l'utilisateur veut visualiser le contenu. Si plusieurs noms de fichiers sont présents dans la ligne de commande, `cat` affiche leur contenu à l'écran à la suite.
`-n`	Numérote les lignes affichées sur l'écran. Utile quand il faut retrouver une ligne qui pose problème.

Exemple d'utilisation de la commande cat

L'utilisateur affiche le contenu de */etc/timezone* d'abord sans option, puis avec numérotation des lignes.

```
francois@raspberrypi /etc $ cat timezone
Europe/Paris
francois@raspberrypi /etc $ cat -n timezone
     1  Europe/Paris
```

Dans le cas d'un fichier comportant un nombre important de lignes, l'affichage fourni par `cat` défile trop vite. Pensez à utiliser `less` pour faciliter la lecture. La séparation entre la commande `cat passwd` et `less` est le signe pipe, obtenu en appuyant simultanément sur [Alt Gr] 6.

```
francois@raspberrypi /etc $ cat passwd | less
```

3.3.5 Modifier le contenu d'un fichier

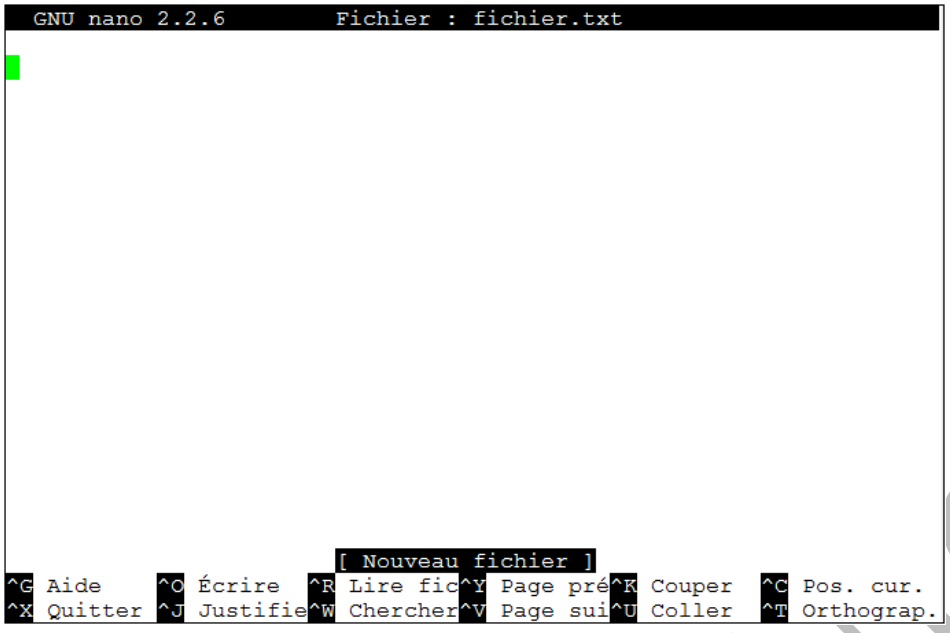
Depuis les origines de Linux, des éditeurs de fichiers ont existé. Plus ou moins pratiques, plus ou moins compliqués à utiliser, ils ont été remplacés vers 2000 par *nano*, un éditeur libre, disponible sur toutes les distributions.

Les éditeurs comme *vi* et *emacs* continuent d'être utilisés pour des besoins plus pointus, notamment pour les recherches et les remplacements.

Syntaxe

```
nano nom_de_fichier
```

Si le fichier existe, *nano* l'ouvre et affiche son contenu. Le déplacement du curseur se fait avec les flèches de direction. Il est possible d'ajouter ou de supprimer des caractères.



En haut à gauche, *nano* affiche son numéro de version. Au centre figure le nom du fichier.

En bas au milieu de l'écran, l'indication **[Nouveau fichier]** montre que le fichier *fichier.txt* n'existait pas et qu'il sera créé par *nano*.

Les deux lignes inférieures de l'écran rappellent les commandes disponibles précédées par le signe **^** qui désigne la touche [Ctrl]. Par exemple pour chercher un mot, l'appui sur [Ctrl] W provoque l'apparition d'une zone intitulée **Recherche**. La saisie d'un mot puis l'appui sur [Entrée] amènent le curseur sur ce mot. La prochaine recherche proposera par défaut le même mot. Un appui sur [Entrée] recherche l'occurrence suivante du mot dans le texte. La saisie d'un autre mot suivi de [Entrée] provoque une nouvelle recherche.

L'écriture du fichier se fait par [Ctrl] O. La combinaison de touches [Ctrl] X sort de *nano* en proposant d'enregistrer le fichier s'il a subi des modifications.

La combinaison de touches [Ctrl] G est la plus importante de *nano*, c'est elle qui permet d'accéder à l'aide du logiciel. La correspondance entre les combinaisons de touches et les touches de fonction y figure, ainsi que d'autres informations utiles.

Pour vous entraîner, créez un fichier avec *nano*. Entrez du texte puis enregistrez le fichier. Rouvrez-le avec *nano* et explorez les fonctions disponibles.

Il faut peu de temps pour prendre en main cet éditeur de texte. Sa maîtrise permet de gagner beaucoup de temps lors d'intervention sur des fichiers de configuration par exemple.

3.3.6 Compresser et décompresser un fichier

Le stockage de nombreux fichiers peu utilisés peut être optimisé par l'utilisation de `gzip`. Cette commande remplace un fichier par sa version compressée, en ajoutant *.gz* après le nom d'origine. Elle réalise également la décompression des fichiers avec l'option `-d`.

Syntaxe

`gzip nom_de_fichier_1 nom_de_fichier_2... nom_de_répertoire`

`nom_de_fichier_n`	Tous les fichiers dont le nom figure dans la ligne de commande seront compressés.
`nom_de_répertoire`	Si un ou plusieurs noms sont des noms de répertoires, ils sont ignorés, sauf si l'option `-r` est présente.
`-d`	Décompresse un fichier compressé.
`-l`	Fournit des informations sur des fichiers compressés (taille du fichier compressé, taille du fichier original, taux de compression, nom du fichier original).
`-r`	Si un ou plusieurs noms de répertoires sont dans la liste, parcourt les répertoires récursivement en compressant individuellement tous les fichiers rencontrés. Le répertoire lui-même n'est pas compressé.
`-t`	Vérifie l'intégrité d'un fichier compressé.
`-v`	Mode bavard (verbose), fournit des informations sur chaque fichier compressé ou décompressé.

Exemple d'utilisation de la commande gzip

L'utilisateur souhaite compresser le fichier *fichier1.txt* qui occupe 35 Ko. La commande `gzip` produit un fichier *fichier1.txt.gz* qui remplace le fichier original mais n'occupe plus que 12 Ko.

```
francois@raspberrypi ~/documents $ ls -al
total 44
drwxr-xr-x 2 francois francois  4096 juil. 22 18:13 .
drwxr-xr-x 7 francois francois  4096 juil. 21 17:32 ..
-rw-r--r-- 1 francois francois 35379 juil. 22 17:48 fichier1.txt
francois@raspberrypi ~/documents $ gzip fichier1.txt
francois@raspberrypi ~/documents $ ls -al
total 24
drwxr-xr-x 2 francois francois  4096 juil. 22 18:13 .

drwxr-xr-x 7 francois francois  4096 juil. 21 17:32 ..
-rw-r--r-- 1 francois francois 12703 juil. 22 17:48
fichier1.txt.gz
```

L'utilisateur veut visualiser les informations sur tous les fichiers *.gz* du répertoire de travail.

```
francois@raspberrypi ~/documents $ gzip -l *.gz
         compressed        uncompressed  ratio uncompressed_name
              12703               35379  64.2% fichier1.txt
```

L'utilisateur lance la décompression de tous les fichiers compressés du répertoire actif.

```
francois@raspberrypi ~/documents $ gzip -d *.gz
```

La compression de *fichier1.txt* est lancée en mode *verbose* (bavard). Les informations relatives à la compression sont affichées.

```
francois@raspberrypi ~/documents $ gzip -v fichier1.txt
```

```
fichier1.txt:    64.2% -- replaced with fichier1.txt.gz
```

L'utilisateur remonte dans son dossier personnel et veut compresser tous les fichiers contenus dans les répertoires et sous-répertoires, avec un affichage à l'écran des opérations réalisées par `gzip`.

```
francois@raspberrypi ~/documents $ cd ..
francois@raspberrypi ~ $ ls
documents  musique  photos  sauvegarde  video
francois@raspberrypi ~ $ gzip -rv *
gzip: documents/fichier1.txt.gz already has .gz suffix --
unchanged
musique/classique/ravel/bolero.wav:      22.2% -- replaced with
musique/classique/ravel/bolero.wav.gz
musique/classique/ravel/alcyone.wav:    -11.1% -- replaced with
musique/classique/ravel/alcyone.wav.gz
sauvegarde/musique/classique/bolero.wav:         22.2% --
replaced with sauvegarde/musique/classique/bolero.wav.gz
sauvegarde/musique/classique/miroirs.wav:       -11.1% --
replaced with sauvegarde/musique/classique/miroirs.wav.gz
sauvegarde/musique/classique/alcyone.wav:        22.2% --
replaced with sauvegarde/musique/classique/alcyone.wav.gz
sauvegarde/musique/classique/ravel/bolero.wav:   22.2% --
replaced with sauvegarde/musique/classique/ravel/bolero.wav.gz
francois@raspberrypi ~ $
```

Il est également possible de décompresser directement les fichiers avec `gunzip` qui accepte les mêmes paramètres que `gzip`. En GNU/Linux, la commande `tar` est également utilisée pour générer des archives compressées au format tar.gz.

