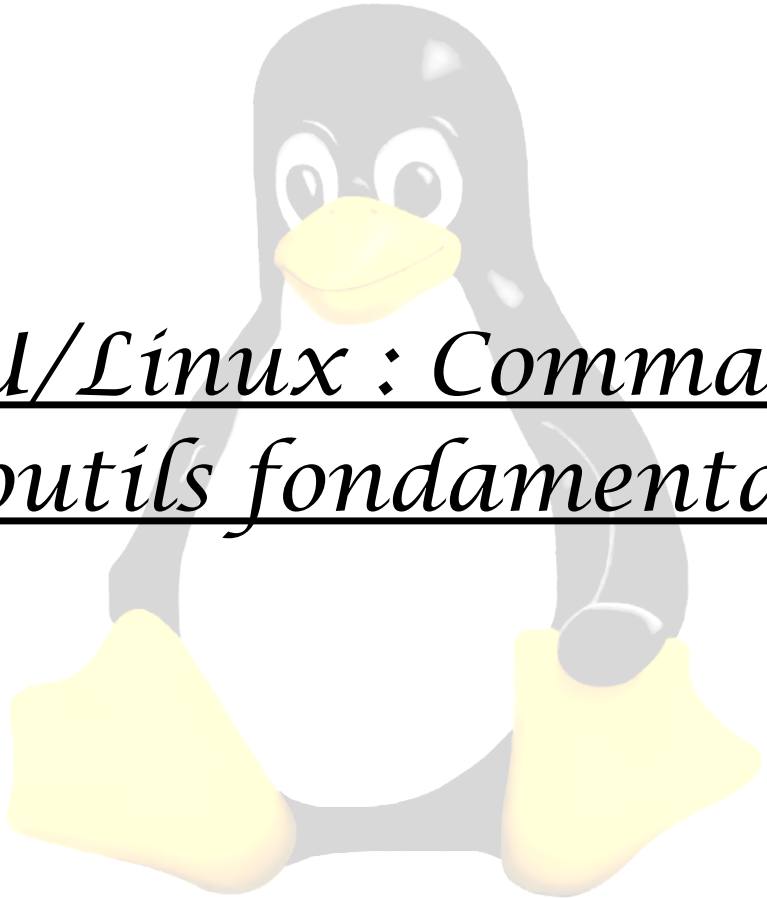
Effectué pour Afric'Edu Projet 2006

How-To Général sur Linux



GNU/Linux : Commandes et outils fondamentaux

Juin 2005

DidRocks

Les commandes et outils fondamentaux de Linux

REMARQUE : SOUS LINUX (comme sous tout système UNIX) LES MINUSCULES ET LES MAJUSCULES NE SONT PAS ÉQUIVALENTES.

1. Commandes fondamentales

1.1 Se déplacer dans l'arborescence de répertoires (cd)

Lorsque vous avez passé le login et le password de linux, vous vous retrouvez devant le prompt shell qui est le plus souvent celui de bash (sinon vous serez devant celui de csh). Il ressemble le plus souvent à ceci :

```
[root@mistra /root]$
```

Le mot **root** signifie que vous vous êtes "logué" sur le compte de l'administrateur système. Vous êtes donc en pleine possession de la machine, vous pouvez faire absolument n'importe quoi, jusqu'à supprimer tous les fichiers ... faites donc très attention ... En théorie *il ne faut utiliser la machine sous ce compte qu'afin de l'administrer*. Des comptes dits « d'utilisateurs » permettent sinon de travailler en temps normal. Nous verrons ci-après comment créer un compte utilisateur.

Le mot **"mistra"**représentera, dans ce document, le nom de votre ordinateur (pour le connaître invoquer la commande "hostname")

Actuellement vous vous trouvez sous le compte de l'administrateur système, c'est-à-dire que vous êtes dans le répertoire **/root** (sous Unix, les composants des noms de répertoires sont séparés par des "slash" "/" et non pas comme sous MS-DOS par des "anti-slash" "\").

Déplaçons-nous dans la "racine" du système :

```
[root@mistra /root]$ cd ..
```

Faites bien attention de séparer par un espace "cd" et "..", UNIX exige une grande précision dans la syntaxe des commandes. Soumettez la commande au système grâce à la touche « Entrée », évidemment !

Vous êtes maintenant dans le répertoire racine :

```
[root@mistra /]#
```

Que contient-il ? Tapez la commande **ls**, et voyez le résultat, vous devez obtenir quelque chose comme :

```
bin boot cdrom etc usr var vmlinux
```

Si certains fichiers ou répertoires manquent ce n'est pas important.

Déplaçons-nous dans le répertoire qui contient une grande partie des programmes (souvent simplement appelés « binaires ») de linux : /usr/bin : **cd usr/bin**. Vous pouvez là aussi obtenir le contenu du répertoire en utilisant la commande **ls**.

Maintenant allons voir ce que contient le répertoire **/etc** (aperçu lorsque nous avons listé le répertoire racine /). Nous avons deux possibilités pour nous y rendre : soit nous revenons dans le répertoire racine et nous nous rendons ensuite dans le répertoire **etc**; soit nous nous rendons immédiatement dans le répertoire **/etc** :

- Méthode no 1 :

cd / (pour se rendre à la racine)
puis
cd etc

Cette méthode est fastidieuse car elle nécessite de taper deux commandes successives. Nous pouvons utiliser la deuxième méthode pour nous rendre directement dans le répertoire **/etc** en écrivant le chemin complet dans la commande **cd** :
- Méthode no 2 :
cd /etc

et nous sommes directement dans le répertoire **/etc**. Dans cette commande nous avons indiqué que pour se rendre dans le répertoire **etc**, il fallait d'abord se rendre dans le répertoire racine. Pour se faire nous avons placé un / devant **etc**.

Lorsque l'on ajoute un ~ au lieu d'un chemin à la commande **cd**, celle-ci nous replace automatiquement dans notre répertoire utilisateur. Si vous êtes en administrateur système la commande par **cd ~** vous placera dans le répertoire **/root**. Dans le cas où je suis (je suis loggé en tant qu'utilisateur *delcros*) je vais automatiquement me retrouver dans le répertoire de l'utilisateur *delcros* qui se trouve dans **/home/delcros**. Les répertoires des utilisateurs sont tous sous **/home**.

[delcros@mistra bin]$cd ~

- Ceci est la méthode orthodoxe, sinon vous pouvez faire simplement :
[delcros@mistra bin]$cd

et vous reviendrez ainsi dans votre répertoire personnel.

1.2 Dans quel répertoire suis-je actuellement ? (pwd)

Lorsque l'on se déplace dans les répertoires, par défaut bash n'affiche que le « nom court » du répertoire où l'on se trouve. Le nom court ne comprend pas le chemin complet. Or il peut arriver qu'un même nom court corresponde à plusieurs répertoires bien distincts, donc que seuls les chemins qui y mènent permettent de les distinguer. C'est par exemple le cas du nom court **bin**, que l'on trouve en **/bin** et en **/usr/local/bin**. Il existe beaucoup d'autres exemples. La solution pour connaître le chemin du répertoire où l'on se trouve est d'utiliser la commande **pwd** :
[delcros@mistra bin]$ pwd
`/usr/bin`
[delcros@mistra bin]$

1.3 Lister les fichiers d'un répertoire (ls)

La commande **ls** et ses très nombreuses options vous permettront d'obtenir beaucoup d'informations sur les fichiers présents dans un répertoire : déplaçons nous par exemple dans le répertoire "/bin" et listons le contenu de ce répertoire :
[delcros@mistra bin]$ cd /bin
[delcros@mistra /bin]$ ls

```
arch            dd              gzip            nisdomainname   su
ash             df              hostname        ping            sync
awk             dmesg           kill            ps              tar
cp              fgrep           mount           sh              ypdomainname
cpio            gawk            mt              sleep           zcat
csh             grep            mv              sort            zsh
date            gunzip          netstat         stty            ls
```

Ceci est un listing "brut" du répertoire /bin qui contient les utilitaires de base de linux. On reconnaît par exemple la commande **ls** ...
De la même manière que sous MS-DOS (avec la commande dir), nous pouvons demander à Linux de lister seulement les fichiers dont les noms contiennent des caractères donnés. Demandons par exemple uniquement les noms des fichiers commençant par la lettre "l" :
[delcros@mistra /bin]$ ls l*

```
ln login ls
```
[delcros@mistra /bin]$

Voici quelques options intéressantes de la commande **ls** (les options sous UNIX suivent la commande et sont le plus souvent précédées d'un tiret) :
L'option **ls -l** permet de lister les attributs des fichiers (les droits de lecture, d'écriture et d'exécution, le propriétaire, le groupe, la taille en octets, sa date de création ou de modification) :
[delcros@mistra /bin]$ ls -l

```
total 3615
-rwxr-xr-x    1 root     root         2716 Apr 23  02:09  arch
-rwxr-xr-x    1 root     root        56380 Dec 23  1996   ash
lrwxrwxrwx    1 root     root            4 May 10  20:01  awk -> gawk
-rwxr-xr-x    1 root     root        18768 Mar  8  19:17  basename
-rwxr-xr-x    1 root     root       300668 Sep  4  1996   bash
lrwxrwxrwx    1 root     root            3 May 10  19:59  bsh -> ash
-rwxr-xr-x    1 root     root        16584 Dec 16  1996   cat
-rwxr-xr-x    1 root     root        17408 Nov 26  1996   chgrp
```

Notes : Ici, tous les fichiers appartiennent à l'administrateur système (root) et à son groupe (root), comme les sections consacrées à chmod (4.1) et à chown (4.2) l'exposerons). Nous traiterons du sens de la fin de chaque ligne, qui contient parfois une flèche visible ici sur la ligne `awk -> gawk`, dans la section consacrée aux liens ln (1.11).

ls -a liste tous les fichiers du répertoire, y compris les fichiers cachés. Cette option est très utile lorsque l'on se trouve dans son répertoire personnel car il contient les fichiers de configuration de l'utilisateur dont les noms commencent généralement par un point et seule l'option **-a** permet de détecter leur existence.

Exemple avec le répertoire de l'administrateur système :
voici une partie des fichiers listés avec la commande **ls** sans option :
[root@mistra /root]# ls

```
bookmarks.sgml     mc.hint            scrsh2             2494.html
Desktop            ftape.o            mc.hlp             scrsh3
FAQ.services.html  kbanner.kssrc      mc.lib             xdm-config
```

Et voici une partie du résultat avec la commande **ls -a**.
[**root@mistra /root]# ls -a**

```
.                                .kvtrc                           .xquadkey
..                               .letter                          .xquadkey~
.BitchX                          .mc.ext                          2494.html
.Xmodmap~                        .peruser-newsrc-working
.amaya                           .peruser_config                  Desktop
.applications                    .peruser_spool                   FAQ.services.html
```

...

On peut maintenant connaître tout (option 'a' : penser au mot "all") le contenu du répertoire.

D'autres options de **ls** sont utiles :
ls -m :
Affiche les fichiers en les séparant par une virgule au lieu de les présenter en colonnes.

ls -t :
Affiche les fichiers par date, c'est-à-dire en les classant du récent au plus ancien.

ls -lu :
Affiche les fichiers par date de dernier accès et indique cette date.

ls -F :
Affiche les fichiers par type. Ainsi un fichier suivi d'un slash (/) est un répertoire, un fichier suivi d'une étoile est un fichier exécutable et un fichier suivi d'un "@" est un lien (nous reviendrons sur les liens dans la section consacrée à **ln**).

ls -S :
Affiche les fichiers triés par ordre de taille décroissante.

ls -X :
Affiche les fichiers par type d'extension.

ls -r :
Affiche les fichier en ordre alphabétique inverse.
Cette option à la particularité d'inverser l'effet de tous les tris requis. Par exemple, la commande **ls -tr** affichera les fichiers par date en commençant par les plus anciens pour finir par les plus récents.

1.4 Voir un fichier (cat et more)

La commande **cat** permet de lire des fichiers. Nous avons vu tout à l'heure que le répertoire **/root** contenait des fichiers de configuration. Ces fichiers sont simplement des fichiers textes avec un agencement et une syntaxe particulière. Regardons le contenu du fichier **.bashrc** qui permet de configurer à souhait son shell :

[root@mistra /root]# cat .bashrc
```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi
source .sd.sh
```
[root@mistra /root]#

Une option utile de **cat** est **-n** qui permet de numéroter les lignes (ne pas oublier que cat permet de *lire* et non de *modifier* un fichier. Ainsi la numérotation de ligne apparaît à l'écran mais le fichier .bashrc n'en est pas pour autant modifié).

[root@mistra /root]# cat -n .bashrc
```
1 # .bashrc
2
3 # User specific aliases and functions
4
5 # Source global definitions
6 if [ -f /etc/bashrc ]; then
7 . /etc/bashrc
8 fi
9 source .sd.sh
```
[root@mistra /root]#

Si vous souhaitez connaître les autres options de **cat**, tapez au prompt "**cat --help**".

Vous pouvez utiliser la commande **more** pour visualiser un fichier. La commande **more** a

l'avantage d'afficher le fichier page par page. Pour passer d'une page à l'autre, tapez sur la touche **ESPACE**.

1.4 Éditer un fichier (vi/vim, emacs, joe)

1. **vi (l'éditeur le plus ancien)**

   vi date des années 70 .... autant dire que cet éditeur a du métier et n'est toujours pas démodé. Ce n'est pas celui que j'utilise mais beaucoup en sont adeptes malgré son apparence fruste. Ceci s'explique par une grande puissance ... Si je m'attarde quelque peu sur **vi**, c'est que dans les moments critiques où rien ne fonctionne, où tout va mal, c'est l'éditeur qu'on ne peut éviter.

   Utilisons maintenant plutôt vim (vi modified), plus simple d'accès. Un petit tuto pour commencer? Ok, alors tapez vimtutor fr, Après, pour un rappel des commandes voir la suite!

   Lançons Vim :
   [root@mistra /root]# vim

   Après le lancement de la commande vous allez vous trouver directement dans l'éditeur ... Pendant ce court apprentissage de vi, nous allons créer un fichier, le modifier, l'enregistrer, ... et quelques autres petites manoeuvres de survie :

   1. **Passer du mode commande aux mode texte, taper du mode texte, enregistrer.**

      vim comprend deux modes : un mode "commande" et un mode "insertion", après le lancement de vi nous sommes en mode commande : appuyez sur la touche "**Echap**" puis sur "**a**" ("a", comme "**a**ppend", permet d'ajouter du texte après le curseur). Vous voyez en bas de l'écran apparaître la ligne "-- INSERT --". Nous pouvons commencer notre texte :

      ```
      linux est gratuit puissant en perpetuelle evolution.
      Linux est stabble. Linux existe depuis 1991 seulement
      et pourtant quel chemin parcouru !
      ```

      N'oubliez pas de placer retour chariot au bout de chaque ligne.

      Sauvons le fichier : nous sortons d'abord du mode texte en appuyant à nouveau sur la touche "**Echap**". La mention "-- INSERT --" disparaît, nous sommes en mode commande. Tapez maintenant " **:w linux-test**" et sur la touche retour chariot (afin d'écrire ("**w**rite") le fichier). Vous devez obtenir en bas de l'écran ceci :
      `"linux-test" [New File] 3 lines, 142 characters written`

   2. **Supprimer du texte et quitter vim**
      Nous voyons qu'à la deuxième ligne, j'ai fait une grosse fôte d'aurtograffe. Nous allons supprimer le "b" qui est en trop dans stab*b*le : déplacez le curseur sur un des "b" en trop, passez en mode commande ("--INSERT --" ne doit pas apparaître à l'écran), appuyez sur "**x**", le b a disparu.

      Quittons vi, mais auparavant, nous devons sauver les modifications effectuées : Passez en mode commande et tapez " **:wq**" (**w**rite et **q**uit). Vous êtes sorti de vi et votre fichier a été sauvegardé sous linux-test. Pour revenir à vi en ouvrant le fichier linux-test au démarrage tapez :
      [root@mistra /root]# vi linux-test

      Si vous souhaitez quitter sans enregistrez les dernières modifications, il vous faudra

passer en mode commande et taper " **: q!**".

Ceci est une présentation très très courte de vi, mais qui vous permettra malgré tout de survivre au cas où vous devriez absolument l'utilisez. Voyons tout de même un rapide descriptif d'autres commandes vi :

3. **D'autres commandes vim.**

   A permet d'ajouter du texte à la fin de la ligne.
   i permet d'ajouter du texte avant le curseur.
   o permet d'ajouter une ligne en dessous du curseur.
   O permet d'ajouter une ligne au dessus du curseur.

   le retour chariot permet d'aller à la ligne suivante.

   - **dd** permet de supprimer la ligne courante.
   - **X** permet de supprimer le caractère avant le curseur.
   - **x** permet de supprimer le caractère après le curseur.
   - **u** permet d'annuler la dernière commande effectuée.

2. **Emacs ... la puissance !**

   Emacs date de la fin des années 70 et ne cesse d'évoluer depuis, ce qui fait de lui, sans aucun doute possible, l'éditeur le plus puissant au monde. Bien plus qu'un éditeur, emacs est un environnement de travail : édition, programmation, mail, news, shell ... bref on peut rester sous emacs sans avoir besoin de quoi que ce soit d'autre.

   Ses adeptes sont très nombreux.

   Et surtout ne leur dites pas qu'emacs est lourd ...vous vous tromperiez lourdement (je sais de quoi je parle ... j'ai fait l'erreur et en ce moment je suis sous emacs ... ;-))
   Lançons emacs :
   [root@mistra root]$ emacs

   Ouvrons maintenant le fichier linux-test que nous avons créé précédemment sous vi :
   Pour cela utilisez la séquence de touches suivante : **Ctrl-x Ctrl-f**

   Vous voyez apparaître en bas de l'écran :
   Find File : ~/

   tapez le nom du fichier et faites un retour chariot.

   Nous retrouvons notre charmant petit texte.
   Vous le comprenez, la touche **Ctrl** permet de passer des commandes et de passer du mode texte au mode commande. Vous pouvez le modifier à souhait. Les touches **Backspace** et **Suppr** fonctionnent comme sous n'importe quel éditeur.

   Pour **sauver** le fichier, tapez la séquence de touches suivante :

   Ctrl-x Ctrl-s

   Si vous êtes bloqués dans la ligne de commande d'emacs après avoir effectué de mauvaises manipulations et que vous souhaitez retrouver le mode texte, tapez la séquence suivante :
   Ctrl-g

   Si vous avez fait des erreurs dans le texte, la séquence suivante permet de **supprimer les dernières modifications** :
   Ctrl-x u

Si vos touches de direction ne fonctionnent pas, voici plusieurs séquences de touches qui vous permettent de vous déplacer dans votre document :
Ctrl-p : monter d'une ligne.

Ctrl-n : descendre d'une ligne.

Ctrl-f : avancer d'un caractère.

Ctrl-b : reculer d'un caractère.

Ctrl-v : avancer d'un écran (ou d'une page si vous préférez).

Alt-v : reculer d'un écran.

Ctrl-d : supprimer le caractère sur lequel le curseur se trouve.

Une commande utile est :
Ctrl-s qui permet de faire une **recherche** "dynamique" ("incrémentale") sur une suite de caractères dans le texte.

Meta-% permet de lancer un "Rechercher et remplacer" . La touche **Meta** est en général confondue avec la touche **Alt** Pour **quitter emacs**, utiliser la combinaison de touches suivante :
Ctrl-x Ctrl-c

Avec cette rapide présentation vous pourrez déjà "barboter" un peu sous emacs. Il m'est impossible de décrire dans ce document les milliers de fonctions disponibles .... si vous souhaitez en savoir plus, cette séquence vous permettra de rentrer dans l'**aide d'emacs** :
Ctrl-h

Ou bien lancez le "tutorial" :
Ctrl-h t

Si vous devenez un mordu d'emacs (ce qui est tout a fait normal : -) vous pourrez trouver quelques ouvrages sur emacs dans toutes les bonnes bibliothèques.