3.4 Accélérer la frappe des commandes

La frappe répétitive des commandes peut devenir lassante. Heureusement Linux propose des solutions pour accélérer la frappe.

3.4.1 Rappel des commandes précédentes

La première méthode est l'utilisation des flèches haut et bas pour rappeler les commandes précédemment saisies.

Utilisation des flèches de direction

Les flèches haut et bas donnent accès à la liste des commandes saisies par un utilisateur. Une fois la commande recherchée affichée, vous pouvez la valider à nouveau en appuyant sur la touche [Entrée]. Si vous souhaitez modifier la commande avant de l'utiliser, déplacez-vous à l'aide des flèches droite et gauche, effacez des caractères, ajoutez-en si nécessaire puis validez à nouveau la commande avec la touche [Entrée].

Si vous changez d'utilisateur sur un même poste, l'historique des commandes reste attaché à l'utilisateur. Par exemple l'utilisateur pi ouvre une session root avec `su`, quand il est dans la session de l'utilisateur root il n'a plus accès à son propre historique. Après la commande `exit` qui le fait sortir de la session root, il a de nouveau accès à son historique.

Utilisation de la commande history

La commande `history` mémorise les dernières commandes d'un utilisateur. C'est elle qui est mise en œuvre lorsque l'utilisateur actionne les touches haut et bas pour faire défiler les commandes mémorisées. Cette commande accepte un certain nombre d'options ou paramètres. Voici les plus utiles.

Syntaxe

```
history [!chiffre] [!lettres] [-c]
```

`history`	`history` sans paramètre affiche la liste des dernières commandes entrées par un utilisateur (jusque 500 commandes par défaut). Pour repérer la commande qui vous intéresse : `history|less` affiche le début de la liste. Déplacez-vous vers le haut et le bas avec les flèches de direction, repérez la commande à réutiliser, la touche **q** permet de sortir.
`!16`	Rappelle la ligne de commande n° 16 de la liste affichée par `history`.
`!ls`	Rappelle la dernière commande commençant par `ls` de la liste affichée par `history`.
`-c`	Efface la liste de commandes stockées par `history`.
`!!`	Rappelle la dernière commande.
`!$`	Rappelle le dernier paramètre. Exemple : `mkdir rep ls -l !$`

*La liste complète des commandes d'un utilisateur est visible dans son dossier personnel **$HOME**. C'est le fichier caché **.bash_history** qui contient cette liste.*

Exemple d'utilisation de la commande history

```
user@raspberrypi ~ $ history
    1  ls
    2  rm fichier2.txt
    3  ls
    4  cp fichier1.txt fichier2.txt
    5  ls
    6  ls -al
    7  date
    8  history
user@raspberrypi ~ $ !7
date
samedi 20 juillet 2013, 19:43:11 (UTC+0200)
user@raspberrypi ~ $ !ls
ls -al
total 32
drwxr-xr-x 2 user user 4096 juil. 20 19:42 .
drwxr-xr-x 4 root root 4096 juil. 20 17:51 ..
-rw-r--r-- 1 user user   19 juil. 20 19:20 fichier1.txt
-rw-r--r-- 1 user user   19 juil. 20 19:42 fichier2.txt
user@raspberrypi ~ $ history -c
user@raspberrypi ~ $ history
    1  history
```

3.4.2 Autocomplétion

L'autocomplétion est une autre des méthodes utilisées pour gagner du temps lors de la saisie des lignes de commande. L'appui sur la touche [Tab] termine automatiquement l'écriture de la fin de la commande et des noms de fichiers et de répertoires si une seule possibilité de complétion existe.

Si plusieurs solutions sont possibles, la liste des choix disponibles est présentée. Il suffit de taper une ou plusieurs lettres et à nouveau la touche [Tab] pour terminer la saisie.

Cette fonction est disponible dans le shell (l'interpréteur de ligne de commande) `bash` qui est normalement le shell par défaut exécuté par Raspbian.

Pour vérifier que le shell utilisé sur votre système est `bash` :

d Saisissez `echo $SHELL` sur la ligne de commande et validez. Respectez bien les majuscules !

```
root@raspberrypi:/home/user# echo $SHELL
/bin/bash
```

La réponse du système `/bin/bash` confirme que l'interpréteur de commande en service est bien `bash`.

Si la commande `echo $SHELL` renvoie autre chose, par exemple :

```
$ echo $SHELL
/bin/sh
```

Le shell par défaut a été modifié. Ici c'est `sh` qui est utilisé.

La commande `cat /etc/shells` liste les interpréteurs de commande disponibles sur votre système :

```
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
```

```
/bin/bash
/bin/rbash
```

Vous pouvez changer l'interpréteur de commande par défaut de l'utilisateur *pi* grâce à la commande `chsh`. Le système vous demandera votre mot de passe, saisissez `raspberry`, le mot de passe de l'utilisateur *pi*. Ensuite, saisissez le chemin et le nom de l'interpréteur de commande que vous souhaitez utiliser :

```
$ chsh
Mot de passe :
Changement d'interpréteur de commandes initial pour pi
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
        Interpréteur de commandes initial [/bin/sh]: /bin/bash
```

N'utilisez pas la commande `sudo chsch`, ceci modifierait le shell par défaut de l'utilisateur root.

Fermez la session ([Ctrl] D) puis reconnectez-vous en tant que l'utilisateur *pi*. Votre shell sera `/bin/bash`.

`SHELL` est une variable système qui contient le nom de l'interpréteur de commande. La commande `echo` avec $ devant la variable envoie le contenu de cette variable à l'écran.

Exemple d'utilisation de l'autocomplétion

Dans le dossier */home/user* se trouvent deux fichiers nommés *fichier1.txt* et *fichier2.txt*. Le super-utilisateur souhaite afficher le contenu de *fichier2.txt*.

```
1 - root@raspberrypi:/home/user# ca
2 - cal         caller     case       catchsegv
3 - calendar    captoinfo  cat        catman
4 - root@raspberrypi:/home/user# cat fichier
5 - fichier1.txt  fichier2.txt
6 - root@raspberrypi:/home/user# cat fichier2.txt
7 - Ceci est le contenu du fichier fichier2.tx
8 -
```

Ligne 1 : saisie de `ca` et appui sur [Tab] (deux fois).

Ligne 2 et 3 : le système affiche toutes les commandes qui commencent par `ca` (ce qui peut être utile quand l'utilisateur ne se rappelle plus exactement de la commande).

Ligne 4 : l'utilisateur complète sa commande en saisissant un `t` puis un [Espace] et un `f`, il appuie ensuite sur [Tab]. Le système complète le mot fichier et s'arrête puisqu'il y a deux possibilités : *fichier1.txt* et *fichier2.txt*, qu'il affiche.

Ligne 5 : affichage des deux possibilités trouvées par le système.

Ligne 6 : l'utilisateur complète sa saisie en tapant un `2` puis appuie sur [Tab]. Le système complète le nom de fichier en ajoutant `.txt`. La ligne de commande est maintenant complète. L'utilisateur valide avec la touche [Entrée].

Lignes 7 et 8 : affichage du contenu de *fichier2.txt*.

Même si la description du fonctionnement de l'autocomplétion peut sembler longue, la combinaison du rappel des commandes précédentes et l'utilisation de l'autocomplétion font gagner énormément de temps avec l'habitude.

3.5 Administrer le système

Apprendre à naviguer dans l'arborescence et à gérer les fichiers est une première étape dans la gestion d'un système Linux. L'administration du système est l'aboutissement de cet apprentissage. La plupart des commandes d'administration ne sont accessibles qu'au super-utilisateur *root*.

3.5.1 Gérer les utilisateurs et les groupes

Un utilisateur est un individu identifié par un nom et authentifié par un mot de passe. Même sur un ordinateur utilisé par une seule personne, il est intéressant de créer des utilisateurs ayant des droits différents. Cela permet de séparer et protéger les activités diverses qui peuvent s'exercer sur la même machine.

– Un utilisateur *webmaster* se connectera directement sur le répertoire du site web pour faire les mises à jour de celui-ci.
– Un utilisateur *sauvegarde* pourra lancer les sauvegardes.

Si plusieurs personnes se partagent la même machine, cela permet de séparer et protéger leurs données.

L'utilisateur *root* ne sera sollicité que pour des opérations de maintenance du système.

Un groupe est un ensemble d'utilisateurs. Lors de la création de chaque utilisateur, Debian crée par défaut un groupe du même nom que cet utilisateur et place l'utilisateur dans ce groupe.

C'est le super-utilisateur ou un utilisateur devenu super-utilisateur qui gère les utilisateurs et les groupes.

Créer un groupe

La création d'un groupe se fait avec la commande `addgroup`. Le groupe créé est vide.

Syntaxe

```
addgroup nom_groupe
```

Exemple d'utilisation de la commande addgroup

```
root@raspberrypi:/etc# addgroup musiciens
Ajout du groupe « musiciens » (GID 1004)...
Fait.
```

Le groupe *musiciens* a été créé. Il s'est vu attribuer le GID (*Group ID* = identifiant de groupe) 1004 par le système.