3. **joe : la simplicité.**

   joe est l'éditeur que j'utilise pour faire des petites modifications dans mes fichiers de configuration par exemple, il est très leger, il ne possède pas la puissance d'emacs mais rend lui aussi service :
   Pour appeler **joe** :
   [root@mistra /root]# joe

   joe est très intuitif (à la Wordstar), pas besoin de s'étendre sur les fonctionnalités textes. Trois opérations fondamentales à connaître :

   Ctrl-k e permet d'ouvrir un fichier

   Ctrl-k d permet de sauvegarder le fichier

   Ctrl-k x permet de sauvegarder le fichier et de quitter joe

   Ctrl-c permet de quitter joe sans sauvegarder les modifications.

   joe possède de nombreuses fonctions possibles qui sont décrites dans le man (nous verrons comment y accéder dans la section consacrée à man (4.8)).

1.5 Copier un fichier (ou un répertoire) : cp.

La syntaxe de la commande **cp** est la suivante :
cp [option] fichier-origine fichier-destination

ou

cp [option] fichier répertoire

par exemple pour faire une copie de notre fichier *linux-test* en un fichier *linux-test2*, il suffit de faire :
[root@mistra /root]# cp linux-test linux-test2

Nous possédons maintenant deux exemplaires de notre fichier dans /root.

ATTENTION ! : si vous effectuez une copie d'un fichier sur un fichier qui existe déjà, celui-ci sera effacé et remplacé par le nouveau fichier.

Si vous souhaitez copier le fichier *linux-test* dans un répertoire (par exemple **/home**) en gardant le nom du fichier, utilisez la commande suivante :
[root@mistra /root]# cp linux-test /home
Pour lui donner un autre nom :
[root@mistra /root]# cp linux-test /home/linux-test2

Nous venons de voir que l'utilisation de **cp** est dangereuse et l'on risque parfois d'effacer des fichiers importants. Les options de **cp** peuvent vous éviter des situations fâcheuses.

cp -i avertit l'utilisateur de l'existence d'un fichier du même nom et lui demande s'il peut ou non remplacer son contenu. Recopions à nouveau le fichier linux-test sur linux-test2 avec l'**option -i** :
[root@mistra /root]# cp -i linux-test linux-test2
cp : overwrite `linux-test2'?

cp vous demande s'il peut écraser linux-test2 : répondre par "y" (yes) ou "n".

Quelques options importantes de cp :
cp -b permet comme l'option -i de s'assurer que la copie n'écrase pas un fichier existant : le fichier écrasé est sauvegardé, seul le nom du fichier d'origine est modifié et **cp** ajoute un tilde (~) à la fin du nom du fichier.

cp -l permet de faire un lien "dur" entre le fichier source et sa copie. Ceci signifie que le fichier copié et sa copie partageront physiquement le même espace. Cela permet des gains de place non négligeables. Plus exactement, sur le disque dur le fichier et sa copie seront le même fichier alors qu'avec une copie classique, le disque dur contiendra deux exemplaires du fichier.

cp -s permet de faire un lien "symbolique" entre le fichier source et sa copie. Le lien symbolique est un pointeur. Ainsi si nous copions le fichier *linux-test* avec l'option **-s**, lorsque par exemple nous voudrons éditer le fichier copié, linux éditera en réalité le fichier original (voir la section consacrée à ln pour un descriptif plus complet des liens).

cp -p permet lors de la copie de préserver toutes les informations concernant le fichier comme le propriétaire, le groupe, la date de création (voir les sections consacrées à chmod et chown pour plus d'informations).

cp -r permet de copier de manière récursive l'ensemble d'un répertoire et de ses sous-répertoires.

Exemple :
Je possède dans mon répertoire **/home/delcros/personnel** un répertoire intitulé "mygale" et qui contient 3 sous répertoires ("echecs", "linux", xcaissa) :

```
/home/delcros/personnel/
/home/delcros/personnel/mygale/
/home/delcros/personnel/mygale/echecs/
/home/delcros/personnel/mygale/linux/
```

```
/home/delcros/personnel/mygale/xcaissa/
```
Je souhaite copier le répertoire mygale ainsi que ses sous-répertoires dans mon répertoire **/home/delcros/**" : j'utilise la commande (en supposant que je me suis au préalable déplacé dans le répertoire **/home/delcros/personnel/** :
[delcros@mistra personnel]$ cp -r mygale /home/delcros

cp -v permet d'afficher le nom des fichiers copiés. Utile si par exemple vous copiez plusieurs fichiers (à l'aide des occurences "*" et/ou "?") et que vous souhaitez voir le bon déroulement de la "multicopie". J'aurais pu par exemple utiliser cette option lors de ma copie récursive du répertoire "mygale".

J'aurais ainsi vu ceci en associant l'option **-v** et **-r** :
[delcros@mistra personnel]$ cp -rv mygale /home/delcros
```
mygale -> /home/delcros/mygale
mygale/index.html -> /home/delcros/mygale/index.html
mygale/logo.gif -> /home/delcros/mygale/logo.gif
mygale/linux -> /home/delcros/mygale/linux
mygale/linux/linux.html -> /home/delcros/mygale/linux/linux.html
.....
```
(c'est une partie du résultat).

1.6 Supprimer un fichier "rm".

PREAMBULE :
Nous entrons maintenant dans une zone à risque, mieux vaut donc se loguer en tant qu'utilisateur de la machine et non pas en tant qu'administrateur système (root), car nous risquerions par une mauvaise manipulation de supprimer des fichiers fondamentaux nécessaires au bon fonctionnement de linux. Nous allons donc créer un compte utilisateur, lui attribuer un mot de passe et nous loguer sur ce compte. Exécutez les commandes suivantes, une explication détaillée interviendra ensuite dans la partie consacrée à l'administration système (4):
[root@mistra /root]#adduser le_nom_de_choix (votre prénom par exemple, mais sans accent et si possible long de moins de 8 caractères)

[root@mistra /root]#passwd le_nom_de_votre_choix (saisir deux fois le même mot de passe, la seconde sert à confirmer)

[root@mistra /root]#cp linux-test /home/le_nom_de_votre_choix (gardons notre fichier pour continuer nos petites expériences ;).

[root@mistra /root]#chow le_nom_de_votre_choix.le_nom_de_votre _choix /home/le_nom_de_votre_choix(L'administrateur donne généreusement le fichier linux-test au nouvel utilisateur avec la commande "chown" que nous verrons dans les commandes d'administration système, pour l'instant ne vous en souciez pas.)

[root@mistra /root]#su le_nom_de_votre_choix (la commande su permet de se loguer sur un autre compte).
Il suffira de saisir **exit** pour « retomber » dans la session de travail root.

Effectuons à nouveau une copie du fichier linux-test (tapez **cd** pour vous retrouver dans votre répertoire personnel) :
[delcros@mistra delcros]$ cp linux-test linux-test2

LA COMMANDE rm
Pour supprimer le fichier "linux-test2" :
[delcros@mistra delcros]$ rm linux-test2

LES OPTIONS de rm

Comme pour **cp**, l'option **cp -i** permet à **rm** de demander à l'utilisateur s'il souhaite vraiment supprimer le ou les fichiers en question :
[delcros@mistra delcros]$ rm -i linux-test2
rm : remove `linux-test2'?

(il vous suffit donc de répondre "y" ou "n")

rm -d permet de supprimer un répertoire qu'il soit plein ou non (attention dangereux ...)

rm -r permet de supprimer un répertoire et ses sous répertoires (attention TRÈS dangereux)

rm -f permet de supprimer les fichiers protégés en écriture et répertoires sans que le prompt demande une confirmation de suppression (à utiliser avec précaution ...)

1.7 Créer un répertoire (mkdir)

Pour créer un répertoire, il suffit de taper la commande suivante (ici je crée le répertoire "personnel" dans /home/delcros :
[delcros@mistra delcros]$ mkdir personnel

Une option de **mkdir** est souvent utile :
mkdir -p permet de créer une suite de répertoire.

Supposons que je veuille créer dans mon répertoire **/home/delcros** la suite de répertoires suivante : **doc/mygale/mail**. Je peux faire soit :
[delcros@mistra delcros]$ mkdir doc

[delcros@mistra delcros]$ cd doc

[delcros@mistra delcros]$ mkdir mygale

[delcros@mistra delcros]$ cd mygale

[delcros@mistra delcros]$ mkdir mail

Ou bien utiliser l'option **-p** qui me permet de créer la suite de répertoires "parents" le plus simplement du monde :
[delcros@mistra delcros]$ mkdir -p doc/mygale/mail

1.8 Déplacer ou renommer un fichier (mv)

Pour comprendre la commande **mv**, voyons une suite de commandes qui effectuent des opérations différentes :
[delcros@mistra delcros]$ mv linux-test perso
renomme le fichier "linux-test" en "perso"

[delcros@mistra delcros]$ mv perso perso
va écraser le fichier existant avec la source.

[delcros@mistra delcros]$ mv personnel mon-répertoire
va renommer le répertoire personnel en mon-répertoire

[delcros@mistra delcros]$ mv perso /home/delcros/mon-répertoire
va déplacer le fichier perso dans le répertoire **/home/delcros/mon-répertoire**

Les options :

- **mv -b** ('b' comme "**b**ackup") va effectuer une sauvegarde des fichiers avant de les déplacer :
  [delcros@mistra delcros]$ mv -b mon-répertoire/perso /mon-répertoire/linux-test

Cette commande va renommer le fichier perso en linux-test, cependant vous trouverez dans le répertoire une sauvegarde de perso (perso~).

- **mv -i** ('i' comme «interactive») demande pour chaque fichier et chaque répertoire s'il peut ou non déplacer fichiers et répertoires.
- **mv -u** ('u' comme «update») demande à mv de ne pas supprimer le fichier si sa date de modification est la même ou est plus récente que son remplaçant. Exemple :

Déplaçons-nous vers notre répertoire personnel puis créons un nouveau fichier avec l'éditeur de texte **joe** :
[delcros@mistra personnel]$ joe linux-test2

saissons un petit texte :
"y en a marre de ces textes stupides !"

et finissons notre session joe par la séquence de touches suivante :
Ctrl-k x

qui permet d'enregister le fichier et de quitter joe.

Notre fichier linux-test2 est plus récent que notre fichier linux-test. Vous pouvez le vérifier en effectuant un "**ls -l**".

Nous souhaitons (naïvement, bien sûr !) renommer le fichier linux-test en linux-test2. Mais nous sommes attentifs et nous ne voulons pas que le fichier linux-test2 soit écrasé si celui-ci est plus récent que linux-test :
[delcros@mistra personnel]$mv -u linux-test linux-test2

L'option **-u** nous a évité d'écraser le fichier linux-test2. La commande **mv** n'a donc pas été effective.

1.9 Retrouver un fichier ("find")

1- La commande find

Exemple simple : comment trouver un fichier portant un nom donné ?

[delcros@mistra delcros]$ find / -name linux-test2 -print

/home/delcros/linux-test2

(Un peu long n'est ce pas pour trouver la reponse dans tout cette grosse arborescence ? :-))

En general on recherche rarement un fichier depuis la racine.

Décomposition de la commande de l'exemple :
"/" indique que nous voulons chercher à partir de la racine notre fichier.
"-name " est l'option qui indique ici que nous voulons spécifier le nom d'un fichier.
"-print" demande à **find** d'afficher le résultat.
Pour chercher tous les fichiers commencant par "linux-tes" et définir à partir de quel répertoire on souhaite effectuer la recherche on utilise cette syntaxe :
[delcros@mistra delcros]$find /home/delcros -name 'linux-tes*' -print

Le nombre d'options de **find** est impressionnant. En voici quelques unes :
-type permet d'indiquer le type de fichier que l'on recherche. Si vous cherchez seulement un répertoire et non pas un fichier vous pourrez utilisez cette option :
[delcros@mistra delcros]$find /usr -type d -name bin -print

Ici, on demande à **find** de trouver les répertoires (l'argument "**d**" (comme "**d**irectory") de l'option **-type** indique que l'on cherche un répertoire) du nom de "bin" à partir du répertoire **/usr**.

-exec ou -ok permet d'exécuter une commande sur les fichiers trouvés. La différence entre **-exec** et **-ok** est que la deuxième vous demandera pour chaque fichier trouvé si vous souhaitez réellement réaliser l'opération :
[delcros@mistra delcros]$find -name 'linux-tes*' -print -ok rm {} \;

```
./linux-test
rm ... ./linux-test ? y
```

[delcros@mistra delcros]$

Dans l'option **-exec**, la paire d'accolades se substitue aux fichiers trouvés, et l'anti-slash lié au point virgule forme une séquence d'échapemment.

On peut dire que cette présentation de **find** est assez , mais j'espère qu'elle vous laisse deviner ses capacités.

2- La commande locate

La commande **locate** a la même mission que **find**. Pourtant vous verrez qu'en utilisant la commande **locate**, le fichier sera trouvé beaucoup plus rapidement. Pourquoi ? Parce que **locate** ne va pas chercher le fichier dans toute l'arborescence des répertoires mais va localiser la position du fichier dans une base de données qui contient la liste des fichiers existants. Cette base de données est en général automatiquement générée une fois par jour par le système grâce à une commande appelée **updatedb**. Sur un système Linux Redhat, cette base de donnée se trouve dans le répertoire **/usr/lib** et se nomme **locatedb**.

La syntaxe est donc simple:

[delcros@mistra delcros]$ locate nom_du_fichier

Bien que la commande **locate** soit très intéressante, elle ne possède pas la puissance des options de **find**. De plus, si vous créez des fichiers pendant la journée et que vous les recherchez avec la commande locate, il n'est pas sûr que la base de donnée ait été remise à jour. Bref, **locate** est un complément de **find**.

3-La commande which

which vous permet simplement de connaître le chemin d'un exécutable. Exemple:

[delcros@mistra delcros]$ which ls
/bin/ls
[delcros@mistra delcros]$

1.10 Trouver du texte dans un fichier (grep)

La commande **grep** est un pivot des commandes UNIX. Elle cherche une expression rationnelle dans un ou plusieurs fichiers, exemple :
[delcros@mistra delcros]$grep fouille linux-commande.html

```
grep, la commande qui vous fouille les fichiers
```

La commande a donc affiché la ligne qui contient le mot "fouille" dans le fichier linux-commande.html.

La richesse de la commande **grep** permet de faire des recherches sur plusieurs fichiers et d'avoir un format de sortie adéquat. Par exemple, le fichier linux-commande.html est déjà assez important et il serait agréable de savoir où se trouve cette ligne qui contient le mot *fouille* dans le fichier :
[delcros@mistra delcros]$grep -n fouille linux-commande.html

```
902: Grep, la commande qui vous fouille les fichiers
```

Le mot fouille se trouve à la ligne numéro 902 et c'est l'option **-n** qui nous a permis de connaître ce numéro.

Une autre option très utile est **-l** qui permet de n'afficher que les noms des fichiers contenant ce que l'on cherche :
[delcros@mistra delcros]$grep -l fouille /home/delcros/personnel/html/*

```
/home/delcros/personnel/html/linux-commande.html
```

Ici, j'ai demandé à la commande **grep** de chercher l'occurence "fouille" dans les fichiers du répertoire /home/delcros/personnel/html/. Le résultat est le nom des fichiers qui contiennent l'occurence. Ici, seul le fichier "linux-commande.html" dans le répertoire contient le mot "fouille".
Quelques-unes des autres options :
-c donne le nombre de fois où l'expression rationnelle a été rencontrée dans le fichier :
[delcros@mistra delcros]$ grep -c fouille linux-commande.html

```
10
```

-n est utile lorsque vous cherchez une expression rationnelle qui commence par un tiret car si vous n'utilisez pas l'option **-n**, **grep** la considèrera comme une option !

1.11 Les liens (ln)

Les liens forment un axe central du fonctionnement de linux. Qu'est ce qu'un lien ?

Un lien est un type spécial de fichier qui permet à plusieurs noms de fichiers de faire référence au même fichier sur le disque.

On doit distinguer deux sortes de liens :

1. **les liens durs** associent deux ou plusieurs fichiers à un même espace sur le disque, les deux fichiers sont pourtant indépendants. On peut dire que physiquement les fichiers sont les mêmes mais que virtuellement ils ne le sont pas. Prenons un exemple :
   [delcros@mistra personnel]$ln linux-test /home/delcros/linux-test-lien-dur

   le fichier *linux-test-lien-dur* est créé dans le répertoire */home/delcros*. si vous faites un **ls -l** vous constaterez que *linux-test* et *linux-test-lien* ont la même taille. Au niveau de leur existence sous linux, ils sont indépendants. Mais sur le disque, il n'existe qu'un seul fichier, simplement *linux-test-lien-dur* et *linux-test* sont sur le même espace (ou inode) sur le disque dur lorsqu'on les appelle.
   Ainsi si nous modifions le *fichier linux-test-lien-dur*, nous aurons automatiquement une modification du fichier *linux-test* (et vice et versa), car la modification s'effectuera physiquement sur le disque dur sur l'inode "partagé" par les deux fichiers.

2. **Les liens symboliques :**
   si nous faisons maitenant un lien symbolique :
   [delcros@mistra personnel]$ln -s linux-test /home/delcros/linux-test-lien-symb

   Faites un **ls -F** dans le répertoire */home/delcros*, vous verrez que le fichier *linux-test-lien-symb* est précédé du signe "@". Ce fichier pointe sur linux-test. Si vous avez fait un peu de programmation en C, nous retrouvons le concept de **pointeur**. Quand on appelle le fichier *linux-test-lien-sym*, il va automatiquement se diriger vers le fichier *linux-test*.

Quelles sont les points communs entre les liens symboliques et les liens durs ?

Le lien symbolique fait référence à un fichier dans un répertoire alors que le lien dur fait référence à un espace sur le disque dur.

- Les liens symboliques sont des fichiers de petite taille qui ont une existence propre sur le disque dur. Ces fichiers contiennent les références des fichiers sources auquels ils correspondent.

- Dans le cas d'un lien dur, la suppression de l'un des deux fichiers n'affectera pas l'autre. Dans le cas d'un lien symbolique, la suppression du fichier source entraînera un changement de comportement du fichier lien qui ne correspondra plus à un fichier valide et sera donc dit "cassé" ("broken").

Utilité des liens

Les liens sont utiles si vous souhaitez qu'un fichier apparaisse dans plusieurs répertoires, ou sous un nom différent. Imaginez que ce fichier fasse quelques megaoctets ... une copie à l'aide "cp" entraînera une perte de place non négligeable alors qu'un lien permettra de limiter l'utilisation de l'espace disque. Mieux :un lien garanti que toute modification effectuée sur ce fichier concernera toutes les apparentes « copies » dispersées.

Syntaxe de **ln** :
ln fichier-source fichier-lien ln -s permet d'effectuer un lien symbolique.

ln -b réalise une sauvegarde d'un fichier existant et dont nous aurions utilisé le nom avant de l'écraser.

ln -i demande à l'utilisateur s'il souhaite écraser le fichier qui a un lien sur le fichier source au cas ou celui-ci existerait déjà.

ln -d effectue des liens durs sur des répertoires ... seuls les utilisateurs possédant les droits adéquats pourront le faire.

1.12 Le compactage et le décompactage des fichiers au format .gz : la commande gzip

Pour compacter un fichier, taper la commande suivante :
[delcros@mistra delcros]$ gzip non_du_fichier

Pour décompacter un fichier, taper la commande suivante :
[delcros@mistra delcros]$ gzip -d non_du_fichier.gz

1.13 Le décompactage des fichiers avec la commande uncompress

Si vous rencontrez un fichier au format **.Z** (un autre type de compression plus ancien, et moins performant), vous pouvez aussi utiliser **gzip -d**.

1.14 Archivage de données : la commande "tar"

La commande **tar** permet d'archiver ou de désarchiver des répertoires et des fichiers de facon optimale.
Une des commandes dont vous aurez certainement le plus besoin est :
[root@mistra /]# tar xzf nom_du_fichier.tar.gz

Cette commande décompacte un fichier au format **.tar.gz** ou **.tgz** ; vous rencontrerez régulièrement ce genre de fichier en voulant par exemple récupérer des logiciels pour linux sur l'Internet. Le format **.tar.gz** indique que le fichier est en réalité une archive (**.tar**), c'est-à-dire que le fichier contient en réalité plusieurs fichiers, et qu'il est compacté (**.gz**). La commande précédente peut être ainsi comprise :
x (**e**x**t**ract) permet d'extraire certains fichiers d'une archive (lorsque l'on ne spécifie pas les noms des fichiers que l'on souhaite extraire de l'archive, **tar** les extrait tous).

z décompacte l'archive

f extrait un fichier donné (ici le fichier est nom_du_fichier.tar.gz).

Une autre commande permet de connaître la liste des fichiers contenus dans un fichier **.tar.gz** ou **tgz** :
[root@mistra /]#tar tvzf nom_du_fichier.tar.gz

t affiche la liste des fichiers contenus dans une archive tar.

v est le mode "verbose", qui affiche les noms des fichiers tel qu'ils ont été archivés à l'origine.

C'est donc l'option **t** qui permet de voir comment les fichiers de l'archive seront désarchivés.

La commande suivante créera une archive de tout mon répertoire **/home/delcros/personnel** :

[delcros@mistra delcros]# tar cvfz personnel.tgz personnel

c indique à tar de créer une archive

z indique a tar de compacter une archive

Ainsi tout mon répertoire personnel, avec les sous répertoires et tous les fichiers, se trouveront rassemblés dans UN fichier archive : personnel.tgz

1.15 Connaître l'espace disque utilisé (df et du)

La commande **df** permet de connaître l'emplacement de montage des systèmes de fichiers (partitions utilisables pour stocker des fichiers) accessibles sur votre système et les capacités restantes sur chacun d'eux.

```
[delcros@mistra delcros]$ df
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sda5             298762  119387   163945     42%   /
/dev/sda1              41166   17116    24050     42%   /dos
/dev/sda6            1745186 1163946   491042     70%   /usr
[delcros@mistra delcros]$
```

La commande **du** permet de connaître l'utilisation disque en kilo-octet par le répertoire spécifié et ses sous répertoires.

```
[delcros@mistra html]$ du
56      ./config
224     ./images
185     ./commandes
28      ./.xvpics
2       ./docs/preparation_debutantlinux
203     ./docs
875     .
[delcros@mistra html]$
```

1.16 Contrôler les ressources utilisées par les processus

1. La commande "**top**" :

La commande **top** vous permet d'afficher des informations en continu sur l'activité du système. Elle permet surtout de suivre les ressources que les processus utilisent (quantité de RAM, pourcentage de CPU, la durée de ce processus depuis son demarrage).
Vous pourrez utiliser l'option **-d** pour spécifier des délais de rafraîchissement (en secondes).
En cours d'utilisation de top, il est possible de stopper un process de manière interactive en tapant **k**. top demande ensuite lquel signal il doit envoyer : 15 (SIGTERM) est le signal par défaut qui met fin à un process, 9 (SIGKILL) est plus brutal.
Pour quitter top, appuyer simplement sur la touche "**q**".

2. La commande "**ps**" :
   La commande **ps** permet de connaître les processus actifs à un moment donné :
   [delcros@mistra delcros]$ ps

```
  PID   TTY STAT  TIME COMMAND
  341   p1    S     0 : 00    bash
  344   p2    S     0 : 00    bash
  1039   p3    S     0 : 00    bash
  1219   p3    R     0 : 00    ps
```

   Le "PID" est l'identificateur d'un processus, c'est un nombre. Chaque processus est identifié dans le système par un nombre unique.

   Le "TTY" indique à quel port de terminal est associé le processus.

   "STAT" indique l'état dans lequel se trouve le processus. Dans l'exemple, trois processus sont endormis (S comme "sleep"), et un processus en cours d'exécution (R comme "run"). Le processus qui est en cours d'exécution n'est autre que la commande "**ps**" que nous venons de lancer.

   Le "TIME" indique depuis combien de temps le processus utilise les ressources du microprocesseur.

   Le "COMMAND" précise, comme son nom l'indique, la commande dont l'état est décrit par PID, TTY, STAT et TIME.

   Ceci dit, une simple commande "**ps**" n'indique pas tous les processus du système. Le simple fait de lancer **ps** nous a juste indiquer les processus associés à un terminal et qui dépendent de l'utilisateur courant (ici "delcros").
   En fait, il est tout a fait probable que d'autres processus non liés à un terminal aient été lancés par "delcros". J'en suis d'ailleurs sur, puisque actuellement j'utilise emacs pour réaliser cette modeste page de documentation et que pour visualiser le résultat, j'utilise netscape :
   [delcros@mistra delcros]$ ps -x

```
    PID TTY STAT  TIME COMMAND
    240  ?  S    0:01 /usr/X11R6/bin/fvwm2
    246  ?  S    0:00 /usr/X11/bin/xautolock -corners ++++ -time 5
  -locker /usr/X
    247  ?  S    0:00 /usr/X11/bin/unclutter -idle 3
    253  ?  S    0:00 /usr/local/bin/Periodic
    254  ?  S    7:34 emacs --background grey79 -geometry 80x58+-
  4+-11
    257  p0 S    0:00 bash
    258  p2 S    0:00 bash
    259  p1 S    0:00 bash
    272  ?  S    0:00 /usr/lib/emacs/19.34/i386-gnu-
  linux/emacsserver
   2134  ?  S    0:00 /usr/bin/ispell -a -m -d francais
   6431  p0 S    1:03 /usr/lib/netscape/netscape-navigator
```

```
6441  p0 S    0:00 (dns helper)
6741  p0 R    0:00 ps -x
```

Les commandes qui ne sont pas associées à un terminal sont reconnaissable par le point d'interrogation qui rempli le champs TTY.

Si vous voulez connaître tous les processus de la machine de tous les utilisateurs, il suffit d'utiliser l'option **ax**. Si en plus vous voulez connaître les utilisateurs associés à chaque processus, il vous suffit d'utiliser l'option **aux**. Vous verrez alors plusieurs colonnes s'ajouter dont "USER" qui indique à quel utilisateur appartient le processus. "%CPU" indique en pourcentage les ressources du microprocesseur utilisées par le processus. "%MEM" montre en pourcentage les ressources en mémoire vive utilisées par le processus. "RSS" donne réellement la mémoire utilisée en kilobytes par le processus. "START" indique l'heure à laquelle le processus a été lancé.

Comment être plus précis ? : -)

3. La commande "**pstree**" :
Cette commande permet d'afficher les processus sous forme d'arborescence et donc de voir leurs inter-dépendances :
[delcros@mistra delcros]$ pstree
init-+-crond
|-emacs---emacsserver
|-gpm
|-inetd
|-kerneld
|-kflushd
|-klodg
|-kswapd
|-loadmeter
|-lpd
|-6*[mingetty]
|-named
|-netscape---netscape
|-4*[nfsiod]
|-nxterm---slrn-gor---slrn
|-portmap
|-pppd |-rc.news---innwatch---sleep
|-rpc.mountd
|-rpc.nfsd
|-rpc.yppasswdd
|-sendmail
|-syslogd
|-update
|-xconsole
|-xdm-+-X
| `-xdm---Xsession---fvwm---FvwmPager
|-xterm---bash---su---bash---tail
|-2*[xterm---bash]
|-xterm---bash---pstree
\ -ypserv

On voit par exemple ici que j'utilise Fvwmpager qui depend en fait lui-même de fvwm et lui même dépend de Xwindow ici lancé grace à xdm (vous n'obtiendrez pas la même chose que

moi si vous lancez Xwindow grâce à la commande startx, en effet xdm permet de lancer automatiquement Xwindow au démarrage de linux).

4. La commande "**kill**" :
   La commande "**kill**" permet d'expédier un signal à un processus en cours.
   Sa syntaxe est la suivante :
   kill [options] PID

   Par exemple, si j'ai lancé une connexion à l'Internet en PPP, un processus pppd sera en cours. Pour tuer le processus, je peux d'abord faire un **ps -ax** pour connaître le numero du PID de pppd et ensuite si par exemple le PID est 592, je peux tuer la connexion en faisant :
   [root@mistra delcros]# kill 592

   Vous remarquerez que je suis logué en utilisateur "root" pour faire ceci, en effet le processus pppd appartenait à l'utilisateur "root" et un autre utilisateur ne peut pas lui expédier de signal.

   Si un processus vous résiste, c'est à dire que vous n'arrivez pas à le tuer, vous devez utiliser la commande : **kill -9 PID** (PID étant toujours le numéro de de processus).

   La commande "**killall**" permet aussi de tuer un processus mais au lieu d'indiquer le PID vous indiquerez le nom du processus.

   Mais **attention**, plusieurs processus peuvent utiliser la même commande. Ainsi, si vous tapez :
   [delcros@mistra delcros]# killall grep

   Vous tuerez tous les processus qui contiennent la commande grep. Je vous recommande donc d'utiliser l'option "-i" qui vous demande une confirmation avant de tenter d'arrêter un processus..

1.17 La connexion de plusieurs commandes : les pipes

Qu'est ce qu'un "pipe" (parfois appelé « tube ») ? Si on le décrit ce n'est rien d'autre que cette barre verticale que vous pouvez obtenir avec la combinaison de touches "Altgr + 6" sur les clavier français classiques, ou "Altgr + 1" sur les claviers franco-belges. Un tube permet de passer le résultat d'une commande à autre commande. Un exemple permettra de comprendre tout cela beaucoup plus facilement :

Je veux savoir quels sont tous les processus "bash" qui fonctionnent sur le système, mais je veux que la commande **ps aux** ne me fournisse les lignes que les lignes qui contiennent le mot "bash" pour m'eviter d'avoir à parcourir toute la longue liste qu'affiche **ps aux** :

```
[delcros@mistra html]$ ps aux | grep bash
delcros    367    0.0   1.8   1600   568   p2   S   18 : 14   0 : 00
bash
delcros    426    0.0   2.2   1624   704   p3   S   18 : 17   0 : 00
bash
delcros   1261   0.0   2.2   1608   692   p6   S   21 : 22   0 : 00
bash
delcros   1332   0.0   2.4   1616   772   ?     S   21 : 41   0 : 00
bash
delcros   1582   0.0   2.7   1604   844   p8   S   22 : 30   0 : 00
bash -rcfile .bashrc
delcros   2796   0.0   0.9     908   300   p3   S   02 : 17   0 : 00
grep bash
root        1162   0.0   2.1   1596   664   ?     S   21 : 06   0 : 00
bash
```

On peut dire que l'on a "connecté" deux commandes entre elles. Mais vous pouvez ainsi en connecter autant que vous voulez en utilisant cette syntaxe :
commande1 | commande2 | commande3 ... | commandeN Si on se rend compte de l'utilité des pipes, progressivement on les utilise et on fini par ne plus s'en passer.

1.18 Les redirections

Quand on parle de redirection, on parle plus précisemment de la redirection des entrées-sorties que traitent ou engendrent les programmes. Par exemple, lorsque vous tapez des commandes au prompt de linux, vous effectuez une entrée de caractère grâce au clavier et linux vous donne une sortie en vous donnant à l'écran le résultat de votre commande. Mais l'entrée de données peut se faire autrement que par le clavier, en indiquant par exemple un fichier qui contient des données à traiter. La sortie peut aussi s'effectuer ailleurs que sur l'écran, sur l'imprimante par exemple.

Ainsi, lorsque nous parlons des entrées sorties, nous parlons aussi des périphériques de l'ordinateur. On considérera que les périphériques sont des fichiers a part entière car, sous UNIX, des fichiers spéciaux permettent l'accès aux périphériques se trouvent dans le répertoire **/dev**. Dans la plupart des cas ce que l'on y copie va vers le périphérique.

Mais comment faire pour rediriger une entrée ou une sortie ?

Comment faire par exemple pour que la commande **cat** qui affiche un fichier à l'écran, sorte plutot le fichier dans un autre fichier ou vers une imprimante ? C'est le signe **>** qui va nous permettre de réaliser ceci.

Il est temps de prendre un exemple....

Dans un premier cas, je veux que linux m'affiche le fichier test à l'ecran :
[delcros@mistra delcros]$ cat test

Vous allez voir s'afficher à l'écran le fichier test.

Dans un deuxième cas, je veux que linux place le fichier test dans un fichier test2 au lien de l'afficher à l'écran :
[delcros@mistra delcros]$ cat test > test2

Dans un troisième cas, je veux que linux imprime le fichier au lieu de l'afficher à l'écran :
[delcros@mistra delcros]$ cat test > /dev/lp0

Quelques constats s'imposent :
1- La sortie sur un autre fichier n'est rien d'autre avec la commande **cat** qu'une copie du fichier "test" en "test2". La commande **cp** nous permet aussi de faire cela.

2- Dans la redirection vers l'imprimante nous avons indiqué le fichier spécial **/dev/lp0** qui correspond au port **LPT1** où est connectée mon imprimante.

La commande **cat** affiche son résultat vers la sortie standard qui est le terminal.

Par défaut le terminal est la sortie standard, ce descripteur de fichier est désigné par le chiffre "1"

L'entrée standard dans un système UN*X est le clavier et est désigné par le chiffre "0".

Il existe un troisième **descripteur de fichier** qui est la sortie des erreurs produites par l'exécution d'une commande.
La sortie des erreurs se fait par défaut sur le terminal et est désigné par le chiffre "2".

Plusieurs types de redirection existent :

- "**> fichier**" qui permet de rediriger le résultat d'une commande vers une sortie que nous choisissons.

- "**< fichier**" permet de spécifier une entrée standard.
- "**>> fichier**" permet comme le signe "**>**" de rediriger la sortie standard vers un fichier, mais si le fichier spécifié existe déjà, la sortie sera ajouté à ce qui existe déjà dans le fichier alors qu'avec un simple "**>** le fichier spécifié serait écrasé.
- "**<> fichier**" permet de spécifier un fichier comme étant en même temps l'entrée standard et la sortie standard.
- "**n> fichier**" permet de rediriger la sortie d'un des descripteurs de fichiers vers un fichier. Par exemple, si vous souhaitez obtenir les erreurs standards dans un fichier vous n'aurez qu'à utiliser cette syntaxe :
  commande 2> erreurs
- "**n< fichier**" permet de spécifier un fichier comme étant un des descripteurs de fichier.
- "**>&n**" permet de dupliquer la sortie standard vers un des descripteurs de fichier.
- "**<&n**" permet de dupliquer l'entrée standard depuis un des descripteurs de fichier.
- "**&> fichier**" permet de rediriger la sortie standard et l'erreur standard vers un seul et même fichier.

À première vue, on se demande bien à quoi peut servir certaines des redirections ...

On les découvre au fur et à mesure, mais une des plus utiles est 2>&1 qui permet de rediriger les erreurs vers la sortie standard. Elle est très appréciée des utilisateurs lorsque par exemple ceux-ci n'arrivent pas à lancer l'interface X-Window. Il est alors courant de recourir à la commande suivante afin d'obliger X à placer tous ses messages dans un fichier nommé **erreursX** que l'on pourra consulter ensuite à loisir :
[delcros@mistra delcros]$ startx 2>&1 erreursX.tmp

2. Bash et ses capacités

2.1 Personnalisation du bash

2.1.1 Où modifier mes paramètres ?

bash contient des variables qui permettent d'adapter son environnement à ses besoins :
Il existe un fichier qui met en place une grande partie des variables d'environnement : le fichier **.bash_profile** (ou .profile).

Pour que les variables d'environnement soit prises en compte vous devez vous reloguer sur votre compte(avec la commande "**su - nom_utilisateur** (si vous avez modifié le **.bash_profile**) ou alors passer les variables directement en ligne de commande (dans ce cas, les variables ne seront pas enregistrées dans le **.bash_profile**).

Il est possible de modifier globalement les variables d'environnement, c'est à dire pour tous les utilisateurs en éditant le fichier /etc/profile. Pour que les modifications soient prises en compte pour vous à la volée, tapez :

source /etc/profile

Sinon au prochain démarrage, se sera le cas.

2.1.2 La personnalisation des variables d'environnement

Pour que toutes ces variables soient prises en compte immédiatement, vous pouvez soit relancer votre terminal, soit tapez dedans export Variable="..."

2.1.2.1 Le PATH

Vous trouverez par exemple la variable **PATH** qui définit les chemins existant pour les exécutables. Si par exemple, votre chemin **PATH** est de la forme :
PATH=$PATH:/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin

et que vous souhaitez ajouter dans ce chemin un répertoire /home/delcros/binaire qui contient votre script bash ou vos programmes personnels, il vous faudra ajouter ce chemin à la variable **PATH** :
PATH=$PATH:/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/home/delcros/binaire

(notez la présence de "**:**" entre chaque nom de répertoire).

2.1.2.2 http_proxy et ftp_proxy

Ces 2 variables d'environnement permettent d'accéder au net depuis un shell. Pour bash par exemple, éditez le fichier .bashrc de votre home et mettez :
export http_proxy="http://x.x.x.x:port/"
export ftp_proxy="ftp://x.x.x.x:port/"

2.1.2.3 L'invite de commande PS1

La variable **PS1** contient la forme de votre invite :
PS1="[\u@\h \w]" affichera votre nom d'utilisateur (\u); "@"; le nom de la machine (\w); un espace; le répertoire de travail courant (\w). Voilà ce que cela donne :

```
[delcros@mistra /usr/X11]
```

Voici une autre configuration d'invite qui contient quasiment toutes les options possibles :
PS1="[\t \d \u@\h \w \$]"

ce qui donne :
```
[21:47 : 13 Sun Apr 26 delcros@mistra /usr/X11 $]
```

On peut faire beaucoup mieux en utilisant les variables prédéfinies :

\d pour ajouter la date (format anglais)
\t pour ajouter l'heure (HH:MM:SS)
\u pour ajouter le nom de l'utilisateur
\r pour un retour à la ligne
\w pour ajouter le chemin complet du répertoire courant
\W pour ajouter le répertoire courant
\h pour ajouter le nom de la machine
\$ pour afficher une terminaison personnalisée ($ pour les utilisateurs et # pour root)

2.1.2.4 Stockage des mail MAIL

Une autre variable utile est **MAIL**. Normalement, vos mails arrivent dans le répertoire /var/spool/mail/nom_utilisateur

Vous pouvez placer cette variable dans votre .bash_profile avec cette forme :
MAIL=/var/spool/mail/nom_utilisateur

2.1.3 Les alias

Les alias sont une des choses les plus pratiques qui soient. Régulièrement on utilise les mêmes commandes avec parfois de nombreuses options. Les alias se placent habituellement dans le fichier de configuration **.bashrc**. Voici un exemple classique d'alias :
`alias l="ls --color=auto"` Avec cet alias, vous n'aurez plus besoin de spécifier systématiquement l'option "--color" qui permet de lister en couleur le contenu d'un répertoire. Il vous suffira simplement de taper l'alias "**l**".