Seul le groupe a été créé, il ne contient aucun membre.

Ajouter un utilisateur

L'ajout d'un utilisateur se fait par la commande `adduser`. Il est possible d'entrer directement les informations sur la ligne de commande, ou de laisser le système agir par défaut.

Syntaxe

```
adduser [options] nom_d_utilisateur
```

`nom_d_utilisateur`	Nom de l'utilisateur à créer.

`--home réper-toire_personnel`	Il est possible d'attribuer un répertoire personnel dont le nom est différent de celui de l'utilisateur. Par défaut, ce répertoire est créé dans */home*.
`--ingroup groupe_primaire`	Indique le groupe principal de l'utilisateur. Le groupe doit exister avant la création du compte de l'utilisateur.

Exemple d'utilisation de la commande adduser

L'utilisateur est créé avec la commande `adduser` sans préciser d'option. Le système demande le mot de passe de l'utilisateur (puis sa confirmation). Le mot de passe ne s'affiche pas par mesure de sécurité.

```
root@raspberrypi:/etc# adduser ravel
Ajout de l'utilisateur « ravel » ...
Ajout du nouveau groupe « ravel » (1005) ...
Ajout du nouvel utilisateur « ravel » (1003) avec le groupe
« ravel » ...
Création du répertoire personnel « /home/ravel »...
Copie des fichiers depuis « /etc/skel »...
Entrez le nouveau mot de passe UNIX
Retapez le nouveau mot de passe UNIX
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur ravel
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur
proposée
        Nom complet []: Maurice RAVEL
        N° de bureau []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Cette information est-elle correcte ? [O/n]o
```

Ensuite, quelques renseignements administratifs facultatifs sont demandés, avant la validation. Sans précision dans les options, le dossier personnel de l'utilisateur est créé dans */home*, et il fait partie du groupe portant le même nom que lui (la commande `groups` ci-dessus affiche les groupes dont fait partie un utilisateur).

Pour vérifier que les opérations se sont bien déroulées, prenez l'identité de *ravel* et vérifiez l'existence de son répertoire personnel et son appartenance au groupe *ravel*.

```
root@raspberrypi:/etc# su ravel
ravel@raspberrypi /home $ cd ~
ravel@raspberrypi ~ $ pwd
/home/ravel
ravel@raspberrypi ~ $ groups  ravel
```

Les commandes `addgroup` et `adduser` sont spécifiques à Debian et aux distributions qui en dérivent. Il existe des équivalents dans les autres branches de Linux.

Changer le mot de passe d'un utilisateur

L'administration est confrontée à un problème récurrent : un utilisateur a perdu/oublié son mot de passe. Sous Debian, les mots de passe sont stockés dans le fichier */etc/shadow* mais sous une forme chiffrée qu'aucun système actuel n'est capable de déchiffrer.
Une attaque avec un dictionnaire contenant des mots de passe usuels peut cependant aboutir sans avoir besoin de déchiffrer le mot de passe.

La commande `passwd` permet de modifier le mot de passe d'un utilisateur. Seuls le super-utilisateur *root* ou le propriétaire d'un mot de passe peuvent le changer.

Syntaxe

```
passwd nom_d_utilisateur
```

`-d`	Seul root peut utiliser cette option qui désactive le mot de passe. L'utilisateur se connecte sans mot de passe.
`-l`	Verrouille (*lock*) le compte de l'utilisateur en mettant un `!` devant le mot de passe chiffré. Le mot de passe n'est plus reconnu.
`-u`	Déverrouille (*unlock*) le compte d'un utilisateur. Enlève le `!` ajouté par l'option `-l`.

Exemple d'utilisation de la commande passwd

L'utilisateur *mozart* essaye de modifier le mot de passe de l'utilisateur *wolfgangamadeus*. Il n'a pas le droit de le faire.

```
mozart@raspberrypi ~ $ passwd wolfgangamadeus
passwd : Vous ne pouvez pas afficher ou modifier les informations
de mot de passe de wolfgangamadeus.
```

Le super-utilisateur modifie le mot de passe de l'utilisateur *wolfgangamadeus*. Il doit saisir deux fois le mot de passe. Si les deux saisies diffèrent, le mot de passe n'est pas modifié.

```
root@raspberrypi:~# passwd wolfgangamadeus
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd : le mot de passe a été mis à jour avec succès
```

Modifier un utilisateur

Les données d'un utilisateur se modifient avec la commande `usermod`.

Syntaxe

```
usermod [options] login_de_l_utilisateur
```

`-a`	Ajoute l'utilisateur à un groupe supplémentaire. Ne s'utilise qu'avec l'option `-G`.
`-d réper-toire_personnel`	Modifie le répertoire personnel de l'utilisateur. Si l'option `-m` est présente, le nouveau répertoire est créé s'il n'existe pas et les fichiers contenus dans le répertoire actuel de l'utilisateur y sont déplacés.
`-g groupe`	Modifie le groupe principal de l'utilisateur. Le groupe doit être créé avant d'utiliser la commande.

`-G groupe1,groupe2,...`	Ajoute l'utilisateur à tous les groupes secondaires listés après `-G`. Attention, si l'utilisateur appartenait à un groupe qui n'est pas listé, il n'en fera plus partie ! Voir l'option `-a` qui permet d'ajouter l'utilisateur à des groupes sans modifier les groupes auxquels il appartient.
`-l nouveau_login`	Le nom de l'utilisateur (son login) devient `nouveau_login`. Rien d'autre ne change, par exemple le répertoire personnel conserve le nom correspondant à l'ancien login. Il faudra le renommer manuellement ou avec l'option `-d`.
`-m`	Déplace le contenu de l'ancien répertoire personnel de l'utilisateur vers son nouveau répertoire personnel.
`-s nouveau_shell`	Modifie le shell de l'utilisateur lors de sa connexion. Pour associer le shell par défaut, laissez le champ `nouveau_shell` vide. Tous les shells disponibles sont listés dans */etc/shells*. Le shell doit être indiqué avec le chemin absolu.

Exemple d'utilisation de la commande usermod

L'utilisateur *user* appartient uniquement au groupe *user* :

```
user@raspberrypi /etc $groups
user
```

L'administrateur ajoute l'utilisateur *user* au groupe *musiciens* tout en le maintenant dans le groupe *user* :

```
root@raspberrypi:/etc# usermod -G musiciens user
```

L'utilisateur *user* appartient maintenant aux deux groupes :

```
user@raspberrypi /etc $ groups
user musiciens
```

Supprimer un utilisateur

La suppression d'un utilisateur se fait avec la commande `userdel`.

Syntaxe

```
userdel [options] login_de_l_utilisateur
```

`-f`	Détruit le compte de l'utilisateur, même s'il est connecté. Détruit également le répertoire principal de cet utilisateur, même s'il est utilisé par un autre utilisateur.

`-r`	Les fichiers du répertoire de l'utilisateur sont supprimés avec le répertoire personnel. Par contre, les fichiers appartenant à cet utilisateur et situés ailleurs devront être recherchés et détruits manuellement.

Exemple de suppression d'un utilisateur

L'administrateur essaye de supprimer l'utilisateur *user* qui est connecté sur le système. Le système signale que l'utilisateur est connecté. Lorsque le super-utilisateur prend l'identité de *user*, il peut se connecter.

```
root@raspberrypi:/etc# userdel user
userdel: user user is currently used by process 2241
root@raspberrypi:/etc# su user
user@raspberrypi /etc $
```

L'administrateur force avec `-f` la suppression de l'utilisateur *user*. Le système lui signale que l'utilisateur est connecté mais le supprime. Quand le super-utilisateur essaye de prendre l'identité de *user*, le système refuse.

```
root@raspberrypi:/etc# userdel -f user
userdel: user user is currently used by process 2241
root@raspberrypi:/etc# su user
No passwd entry for user 'user'
root@raspberrypi:/etc#
```

3.5.2 Gérer les dépôts

Linux utilise un système de mise à jour basé sur des *repositories* (dépôts en français). Un dépôt est un miroir (accessible via le protocole HTTP), une copie du dépôt d'origine de la distribution. Lors de l'installation, il est possible de choisir un miroir sur certaines distributions. Sur d'autres distributions, le miroir est attribué automatiquement en fonction de la situation géographique de la station de travail.

Sur la distribution Debian et ses dérivés, le fichier */etc/sources.list* contient les sites miroir sur lesquels le système doit aller chercher la liste des fichiers disponibles, leur version… Les fichiers et leurs dépendances sont enregistrés sous forme de paquets contenant tout ce qui est nécessaire à l'installation.

APT (*Advanced Packaging Tool* = gestionnaire avancé de paquets) est l'outil utilisé par Debian pour mettre à jour et installer les paquets. Les paquets Debian sont au format *.deb*. Ce format a également été adopté par les distributions dérivées de Debian.

Dans Raspbian, le fichier *sources.list* contient par défaut :

– La branche *main* qui est la branche principale de Raspbian, complètement libre.
– La branche *contrib* dans laquelle les paquets eux-mêmes sont libres, mais utilisent des dépendances non libres.
– La branche *non-free* dont les paquets ne sont pas libres mais simplement redistribuables. Debian considère que la branche non libre ne fait pas partie de Debian. Elle n'est fournie que par commodité pour les utilisateurs.
– *rpi* destiné à recevoir les paquets spécifiques à Raspbian.