Ainsi, le mini script que nous avions réalisé au début de cette section pourrait aussi se faire grâce à un simple alias :
```
alias montar="tar xvzf"
```

2.2 Programmation en bash (scripts)

2.2.1 Variables et évaluation

Les variables sont stockées comme des chaînes de caractères.
Les variables *d'environnement* sont des variables exportées aux processus (programmes) lancés par le shell. Les variables d'environnement sont gérées par UNIX, elles sont donc accessibles dans tous les langages de programmation (voir plus loin).
Pour définir une variable :

```
$ var='ceci est une variable'
```

Attention : pas d 'espaces autour du signe égal. Les quotes (apostrophes) sont nécessaires si la valeur contient des espaces. C'est une bonne habitude de toujours entourer les chaînes de caractères de quotes.
Pour utiliser une variable, on fait précéder son nom du caractère "`$`" :

```
$ echo $var
```

ou encore :

```
$ echo 'bonjour, ' $var
```

Dans certains cas, on doit entourer le nom de la variable par des accolades :

```
$ X=22
$ echo Le prix est ${X}0 euros
```

affiche "Le prix est de 220 euros" (sans accolades autour de X, le shell ne pourrait pas savoir que l'on désigne la variable X et non la variable X0).
Lorsqu'on utilise une variable qui n'existe pas, le bash renvoie une chaîne de caractère vide, et n'affiche pas de message d'erreur (contrairement à la plupart des langages de programmation, y compris csh) :

```
$ echo Voici${UNE_DROLE_DE_VARIABLE}!
```

affiche "Voici!".
Pour indiquer qu'une variable doit être exportée dans l'environnement (pour la passer aux commandes lancée depuis ce shell), on utilise la commande `export` :

```
$ export X
```

ou directement :

```
$ export X=22
```

Accès aux variables d'environnement dans des programmes

- **En langage Python** : on accède aux variables via un *dictionnaire* défini dans le module `os` : `os.environ`.
  Ainsi, la valeur de la variable `$PATH` `os.environ['PATH']`.
- **En langage C** : les fonctions `getenv` et `setenv` permettent de manipuler les variables d'environnement. Le programme suivant affiche la valeur de la variable `PATH` :

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    char *var = getenv("PATH");
    printf("la valeur de PATH est %s\n", var );
}
```

Évaluation, guillemets et quotes

Avant évaluation (interprétation) d'un texte, le shell substitue les valeurs des variables. On peut utiliser les guillemets (`"`) et les quotes (`'`) pour modifier l'évaluation :

- les guillemets permettent de grouper des mots, sans supprimer le remplacement des variables. Par exemple, la commande suivante ne fonctionne pas :

```
$ x=Hauru no
bash: no: command not found
```

Avec des guillemets c'est bon :

```
$ x="Hauru no"
```

On peut utiliser une variable entre guillemets :

```
$ y="Titre: $x Ugoku Shiro"
$ echo $x
Titre: Hauru no Ugoku Shiro
```

- les quotes (apostrophes) groupes les mots et suppriment toute évaluation :

```
$ z='Titre: $x Ugoku Shiro'
$ echo $z
Titre: $x Ugoku Shiro
```

Expressions arithmétiques

Normalement, bash traite les valeurs des variables comme des chaînes de caractères. On peut effectuer des calculs sur des nombres entiers, en utilisant la syntaxe `$(( ... ))` pour délimiter les expressions arithmétiques:

```
$ n=1
$ echo $(( n + 1 ))
2
$ p=$((n * 5 / 2 ))
$ echo $p
2
```

Découpage des chemins

Les scripts shell manipulent souvent des chemins (*pathnames*) et des noms de fichiers. Les commandes `basename` et `dirname` sont très commodes pour découper un chemin en deux partie (répertoires, nom de fichier) :

```
$ dirname /un/long/chemin/vers/toto.txt
/un/long/chemin/vers
$ basename /un/long/chemin/vers/toto.txt
toto.txt
```

Évaluation de commandes

Il est courant de stocker le résultat d'une commande dans une variable. Nous entendons ici par "résultat" la *chaîne affichée par la commande*, et non son code de retour.
Bash utilise plusieurs notations pour cela : les *back quotes* (`) ou les parenthèses :

```
$ REP=`dirname /un/long/chemin/vers/toto.txt`
$ echo $REP
```

```
/un/long/chemin/vers
```

ou, de manière équivalente :

```
$ REP=$(dirname /un/long/chemin/vers/toto.txt)
$ echo $REP
/un/long/chemin/vers
```

(attention encore une fois, pas d'espaces autour du signe égal).
La commande peut être compliquée, par exemple avec un tube :

```
$ Fichiers=$(ls /usr/include/*.h | grep std)
$ echo $Fichiers
/usr/include/stdint.h /usr/include/stdio_ext.h /usr/include/stdio.h
/usr/include/stdlib.h /usr/include/unistd.h
```

Découpage de chaînes

Bash possède de nombreuses fonctionnalités pour découper des chaînes de caractères. L'une des plus pratiques est basée sur des motifs.
La notation ## permet d'éliminer la *plus longue* chaîne en correspondance avec le motif :

```
$ Var='tonari no totoro'
$ echo ${Var##*to}
ro
```

ici le motif est `*to`, et la plus longue correspondance "tonari no toto". Cette forme est utile pour récupérer l'extension (suffixe) d'un nom de fichier :

```
$ F='rep/bidule.tgz'
$ echo ${F##*.}
tgz
```

La notation # (un seul #) est similaire mais élimine la *plus courte* chaîne en correspondance :

```
$ Var='tonari no totoro'
$ echo ${Var#*to}
nari no totoro
```

De façon similaire, on peut éliminer la fin d'une chaîne :

```
$ Var='tonari no totoro'
$ echo ${Var%no*}
tonari
```

Ce qui permet de supprimer l'extension d'un nom de fichier :

```
$ F='rep/bidule.tgz'
$ echo ${F%.*}
rep/bidule
```

`%` prend la plus courte correspondance, et `%%` prend la plus longue :

```
$ Y='archive.tar.gz'
$ echo ${Y%.*}
```

```
archive.tar
$ echo ${Y%%.*}
archive
```

2.2.2 Exécution conditionnelle

L'instruction `if` permet d'exécuter des instructions si une condition est vraie. Sa syntaxe est la suivante :

```
if [ condition ]
then
    action
fi
```

`action` est une suite de commandes quelconques. On peut aussi utiliser la forme complète :

```
if [ condition ]
then
    action1
else
    action2
fi
```

ou encore enchaîner plusieurs conditions :

```
if [ condition1 ]
then
    action1
elif [ condition2 ]
then
    action2
elif [ condition3 ]
then
    action3
else
    action4
fi
```

Opérateurs de comparaison

Le shell étant souvent utilisé pour manipuler des fichiers, il offre plusieurs opérateurs permettant de vérifier diverses conditions sur ceux-ci : existence, dates, droits. D'autres opérateurs permettent de tester des valeurs, chaînes ou numériques. Le tableau ci-dessous donne un aperçu des principaux opérateurs :

Opérateur	**Description**	**Exemple**
Opérateurs sur des fichiers		
-e *filename*	vrai si *filename* existe	`[ -e /etc/shadow ]`
-d *filename*	vrai si *filename* est un répertoire	`[ -d /tmp/trash ]`
-f *filename*	vrai si *filename* est un fichier ordinaire	`[ -f /tmp/glop ]`
-L *filename*	vrai si *filename* est un lien symbolique	`[ -L /home ]`
-r *filename*	vrai si *filename* est lisible (r)	`[ -r /boot/vmlinuz ]`
-w *filename*	vrai si *filename* est modifiable (w)	`[ -w /var/log ]`
-x *filename*	vrai si *filename* est exécutable (x)	`[ -x /sbin/halt ]`
file1 -nt *file2*	vrai si *file1* plus récent que *file2*	`[ /tmp/foo -nt /tmp/bar ]`

file1 -ot *file2*	vrai si *file1* plus ancien que *file2*	`[ /tmp/foo -ot /tmp/bar ]`
Opérateurs sur les chaînes		
-z *chaine*	vrai si la *chaine* est vide	`[ -z "$VAR" ]`
-n *chaine*	vrai si la *chaine* est non vide	`[ -n "$VAR" ]`
chaine1 = *chaine2*	vrai si les deux chaînes sont égales	`[ "$VAR" = "totoro" ]`
chaine1 != *chaine2*	vrai si les deux chaînes sont différentes	`[ "$VAR" != "tonari" ]`
Opérateurs de comparaison numérique		
num1 -eq *num2*	égalité	`[ $nombre -eq 27 ]`
num1 -ne *num2*	inégalité	`[ $nombre -ne 27 ]`
num1 -lt *num2*	inférieur (<)	`[ $nombre -lt 27 ]`
num1 -le *num2*	inférieur ou égal (< =)	`[ $nombre -le 27 ]`
num1 -gt *num2*	supérieur (>)	`[ $nombre -gt 27 ]`
num1 -ge *num2*	supérieur ou égal (> =)	`[ $nombre -ge 27 ]`

Quelques points délicats doivent être soulignés :

- Toutes les variables sont de type chaîne de caractères. La valeur est juste convertie en nombre pour les opérateurs de conversion numérique.
- Il est nécessaire d'entourer les variables de guillemets (`"`) dans les comparaisons. Le code suivant affiche "OK" si `$var` est égale à "tonari no totoro" :

  ```
  if [ "$myvar" = "tonari no totoro" ]
  then
     echo "OK"
  fi
  ```

  Par contre, si on écrit la comparaison comme
  `if [ $myvar = "tonari no totoro" ]` le shell déclenche une erreur si `$myvar` contient plusieurs mots. En effet, la substitution des variables a lieu avant l'interprétation de la condition.

2.2.3 Scripts shell

Un script bash est un simple fichier texte exécutable (droit `x`) commençant par les caractères `#!/bin/bash` (doivent être les premiers caractères du fichier).
Voici un exemple de script :

```
#!/bin/bash

if [ "${1##*.}" = "tar" ]
then
        echo $1 est une archive tar
else
        echo $1 n\'est pas une archive tar
fi
```

Ce script utilise la variable `$1`, qui est le premier argument passé sur la ligne de commande.

Arguments de la ligne de commande

Lorsqu'on entre une commande dans un shell, ce dernier sépare le nom de la commande (fichier exécutable ou commande interne au shell) des arguments (tout ce qui suit le nom de la commande, séparés par un ou plusieurs espaces). Les programmes peuvent utiliser les arguments (options, noms de fichiers à traiter, etc).
En bash, les arguments de la ligne de commande sont stockés dans des variables spéciales :

`$0, $1, ...`	les arguments
`$#`	le nombre d'arguments
`$*`	tous les arguments

Le programme suivant illustre l'utilisation de ces variables :

```
#!/bin/bash

echo 'programme  :' $0
echo 'argument 1 :' $1
echo 'argument 2 :' $2
echo 'argument 3 :' $3
echo 'argument 4 :' $4
echo "nombre d'arguments :" $#
echo "tous:" $*
```

Exemple d'utilisation, si le script s'appelle "`myargs.sh`" :

```
$ ./myargs.sh un deux trois
programme  : ./myargs.sh
argument 1 : un
argument 2 : deux
argument 3 : trois
argument 4 :
nombre d'arguments : 3
tous: un deux trois
```

2.2.4 Autres structures de contrôle

Nous avons déjà évoqué l'instruction `if` et les conditions. On utilise souvent des répétitions (for) et des choix multiples (case).

Boucle for

Comme dans d'autre langages, la boucle for permet d'exécuter une suite d'instructions avec une variable parcourant une suite de valeurs. Exemple :

```
for x in un deux trois quatre
do
    echo x= $x
done
```

affichera :

```
x= un
x= deux
x= trois
x= quatre
```

On utilise fréquemment for pour énumérer des noms de fichiers, comme dans cet exemple :

```
for fichier in /etc/rc*
do
   if [ -d "$fichier" ]
   then
```

```
        echo "$fichier (repertoire)"
    else
        echo "$fichier"
    fi
done
```

Ou encore, pour traiter les arguments passés sur la ligne de commande :

```
#!/bin/bash

for arg in $*
do
  echo $arg
done
```

Instruction case

L'instruction case permet de choisir une suite d'instruction suivant la valeur d'une expression :

```
case "$x" in
  go)
      echo "demarrage"
      ;;
  stop)
      echo "arret"
      ;;
   *)
      echo "valeur invalide de x ($x)''
esac
```

Noter les deux `;` pour signaler la fin de chaque séquence d'instructions.

Création de menus

Nous pourrions utiliser aussi un menu qui nous permettrait de choisir entre une décompression immédiate ou une visualisation du contenu de l'archive :

```
#!/bin/bash
PS3='votre choix ?'
select choix in "tar tvzf" "tar xvzf"
do
$choix $1;
done
```

La construction **select** permet de générer des menus avec une grande facilité.
PS3 est une variable qui permet de stocker une chaîne d'invite qui est utilisée par **select**.

"choix" est le nom de la variable qui contiendra un des éléments de la suite qui suit le mot clé **in**. Dans notre cas, "choix" contiendra soit la chaîne "tar tvzf" ou la chaîne "tar xvzf".

Dans la construction **do... done**, nous placons les commandes que nous voulons executer. Ici "$choix" contiendra donc soit "tar tvzf" soit "tar xvzf" et "$1" contiendra l'argument (ici le nom du fichier compressé) que l'on aura indiqué à l'execution de notre script.

Si notre script s'appelle "ctgz", son exécution se déroulera ainsi :

```
[delcros@mistra binaire]$ ./ctgz fichier.tar.gz
1) tar tvzf
2) tar xvzf
votre choix ?
```

L'utilisateur n'a plus qu'à taper "1" ou "2".

```
select nom [in liste]
do
instructions utilisant la $nom
done
```

2.2.5 Les Metacaractères, Opérateurs et commandes spéciales

- ***** désigne une chaîne de caractères quelconque ne commençant pas par un metacaractère .
- **?** désigne un caractère quelconque.
- **#** précède une ligne de commentaires.
- **[aA]** désigne les caractères A et a
- **[0-9a-zA-Z]** désigne un caractère alphanumérique quelconque.
- **[!0-9]** désigne l'ensemble des caractères qui ne sont pas des chiffres.
- **;** sépare deux commandes (ou plus) situées sur une même ligne.
- **&&**

  Dans l'exemple : **commande1 && commande2**, la commande2 ne sera exécutée que si la commande1 se termine par un succès.
- ||

  Dans l'exemple : **commande1** || **commande2**, la commande2 ne sera exécutée que si la commande1 se termine par un échec.
- |

  Dans l'exemple : **commande1** | **commande2**, l'entrée de la commande2 est la sortie de la commande1
- \ protège le caractère qui suit, que ce soit un caractère normal ou un métacaractère du shell (sauf à l'intérieur d'une chaîne délimitée par des ').
- **(** et **)** permettent de regrouper un ensemble de commandes et de les exécuter dans un "shell fils"
- **exit** Vous permet de sortir d'un programme en cours d'exécution.

2.2.6 Définition de fonctions

Il est souvent utile de définir des fonctions. La syntaxe est simple :

```
mafonction() {
   echo "appel de mafonction..."
}

mafonction
mafonction
```

qui donne :

```
appel de mafonction...
appel de mafonction...
```

Ne pas oublier qu'une variable utilisée à l'intérieur d'une fonction est globale! Pour éviter ce phénomène, il faut rajouter local à la déclaration de la variable :

local MA_VARIABLE

Voici pour terminer un exemple de fonction plus intéressant :

```
tarview() {
    echo -n "Affichage du contenu de l'archive $1 "
    case "${1##*.}" in
        tar)
           echo "(tar compresse)"
           tar tvf $1
        ;;
        tgz)
           echo "(tar compresse gzip)"
           tar tzvf $1
        ;;
        bz2)
           echo "(tar compresse bzip2)"
           cat $1 | bzip2 -d | tar tvf -
        ;;
        *)
           echo "Erreur, ce n'est pas une archive"
        ;;
    esac
}
```

Plusieurs points sont à noter :

- `echo -n` permet d'éviter le passage à la ligne;
- La fonction s'appelle avec un argument (`$1`)

```
tarview toto.tar
```

3. Organisation des répertoires

Voici l'arborescence d'un système UNIX classique :
/ est le répertoire racine, tous les autres répertoires en dépendent. Par exemple le répertoire où est "monté" mon CD-ROM est sur /mnt/cdrom. On n'a donc pas comme sous MS-DOS, différentes lettres qui correspondent à différents lecteurs disctincts physiquement. Les lecteurs sont harmonieusement montés en répertoires dans l'arborescence UNIX.

/bin contient les binaires fondamentaux à la gestion de Linux. On y retrouve par exemple les commandes précédemment étudiées.

/dev contient une multitudes de fichiers dits spéciaux. L'un deux correspond à mon modem. Je dois indiquer ce fichier dans la configuration de mes outils de communication. De même **/dev/hda1** correspond à la première partition de mon disque dur IDE, si mon disque dur est un SCSI, son nom sera **/dev/sda1**. Un dernière exemple : /dev/fd0 correspond à mon lecteur de disquettes. Pour une application, allez voir la "section consacrée à mount".

/etc contient tous les fichiers de configuration de linux. On y retrouve par exemple le fichier /etc/passwd, qui définit les mots de passe des utilisateurs.

/sbin contient les binaires du système. On y trouve par exemple la commande **shutdown** qui permet d'arrêter l'ordinateur.

/home est le répertoire qui contient les répertoires des utilisateurs du système. Le répertoire des utilisateurs est automatiquement créé avec la création d'un compte. J'ai par exemple dans mon ordinateur un compte que j'utilise en permanence (comme maintenant, pendant la rédaction de ce petit guide), tous mes fichiers personnels sont dans /home/delcros. J'ai un autre utilisateur de ma machine, lui se logue en tant que « gorka ». Il stocke ses fichiers dans le répertoire **/home/gorka**).

/lost+found est le répertoire des fichiers perdus. Ces fameux fichiers qui, du fait d'erreur disque, se retrouvent sans chemin d'accès. Le binaire **fsck**, qui est lancé régulièrement au démarrage de linux, se charge de les détecter et de les stocker dans le répertoire /lost+found

/tmp est un répertoire accessible par tous les utilisateurs du système, il permet de ne pas encombrer son répertoire personnel par des fichiers que l'on souhaite de toute manière rapidemment détruire ou modifier.

/var/spool est le répertoire des fichiers qui servent de file d'attente. Par exemple, les files d'attente de l'imprimante se trouvent sous ce répertoire. Les données à imprimer, envoyer, ... sont stockées dans ces files d'attentes jusqu'à ce qu'elles soient traitées.

/usr contient grosso modo tout ce qui concerne les binaires utiles à tous les utilisateurs et quelques commandes d'administration. On y trouve cependant d'autres choses :
/usr/bin contient donc les binaires disponibles pour les utilisateurs et les scripts.

/usr/X11R6 contient tout ce qui concerne Xfree86 (les bibliothèques, les binaires, la documentation).

/usr/include contient tous les "headers" nécessaires à la programmation dans les différents languages.

/usr/lib contient toutes les bibliothèques nécessaires au fontionnement des logiciels. (comme par exemple la bibliothèque C ou C++ ou tcl/tk).

/usr/local on y met ce qu'on veut, mais surtout les fichiers d'usage local. J'y place les logiciels qui ne sont pas habituellement livrés avec linux et que j'ai trouvé dans d'autres CD-ROM ou sur l'Internet.

4. Quelques commandes d'administration système

4.1 Placer les propriétés (chmod)

Introduction : linux permet de spécifier les droits qu'ont les utilisateurs sur un fichier. Pour voir ces droits, il suffit d'utiliser la commande **ls -l** :
[delcros@mistra delcros]$ ls -l perso
-rw-r--r-- 1 delcros delcros 9 Jul 19 12 : 39 perso

c'est la partie qui contient : *-rw-r--r--* qui nous intéresse pour l'instant.

Le premier tiret signifie que perso est un fichier tout ce qu'il y a de plus classique. Si à la place du premier tiret on observait un "**d**" cela signifierait qu'en réalité le fichier est un répertoire. Si à la place du premier tiret on observe un "**l**", cela signifie que le fichier est un lien.

Ensuite nous devons décomposer en trois parties les 9 dernières caractères :
rw- | r-- | r--

1. La première partie fixe les droits de propriétés pour le propriétaire du fichier.
2. La deuxième partie fixe les droits accordés aux utilisateurs faisant partie du groupe auquel appartient le fichier.
3. La dernière partie fixe les droits des autres utilisateurs.

Dans chaque partie, le premier caractère correspond au droit de lecture ("**r**"), la deuxième caractère correspond au droit d'écriture ("**w**"), le troisième caractère correspond au droit d'exécution ("**x**"). Si à la place d'un des caractères nous ne voyons qu'un tiret "**-**", c'est que le droit n'est pas autorisé.

On voit ainsi que tous les utilisateurs ont le droit de lire ("**r**" comme "read") le fichier et que seul son propriétaire a le droit de le modifier ("**w**" comme "write").

Par contre personne ne peut exécuter ce fichier (normal ce n'est ni un script, ni un binaire). Si par exemple tout le monde pouvait exécuter le fichier on aurait le dernier tiret de chaque partie remplacé par un "**x**" comme "eXécutable".

rwx | r-x | r-x

Cette spécificité d'UNIX sur la méthode de fixation des permissions sur un fichier assure une très grande sécurité et une très grande souplesse.

Dès maintenant, nous donnerons la lettre "**u**" pour le propriétaire du fichier, la lettre "**g**" pour le groupe d'utilisateur qui possède le fichier, la lettre "**o**" pour les autres utilisateurs. La lettre "**a**" nous permettra de faire référence à tous les utilisateurs. Cette notation est nécessaire car c'est celle que l'on doit utiliser avec la commande **chmod**.

C'est donc la commande **chmod** qui permet de modifier ces permissions qu'ont les utilisateurs sur le fichier. Évidemment, seul le propriétaire du fichier a le pouvoir de modifier ces permissions (à part bien sur le superutilisateur "root" qui peut faire absolulement tout ce que bon lui semble ...

Par exemple, nous décidons que n'importe qui pourra modifier notre fichier linux-test :
[delcros@mistra delcros]$ chmod a+w linux-test

"**a**" indique que tous les utilisateurs seront touchés par la modification des permissions

"+" signifie que c'est une permission supplémentaire que l'on donne. Pour en supprimer une il suffit de remplacer le signe "+" par "**-**".

"**w**" signifie que c'est la permission d'écriture que nous donnons.

Pour vérifier que tout a bien fonctionné, faites un "**ls -l linux-test**", nous obtenons :
-rw-rw-rw- 1 delcros delcros 9 Jul 19 19 : 03 linux-test

Si maintenant nous voulons supprimer ce droit d'écriture mais aussi le droit de lecture pour le groupe propriétaire et les autres utilisateurs nous utilisons la syntaxe suivante :
[delcros@mistra delcros]$ chmod go-wr linux-test

"**go**" signifie que la commande affectera le groupe propriétaire et les autres utilisateurs.

"**wr**" signifie que la modification portera sur les droits d'écriture ou de lecture. (on aurait pu aussi écrire la commande en mettant "rw", l'ordre n'a pas d'importance).

Dernier exemple : je souhaite que le propriétaire du fichier puisse exécuter ce fichier :
[delcros@mistra delcros]$ chmod u+x linux-test

Ainsi le propriétaire du fichier a le droit d'exécuter linux-test (ce qui de toute manière dans ce cas ci ne servira pas à grand chose puisque linux-test n'est ni un binaire ni un script ...)

Si nous souhaitons définir d'un seul mouvement toutes les permissions d'un fichier, on peut utiliser la syntaxe suivante (nous voulons que linux-test soit en lecture, en écriture et en exécution pour le propriétaire, que le groupe n'ait le droit que de le lire et d'écrire et que les autres utilisateurs ne puissent que le lire) :
[delcros@mistra delcros]$ chmod u=rwx,g=rw,o=r linux-test

En une seule ligne grâce au signe "=" nous avons définit l'ensemble des droits. Il existe une autre facon d'indiquer les permissions, nous aurions pu utiliser la syntaxe suivante pour l'exemple précédent :
chmod 764 linux-test

La syntaxe est vraiment très différente ...

En réalité, nous venons d'utiliser la notation binaire pour définir les droits :
Petit rappel :