Le type *deb* contient des programmes sous forme binaire exécutables par le microprocesseur. Le type *deb-src* correspond aux sources des paquets. Les programmes sont fournis sous forme de texte et sont appelés sources. Ils pourront être utilisés par exemple pour modifier et recompiler certains paquets. Le type *deb-src* n'est généralement pas nécessaire pour une utilisation basique du Raspberry Pi.

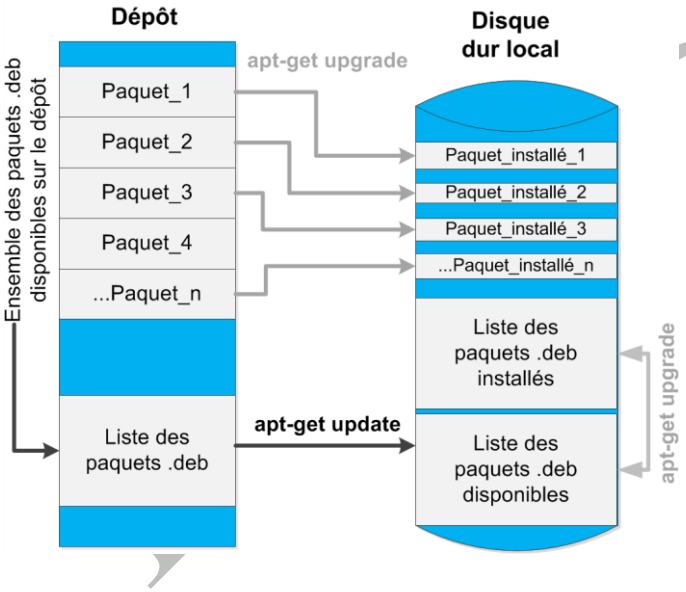
Contenu du fichier /etc/sources.list de la distribution Raspbian wheezy

```
deb http://mirrordirector.raspbian.org/raspbian wheezy main
contrib non-free rpi
```

http://... est le lien vers le miroir et wheezy est le nom de la version de Debian dont dérive Raspbian.

3.5.3 Tenir le système à jour

APT se compose de plusieurs programmes. L'un d'eux `apt-get` est utilisé pour la mise à jour du système et l'installation de nouveaux paquets.



La mise à jour commence obligatoirement par la commande `apt-get update` qui va synchroniser la liste des paquets disponibles, présente sur le disque dur local, avec la liste présente sur le dépôt.

Une fois la synchronisation achevée les commandes `apt-get upgrade` et `apt-get dist-upgrade` peuvent être exécutées.

Dans certains cas il est utile de mettre à jour le firmware du Raspberry Pi. Ce programme est l'équivalent du BIOS d'un PC, c'est par son intermédiaire que le SoC s'adresse au matériel de la carte. Dans ce cas la commande `rpi-update` sera exécutée avant `apt-get dist-upgrade`.

Si vous avez des données importantes sur la carte micro SD, réalisez systématiquement une sauvegarde de la carte avant de lancer les opérations de mise à jour.

Syntaxe

`apt-get update`	Synchronise les fichiers énumérant les paquets disponibles côté dépôt et côté station de travail.
`apt-get upgrade`	Installe les versions les plus récentes des paquets installés sur le disque dur local. N'ajoute pas de paquets qui n'auraient pas été installés. Ne supprime pas de paquets installés sur la machine.
`apt-get dist-upgrade`	Réalise la mise à jour des paquets en gérant les éventuels changements de dépendance des nouveaux paquets. Favorise la mise à jour des paquets les plus importants au détriment des paquets mineurs.

Exemple d'utilisation de la commande apt-get

```
root@raspberrypi:~# apt-get update
0% [Working]
Hit http://mirrordirector.raspbian.org wheezy InRelease
0% [Waiting for headers]
Hit http://archive.raspberrypi.org wheezy InRelease
0% [Working]
0% [InRelease gpgv 14.9 kB]
7% [Working]
7% [InRelease gpgv 7,737 B] [Waiting for headers]
Hit http://mirrordirector.raspbian.org wheezy/main armhf Packages
13% [InRelease gpgv 7,737 B]
...
Reading package lists... 83%
Reading package lists... Done...
```

Mise à jour de la liste des paquets disponibles. Une fois la liste des paquets mise à jour avec `apt-get update`, il est possible de lancer la mise à jour des paquets avec `apt-get upgrade`.

```
root@raspberrypi:~# apt-get upgrade
Reading package lists... Done
Building dependency tree... 87%
...
Reading state information... Done

The following packages have been kept back:
  alsa-base gnome-themes-standard lxde lxsession omxplayer
openssh-client
  openssh-server scratch wpasupplicant
The following packages will be upgraded:
  alsa-utils apt apt-utils aptitude aptitude-common base-files
bash binutils
  bsdutils bzip2 ca-certificates console-setup console-setup-
linux consolekit
  coreutils cpio cpp cpp-4.6 cryptsetup-bin curl dbus dbus-x11
...
312 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
Need to get 263 MB of archives.
After this operation, 25.8 MB disk space will be freed.
Do you want to continue [Y/n]? y
0% [Working]
Get:1 http://archive.raspberrypi.org/debian/ wheezy/main
libpixman-1-0 armhf 0.26.0-4+raspi [1,200 kB]
```

```
0% [Waiting for headers] [1 libpixman-1-0 156 kB/1,200 kB 13%]
...
```

Dans le cas ci-dessus, la mise à jour va nécessiter le téléchargement de 312 paquets, soit 263 Mo. Les paquets sont ensuite décompressés, enfin la mise à jour est effectuée. Cette opération peut durer près de 10 minutes sur le Raspberry Pi. La durée dépend des différences existantes entre les paquets installés et les paquets disponibles.

Il est conseillé de mettre régulièrement la distribution à jour, pour installer les dernières versions des paquets qui corrigent des failles ou des bugs.

3.5.4 Installer/supprimer un programme

L'installation d'un programme sur Raspbian passe également par l'utilisation de la commande `apt-get`. L'exécution de la commande `apt-get update` avant l'installation d'un programme garantit que la version du programme qui va être installé sera bien la plus récente.

Le programme à installer doit être disponible dans un dépôt présent dans sources.list. À défaut il faudra ajouter, avec nano ou avec la commande `add-apt-repository`, le dépôt susceptible de fournir le paquet correspondant, ou installer le programme en suivant les indications du développeur.

Syntaxe

```
apt-get install nom_du_programme
apt-get remove nom_du_programme
```

Exemple d'installation et de suppression de programme

L'utilisateur souhaite exécuter le programme `tree`, qui est un utilitaire affichant à l'écran l'arborescence du système de fichiers.

```
root@raspberrypi:~# tree /home/user
-bash: /usr/bin/tree: Aucun fichier ou dossier de ce type
```

L'utilisateur installe le programme `tree`, disponible dans le dépôt Raspbian.

```
root@raspberrypi:~# apt-get install tree
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les NOUVEAUX paquets suivants seront installés :
  tree
0 mis à jour, 1 nouvellement installés, 0 à enlever et 10 non mis
à jour.
Il est nécessaire de prendre 0 o/43,4 ko dans les archives.
Après cette opération, 108 ko d'espace disque supplémentaires
seront utilisés.
Sélection du paquet tree précédemment désélectionné.
(Lecture de la base de données... 54271 fichiers et répertoires
déjà installés.)
Dépaquetage de tree (à partir de .../tree_1.6.0-1_armhf.deb) ...
Traitement des actions différées (« triggers ») pour « man-db »...
Paramétrage de tree (1.6.0-1) ...
```

`tree` est maintenant disponible pour afficher l'arborescence du répertoire personnel de l'utilisateur *user*.

```
root@raspberrypi:~# tree /home/user
/home/user
├── documents
│   ├── fichier1.txt
│   └── fichier2.txt
├── musique
│   └── bolero_de_ravel.txt
└── videos
```

```
3 directories, 3 files
```

L'utilisateur désinstalle maintenant le programme `tree` et confirme la suppression du programme.

```
root@raspberrypi:~# apt-get remove tree
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
0 mis à jour, 0 nouvellement installés, 1 à enlever et 10 non mis
à jour.
Après cette opération, 108 ko d'espace disque seront libérés.
Souhaitez-vous continuer [O/n] ? o
(Lecture de la base de données... 54279 fichiers et répertoires
déjà installés.)
Suppression de tree ...
Traitement des actions différées (« triggers ») pour « mandb »...
```

Le programme est supprimé : la commande `tree` ne fonctionne plus.

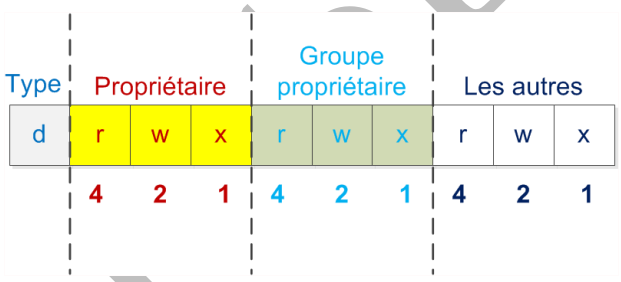
```
root@raspberrypi:~# tree
-bash: /usr/bin/tree: Aucun fichier ou dossier de ce type
```

3.5.5 Gérer les droits

La commande `ls -l` affiche des informations sur les fichiers et répertoires, en particulier les droits attachés à chaque fichier, ainsi que le propriétaire du fichier et le groupe propriétaire.

```
root@raspberrypi:/home/user# ls -l
total 12
-rwxrw-rw- 1 user user  120 Jul 24 13:14 fichier1.txt
-rwxrw-rw- 1 user user  180 Jul 24 13:14 fichier2.txt
drw-r--r-- 2 user user 4096 Jul 24 13:15 lettres
root@raspberrypi:/home/user#
```

Le début de chacune des lignes donne des informations sur le fichier ou le répertoire dont le nom figure à l'extrémité droite de la ligne. Le premier bloc de 10 caractères concerne les droits attachés au fichier ou au répertoire listé.



La première lettre indique le type du fichier :

d pour un répertoire (*directory* en anglais).

- pour un fichier ordinaire.

c pour un périphérique de type caractère (liaison série, modem...).

b pour un périphérique de type bloc (disque dur).

Les neuf caractères suivants sont répartis en trois blocs :

– Les droits du propriétaire du fichier.

– Les droits du groupe propriétaire du fichier.

– Les droits de tous les autres.

Chaque bloc comprend trois lettres qui, lorsqu'elles sont présentes, indiquent que le droit correspondant est disponible. Quand le droit n'existe pas, la lettre est remplacée par un tiret **-**.

r droit de lire.

w droit d'écrire.

x droit d'exécuter.

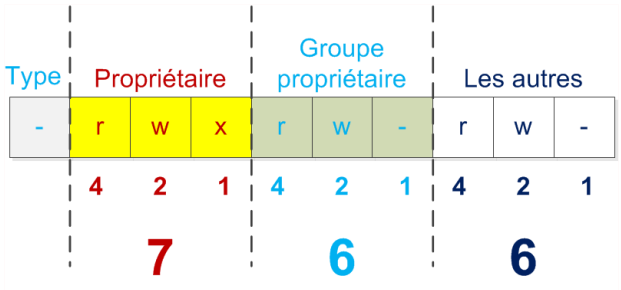
- le droit n'existe pas.

Les droits peuvent également être indiqués en utilisant les lettres ou les chiffres inscrits sous chacune des cases.

Suivent ensuite deux noms : ***user user*** qui indiquent pour le premier le nom du propriétaire du fichier, pour le second le nom du groupe propriétaire du fichier. Ici l'utilisateur *user* appartient au groupe *user*, d'où la répétition du nom. Ces informations proviennent de l'utilisateur qui a créé le fichier : son login et son groupe principal.

Analyse des droits sur un fichier

```
-rwxrw-rw- 1 user user  120 Jul 24 13:14 fichier1.txt
```



Le fichier appartient à l'utilisateur *user* et au groupe *user*.

Les droits correspondent à un fichier car la première lettre est un -.

Le propriétaire possède les droits `r`, `w` et `x` : il peut lire, écrire (modifier, supprimer) et exécuter ce fichier (ce qui a peu d'intérêt dans le cas présent d'un simple fichier texte). Les droits `rwx` correspondent à 4+2+1 et peuvent s'exprimer par le chiffre 7.

Les membres du groupe *user* autres que le propriétaire *user* ont le droit de lire et écrire ce fichier (`rw`). Les droits `rw` correspondent à 4+2 et peuvent s'exprimer par le chiffre 6.

Les utilisateurs qui ne sont ni *user* ni membres du groupe *user* (tous les autres) ont également le droit de lire et écrire ce fichier (`rw`). Les droits `rw` correspondent à 4+2 et peuvent s'exprimer par le chiffre 6.

Les droits de ce fichier peuvent s'écrire `rwxrw-rw-` ou `766`, qui est équivalent.

Les droits sous Linux sont exprimés en octal, système de numération à base 8. Chaque groupe de 3 signes représente 1 bit et la valeur numérique est la traduction en octal du nombre binaire correspondant. rwxrw-rw- est équivalent à 111110110 en binaire et 766 en octal.

Droits sur un répertoire

Les droits sur un répertoire ont une signification différente des droits sur un fichier.

r afficher le contenu du répertoire avec **ls**.

w créer ou supprimer des fichiers dans le répertoire avec **mv**, **cp**, **rm**.

x accéder au répertoire avec la commande **cd**.

- le droit n'existe pas.

Si le droit `w` est positionné sur un répertoire, le droit `x` doit être positionné pour autoriser l'accès au répertoire, et r doit également être positionné pour pouvoir accéder aux fichiers contenus dans le répertoire. Lorsque `w` est positionné sur un répertoire, un utilisateur peut supprimer tous les fichiers qu'il contient, même s'il n'a pas de droits explicites sur ces derniers. Le Sticky Bit permet de limiter

ce droit en limitant au seul propriétaire d'un fichier le droit de le supprimer.

Droits étendus

SUID (*Set owner User ID up on execution* = exécuter avec l'identité du propriétaire) permet d'exécuter un programme avec les droits de son propriétaire au lieu des droits de celui qui lance l'exécution. Il n'a pas d'effet sur les répertoires. L'application de SUID à un fichier dont root est propriétaire pourrait permettre à un attaquant d'exécuter un programme avec les droits de root, ce qui constitue un risque pour la sécurité.

SGID (*Set Group ID up on execution* = exécuter avec l'identité du groupe) appliqué à un fichier a le même effet que SUID mais au niveau du groupe, avec le même risque pour la sécurité. Appliqué à un répertoire, SGID fait hériter son groupe propriétaire à tous les fichiers et répertoires créés dans ce répertoire. Dans un partage, ceci permet à tous les membres d'un groupe de modifier des fichiers communs.

Sticky Bit positionné sur un répertoire, empêche un utilisateur qui ne serait pas propriétaire d'un fichier de le supprimer.

Changement du propriétaire et du groupe propriétaire

Le propriétaire et le groupe propriétaire d'un fichier ou d'un répertoire sont positionnés à la création, avec le login et le groupe principal de celui qui les crée.

Le changement de propriétaire se fait avec la commande `chown`, celui du groupe propriétaire avec `chgrp`. Seul le super-utilisateur peut changer le propriétaire et le groupe d'un fichier ou d'un répertoire dont il n'est pas propriétaire.

Lorsque le propriétaire d'un fichier est changé, l'ancien propriétaire n'a plus de droits sur ce fichier. S'il appartient au même groupe que le nouveau propriétaire, il aura les droits du groupe. S'il n'appartient pas au même groupe que le nouveau propriétaire, il aura les droits de tous les autres.

Syntaxe

```
chown nouveau_propriétaire nom_du_fichier
chgrp nouveau_propriétaire nom_du_fichier
```

Exemple de changement de l'utilisateur propriétaire d'un fichier

L'utilisateur *user* essaye de rendre *mozart* propriétaire de son fichier. Le système indique que cette opération n'est pas permise.

```
user@raspberrypi ~ $ chown mozart fichier1.txt
chown: changing ownership of `fichier1.txt': Operation not
permitted
```

Seul le super-utilisateur peut changer le propriétaire d'un fichier

```
root@raspberrypi:/home/user# chown mozart fichier1.txt
root@raspberrypi:/home/user# ls -l
total 12
-rw-r--r-- 1 mozart user  120 Jul 24 13:14 fichier1.txt
-rw-r--r-- 1 user   user  180 Jul 24 13:14 fichier2.txt
drw-r--r-- 2 user   user 4096 Jul 24 13:15 lettres
```

Exemple de changement du groupe propriétaire d'un fichier

Le super-utilisateur change le groupe propriétaire de *fichier1.txt*.

```
root@raspberrypi:/home/user# chgrp mozart fichier1.txt
root@raspberrypi:/home/user# ls -l
total 12
-rw-r--r-- 1 mozart mozart  120 Jul 24 13:14 fichier1.txt
-rw-r--r-- 1 user   user    180 Jul 24 13:14 fichier2.txt
drw-r--r-- 2 user   user   4096 Jul 24 13:15 lettres
```

Modification des droits

La modification des droits des fichiers et répertoires se fait avec la commande `chmod`. Le propriétaire d'un fichier et le super-utilisateur peuvent tous deux exécuter la commande `chmod`.

`chmod` fonctionne de façon littérale (mode symbolique) en faisant des additions et des soustractions de droits, ou de façon numérique en appliquant directement l'ensemble des droits.

Syntaxe

```
chmod [u g o a] [+ - =] [r w x] nom_du_fichier
chmod NNN nom_du_fichier
```

u	Appliquer les modifications à l'utilisateur propriétaire (*user*).
g	Appliquer les modifications au groupe propriétaire (*group)*.
o	Appliquer les modifications aux autres (*others*).
a	Appliquer les modifications aux trois blocs (*all*).
+	Ajouter un droit.
−	Retirer un droit.
=	Écraser un droit.
r	Droit de lire.
w	Droit d'écrire.
x	Droit d'exécuter.