```
Binaire ------- Logique ------ Décimal
000     -------- (---) ------- 0
001     -------- (--x) ------- 1
010      -------- (-w-) ------- 2
011      -------- (-wx) ------- 3
100      -------- (r--) ------- 4
101      -------- (r-x) ------- 5
110      -------- (rw-) ------- 6
111      -------- (rwx) ------- 7
```

Le 0 indique donc un tiret et le 1 indique que la lettre correspondant à la position doit être inscrite. Donc pour notre exemple, rwx (pour le propriétaire) correspond à 7, rw (pour le groupe correspond à 6, et r (pour les autres utilisateurs) correspond à 4. Nous avons bien la séquence 764. les chiffres doivent être dans l'ordre, le premier pour le propriétaire, le deuxième pour le groupe, le troisième pour les autres utilisateurs.

4.2 Définir le propriétaire et le groupe d'un fichier (chown)

Préambule : cette commande nécessite d'être administrateur système, il vous faut donc vous loguer en root (utiliser la commande "**su**" pour vous loguer en root) :
[delcros@mistra /home]$ su root
Password :

lorsque nous avons effectué un **ls -l** sur le fichier linux-test, nous avons obtenu :
-rw-r-r-- 1 delcros delcros 9 Jul 19 19 : 03 linux-test

Le premier nom "delcros" est le propriétaire du fichier, c'est lui qui peut placer les droits de propriété sur le fichier. Le deuxième nom "delcros" indique le groupe utilisateur du fichier. C'est l'administrateur système qui peut décider des utilisateurs qui feront partie du groupe (dans certains cas, l'administrateur système peut permettre à un utilisateur de déterminer lui même qui fera partie du groupe). Le fichier */etc/group* montre les différents groupes qui existent dans le système).

Je peux décider par exemple que le fichier linux-test n'appartienne plus à l'utilisateur "delcros" mais à l'utilisateur "thomas" :
[root@mistra delcros]# chown thomas.delcros linux-test

Vérifions :
[root@mistra delcros]# ls -l linux-test
-rwxrw-r-- 1 thomas delcros 9 Jul 19 19 : 03 linux-test

Le nouveau propriétaire du fichier est bien thomas.

Une option de **chown** est à connaître :
chown -R (récursif) permet de modifier les permissions de d'un répertoire et de ses sous-répertoires :
Il m'est arrivé par exemple de copier de la documentation qui se trouvait dans un répertoire "doc" dont le propriétaire était l'administrateur système dans le répertoire d'un utilisateur pour qu'il en ait la plus totale disposition.

J'ai donc d'une part copié tout le répertoire et ses sous répertoires dans le répertoire de l'utilisateur grâce à la commande "**cp**" et son option "**-r**" (voir la section consacrée à **cp**) et j'ai donc dû aussi modifier les droits de propriétés de tout ce répertoire et de ses sous répertoires grâce à la commande **chown** et son option **-R** :
[root@mistra delcros]# chown -R delcros.delcros doc

ceci a permis de fixer en une seule fois le propriétaire de plusieurs sous répertoires et de fichiers.

4.3 Ajouter un utilisateur et changer le mot de passe

Utilisez la commande **adduser** pour ajouter un utilisateur :

Je veux par exemple créer un compte utilisateur "ernest" :
[root@mistra /]# adduser ernest

Le compte est créé, c'est-à-dire qu'un répertoire ernest a été créé dans le répertoire

/home et l'utilisateur ernest a été ajouté dans le fichier de configuration /etc/passwd.

Il ne vous reste plus qu'à déterminer un mot de passe pour l'utilisateur ernest à l'aide de la commande **passwd**

[root@mistra /]# passwd ernest

passwd vous demande de rentrer deux fois le même password.

Vous pouvez maintenant quitter la session en cours (commande "exit") puis vous loguer en tant qu'"ernest", ou bien utiliser la commande "**su**" :
[root@mistra /]# su ernest

Ou encore en ouvrant une nouvelle console (linux permet d'ouvrir plusieurs consoles) en utilisant la combinaison de touches suivante :
Alt-F2

pour revenir sur la première console vous devez simplement faire :
Alt-F1

(Sous l'environnement graphique X, on utilisera **Ctrl-Alt-F1**, **Ctrl-Alt-F2**, etc ...)

4.3 Décrire un utilisateur : "chfn"

Cette commande vous permet d'indiquer dans le fichier /etc/passwd différentes informations sur un utilisateur dont son nom, son bureau, ses numeros de téléphone, exemple :

```
[delcros@mistra html]$ chfn
Changing finger information for delcros.
Password :
Name [Armand Delcros] : Armand Delcros
Office [Farniente] : Le Mont Olympe
Office Phone [] : France telecom ?
Home Phone [] : Aie mes factures
```

4.4 Supprimer un utilisateur (userdel)

La suppression d'un compte utilisateur se décompose en deux phases :

1. La suppression de l'utilisateur dans les fichiers de configuration (/etc/passwd, /etc/group ...)
2. La suppression du répertoire et des fichiers de l'utilisateur.

la commande **userdel** permet de faire soit la première étape soit de réaliser les deux d'un coup.

Pour supprimer l'utilisateur ernest des fichiers de configuration du système, utilisez la commande suivante :
[root@mistra /]# userdel ernest

Pour supprimer d'un coup l'utilisateur et son répertoire (ici /home/ernest), utilisez la commande suivante :
[root@mistra /]# userdel -r ernest

4.5 Affichage des dernières lignes ou des premières lignes d'un fichier

La commande **tail** est tout simplement inévitable.
Elle permet d'afficher les dernières lignes d'un fichier. Jusque là on pourrait se dire qu'après tout il suffit d'éditer le fichier et de se déplacer à la fin. D'une part c'est une méthode fastidieuse mais d'autre part, l'option **-f** va définitivement vous convaincre de l'utiliser :
L'option **-f** demande à **tail** de ne pas s'arrêter lorsqu'elle a affiché les dernières lignes du fichier et de continuer à afficher la suite du fichier au fur et à mesure que celui-ci grossit jusqu'à ce que l'utilisateur interrompe la commande avec la combinaison de touches d'interruption **Ctrl-c**.

Les deux grands cas classique de l'utilisation de **tail** avec l'option **-f** est le suivi des fichiers de log */var/log/secure* et */var/log/messages*. Le premier fichier permet de surveiller les connexions que peuvent effectuer d'autres utilisateurs sur votre machine et le deuxième fichier permet de connaître les différents événements qui se produisent sur le système (impression, connexion à l'Internet, tâche de maintenance système...) :

```
[root@mistra /]# tail -f /var/log/messages
Apr 26 14 : 34 : 39 mistra kernel :  PPP line discipline registered.
Apr 26 14 : 34 : 39 mistra kernel :  registered device ppp0
Apr 26 14 : 34 : 40 mistra pppd[26252] :  pppd 2.2.0 started by root, uid 0
Apr 26 14 : 34 : 41 mistra chat[26254] :  send (ATZ^M)
Apr 26 14 : 34 : 41 mistra chat[26254] :  expect (OK)
Apr 26 14 : 34 : 43 mistra chat[26254] :  ATZ^M^M
Apr 26 14 : 34 : 43 mistra chat[26254] :  OK -- got it
```

Ici, on voit le déroulement d'une connexion à l'Internet.

la commande **head** réalise la même chose que tail mais elle affiche les premières lignes du fichier au lieu d'afficher les dernières. **tail** et **head** ont une option commune qui permet d'afficher le nombre de ligne que l'on souhaite :
"tail -5 nom_du_fichier" affichera les 5 dernières lignes du fichier

"head -15 nom_du_fichier" affichera les 15 premières lignes du fichier.

Par défaut, **tail** et **head** affichent 10 lignes.

4.5 Utilisez votre cdrom, votre lecteur de disquette ... etc .. (mount)

La commande **mount** est utilisée par linux dès son démarrage. Elle permet de monter une système de fichier, c'est-à-dire de le rendre accessible. Ce montage est parfois effectué automatiquement grâce au fichier de configuration /etc/fstab. Ce fichier contient tout ce que linux doit monter lors de son démarrage.
Une question souvent posée dans les forums est "comment puis-je lire un CD-ROM ou une disquette". Il faut d'une part créer un point de montage, puis monter le medium et enfin savoir le démonter si on veut pouvoir en mettre un autre.

- **Créer un point de montage**

  Créer un point de montage signifie tout simplement créer un répertoire où l'on pourra à chaque fois qu'on le souhaite regarder le contenu d'un CD-ROM. Le plus souvent ce répertoire est créé dans le répertoire /mnt. Pour ma part je l'ai monté dans la racine et je l'ai appelé tout simplement cdrom :
  [root@mistra /]# mkdir /mnt/cdrom

- **Monter le cdrom :**
  La première chose à connaître est le nom du fichier spécial qui correspond à votre cdrom. Les fichiers spéciaux sont ces fameux fichiers "device" ("dipositif" en francais ...) que l'on trouve dans le répertoire /dev. C'est en quelque sorte des drivers.
  Les lecteurs IDE commencent par les lettres "hd" alors que les lecteurs scsi commence par les lettres "sd". Si vous avez deux lecteurs IDE (un disque dur et un cdrom par exemple), le disque dur s'appelera normalement hda et le cdrom hdb. Si par exemple le disque dur contient 4 partitions, la première s'appelera hda1, la deuxième hda2 , etc ...
  Donc logiquement si vous êtes dans la situation classique où vous possédez un disque dur et un cdrom, la commande suivante vous permettra de monter le cdrom sur le point de montage /mnt/cdrom :
  [root@mistra /]# mount -t iso9660 /dev/hdb /mnt/cdrom

  iso9660 : est le type de formatage du support : pour les cdrom c'est le format "iso9660", pour une disquette MS-DOS, c'est le format "ms-dos", "hpfs" pour une partition OS/2 et pour linux c'est le format "ext2", etc

  - **/dev/hdb** est le "device" du cdrom
  - **/mnt/cdrom** est le point de montage.

  Vous n'avez plus qu'à vous déplacer dans le répertoire /mnt/cdrom et lister le contenu de ce répertoire.

- **Démonter un cdrom : umount**

  Pour changer de CD-ROM, il ne suffit pas d'appuyer sur le bouton eject du lecteur, de changer le CD-ROM et de relister le contenu du point de montage. Il faut d'une part

démonter le CD-ROM en place pour ensuite le remplacer par un autre qui devra lui même être "monté" de la manière qui a été expliquée au point 2. La commande pour démonter le cdrom est :
[root@mistra /]# umount /mnt/cdrom

Ne restez pas dans le répertoire /mnt/cdrom pour le faire, soyez par exemple à la racine.

4.6 Mettre à jour le cache et les liens des bibliothèques (ou comment évitez les "can't load lib..." au démarrage d'un logiciel)

Linux fonctionne maintenant avec un système de bibliothèques dynamiques. Les logiciels utilisant la même bibliothèque pourront accéder tous les deux à la même copie placée en mémoire, ce qui permet un gain de mémoire important.

Il vous est peut-être déjà arrivé d'avoir un problème au lancement d'un logiciel avec un message d'erreur qui peut revêtir cette forme :
"can't load libXpm.so.4.7"

Il vous faudra donc récupérer et installer la bibliothèque manquante sur votre système. Mais une fois installée, la bibliothèque devra être signalée au système. La commande **ldconfig** permettra de mettre à jour les liens symboliques des bibliothèques et des caches.
Vous pourrez enfin lancez votre application normalement.

[root@mistra /etc]# ldconfig -v

4.7 Arrêter le système : la commande shutdown

Je ne vous montrerai que les deux options que j'utilise sous linux :
[root@mistra /root]# shutdown -r now
Cette commande vous permet de rebooter l'ordinateur.

[root@mistra /root]# shutdown -h now
Cette commande vous permet d'arrêter complètement le système. Vous pouvez éteindre l'ordinateur lorsque vous verrez affiché :
"System halted
The system is halted"

4.8 Voilà, c'est fini mais comment puis-je en savoir plus sur les commandes ?

La commande "**man**" est là pour vous aider. Toutes les commandes possèdent une "page de manuel" qui vous est livrée avec linux :
[delcros@mistra delcros]$ man cp

Et vous obtiendrez toute la documentation de **cp**.
Pour quitter la page de manuel, vous pouvez appuyer à n'importe quel moment sur la touche "**q**".

5. Multitâches

5.1 Généralités

En ligne de commande, vous pouvez lancer un logiciel en tâche de fond en rajoutant `&` après son invocation:

```
toto &
```

Si un processus tourne déjà en avant-plan, vous pouvez le suspendre temporairement en pressant les touches Ctrl+Z. Une fois le processus suspendu, vous pouvez le ramener en avant-plan en tapant `fg`, ou le forcer à continuer son exécution en arrière-plan en tapant `bg`.

Pour voir la liste des processus actuellement en cours (en arrière-plan ou suspendus), utilisez la commande `jobs`. Chaque processus est listé avec un numéro:

```
arnaud@pumpkin:~ > jobs
[1]- Stopped                 find / -name xemacs
[2]+ Stopped                 cat
```

Pour reprendre le processus numéro *n*, utilisez `fg %n` (pour le remettre en avant plan) ou `bg %n`. (pour le mettre en arrière-plan), Si vous omettez l'argument, c'est le processus marqué d'un + dans la liste qui reprend. Pour terminer le processus numéro *n*, utilisez `kill %n ou killall nom du processus.` ATTENTION, cette dernière commande supprime tous les processus ayant le même nom.

Bonus: Priorité d'un processus

Chaque processus se voit affecter une **priorité** qui correspond à un numéro. Lorsqu'une commande est lancée, le processus a une priorité maximale. Plus le processus occupe de temps d'éxecution pour le processeur, plus son numéro de priorité baisse, et moins le processus occupe de temps d'éxecution pour le processeur, plus son numéro de priorité augmente. Ainsi, plusieurs processus peuvent être éxecutés en même temps. La commande `nice` permet de diminuer la priorité du processus (pour les commandes longues et peu urgentes, par exemple). Le paramétre spécifié après l'option *-n* est un nombre compris entre 0 et 20 qui indique le facteur de diminution :

```
nice -n 20 find / -type f -name "install*" -print
> liste 2> /dev/null &
```

L'administarteur système (compte **root**) peut également augmenter la priorité avec un nombre négatif :

```
nice -n -20 find / -type f -name "install*" -print
> liste 2> /dev/null &
```

et la commande `renice` permet de changer le facteur de priorité en cours d'exécution de la commande, en spécifiant le nouveau facteur de priorité et le numéro de processus :

```
renice 10 733
```

Le résultat informe alors le super utilisateur du changement :