Exemple d'utilisation de chmod en mode symbolique

Un fichier nommé *fichier1.txt* possède les droits *rwxrwxrwx*.

```
root@raspberrypi:/home/user# ls -l
total 4
-rwxrwxrwx 1 user user 180 Jul 24 13:14 fichier1.txt
```

Pour tous les utilisateurs (*a*), le super-utilisateur écrase les droits (=) et les remplace par lecture (*r*).

```
root@raspberrypi:/home/user# chmod a=r fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-r--r--r-- 1 user user 180 Jul 24 13:14 fichier1.txt
```

Pour le groupe seulement (*g*), le super-utilisateur ajoute (+) le droit d'exécuter (*x*).

```
root@raspberrypi:/home/user# chmod g+x fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-r--r-xr-- 1 user user 180 Jul 24 13:14 fichier1.txt
```

Pour le propriétaire seulement (*u*), le super-utilisateur ajoute (+) les droits d'écrire et d'exécuter (*wx*).

```
root@raspberrypi:/home/user# chmod u+wx fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-rwxr-xr-- 1 user user 180 Jul 24 13:14 fichier1.txt
```

Pour le propriétaire et le groupe (*ug*), le super-utilisateur retire (-) le droit d'exécuter (*x*).

```
root@raspberrypi:/home/user# chmod ug-x fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-rw-r--r-- 1 user user 180 Jul 24 13:14 fichier1.txt
root@raspberrypi:/home/user#
```

Exemple d'utilisation de `chmod` en mode numérique

Avec un peu d'habitude cette méthode est très rapide. Elle permet de configurer en une seule fois tous les droits sur un fichier ou un répertoire. Combinée avec le rappel des commandes précédentes, elle permet un gain de temps appréciable.

Le fichier *fichier1.txt* a les droits *rw-r--r--*.

```
root@raspberrypi:/home/user# ls -l
total 4
-rw-r--r-- 1 user user 180 Jul 24 13:14 fichier1.txt
```

Le super-utilisateur donne tous les droits à tout le monde.

```
root@raspberrypi:/home/user# chmod 777 fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-rwxrwxrwx 1 user user 180 Jul 24 13:14 fichier1.txt
```

Le super-utilisateur donne tous les droits mais seulement au propriétaire, le groupe et les autres n'ont aucun droit.

```
root@raspberrypi:/home/user# chmod 700 fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-rwx------ 1 user user 180 Jul 24 13:14 fichier1.txt
```

Le super-utilisateur laisse tous les droits au propriétaire, et le droit de lire et d'écrire au groupe et aux autres.

```
root@raspberrypi:/home/user# chmod 766 fichier1.txt
root@raspberrypi:/home/user# ls -l
total 4
-rwxrw-rw- 1 user user 180 Jul 24 13:14 fichier1.txt
```

Lors de la création d'un fichier, un masque est appliqué aux droits par défaut de Raspbian. La commande `umask` retourne la valeur 0022 qui correspond à ----w--w-. Ce masque est soustrait des droits fondamentaux de la distribution. Les fichiers créés ne sont modifiables ni par le groupe propriétaire ni par les autres.

3.5.6 Connaître l'occupation de la carte micro SD

Pour connaître l'occupation de la carte micro SD et des éventuels périphériques de stockage de masse connectés au Raspberry Pi, Linux dispose de la commande `df` qui indique la taille de

chaque système de fichier, l'espace occupé et l'espace restant. Par défaut df affiche les résultats en nombre de blocs. Les options permettent d'obtenir un affichage plus lisible.

Syntaxe

```
df [-hkm]
```

-h	Ajoute à chaque valeur affichée une lettre d'unité facilitant la lecture. C'est la forme le plus lisible pour l'utilisateur.
-k	Affiche les valeurs en nombre de blocs de 1 Ko
-m	Affiche les valeurs en nombre de blocs de 1 Mo

Exemple d'utilisation de la commande df

```
pi@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        15G  4.0G   11G  29% /
devtmpfs        459M     0  459M   0% /dev
tmpfs           463M     0  463M   0% /dev/shm
tmpfs           463M  6.6M  457M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           463M     0  463M   0% /sys/fs/cgroup
/dev/mmcblk0p1   63M   21M   43M  33% /boot
tmpfs            93M     0   93M   0% /run/user/1000
```

A partir de ces données il est possible de déterminer quand un système de fichier ou un périphérique de stockage nécessite une intervention parce que trop rempli. Par exemple ci-dessus on voit que / a 4Go occupés sur les 15 Go disponibles.

3.5.7 Écrire un script shell

Un script shell est un fichier texte contenant un ensemble de commandes, habituellement saisies sur la ligne de commande. Elles sont regroupées sous la forme d'un fichier dont l'exécution peut être lancée comme une commande Linux habituelle.

Ce paragraphe ne prétend pas fournir en quelques lignes une formation à l'écriture de scripts shell. L'objectif plus modeste est de montrer le mécanisme depuis l'écriture d'un script jusqu'à son exécution. Une fois l'intérêt de ce type de programmation découvert, un ouvrage comme *Scripts Shell* de J.M Barabgér et T. Schomaker - Éditions ENI sera un complément indispensable.

À quoi sert un script shell ?

Un script shell sert à automatiser des tâches qu'un administrateur doit faire de façon répétitive. Cela peut être par exemple une sauvegarde régulière ou une action déclenchée par la présence d'un fichier particulier.

Un script de sauvegarde

Pour montrer de façon simple l'utilité d'un script Shell, voici un script de sauvegarde du répertoire

personnel d'un utilisateur.

Cahier des charges

Un utilisateur se sert régulièrement de son Raspberry Pi. Les documents qu'il réalise sont enregistrés dans son répertoire personnel. Il souhaite sauvegarder, lorsqu'il a fini de travailler, l'ensemble de ses répertoires ainsi que les fichiers qu'ils contiennent sur une clé USB. Le nom du fichier de sauvegarde doit contenir la date du jour pour faciliter le classement et la recherche.

La clé USB doit être montée au préalable. Le montage de la clé USB sur le système de fichiers est détaillé dans le chapitre Utiliser une mémoire de masse externe.

Contenu du script sauve.sh

```
#!/bin/bash
# Fichier de sauvegarde du répertoire personnel de l'utilisateur pi

# Le nom du fichier de sauvegarde est la date du jour au format AAAAMMJJ
# La clé USB est montée sur le répertoire /mnt/cle_usb

# Enregistrement de la date dans la variable $DATE
# En précisant le format demandé : AAAAMMJJ
DATE=$(date +"%Y%m%d")

# Répertoire personnel de l'utilisateur pi
REP_PERSONNEL="/home/pi"

# Message indiquant que la sauvegarde commence
echo "Début de la sauvegarde"
echo "----------------------"

# Création de l'archive avec compression des fichiers
# La commande tar crée un fichier regroupant tous les fichiers du répertoire.
# L'option j indique que l'archive doit être compressée
# L'option v indique le mode bavard
# L'option f indique que le nom du fichier archive suit les options
# L'option c indique qu'il faut créer un nouveau fichier archive
tar -cjvf /mnt/cle_usb/$DATE.tar.bz2 $REP_PERSONNEL

# Annonce de la fin de la sauvegarde
echo "Sauvegarde terminée"
echo "-------------------"
```

Quelques explications

Les # en début de ligne indiquent des commentaires. La ligne qui suit un # est ignorée par l'interpréteur de commandes.

`DATE=`… crée une variable appelée DATE et range dans cette variable ce qui est à droite du signe =, ici la date au format AAAAMMJJ, à l'aide de la commande `date` et du format adapté.

`REP_PERSONNEL=`… crée la variable REP_PERSONNEL et y range */home/pi*.

`echo "..."` écrit simplement à l'écran le texte entre guillemets.

`tar` crée un fichier archive en fonction des paramètres qui suivent.

`/mnt/cle_usb/$DATE.tar.gz2` est le chemin absolu du fichier archive. La variable DATE est remplacée par la date du jour qui y a été placée au début du programme. Le `$` placé devant la variable permet d'appeler sa valeur.

Mise en œuvre du script

d Connectez-vous en super-utilisateur *root*. Créez un fichier *sauve.sh* avec *nano*. Saisissez le

texte du script. Éventuellement, modifiez le chemin du répertoire où sera enregistrée l'archive. Rendez le script exécutable avec la commande `chmod +x sauve.sh`. Exécutez ensuite le script (il faut obligatoirement saisir ./ devant le nom du fichier, cela est nécessaire pour pouvoir lancer le script depuis le répertoire courant. Il peut également être lancé avec le chemin absolu).

```
pi@raspberrypi ~ $ sudo ./sauve.sh
Début de la sauvegarde
----------------------
/20130725.tar.bz2
tar: Suppression de « / » au début des noms des membres
/home/pi/
/home/pi/.profile
...
...        #Liste de tous les fichiers ajoutés à l'archive
...
/home/pi/python_games/inkspilllogo.png
/home/pi/python_games/catgirl.png
/home/pi/.bash_history
Sauvegarde terminée
-------------------
pi@raspberrypi ~ $
```

Le fichier archive a été créé et contient tous les fichiers de l'utilisateur *pi*. Le script remplit donc son rôle puisqu'une seule commande réalise toute une série d'opérations de façon automatique.

Ce script est minimaliste, il ne réalise aucun test avant d'effectuer la sauvegarde, pour vérifier que le répertoire de destination est disponible par exemple.

3.5.8 Planifier des tâches

S'il est intéressant de disposer d'un script automatisant certaines opérations, il serait également pratique que ce script se lance automatiquement à certaines heures. Linux dispose de cette fonction, c'est `crontab` qui permet l'exécution d'un script ou d'une commande à une heure donnée ou selon un cycle défini par l'utilisateur. `crontab` est le diminutif de chrono table (table de planification). Cette table spécifie les tâches à exécuter ainsi que l'heure à laquelle l'exécution doit être déclenchée. Une périodicité peut également être indiquée pour chacune des tâches.

Pour créer une table ou l'éditer, la commande est `crontab -e`.

Cette commande ouvre l'éditeur de texte par défaut (*nano* sur Raspbian) sur la *crontab* personnelle de l'utilisateur. Lors de la première ouverture, ce fichier ne contient que des lignes d'explications commençant par un #.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

À la fin du fichier, sous les commentaires, l'utilisateur peut programmer des tâches et le moment de leur exécution.

Syntaxe des lignes dans la crontab utilisateur

```
m   h   dom   mon   dow   commande
```

`m`	Minutes de 0 à 59.
`h`	Heure de 0 à 23.
`dom`	*day of month* = jour du mois de 1 à 31.
`mon`	*month* = mois de 1 à 12 ou l'abréviation du mois (en anglais soit : jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov et dec).
`dow`	*day of week* = jour de la semaine avec 0 et 7 pour dimanche, 1 pour lundi…
`commande`	Le script doit au moins avoir le droit exécution pour l'utilisateur qui lance la tâche.

Pour chaque unité de temps, plusieurs notations existent :

`*` la tâche est exécutée à chaque unité de temps.

`3-6` la tâche est exécutée aux unités de temps 3, 4, 5 et 6.

`*/2` la tâche est exécutée toutes les deux unités de temps 0, 2, 4, 6, 8…

`2,7` la tâche est exécutée aux unités de temps 2 et 7.

Exemples de programmation de la tâche sauve.sh

La sauvegarde se lance à la demi de toutes les heures :

```
30   *    *    *    *    /home/pi/sauve.sh
```

La sauvegarde se lance tous les jours à 18 H 00 :

```
00   18   *    *    *    /home/pi/sauve.sh
```

La sauvegarde se lance toutes les 10 minutes :

```
10,20,30,40,50,0    *    *    *    *    /home/pi/sauve.sh
*/10    *    *    *    *    /home/pi/sauve.sh
```

La sauvegarde se lance le jeudi à 12 H 30 :

```
30    12    *    *    4    /home/pi/sauve.sh
```

La vérification des tâches programmées se fait avec la commande `crontab -l` qui affiche la liste des tâches présentes dans la *crontab* utilisateur.

Il est également possible de programmer une tâche pour qu'elle s'exécute au démarrage de la machine :

Syntaxe

```
@reboot commande
```

La directive `@reboot` remplace les cinq champs de date et d'heure.

3.5.9 Gérer les processus

Chaque programme lancé sous Linux devient un processus, le temps de son exécution par le microprocesseur. Un processus s'exécute avec les droits de celui qui le lance. C'est une des raisons qui expliquent la presque totale absence de virus sous Linux. Si un utilisateur normal lance un programme malveillant, celui-ci ne pourra effectuer que les opérations autorisées à cet utilisateur. Il ne pourra pas modifier les fichiers système par exemple.

Au démarrage de Linux, un premier processus est démarré : *init*. Il porte le PID 1 (PID = *Processus ID* = identifiant de processus). Ce processus est le père de tous les autres processus. Il démarre les processus du noyau et les processus système qui reçoivent chacun un PID différent. Chaque processus a un père, le processus qui l'a créé, et il peut lui-même être père d'autres processus.

Plusieurs commandes sont disponibles pour lister et analyser les processus en cours d'exécution.

pstree

La commande `pstree` permet de lister sous forme d'arborescence les processus actifs sur un système Linux. L'option `-p` affiche le numéro des PID entre parenthèses.

```
root@raspberrypi:~# pstree -p
init(1)─┬─console-kit-dae(2437)─┬─{console-kit-dae}(2438)
        │                       ├─{console-kit-dae}(2439
...
        ├─cron(2282)
        ├─dbus-daemon(2317)
        ├─dhclient(2065)
        ├─getty(2430)
        ├─getty(2431)
        ├─getty(2432)
        ├─getty(2433)
        ├─getty(2434)
        ├─getty(2435)
        ├─ifplugd(1885)
        ├─ifplugd(1912)
        ├─login(2429)───bash(2514)
        ├─ntpd(2361)
        ├─polkitd(2504)───{polkitd}(2506)
        ├─rsyslogd(2235)─┬─{rsyslogd}(2236)
        │                ├─{rsyslogd}(2238)
        │                └─{rsyslogd}(2239)
        ├─sshd(2395)───sshd(2549)───bash(2556)───pstree(2562)
        ├─thd(2420)
        └─udevd(145)─┬─udevd(235)
                     └─udevd(236)
```

top

Pour avoir une idée des processus les plus gourmands en ressources, la commande `top` affiche en temps réel les processus et leur consommation de ressources système. Dans les premières lignes sont affichées des informations sur le temps de fonctionnement du système, les tâches actives, l'occupation du CPU et de la mémoire.

```
top - 19:05:33 up 21 min,  2 users,  load average: 0,02, 0,07, 0,13
Tasks:  49 total,   1 running,  48 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6,8 us,  3,5 sy,  0,0 ni, 89,6 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem:    190020 total,    53736 used,   136284 free,    10312 buffers
KiB Swap:   102396 total,        0 used,   102396 free,    26436 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2581 root      20   0  4688 1264  952 R  14,4  0,7   0:00.15 top
    1 root      20   0  2144  724  616 S   0,0  0,4   0:00.71 init
    2 root      20   0     0    0    0 S   0,0  0,0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S   0,0  0,0   0:00.04 ksoftirqd/0
    4 root      20   0     0    0    0 S   0,0  0,0   0:00.37 kworker/0:0
    5 root      20   0     0    0    0 S   0,0  0,0   0:00.03 kworker/u:0
    6 root      -2   0     0    0    0 S   0,0  0,0   0:02.03 rcu_kthread
    7 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 khelper
    8 root      20   0     0    0    0 S   0,0  0,0   0:00.02 kdevtmpfs
    9 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 netns
   10 root      20   0     0    0    0 S   0,0  0,0   0:00.01 sync_supers
   11 root      20   0     0    0    0 S   0,0  0,0   0:00.00 bdi-default
   12 root       0 -20     0    0    0 S   0,0  0,0   0:00.01 kblockd
   13 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 rpciod
   14 root      20   0     0    0    0 S   0,0  0,0   0:00.14 kswapd0
   16 root      20   0     0    0    0 S   0,0  0,0   0:00.00 fsnotify_mark
   17 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 nfsiod
```

Cette commande dispose d'une grande quantité d'options (voir `man top`) mais les plus utilisées sont généralement les lettres (respecter la casse) :

– `P` : classer les processus par pourcentage d'occupation du processeur.

– `M` : classer les processus par pourcentage d'occupation mémoire.

– `k` : "tuer" un processus.

– `q` : quitter `top`.

Ces tris sont utiles pour repérer un processus qui "plante" le Raspberry Pi, en monopolisant le temps processeur par exemple. La lettre k saisie au clavier ouvre une demande sur la sixième ligne de `top` :

```
pid to signal/kill
```

Il suffit de saisir le PID du processus à supprimer puis d'appuyer sur la touche [Entrée]. La ligne devient :

```
Send pid 2602 signal [15/sigterm].
```

Par défaut, `top` propose d'envoyer *SIGTERM* au processus. Ce signal qui porte le numéro 15, demande au processus de sauvegarder ses données et de s'arrêter de lui-même. C'est parfois suffisant pour provoquer l'arrêt d'un processus, mais *SIGTERM* peut aussi être intercepté ou ignoré par le processus.
Pour envoyer *SIGTERM* il faut simplement appuyer sur la touche [Entrée].

Si l'envoi de *SIGTERM* n'a pas permis l'arrêt du processus, il va falloir forcer l'arrêt de celui-ci en lui envoyant *SIGKILL*, le signal numéro 9. Il suffit de saisir 9 et de valider pour supprimer à coup sûr le processus.

ps

La commande `ps` affiche les processus en cours mais sous forme d'une liste statique.

Syntaxe

```
ps [options]
```

`ps`	Lancée sans option, la commande `ps` affiche les processus lancés par l'utilisateur connecté sur la console en cours d'exécution. Ils sont en général peu nombreux.

`ps -ef`	Liste tous les processus lancés par l'ensemble des utilisateurs sur toutes les consoles du système. Il faut souvent lancer `ps -ef | less` pour exploiter correctement ces informations.
`ps -aux`	`a` liste tous les processus. `u` affiche l'utilisateur des processus. `x` affiche la liste des processus sans terminal.
`ps -u utilisateur`	Liste les processus lancés par un utilisateur.