```
733: old priority 19, new priority 10
```

5.2 Gestion des services

Comment ajouter et supprimer des services (daemons, etc.) au démarrage.

5.2.1 Définitions

Les daemons (ou démons) sont des programmes résidents chargés au démarrage. A chaque runlevel, correspond une liste de daemons à lancer (1 à 5) ou à arrêter (6 ou 0).

D'autres programmes que des démons peuvent également être lancés dès le démarrage de la machine, avec le même mécanisme.

Les runlevels ou "niveaux d'exécution", correspondent aux services qui vont être lancés au démarrage de la machine. En général (mais toutes les distributions n'utilisent pas la même numérotation), on peut avoir les niveaux d'exécution suivant :

- 0 : arrête la machine.
- **1 : mode simple utilisateur (ou single, ou encore failsafe). Ce mode est utile pour dépanner une machine plantée.**
- 3 : mode console. Les services sont lancés, mais le serveur X n'est pas activé (ainsi que les services dont il dépend).
- 4 (Slackware) ou 5 (Mandrake, RedHat..). : mode graphique : le serveur X et les services dont ils dépend sont lancés.
- 6 : redémarre la machine.

Le niveau d'exécution est déterminé (dans l'ordre) soit :

- lors du boot : si vous précisez un niveau sur la ligne de commande du noyau (par exemple, au prompt LILO, taper "linux 1"),
- dans le fichier `/etc/inittab`, où le runlevel par défaut est défini,
- par la commande `init <runlevel>` qui permet de changer de runlevel en cours de fonctionnement.

Remarque : Le fonctionnement des services sur une gentoo, bien qu'assez similaire, est différent.

5.2.2 Fonctionnement

Les services sont lancés par des scripts situés dans `/etc/init.d` (ou dans `/etc/rc.d/init.d`, (qui sont souvent le même fichier, l'un étant un lien vers l'autre). Chaque script contient une description ce qui permet de savoir ce que fait chaque daemon en début de script.

Le répertoire `/etc/rc.d/` contient aussi des répertoires nommés `rcX.d` (avec X numéro de runlevel). Chacun de ces répertoires contient un lien vers les scripts situés dans init.d.

Exemple:

$ ls -l /etc/rc.d/rc5.d
rwxrwxrwx 1 root root 13 Jun 16 23:23 K30usb -> /etc/rc.d/init.d/usb*
lrwxrwxrwx 1 root root 16 Jun 17 00:03 S30syslog -> /etc/rc.d/init.d/syslog*
lrwxrwxrwx 1 root root 18 Jun 17 00:05 S75keytable -> /etc/rc.d/init.d/keytable*

La 1ère lettre détermine si le daemon est activé (S comme start) dans ce niveau d'exécution (runlevel) ou arrêté (K comme kill).
Les 2 chiffres permettent de trier l'ordre d'exécution des services (dans l'exemple, `syslog` est démarré avant `keytable`).

Remarque : sur une gentoo, les répertoires correspondant à un runlevel particulier sont stockés dans

`/etc/runlevels/`. La gestion des runlevels par gentoo est complétement différente : il n'est pas nécessaire de se préoccuper de l'ordre de démarrage des services car chaque service précise quels sont les services qui lui sont nécessaires, le reste (dépendances...) est géré automatiquement. De plus les runlevels ont des noms plutôt que des numéros. Par default, une Gentoo arrive avec les runlevels : `boot`, `default`, `nonetwork`, `single`.

5.2.3 Commandes utiles

Plutôt que de modifier directement les liens, on va utiliser la commande suivante : `/etc/init.d/nom_service {start|stop|restart|reload|status}` ou `service nom_service {start|stop|restart|reload|status}`. À ce propos, je recommande d'utiliser "bash-completion" qui permet d'améliorer la complétion automatique de la ligne de commande, et par exemple de lister tous les services disponibles, en tapant "service" puis la touche [Tab].

Exemple:

/etc/rc.d/init.d/linuxconf start
Note: les options `start/stop/restart` lancent/arrêtent/redémarrent le service spécifié pour tous les niveaux.
Note (Mandrake, RedHat...) : on peut utiliser la commande `service` pour démarrer un service particulier,
$ service nomduservice start

ou :

$ service nomduservice stop

pour l'arrêter.

chkconfig

pour Mandrake et RedHat

La commande `chkconfig` est un peu plus puissante.

Pour ajouter un daemon dans tous les niveaux de demarrage:
`/sbin/chkconfig --add le_service`

Note: le daemon doit obligatoirement se trouver dans `/etc/rc.d/init.d` ou `/etc/init.d`.

Pour lister tous les daemons avec leurs status:

/sbin/chkconfig --list
atd   0:off 1:off 2:off 3:on   4:on   5:on   6:off
xfs   0:on   1:on   2:on   3:on   4:on   5:on   6:on
keytable 0:off 1:off 2:on   3:on   4:on   5:on   6:off
gpm   0:off 1:off 2:on   3:on   4:on   5:on   6:off

Autre option:
`/sbin/chkconfig --list le_service`
pour ne lister que celui souhaité.

Pour activer ou désactiver un service au démarrage :
`/sbin/chkconfig --level 123456 mon_service on/off`
(avec 123456 le(s) runlevel(s) pour le(s)quel(s) le service doit être ou non activé).

Pour plus de détails, voir la man page.

C'est bien beau, mais si je dois me tapper tout ça à la mimine ?! Stop ! Linux a pensé à nous, et pour se simplifier la tache, on a plusieurs outils : Linuxconf via Panneau de configuration/gestion des services (qui stoppe ou arrête un daemon pour tous les runlevels), Runleveleditor (qui permet de choisir pour chaque runlevel les daemons à activer ou non), ksysv, etc. Fais ton choix camarade ;-)

5.2.4 Ajouter ses propres services

Nous allons détailler les étapes nécessaires pour configurer un nouveau service et l'ajouter à la base de `chkconfig`. Nous allons prendre l'exemple d'un service nommé `bidule`, qui devra être démarré aux runlevels 3, 4 et 5.

1 - Ecrire le script de configuration du démarrage

Pour écrire le script `/etc/rc.d/init.d/biduled`, on peut s'inspirer d'un script existant dans ce répertoire. Ci-dessous un exemple de ce script que nous allons détailler

```
root@pingu# cat /etc/rc.d/init.d/biduled
#!/bin/sh
# description: exemple de script pour Léa
# chkconfig: 345 99 0
```

Attention la syntaxe des 2 dernières lignes est à respecter à la lettre pour que le service puisse être ajouté correctement par la commande `chkconfig`. Nous devons spécifier 2 mots clé :

- `description` : décrire le service en quelques mots. Si votre description utilise plus d'une ligne, utiliser un "\" pour respecter la continuité.
- `chkconfig` : cette ligne contient 3 informations. Ici `345` indique les runlevels auxquels sera démarré le service, `99` indique le numéro de séquence des liens S, `0` indique le numéro de séquence des liens K.

```
#source function library
. /etc/init.d/functions
```

On charge ici en mémoire les fonctions définies dans le fichier `/etc/init.d/functions`. Nous allons en effet utiliser certaines d'entre elles comme `daemon`, `killproc`, `status`.

```
case $1 in

'start')
[ -f /var/lock/subsys/biduled ] &&
exit 0
echo -n "exécute bidule"
daemon /usr/bin/biduled
echo
touch /var/lock/subsys/biduled
 ;;
```

On démarre ici le traitement des arguments de gestion du service. Pour le **démarrage**, on vérifie d'abord qu'il n'existe pas de fichier de lock, ce qui permet d'éviter de démarrer 2 fois le même service. S'il existe alors on sort du script. Dans le cas contraire, on affiche un message sur la console indiquant le démarrage de `bidule` et c'est la fonction `daemon` qui se charge de lancer le service. Enfin au moment du lancement, on crée le fichier de lock.

```
'stop')
echo -n "arrête bidule"
```

```
killproc biduled
echo
rm -f /var/lock/subsys/biduled
 ;;
```

En ce qui concerne l'**arrêt** du service, on utilise la fonction `killproc` qui récupère le numéro de processus de notre service et le tue avec la commande `kill`. On efface également le fichier de lock afin de permettre éventuellement le redémarrage du service.

```
'restart')
$0 stop
$0 start
 ;;
```

Le **redémarrage** du service consiste simplement à utiliser ce même script pour l'arrêter puis le démarrer

```
'status')
status biduled
 ;;
```

Cet argument permet de vérifier si le service est lancé, au moyen de la fonction `status`.

```
*)
echo "Usage : biduled \
{start|stop|restart|status}"
exit 1
 ;;
esac
exit 0
```

Ce cas de figure traite le cas où l'utilisateur ne rentre pas le bon argument. On lui renvoie donc un message d'erreur lui indiquant ceux qui doivent être utilisés.

Ouf ! Nous avons terminé le script. Il ne reste plus qu'à **le rendre exécutable** :

```
root@pingu# chmod +x /etc/rc.d/init.d/biduled
```

2 - Ajouter le service à la base de chkconfig

Le plus gros du travail est fait ! Il ne reste plus qu'à ajouter le service `biduled` à la base des services gérée par `chkconfig`. Pour ce faire, rien de plus simple :

```
root@pingu# chkconfig --add biduled
```

Le service fait maintenant partie de la base, les liens symboliques de démarrage ont été créés automatiquement de la manière suivante :

```
# ll /etc/rc.d/rc?.d/*biduled lrwxrwxrwx 1 root root 17 oct 11 09:37 rc0.d/K00biduled ->
/etc/rc.d/init.d/biduled
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc1.d/K00biduled -> /etc/rc.d/init.d/biduled
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc2.d/K00biduled -> /etc/rc.d/init.d/biduled
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc3.d/S99biduled -> /etc/rc.d/init.d/biduled
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc4.d/S99biduled -> /etc/rc.d/init.d/biduled
lrwxrwxrwx 1 root root 17 oct 11 09:32 rc5.d/S99biduled -> /etc/rc.d/init.d/biduled
lrwxrwxrwx 1 root root 17 oct 11 09:37 rc6.d/K00biduled -> /etc/rc.d/init.d/biduled
```

Si vous n'avez pas chkconfig ou désirez faire cette opération manuellement, il vous suffit de créer les liens symboliques listés ci-dessus.

Nous pouvons également vérifier de la manière suivante :

root@pingu# **chkconfig --list** | grep biduled
biduled 0:Arrêt 1:Arrêt 2:Arrêt 3:Marche 4:Marche 5:Marche 6:Arrêt

5.2.5 Faire du ménage

Dans mon cas et à titre d'exemple (internet, pas d'imprimante, travail sous X, son) je ne garde en activité que : `syslog`, `xfs`, `keytable`.

Pour information, voici une liste (non exhaustive) de quelques daemons et de leur fonctions.

`apmd`: nécessaire uniquement pour les ordinateurs portables
`xntpd`: Network time protocol
`portmap`: nécessaire si vous utilisez un service rpc, comme NIS ou NFS
`sound`: configuration des sons (ma carte fonctionne très bien sans ??? ndlr:normal si le fichier `/etc/modules.conf` est bien conçu)
`netfs`: c'est le client `nfs`, utilisé pour mounter des filesystems depuis un serveur `nfs`
`rstatd` , `rusersd`, `rwhod`, `rwalld`: ne pas exécuter tous les services car ils fournissent trop d'informations aux utilisateurs à distance
`bootparamd`: Utilisé par les clients sans lecteur de disquette (vulnérable)
`squid`: serveur proxy
`yppasswdd`: nécessaire si vous êtes un serveur NIS (extrêmement vulnérable)
`ypserv`: nécessaire si vous êtes un serveur NIS (extrêmement vulnérable)
`dhcpd`: démarre le daemon du serveur dhcp
`atd`: utilisé pour le service at, similaire à cron, mais n'est pas nécessaire
`pcmcia`: parle de lui-même
`snmpd`: daemon SNMP, peut donner à des utilisateurs distants des informations détaillées sur votre système
`named`: serveur DNS
`routed`: RIP, n'exécutez cela que si vous en avez vraiment besoin
`lpd` : services d'impression
`mars-nwe`: fichier Netware et serveur d'impression
`nfs`: Utilisé pour le serveur NFS, lancez le que si vous en avez absolument besoin
`amd`: daemon AutoMount, sert à mounter les filesystems distants
`gated`: sert à lancer d'autres protocoles de routage comme OSPF
`sendmail`: Vous pourrez toujours envoyer/recevoir des emails par Netscape (ou autre) sans lui.
`httpd`: serveur web Apache
`ypbind`: nécessaire si vous êtes un client NIS
`xfs`: xfont server (indispensable si vous êtes sous X).
`innd`: serveur de news
`arpwatch`: off par défaut. Rapport d'activité de datagrammes IP via mail
`kudzu`: détection des periphériques. A réactiver à l'occasion
`anacron`: reprise de jobs de la crontab après un crash
`crond`: si vous ne savez pas ce qu'est une crontab, désactivez-le.
`rawdevices`: partitions spécifique sous ORACLE ou autre SGBD
`random`: améliore la génération aléatoire de nombres (peut être utile pour les joueurs)
`rhnd`: redhat network

`linuxconf`: j'utilise linuxconf sans ce daemon (peut être est-ce utile pour l'administration à distance ?)
`nfslock`: si vous n'êtes pas serveur NFS, désactivez-le
`usb`: parle de lui-même
`gpm`: fournit des fonctions pour le support de la souris en mode texte (utile pour midnight commander en particulier)

Ouf, c'est fini...

5.3 Screen – rendre un processus libre

5.3.1. Le concept

Le problème à résoudre est le suivant : vous avez lancé sur votre système un programme qui fonctionne en mode console (un client IRC par exemple). Vous voulez vous délogguer en laissant tourner le programme... et pouvoir vous relogguer plus tard (en local ou à distance) et récupérer le programme à l'écran.

Pour cela, il faut lancer le programme dans un **screen**, qui est une sorte écran virtuel que l'on peut détacher et rattacher :

1. vous ouvrez un screen,
2. vous lancez le programme dedans,
3. si vous voulez vous délogguer et laisser tourner le programme, vous détachez le screen,
4. vous pouvez rattacher le screen et donc retrouver le programme depuis n'importe quelle connexion au système (console locale ou accès distant).

5.3.2. Installer et utiliser

5.3.2.1. Installer le package

```
# apt-get install screen
```

Nous allons prendre l'exemple de 2 scénarios pour expliquer comment ça fonctionne :

5.3.2.2. 1er scénario

1. Depuis un premier ordinateur, ouvrez un screen depuis une console locale en lui donnant un nom. Pour cela, tapez la commande :

   ```
   % screen -S nom_du_screen
   ```

   Un message d'explication apparaît : appuyez sur **Entrée** pour le zapper. Vous avez alors un prompt normal à l'intérieur du screen. Lancez une application qui tourne en mode console (un client IRC par exemple).
2. Quittez le premier ordinateur en laissant le programme tourner et votre console ouverte (on dit que le screen reste attaché). Vous ouvrez une console sur un deuxième ordinateur et vous vous connectez au premier ordinateur (par une connexion SSH par exemple). Pour rattacher le screen, c'est-à-dire retrouver à l'écran le programme que vous avez lançé dans le screen sur le premier ordinateur, tapez la commande :

```
% screen -x nom_du_screen
```

Si vous ne vous souvenez plus du nom que vous aviez donné à votre screen, lancer simplement la commande **screen -x** et vous verrez la liste des screens avec leurs noms associés.

3. Si la taille de votre console sur le deuxième ordinateur n'est pas la même que la taille de votre console sur le premier ordinateur, ce qui se traduit par un programme qui occupe plus ou moins de place que la taille de l'écran : utilisez alors la combinaison de touches **Ctrl-a** puis **F**, ce qui a pour effet de redimensionner le programme à la taille de votre nouvelle console.
4. Vous voulez quitter le deuxième ordinateur : détachez le screen par la combinaison de touches **Ctrl-a** puis **d**. Le message suivant apparaît sur la console :

```
[detached]
```

et vous pouvez vous délogguer du deuxième ordinateur.

5. Vous revenez sur le premier ordinateur et vous retrouvez votre console avec le programme qui tourne à l'intérieur. Si vous avez redimensionné le programme sur le deuxième ordinateur, vous devrez le redimensionner à nouveau avec la même combinaison de touches pour le remettre aux dimensions de votre console initiale.
6. Vous voulez quitter le programme qui ne vous sert plus : quittez le programme normalement puis fermez le screen en faisant comme si vous vous délogguiez (combinaison de touches **Ctrl-d** ou commande **logout**). Le message suivant s'affiche sur la console :

```
[screen is terminating]
```

5.3.2.3. 2ème scénario

1. Depuis un premier ordinateur, vous vous connectez à distance sur un deuxième ordinateur. Vous voulez lancer un programme sur ce deuxième ordinateur et pouvoir le récupérer quand vous voulez et depuis n'importe quel ordinateur. Pour cela, lancez le programme dans un screen : pour faire d'une pierre deux coups, c'est à dire ouvrir le screen et lancer le programme en même temps, tapez :

```
% screen -S nom_du_screen commande_qui_lance_le_programme
```

2. Vous voulez vous délogguer du premier ordinateur : détachez le screen avec la combinaison de touches **Ctrl-a** puis **d**, déconnectez-vous du deuxième ordinateur puis délogguez-vous du premier ordinateur.
3. Vous voulez retrouver le programme que vous aviez lançé dans le screen : logguez-vous en local sur le deuxième ordinateur ou connectez-vous sur le deuxième ordinateur à distance depuis un autre ordinateur et tapez la commande suivante pour rattacher le screen que vous aviez détaché :

```
% screen -r nom_du_screen
```

Si vous ne vous souvenez plus du nom que vous aviez donné à votre screen, lancez simplement la commande **screen -r** et vous verrez la liste des screens avec leurs noms associés.

Vous aurez peut-être besoin de redimensionner le programme avec la combinaison de touches **Ctrl-a** puis **F**.

4. Vous voulez quitter le programme qui ne vous sert plus : quittez le programme normalement et le screen se fermera tout seul car vous aviez ouvert le screen et lançé le programme en même temps. Le message suivant s'affiche sur la console :

```
[screen is terminating]
```

5.3.2.4. Plusieurs fenêtres dans un screen

A l'intérieur d'un screen, vous pouvez avoir une deuxième fenêtre avec un nouveau shell à l'intérieur. Pour cela, utilisez la combinaison de touches **Ctrl-a** puis **c**. Vous pouvez en ouvrir autant que vous voulez en répétant cette combinaison de touches. Vous pouvez ensuite passer d'une fenêtre à la suivante par la combinaison de touches **Ctrl-a** puis **n** (*n* comme *Next*) et passer à la fenêtre précédente par la combinaison de touches **Ctrl-a** puis **p** (*p* comme *Previous*).

Pour fermer une fenêtre, il suffit de fermer le shell qu'elle contient (combinaison de touches **Ctrl-d** ou commande **logout**). Le fait de fermer la dernière fenêtre restante provoque la fermeture du screen.

6. Les logs

Beaucoup de logiciels laissent une trace de leurs activités dans les logs du système. Certains logiciels, comme le serveur web Apache ou les logiciels de communication Taylor UUCP, gèrent eux-même leurs logs. La plupart, cependant, utilisent le serveur `syslogd`, et le noyau utilise son frère `klogd`.

6.1 Où se trouvent les logs?

Cela dépend de la distribution utilisée. Sous Red Hat, ils se trouvent par défaut dans le répertoire `/var/log`.

6.2 Configurer `syslogd`

Le fichier de configuration de `syslogd` est `/etc/syslog.conf`. La configuration par défaut est satisfaisante; toutefois, j'aime bien afficher tous les logs sur une console virtuelle inutilisée. Pour cela, j'ai rajouté la ligne suivante dans mon `syslog.conf`:

```
*.*                                                     /dev/tty12
```

Attention, il faut utiliser des tabulations, et pas des espaces. Une fois la modification effectuée, il faut relancer `syslogd` et `klogd`:

```
/etc/rc.d/init.d/syslog restart
```

Ensuite, pour voir les logs, il suffit de faire Alt-F12, ou Ctrl-Alt-F12 depuis X.

7. Automatiser des tâches

7.1 Tâches périodiques: `cron` et la *crontab*

Pour exécuter un programme périodiquement, il faut indiquer la marche à suivre à `cron` au moyen d'une *crontab*. Si le programme doit afficher quelque chose à l'écran, `cron` vous envoie le résultat par courrier électronique.

Pour éditer votre *crontab*, utilisez la commande `crontab -e`. La *crontab* se compose d'une série de lignes de texte. Chaque ligne décrit une tâche à effectuer et le moment auquel elle doit l'être.

Une ligne se compose de six champs, séparés par des blancs (espaces ou tabulations). Le sixième champ se compose de tout ce qui suit le cinquième blanc; il peut lui-même comporter des blancs. Les champs sont, dans l'ordre:

1. minute: entre 0 et 59
2. heure: entre 0 et 23
3. jour du mois: entre 0 et 31
4. mois: entre 0 et 12
5. jour de la semaine: 0=dimanche, 1=lundi... 7=dimanche. Oui, 0 et 7 ont la même signification.
6. instruction à exécuter.

Les 5 premiers champs peuvent contenir une étoile, pour ignorer le champ, ou plusieurs valeurs séparées par des virgules.

Quelques exemples:

- Pour avoir les statistiques de mes filtres de courrier électronique, tous les lundi à 21h18 :

```
18 21 * * 1 mailstat $HOME/data/maillog
```

- Pour poster la FAQ du groupe `mygale.linux`, le premier et le 16 de chaque mois, à 20h20 :

```
20 20 1,16 * * $HOME/bin/nsotpp.sh
$HOME/.nsotpp/mygale.linux.faq mygale.linux
```

- Pour lancer `atrun` toutes les minutes:

```
* * * * * atrun
```

7.2 Un `cron` pour les machines personnelles: `ucrond`

Vixie cron, le `cron` par defaut de la Red Hat et de la plupart des autres distributions, présente un gros défaut: une tâche programmée n'est effectuée que si la machine est allumée au bon moment. Si ça ne pose guère de problemes pour les serveurs et les autres machines allumées en permanence, c'est beaucoup plus gênant dans le cas d'un PC personnel.

Pour résoudre ce problème, vous pouvez remplacer *Vixie cron* par `ucrond`. Vous pouvez en télécharger un package RPM par exemple à l'adresse http://www.chez.com/gomesdv/linux/RPMS/.

Une fois `ucrond` installé, il vous suffit de rajouter l'option `-D` avant une commande quelconque dans votre *crontab* pour que la tâche correspondante soit effectuée au prochain démarrage de la machine si elle n'a pas pu l'être au moment prévu.

Par exemple, la ligne de ma *crontab* pour poster la FAQ de `mygale.linux` devient:

```
20 20 1,16 * * -D $HOME/bin/nsotpp.sh $HOME/.nsotpp/mygale.linux.faq
mygale.linux
```

7.3 Tâches ponctuelles: `at`

Pour programmer l'exécution ponctuelle d'un programme, tapez `at` *`date`*. Différents formats sont possibles pour la date; reportez-vous à la page de manuel de `at` pour les connaître.

Ensuite, au prompt `at>`, entrez les instructions à exécuter. Vous pouvez utiliser plusieurs lignes. Tapez Ctrl-d quand vous avez terminé.

Quelques exemples:

- Pour souhaiter la bonne année aux aminautes:

```
arnaud@pumpkin:~ > at 01.01.99 00:00
at>echo "Bonne annee!" | mail -s "Joyeuses paques! :-)"
aminautes@concar.net
at>
```

- Pour ne pas oublier la pause-déjeûner:

```
arnaud@pumpkin:~ > at noon
at>cat $HOME/data/reveil.wav > /dev/dsp
at>
```

  où `reveil.wav` est un fichier son bien choisi `:-)`

La commande `atq` permet de voir la liste des tâchez programmées. `at -c` *`numéro_de_la_tâche`* affiche à l'écran une tâche programmée, et `atrm` *`numéro`* supprime une tâche non encore exécutée.

8 APT

8.1 Introduction

Au commencement, il y avait le .tar.gz. Les utilisateurs devaient compiler chaque programme qu'ils voulaient utiliser sur leur système GNU/Linux. Quand Debian fut créée, il a été estimé nécessaire que le système s'occupe aussi de la gestion des paquets installés sur la machine. Le nom `dpkg` fut donné à ce système. Ainsi, ce fut la première fois qu'un « système de paquets » était inclus dans GNU/Linux, longtemps avant que Red Hat ne décide de créer son système de « rpm ».

Un nouveau dilemme est rapidement survenu dans l'esprit des développeurs GNU/Linux. Ils avaient besoin d'une solution rapide, pratique et efficace pour installer les paquets qui gérerait automatiquement les dépendances et qui prendrait en compte les fichiers de configuration des paquets lors de mises à niveau. Ici encore, Debian a ouvert la voie et a donné naissance à `apt`, *Advanced Packaging Tool*, qui depuis a été porté par Conectiva pour l'utiliser avec les rpm et a été adopté par quelques autres distributions.

8.2 Configuration de base

8.2.1 Le fichier /etc/apt/sources.list

Pour réaliser ses opérations, `apt` utilise un fichier qui liste les « sources » d'où peuvent provenir les paquets. Ce fichier est `/etc/apt/sources.list`.

Les entrées dans ce fichier suivent ce format :

```
deb http://host/debian distribution section1 section2 section3
deb-src http://host/debian distribution section1 section2 section3
```

Bien sûr, les entrées ci-dessus sont fictives et ne doivent pas être utilisées. Le premier mot de chaque ligne, `deb` ou `deb-src`, indique le type de l'archive : `deb` pour les paquets binaires, ce sont les paquets pré-compilés que l'on utilise habituellement, ou `deb-src` pour les paquets sources, qui sont les sources originales du programme, plus les fichiers de contrôle Debian (`.dsc`) et le `diff.gz` contenant les changements nécessaires pour « debianiser » le programme.

Nous trouvons généralement les lignes suivantes dans le `sources.list` Debian par défaut :

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-
free
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib
non-free
```

Ce sont les lignes nécessaires à une installation Debian de base. La première ligne `deb` pointe vers l'archive officielle, la seconde vers l'archive non-US et la troisième vers l'archive des mises à jour de sécurité Debian.

Les deux dernières lignes sont commentées (avec un « # » devant), `apt-get` les ignorera. Ce sont les lignes `deb-src`, elles pointent vers les paquets source Debian. Si vous téléchargez souvent les sources de programmes pour tester ou recompiler, décommentez-les.

Le fichier `/etc/apt/sources.list` peut contenir plusieurs types de lignes. `apt` sait traiter les archives de type `http`, `ftp`, `file` (les fichiers locaux, c'est-à-dire, un répertoire contenant une image ISO9660 montée) et `ssh`.

8.2.2 Comment utiliser `apt` localement ?

Parfois vous avez beaucoup de paquets .deb et vous aimeriez utiliser `apt` pour les installer afin que les dépendances soient automatiquement résolues.

Pour ce faire, créez un répertoire et ajoutez-y vos fichiers .deb. Par exemple :

```
# mkdir /root/debs
```

Vous pouvez modifier directement l'ensemble des définitions dans le fichier de contrôle du paquet en utilisant un fichier `override` dans votre référentiel. À l'intérieur de ce fichier, vous pouvez vouloir définir certaines options pour modifier celles incluses dans le paquet original. Un tel fichier ressemble à ce qui suit :

```
paquet priorité section
```

paquet est le nom du paquet, priorité est faible, moyenne ou haute, et section est la section à laquelle il appartient. Le nom du fichier importe peu, vous devrez le passer après en argument à `dpkg-scanpackages`. Si vous ne souhaitez pas écrire de fichier `override`, utilisez simplement `/dev/null` lorsque vous appelez `dpkg-scanpackages`.

Toujours dans le répertoire /root, faites :

```
# dpkg-scanpackages debs fichier | gzip > debs/Packages.gz
```

Dans la ligne ci-dessus *fichier* est le fichier « `override` », la commande produit un fichier `Packages.gz` qui contient diverses informations sur les paquets utilisées par `apt`. Pour utiliser les paquets, ajoutez enfin :

```
deb file:/root debs/
```

Ensuite, utilisez `apt` normalement. Vous pourriez aussi fabriquer un dépôt de sources. Pour ce faire utilisez la même procédure, mais rappelez-vous que vous avez besoin des fichiers `.orig.tar.gz`,`.dsc` et `.diff.gz` dans le répertoire. De plus, vous devez utiliser `Sources.gz` au lieu de `Packages.gz`. Le programme utilisé est aussi différent. C'est `dpkg-scansources`. La ligne de commande ressemblera à :

```
# dpkg-scansources debs | gzip > debs/Sources.gz
```

Remarquez que `dpkg-scansources` n'a pas besoin d'un fichier `override`. La ligne correspondante du sources.list est :

```
deb-src file:/root debs/
```

8.2.3 Comment décider quel est le meilleur miroir pour le fichier sources.list : netselect, netselect-apt ?

Un doute très fréquent, principalement parmi les nouveaux utilisateurs est : « Quel miroir Debian à mettre dans le`sources.list` ? ». Il y a plusieurs façons de décider. Les experts ont probablement un script qui mesure le temps de réponse des principaux miroirs. Mais il y a un programme qui fait ça pour nous : **netselect**.

Pour installer netselect, faites comme d'habitude :

```
# apt-get install netselect
```

En lançant `netselect` sans paramètre, l'aide est affichée. En le lançant avec une liste de serveurs (miroirs) séparés par un espace, il retournera un résultat et un des serveurs. Ce résultat prend en considération l'estimation du temps de réponse et le nombre de sauts (les serveurs par lesquels une requête réseau passera pour atteindre la destination) et il est inversement proportionnel à la vitesse de téléchargement (donc le plus bas est le meilleur). Le serveur retourné est celui qui a eu le plus petit score (la liste complète des résultats peut être vue en ajoutant l'option -vv). Regardez cet exemple :

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org
download.unesp.br ftp.debian.org.br
  365 ftp.debian.org.br
#
```

Cela signifie que, d'après les miroirs donnés en paramètre à `netselect`, `ftp.debian.org.br` était le meilleur, avec un résultat de 365. (Attention ! Comme cela a été fait de mon ordinateur et que la topographie du réseau est très différente selon le point de contact, cette valeur n'est pas forcément la bonne vitesse sur d'autres ordinateurs).

Maintenant, mettez seulement le miroir le plus rapide trouvé par `netselect` dans le fichier `/etc/apt/sources.list` (voir Le fichier /etc/apt/sources.list, Section 8.2.1) et suivez les conseils dans Gestion des paquets, 8.3

Note : on peut toujours trouver la liste des miroirs sur `http://www.debian.org/mirror/mirrors_full`.

À partir de la version 0.3.ds1, le paquet source `netselect` inclut le paquet binaire **netselect-apt**, qui effectue la procédure ci-dessus automatiquement. Entrez seulement la distribution en paramètre (celle par défaut est `stable`) et le fichier `sources.list` sera créé avec les meilleurs miroirs pour main et non-US et sera sauvegardé dans le répertoire courant. L'exemple suivant crée un sources.list de la distribution `stable` :

```
# ls sources.list
ls: sources.list: File or directory not found
# netselect-apt stable
(...)
# ls -l sources.list
sources.list
#
```

Rappelez-vous : Le fichier `sources.list` est créé dans le répertoire courant et doit être déplacé dans le répertoire `/etc/apt`.

Ensuite, suivez les conseils dans Gestion des paquets, 8.3

8.2.4 Ajouter un cédérom dans le fichier sources.list

Si vous voulez plutôt utiliser vos cédéroms pour installer vos paquets ou mettre à jour votre système automatiquement avec `apt`, vous pouvez les mettre dans votre `sources.list`. Pour le faire, vous pouvez utiliser le programme `apt-cdrom` comme ceci :

```
# apt-cdrom add
```

avec le cédérom Debian dans le lecteur. Il montera le cédérom, et si c'est un cédérom Debian valide, il regardera les informations des paquets sur le disque. Si la configuration de votre cédérom est inhabituelle, vous pouvez aussi utiliser les options suivantes :

```
-h           - Aide du programme
-d directory - Point de montage du cédérom
-r           - Renommer un cédérom reconnu
-m           - Pas de montage
-f           - Mode rapide, ne vérifie pas les fichiers paquet
-a           - Mode de vérification minutieux
```

Par exemple :

```
# apt-cdrom -d /home/kov/moncdrom add
```

Vous pouvez aussi identifier un cédérom sans l'ajouter à votre sources.list :

```
# apt-cdrom ident
```

Remarquez que ce programme fonctionne seulement si votre cédérom est correctement configuré dans votre `/etc/fstab`.

8.3 Gestion des paquets

8.3.1 Mise à jour de la liste des paquets disponibles.

Le système de paquets utilise sa propre base de données pour garder une trace des paquets qui sont installés, de ceux qui ne sont pas installés et de ceux qui peuvent être installés. Le programme `apt-get` utilise cette base de données pour retrouver comment installer les paquets demandés par l'utilisateur ainsi que pour retrouver les paquets supplémentaires nécessaires afin qu'un paquet sélectionné fonctionne correctement.

Pour mettre à jour cette liste, vous pouvez utiliser la commande `apt-get update`. Cette commande vérifie la liste des paquets trouvés dans les archives dans `/etc/apt/sources.list` : voir Le fichier /etc/apt/sources.list, Section 8.2.1 pour plus d'informations sur ce fichier.

C'est une bonne idée d'exécuter cette commande de temps en temps pour vous garder, vous et votre système, informés des possibilités de mise à jour des paquets, particulièrement les mises à jour de sécurité.

8.3.2 Installation de paquets

Finalement, la procédure que vous attendiez tous ! Avec votre sources.list prêt et votre liste de paquets disponibles à jour, tout ce que vous avez à faire est de lancer `apt-get` pour obtenir le paquet que vous désirez installer. Par exemple, vous pouvez lancer :

```
apt-get install xchat
```

`apt` cherchera dans sa base de données la version la plus récente de ce paquet et le récupérera à

partir de l'archive correspondante indiquée dans `sources.list`. Dans le cas où ce paquet dépend d'autres — comme c'est le cas ici — `apt` vérifiera les dépendances et installera les paquets nécessaires. Par exemple :

```
# apt-get install nautilus
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

Le paquet `nautilus` dépend des bibliothèques partagées citées, et `apt` les mettra dans l'archive. Si vous aviez donné les noms de ces bibliothèques sur la ligne de commande `apt-get`, `apt` ne vous aurait pas demandé si vous vouliez continuer : il aurait supposé que vous vouliez installer tous ces paquets.

Cela signifie que `apt` demande confirmation seulement quand il est nécessaire d'installer des paquets qui n'étaient pas spécifiés sur la ligne de commande.

Les options suivantes d'`apt-get` peuvent être utiles :

```
     -h  Ce texte d'aide
     -d  télécharge seulement - N'installe PAS, ni ne décompresse les paquets
     -f  Essaie de continuer si la vérification de l'intégrité échoue
     -s  Pas d'action. Effectue seulement une simulation
     -y  Suppose une réponse affirmative à toutes les requêtes et n'interroge
pas
     -u  Affiche une liste des paquets à mettre à jour
```

Plusieurs paquets peuvent être sélectionnés pour installation sur une seule ligne. Les fichiers téléchargés sur le réseau sont placés dans le répertoire `/var/cache/apt/archives` pour une installation ultérieure.

Vous pouvez indiquer les paquets à retirer sur la même ligne de commande. Mettez seulement un « - » juste après le nom du paquet à enlever, comme ceci :

```
# apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Voyez la section Suppression de paquets, Section 8.3.3 pour plus de détails sur la suppression de paquets.

Si vous endommagez d'une manière ou d'une autre un paquet installé, ou si vous voulez simplement que les fichiers d'un paquet soient réinstallés avec la nouvelle version disponible, vous pouvez utiliser l'option `--reinstall` comme ça :

```
# apt-get --reinstall install gdm
```

```
     Reading Package Lists... Done
     Building Dependency Tree... Done
     0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1
not
      upgraded.
     Need to get 0B/182kB of archives. After unpacking 0B will be used.
     Do you want to continue? [Y/n]
```

8.3.3 Suppression de paquets

Si vous ne voulez plus utiliser un paquet, vous pouvez le supprimer de votre système en utilisant `apt`. Pour ce faire, tapez seulement `apt-get remove paquet`. Par exemple :

```
     # apt-get remove gnome-panel
     Reading Package Lists... Done
     Building Dependency Tree... Done
     The following packages will be REMOVED:
       gnome-applets gnome-panel gnome-panel-data gnome-session
     0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
     Need to get 0B of archives. After unpacking 14.6MB will be freed.
     Do you want to continue? [Y/n]
```

Comme vous pouvez le voir dans l'exemple ci-dessus, `apt` prend soin de supprimer les paquets qui dépendent du paquet dont vous avez demandé la suppression. Avec `apt`, il n'est pas possible de supprimer un paquet sans supprimer aussi les paquets dépendant de celui-ci.

Exécuter `apt-get` comme ci-dessus entraînera la suppression des paquets, mais leurs fichiers de configuration, s'il y en a, resteront intacts sur le système. Pour une suppression complète du paquet, lancez :

```
     # apt-get --purge remove gnome-panel
     Reading Package Lists... Done
     Building Dependency Tree... Done
     The following packages will be REMOVED:
       gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
     0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
     Need to get 0B of archives. After unpacking 14.6MB will be freed.
     Do you want to continue? [Y/n]
```

Remarquez après les noms le signe « * ». Cela indique que les fichiers de configuration vont aussi être supprimés.

Comme avec la méthode `install`, vous pouvez utiliser un symbole avec `remove` pour inverser le sens d'un paquet en particulier. Dans le cas de la suppression, si vous ajoutez un « + » juste après le nom du paquet, le paquet sera installé au lieu d'être supprimé.

```
     # apt-get --purge remove gnome-panel nautilus+
     Reading Package Lists... Done
     Building Dependency Tree... Done
     The following extra packages will be installed:
       bonobo libmedusa0 libnautilus0 nautilus
     The following packages will be REMOVED:
       gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
     The following NEW packages will be installed:
       bonobo libmedusa0 libnautilus0 nautilus
     0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
     Need to get 8329kB of archives. After unpacking 2594kB will be used.
     Do you want to continue? [Y/n]
```

Notez que `apt-get` liste les paquets supplémentaires qui seront installés (c'est-à-dire, les paquets dont l'installation est nécessaire au bon fonctionnement du paquet dont l'installation a été demandée), ceux qui seront supprimés, et ceux qui seront installés (toujours dans les paquets supplémentaires).

8.3.4 Mise à niveau des paquets

La mise à niveau de paquets est une vraie réussite du système `apt`. Une simple commande suffit : `apt-get upgrade`. Vous pouvez utilisez cette commande pour mettre à niveau les paquets d'une même distribution, et aussi pour mettre à niveau vers une nouvelle distribution, bien que la commande `apt-get dist-upgrade` soit préférée à cette dernière ; voir Mettre à niveau vers une nouvelle distribution, Section 8.3.5 pour plus de détails.

Il est utile d'exécuter cette commande avec l'option `-u`. Cette option oblige `apt` à afficher la liste complète des paquets qui seront mis à niveau. Sans elle, vous ferez vos mises à niveau en aveugle. `apt` téléchargera les dernières version de chaque paquet et les installera dans le bon ordre. C'est important de toujours lancer `apt-get update` avant de l'essayer. Voir la section Mise à jour de la liste des paquets disponibles., Section 8.3.1. Observez cet exemple :

```
     # apt-get -u upgrade
     Reading Package Lists... Done
     Building Dependency Tree... Done
     The following packages have been kept back
       cpp gcc lilo
     The following packages will be upgraded
       adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp
indent
       ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
       libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
       libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps
psmisc
     29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
     Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
     Do you want to continue? [Y/n]
```

Le processus est très simple. Notez que dans les premières lignes, `apt-get` dit que certains paquets ont été `laissés de côté` (NdT : « kept back »). Cela signifie qu'il y a de nouvelles versions de ces paquets qui ne seront pas installées pour plusieurs raisons. Les raisons possibles sont des dépendances cassées (un paquet dont il dépend n'a pas de version disponible en téléchargement) ou de nouvelles dépendances (le paquet dépend de nouveaux paquets depuis la dernière version).

Pour le premier cas, il n'y a pas de solution propre. Pour le second, il suffit d'exécuter `apt-get install` sur le paquet en question il téléchargera ainsi les dépendances. Une meilleure solution est d'utiliser `dist-upgrade`. Voir la section Mettre à niveau vers une nouvelle distribution, Section 8.3.5.

8.3.5 Mettre à niveau vers une nouvelle distribution

Cette caractéristique d'`apt` permet de mettre à niveau tout un système Debian en une seule fois, soit par Internet ou soit par un nouveau cédérom (commandé ou téléchargé en image ISO).

Elle est aussi utilisée quand des changements dans les dépendances des paquets installés sont faits. Avec `apt-get upgrade`, ces paquets seraient laissés de côté (NdT :`kept back`).

Par exemple, supposons que vous utilisiez la révision 0 de la version stable de Debian et que vous

achetiez un cédérom de la révision 3. Vous pouvez utiliser `apt` pour mettre à niveau votre système avec ce nouveau cédérom. Pour le faire, utilisez `apt-cdrom` (voir la section Ajouter un cédérom dans le fichier sources.list, Section 8.2.4) pour ajouter le cédérom à votre `/etc/apt/sources.list` et lancez `apt-get dist-upgrade`.

Il est important de noter que `apt` cherche toujours la version du paquet la plus récente. Ainsi, si votre `/etc/apt/sources.list` listait une archive qui a une version plus récente que la version sur le cédérom, `apt` téléchargerait le paquet à partir de là.

Dans l'exemple de la section Mise à niveau des paquets, Section 8.3.4, nous avons vu que quelques paquets avaient été `laissés de côté` (NdT « kept back »). Nous allons maintenant résoudre ce problème par la méthode `dist-upgrade` :

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp
gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

Notez maintenant que le paquet va être mis à niveau et des nouveaux paquets vont aussi être installés (les dépendances des paquets). Notez également que lilo est toujours `laissé de côté`. Il a probablement un problème plus important qu'une nouvelle dépendance. Nous pouvons le découvrir en lançant :

```
# apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

Comme affiché ci-dessus, lilo a un nouveau conflit avec le paquet `debconf-tiny`, ce qui signifie qu'il ne peut pas être installé (ou mis à niveau) sans suppression de debconf-tiny.