Exemple d'utilisation de ps

Processus de l'utilisateur connecté à la console :

```
root@raspberrypi:~# ps
  PID TTY          TIME CMD
 2556 pts/0    00:00:01 bash
 2659 pts/0    00:00:00 ps
```

Processus de l'utilisateur *pi* :

```
root@raspberrypi:~# ps -u pi
  PID TTY          TIME CMD
 2672 tty1     00:00:00 bash
 2679 tty1     00:00:00 nano
```

Tous les processus de tous les utilisateurs, y compris le système :

```
root@raspberrypi:~# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 18:44 ?        00:00:00 init [2]
root         2     0  0 18:44 ?        00:00:00 [kthreadd]
root         3     2  0 18:44 ?        00:00:00 [ksoftirqd/0]
root         4     2  0 18:44 ?        00:00:00 [kworker/0:0]
root         5     2  0 18:44 ?        00:00:00 [kworker/u:0]
root         6     2  0 18:44 ?        00:00:02 [rcu_kthread]
root        28     2  0 18:44 ?        00:00:00 [jbd2/sda2-8]
root        29     2  0 18:44 ?        00:00:00 [ext4-dio-unwrit]
root       145     1  0 18:44 ?        00:00:00 udevd -daemon
root       235   145  0 18:44 ?        00:00:00 udevd -daemon
root       236   145  0 18:44 ?        00:00:00 udevd -daemon
|
|               LISTE REDUITE CAR TROP LONGUE…
|
/usr/lib/policykit-1/polkitd -n
root      2514  2429  0 18:58 tty1     00:00:01 -bash
root      2549  2395  0 19:00 ?        00:00:03 sshd: root@pts/0
root      2556  2549  0 19:01 pts/0    00:00:01 -bash
root      2665  2514  0 19:58 tty1     00:00:00 su pi
pi        2672  2665  0 19:58 tty1     00:00:00 bash
root      2678     2  0 19:58 ?        00:00:00 [flush-8:0]
pi        2679  2672  0 19:58 tty1     00:00:00 nano fichier1.txt
root      2686  2556  0 20:00 pts/0    00:00:00 ps -ef
```

On peut repérer sur cette liste les informations suivantes (première ligne de la liste) :

```
UID        PID  PPID  C STIME TTY          TIME CMD
```

– UID = utilisateur ayant lancé ce processus.
– PID = identifiant du processus.
– PPID = identifiant du père de ce processus.
– C = priorité du processus (grande valeur = grande priorité).
– STIME = *start time* = heure de début d'exécution du processus.
– TTY = terminal sur lequel le processus a été lancé.
– TIME = temps d'occupation du CPU par ce processus.
– CMD = commande ayant lancé ce processus.

Suppression de processus

La commande `kill` permet de supprimer un processus.

Syntaxe

```
kill -numero_signal PID_du_processus
```

Exemple de suppression de processus

L'utilisateur pi a ouvert l'éditeur *nano* sur un terminal.

```
root@raspberrypi:~# ps -upi
  PID TTY          TIME CMD
 2672 tty1     00:00:00 bash
 2750 tty1     00:00:00 nano
```

Le super-utilisateur souhaite fermer la fenêtre de *nano*.

```
root@raspberrypi:~# kill -9 2750
```

La fenêtre *nano* du terminal utilisé par l'utilisateur *pi* se ferme, un message **`Processus arrêté`** est affiché.

`kill PID` provoque un arrêt normal du processus (`-15` par défaut). `kill -9 PID` force l'arrêt du processus et risque de provoquer des pertes d'informations.

3.5.10 Configurer la date du système

Le Raspberry Pi 3 ne possède pas d'horloge RTC maintenue par pile. Lorsqu'il est connecté à Internet, il récupère l'heure sur un serveur de temps. Sur le Raspberry Pi Zero, il est parfois nécessaire de régler la date manuellement.

La commande `date` utilisée sans argument permet d'afficher l'heure du système.

Affichage de l'heure système :

```
pi@raspberrypi ~ $ date
mercredi 10 août 2016, 08:22:00 (UTC+0200)
```

La commande `date` utilisée avec des arguments permet de configurer l'heure du système. Seul le super-utilisateur peut modifier l'heure système.

Mise à l'heure du système :

Syntaxe

```
date MMJJhhmm[YYAA.ss]
```

`MM`	Mois
`JJ`	Jour du mois
`hh`	Heure
`mm`	Minute
`YY`	Deux premiers chiffres de l'année (facultatif)
`AA`	Deux derniers chiffres de l'année (facultatif)
`.ss`	Secondes (facultatif)

Exemple :

```
pi@raspberrypi:~ $ sudo date 081008242016.00
mercredi 10 août 2016, 08:24:00 (UTC+0200)
```

4. Sauvegarder votre configuration

Après l'application des options, la mise à jour et la personnalisation du système, Raspbian est prêt pour une utilisation sur le Raspberry Pi.

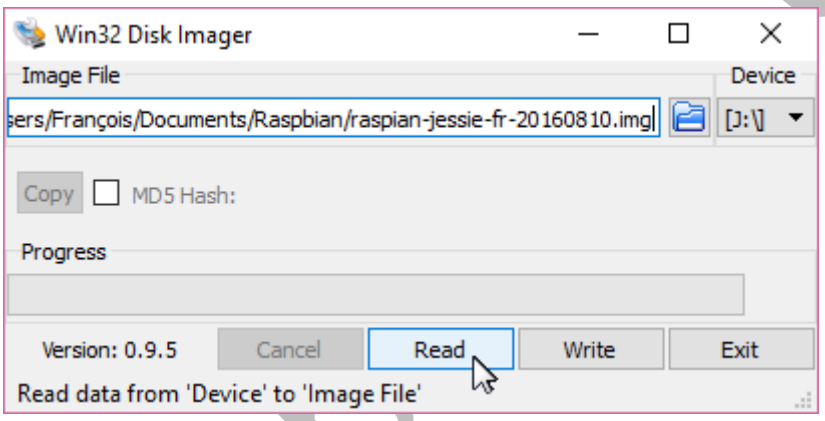
La carte micro SD contient une version de Raspbian en bon état. Il est prudent d'en faire une sauvegarde qui évitera de repasser par toutes les étapes précédentes. Le Raspberry Pi est un outil formidable pour l'expérimentation, ce qui aboutit parfois à une impossibilité de redémarrer avec un système "modifié".

Exactement comme pour les images mises à disposition sur le site de la Fondation, le système présent sur la carte micro SD va être sauvegardé dans son état actuel sous forme d'un fichier *.img*.

4.1 Sauvegarde de la carte micro SD sous Windows

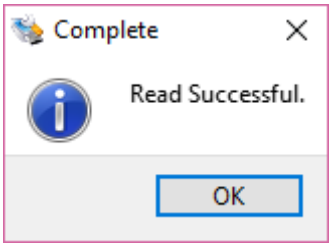
d Éteignez proprement le Raspberry Pi. Sortez la carte micro SD du connecteur et introduisez-la dans le lecteur de carte de votre PC.

d Lancez l'utilitaire *Win32DiskImager* qui a servi à créer la carte micro SD. Vérifiez soigneusement que la carte indiquée sous l'étiquette **Device** de *Win32DiskImager* correspond bien à votre carte micro SD. Cliquez sur le dossier à droite de la zone **Image File**, choisissez le répertoire dans lequel vous souhaitez enregistrer l'image et donnez-lui un nom. Cette fois, au lieu d'écrire l'image depuis le disque vers la carte, cliquez sur le bouton **Read**. Cela va provoquer la lecture de la carte micro SD, le transfert des octets qu'elle contient vers le PC et la création d'un fichier *.img* contenant une copie conforme du système.



07RI02N

Lorsque la sauvegarde complète est terminée, *Win32DiskImager* affiche :



07RI03N

L'image est maintenant enregistrée sur votre disque dur à l'emplacement que vous avez choisi. Vous pouvez sortir la carte micro SD du lecteur et l'utiliser. En cas de problème vous la régénérerez avec l'image.

4.2 Sauvegarde de la carte micro SD sous Debian

d Éteignez proprement le Raspberry Pi. Sortez la carte micro SD du connecteur et introduisez-la dans le lecteur de carte de votre PC.

Sous Linux, la commande `dd` qui a servi à la création de la carte micro SD va également être utilisée pour réaliser une image de celle-ci.

d Identifiez votre carte micro SD avec la commande : `fdisk -l`

```
root@debian:~# fdisk -l

.../...

Disque /dev/sdb : 15.7 Go, 15720251392 octets
64 têtes, 32 secteurs/piste, 14992 cylindres, total 30703616 secteurs
Unités = secteurs de 1 * 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Identifiant de disque : 0x000c7b31

Périphérique Amorce  Début    Fin      Blocs    Id  Système
/dev/sdb1            8192     122879   57344    c   W95 FAT32 (LBA)
/dev/sdb2            122880   3788799  1832960  83  Linux
```

d Contrôlez la taille des partitions et la taille totale de la carte pour éviter toute erreur.

d Votre carte micro SD peut apparaître par exemple en */dev/sda* ou */dev/sdb*. Saisissez la commande suivante avec une carte détectée en */dev/sdb*, puis validez pour démarrer la copie de la carte vers le fichier indiqué en paramètre après `of=` (*Output File* = fichier de destination).

```
sudo dd if=/dev/sdb of=/home/user/raspbian-jessie-fr-20160810.img bs=1M
```

La copie de la carte aura la dimension de la carte. L'image d'une carte micro SD de 16 Go occupera donc 16 Go d'espace disque. Vous pouvez compresser le fichier pour réduire la place occupée, avec la commande :

```
gzip /home/user/raspbian-jessie-fr-20160810.img
```

5. Conclusion

La totalité des commandes présentées dans ce chapitre constitue une sorte de trousse à outils pour démarrer l'utilisation de Linux.

De nombreuses options n'ont pas été évoquées, d'autres commandes existent. Seule la pratique permet à un utilisateur/administrateur de se forger une trousse à outils personnelle. La consultation

des pages `man` de ces commandes est un complément indispensable à la pratique régulière de la ligne de commande.