Pour savoir ce qu'un paquet ajoute ou supprime, vous pouvez utiliser ceci :

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
 Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2  not upgraded.
```

Cette fois, il est facile de remarquer que le paquet python1.5-dev ne peut pas être installé à cause de dépendances non satisfaites : python1.5.

8.3.6 Supprimer des paquets non utilisés : apt-get clean et autoclean

Lorsque vous installez un paquet, APT récupère les fichiers nécessaires depuis les hôtes listés dans /etc/apt/sources.list, les stocke dans un référentiel local (`/var/cache/apt/archives/`), et ensuite procède à l'installation, voir Installation de paquets, Section 8.3.2.

Le référentiel local peut grandir dans le temps et utiliser beaucoup d'espace disque. Heureusement, APT fournit des outils pour gérer son référentiel local : les méthodes `apt-get clean` et `autoclean`.

`apt-get clean` supprime tout à part les fichiers verrou dans `/var/cache/apt/archives/` et `/var/cache/apt/archives/partial/`. Ainsi, si vous avez besoin de réinstaller un paquet, APT devra le retélécharger.

`apt-get autoclean` supprime seulement les paquets qui ne peuvent plus être téléchargés.

L'exemple suivant montre comment apt-get autoclean fonctionne :

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

Il y a deux fichiers pour le paquet `logrotate` et un pour le paquet `gpm` dans /var/cache/apt/archives.

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

`apt-show-versions` montre que `logrotate_3.5.9-8_i386.deb` fournit la version à jour de `logrotate`, donc `logrotate_3.5.9-7_i386.deb` est inutile car une version plus récente du paquet peut être rapatriée.

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

Finalement, `apt-get autoclean` ne supprime que les anciens fichiers. Pour plus d'informations sur apt-show-versions, voyez Comment mettre à niveau un paquet d'une distribution Debian spécifique ?, Section 8.3.9.

8.3.7 Utiliser `apt` avec dselect

`dselect` est un programme qui aide les utilisateurs à sélectionner des paquets Debian pour les installer. Il est considéré comme un peu difficile et plutôt rébarbatif mais avec de la pratique vous pouvez maîtriser son interface console basée sur ncurses.

Un avantage de dselect est qu'il sait comment utiliser le fait que les paquets Debian « recommandent » et « suggèrent » d'autres paquets à installer. Pour utiliser ce programme, exécutez « `dselect` » en tant que root. Choisissez « apt » comme méthode d'accès. Ce n'est pas vraiment nécessaire, mais si vous n'utilisez pas de cédérom et que vous voulez télécharger des paquets à partir d'Internet, c'est la meilleure façon d'utiliser `dselect`.

Pour acquérir une meilleure compréhension de l'usage de `dselect`, lisez la documentation de `dselect` sur la page Debian `http://www.debian.org/doc/ddp`.

Après avoir fait votre sélection avec `dselect`, utilisez :

```
# apt-get -u dselect-upgrade
```

comme dans l'exemple ci-dessous :

```
# apt-get -u dselect-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

Comparez avec ce que nous avons vu lorsque nous avons lancé `apt-get dist-upgrade` sur le même système :

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

Notez que beaucoup de paquets ci-dessus vont être installés parce que les autres paquets les « suggèrent » ou les « recommandent ». D'autres seront installés ou supprimés (dans le cas de lbxproxy, par exemple) grâce aux choix que nous avons faits lorsque nous naviguions dans la liste de paquets de `dselect`. `dselect` peut être un outil très puissant lorsqu'il est utilisé conjointement avec `apt`.

8.3.8 Comment garder un système mixte ?

L'utilisation d'une des versions de Debian en tant que distribution principale et un ou plusieurs paquets d'une autre branche peut être parfois intéressante pour certains utilisateurs.

Pour définir votre version principale de Debian, vous devriez modifier le fichier `/etc/apt/apt.conf` afin qu'il contienne la ligne suivante :

```
APT::Default-Release "version";
```

Où, *version* est la version de Debian que vous souhaitez utiliser comme distribution principale. Les versions que vous pouvez utiliser sont `stable`n `testing` et `unstable`. Pour installer des paquets d'une autre version, vous devez utiliser APT de la manière suivante :

```
# apt-get -t distribution install paquet
```

Pour que cela fonctionne, il est nécessaire que vous ayez au moins une source APT de la distribution que vous voulez utiliser dans votre `/etc/apt/sources.list`, les paquets doivent exister dans cette source.

Vous pouvez aussi demander une version spécifique d'un paquet en utilisant la syntaxe suivante :

```
# apt-get install paquet= version
```

Par exemple, la ligne ci-dessous installera la version `2.2.4-1` du paquet `nautilus` :

```
# apt-get install nautilus=2.2.4-1
```

IMPORTANT : la distribution « unstable » de Debian est la version où les nouvelles versions des paquets Debian sont d'abord déposées. Cette distribution voit tous les changements des paquets, petits et plus importants qui affectent beaucoup de paquets ou tout le système. Pour cette raison, cette version *ne* devrait *pas* être utilisée par des utilisateurs non expérimentés ni par ceux qui ont besoin de stabilité.

La distribution « testing » n'est pas nécessairement meilleure que « unstable » car elle ne reçoit pas rapidement les mises à jour de sécurité. Pour un serveur ou un système en production, préférez toujours la distribution stable.

8.3.9 Comment mettre à niveau un paquet d'une distribution Debian spécifique ?

Pour les utilisateurs de distribution mixte, `apt-show-versions` est un moyen sûr pour mettre à niveau leur système, en contrôlant la part de la distribution la moins stable qu'ils veulent. Pour l'instant, il est possible de mettre à niveau seulement vos paquets unstable en exécutant après avoir installé le paquet `apt-show-versions` :

```
# apt-get install `apt-show-versions -u -b | grep unstable | cut -d ' ' -f
1`
```

8.3.10 Comment garder des versions spécifiques de paquets installés (complexe)

Vous avez sûrement eu l'occasion de modifier quelque chose dans un paquet et vous n'avez pas eu le temps ou l'envie de les transférer dans une nouvelle version du programme. Ou, par exemple, vous venez juste de passer à la distribution 3.0 de Debian, mais vous voulez continuer avec la version 2.2 pour certains paquets. Vous pouvez « étiqueter » la version que vous avez installée de manière à ce qu'elle ne soit pas mise à niveau.

Utiliser cette fonctionnalité est simple. Vous avez seulement besoin de modifier le fichier `/etc/apt/preferences`.

Le format est simple :

```
Package: <package>
Pin: <pin definition>
Pin-Priority: <pin's priority>
```

Toutes les entrées doivent être séparées par une ligne vide. Par exemple, pour garder le paquet `sylpheed` que j'ai modifié pour utiliser « reply-to-list » dans la version 0.4.99, j'ai ajouté :

```
Package: sylpheed
Pin: version 0.4.99*
```

Notez que j'utilise un `*` (astérisque). C'est un « joker » ; cela signifie que je veux que cette « étiquette » soit valable pour toutes les versions commençant par 0.4.99. C'est parce que Debian donne un numéro de révision à ses paquets et je ne veux pas éviter l'installation de ces révisions. Et, par exemple, les versions 0.4.99-1 et 0.4.99-10 seront installées dès qu'elles seront disponibles. Notez que si vous avez modifié le paquet, vous ne voudrez pas procéder ainsi.

La priorité de l'étiquettage aide à déterminer si un paquet correspondant aux lignes « Packages: » et « Pin: » sera installé, plus la priorité d'un paquet est grande, plus il est probable que le paquet correspondant sera installé. Si vous souhaitez plus de détails, vous pouvez lire `apt_preferences(7)`, mais quelques exemples devraient vous donner quelques idées de base. Les exemples suivant décrivent les effets du positionnement du champ de priorité à différentes valeurs dans l'exemple sur sylpheed ci-dessus.

1001
: La version 0.4.99 de sylpheed ne sera jamais remplacée par apt. Si elle est disponible, apt installera la version 0.4.99 même s'il doit remplacer un paquet avec une version supérieure. Seuls les paquets avec une priorité supérieure à 1000 peuvent remplacer un paquet existant par une version inférieure.

1000
: L'effet est le même qu'avec une priorité de 1001, à l'exception qu'apt refusera d'installer la version 0.4.99 si une autre version est installée.

990
: La version 0.4.99 sera remplacée seulement si une version supérieure est disponible dans la version de la distribution « préférée » en utilisant la variable *APT::Default-Release* (voir Comment garder un système mixte ?, Section 8.3.8 ci-dessus).

500
: Toute version supérieure à 0.4.99 disponible dans n'importe quelle version de la distribution

sera préférée à la version 0.4.99, mais la version 0.4.99 sera toujours préférée à une version moins élevée.

100
: Toute version de sylpheed disponible dans n'importe quelle version de la distribution sera préférée à la version 0.4.99, comme toute version supérieure de sylpheed installée ; la version 0.4.99 sera donc installée seulement si aucune version n'est déjà installée. C'est la priorité des paquets installés.

-1
: Les priorités négatives sont aussi permises et empêchent la version 0.4.99 d'être installée.

Une étiquette peut être spécifiée sur `la version`, `la distribution` ou `l'origine` d'un paquet.

Pour étiqueter une `version`, nous avons vu qu'on pouvait utiliser les numéros de version de manière littérale aussi bien que les jokers pour spécifier plusieurs versions en une fois.

L'option `release` dépend du fichier Release d'un référentiel d'`apt` ou d'un cédérom. Cette option peut être sans intérêt si vous utilisez des référentiels de paquets qui ne contiennent pas ce fichier. Vous pouvez voir le contenu de ce fichier Release dans `/var/lib/apt/lists`. Les paramètres de la distribution sont : `a` (archive), `c` (composants), `v` (version), `o` (origine) et `l` (label).

Un exemple :

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

Dans cet exemple, nous avons choisi les versions Debian 2.2* (qui peuvent être 2.2r2, 2.2r3 — cela satisfait les distributions qui incluent les correctifs de sécurité et d'autres mises à jour très importantes), le référentiel `stable`, la section `main` (opposée à `contrib` ou `non-free`) et l'origine et le label Debian. L'origine (o=) définit qui a produit ce fichier Release, le label (l=) définit le nom de la distribution : Debian pour Debian elle-même et Progeny pour Progeny, par exemple. Un simple fichier Release :

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-
i386_Release
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

8.4 Aides très utiles

8.4.1 Installation multiposte d'une collection de paquets

Comment installer une collection de paquets sur un autre PC ? Rien de plus facile :

```
1. Récupérer la liste des paquest installé sur pc n°1:

sudo dpkg --get-selections >liste-pkg
```

```
=> vous avez maintenant un fichier "list-pkg" créé dans le répertoire
courant. Celui-ci contient la liste des différents paquets installés

2. Vous copiez le fichier sur le pc n°2

3. Vous indiquez les paquets à installer:

sudo dpkg --set-selections <liste-pkg

4. Vous installez les paquets:

sudo apt-get dselect-upgrade
```

8.4.2 Gérer un Proxy (APT-Proxy)

> apt-proxy est un logiciel serveur permettant de simuler un miroir debian. Ce logiciel permet de centraliser les demandes de paquets au niveau d'un petit réseau et de ne les télécharger qu'une unique fois.

Pour l'utilisateur, apt-proxy ne change absolument rien, il n'a même pas besoin de savoir qu'apt-proxy est installé. La seule chose qu'il doit faire, c'est de modifier son fichier sources.list (cf 8.2.1) pour pointer sur le serveur apt-proxy.

8.4.2.1 Configuration client

voici des exemples de lignes à utiliser comme source de paquet:

```
# Debian main
deb http://APTPROXY:9999/debian woody main contrib non-free
deb http://APTPROXY:9999/debian sarge main contrib non-free
deb http://APTPROXY:9999/debian sid main contrib non-free

# Debian non-US
deb http://APTPROXY:9999/non-US woody/non-US main contrib non-free
deb http://APTPROXY:9999/non-US sarge/non-US main contrib non-free
deb http://APTPROXY:9999/non-US sid/non-US main contrib non-free

# Debian security
deb http://APTPROXY:9999/security stable/updates main contrib non-free
deb http://APTPROXY:9999/security woody/updates main contrib non-free
deb http://APTPROXY:9999/security sarge/updates main contrib non-free
```

Chacunes de ces lignes correspond à un back-end configuré sur le serveur. Il est possible d'en ajouter pour des serveurs plus spécialisés.

Pour chaque back-end, le serveur utilise un ou plusieurs "vrais" miroirs.

8.4.2.2 Configuration Proxy

apt-get utilise les variables d'environement http_proxy et ftp_proxy pour savoir si un proxyu doit être utilisé ou non. Un problème peut se poser quand le serveur apt-proxy et les autres serveurs sont de part et d'autre d'un proxy.

Dans ce cas, il est nécéssaire d'utiliser la variable no_proxy.

Exemple:

```
export http_proxy=http://login:passwd@mon.proxy:3128
export ftp_proxy=http://login:passwd@mon.proxy:3128
```

```
export no_proxy='mon.sous.domaine,mon.domaine'
```

8.4.2.3 A savoir

- L'update est automatiquement effectuée par le serveur toutes les deux heures, il y a donc peu de chances pour que le serveur ne soit pas à jour.
- apt-proxy est utilisable en parallèle avec d'autres sources.
- apt-proxy est configurable pour qu'il conserve des différentes versions d'un paquet en cas de problème avec une upgrade.

8.4.2 Comment installer un paquet localement : equivs

Parfois, des personnes veulent utiliser une version spécifique d'un programme dont est disponible seulement le code source, sans paquet Debian. Mais le système de paquets peut être perturbé quand vous le faites. Supposons que vous voulez compiler une nouvelle version de votre serveur de courriels. Tout est bien, mais beaucoup de paquets dépendent d'un MTA (« Mail Transport Agent ») dans Debian. Parce que vous avez installé quelque chose que vous avez compilé vous-même, le système de paquet ne connaît rien sur lui.

C'est là que `equivs` entre en scène. Pour l'utiliser, installez le paquet du même nom. `Equivs` crée un paquet vide capable de réaliser toutes les dépendances, en faisant croire au système de paquets que les dépendances sont satisfaites.

Avant que nous ne commencions, il est bon de vous rappeler qu'il y a d'autres méthodes plus sûres pour compiler un programme déjà empaqueté pour Debian avec différentes options, et l'on ne devrait pas utiliser equivs pour remplacer les dépendances quand on ne sait pas ce que l'on fait. Voir la section Travailler avec des paquets sources, 8.6 pour plus d'informations.

Continuons avec l'exemple du MTA, vous venez juste d'installer votre `postfix` nouvellement compilé et vous continuez avec l'installation de `mutt`. Tout à coup vous découvrez que `mutt` veut installer un autre MTA. Mais vous avez déjà le vôtre.

Allez dans un répertoire (`/tmp`, par exemple) et lancez :

```
# equivs-control nom
```

Remplacez *nom* par le nom du fichier de contrôle que vous voulez créer. Le fichier sera créé comme suit :

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: <entrer le nom du paquet  par défaut equivs-dummy>
Version: <entrer ici la version  par défaut 1.0>
Maintainer: <votre nom et votre adresse mail par défaut nom d'utilisateur>
Pre-Depends: <paquets>
Depends: <paquets>
Recommends: <paquets>
Suggests: <paquet>
Provides: <paquet (virtuel)>
Architecture: all
Copyright: <fichier copyright  par défaut GPL2>
Changelog: <fichier changelog  par défaut un changelog générique>
Readme: <fichier README.Debian  par défaut un générique>
Extra-Files: <fichiers supplémentaires pour le répertoire doc, séparés par
des virgules>
Description: <description courte  par défaut quelques mots judicieux>
```

```
 description longue et informations
 .
 second paragraphe
```

Nous avons juste besoin de modifier cela pour faire ce que nous voulons. Examinons le format des champs et leurs descriptions. Il n'y a pas besoin ici de tout expliquer, faisons ce qui est nécessaire :

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

Oui, c'est tout. `mutt` dépend de `mail-transport-agent`, c'est un paquet virtuel fourni par tous les MTA, je peux simplement nommer le paquet `mail-transport-agent`, mais je préfère utiliser le schéma du paquet virtuel, en utilisant Provides.

Maintenant vous avez seulement besoin de construire le paquet :

```
# equivs-build nom
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Ajoutez ici les commandes pour installer le paquet dans debian/tmp.
touch install-stamp
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package `nom' in
 `../nom_1.0_all.deb'.

The package has been created.
Attention, the package has been created in the current directory,
```

Et installer le `.deb` résultant.

Comme vous pouvez le voir, il y a plusieurs utilisations d'`equivs`. L'une d'entre elles peut éventuellement créer un paquet `mes-favoris`, qui dépend des programmes que vous installez habituellement, par exemple. Laisser aller simplement votre imagination, mais soyez prudent.

Il est important de noter qu'il y a des exemples de fichiers contrôle dans `/usr/share/doc/equivs/examples`. Allez-y jeter un oeil.

8.4.3 Suppression des fichiers de locale inutiles : localepurge

Beaucoup d'utilisateurs Debian utilisent une seule locale. Par exemple, un utilisateur brésilien utilise, généralement, la locale `pt_BR` tout le temps et ne se soucie pas de l'`es`.

`localepurge` est un outil très utile pour ces utilisateurs. Vous pouvez libérer beaucoup d'espace en ayant seulement la locale que vous utilisez. Taper simplement `apt-get install`

`localepurge`.

C'est très simple à configurer, les questions de debconf guident l'utilisateur dans une configuration pas-à-pas. Soyez très attentif à la réponse de la première question, une mauvaise réponse peut supprimer tous les fichiers de locale, y compris ceux que vous utilisez. Le seul moyen de récupérer ces fichiers est de réinstaller tous les paquets qui les fournissent.

8.4.4 Comment savoir quels paquets peuvent être mis à niveau

`apt-show-versions` est un programme qui affiche quels paquets du système peuvent être mis à jour et diverses informations utiles. l'option `-u` affiche une liste des paquets pouvant être mis à niveau :

```
$ apt-show-versions -u
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

8.5 Avoir des informations sur les paquets

Il y a quelques interfaces pour le système `apt` qui facilitent indubitablement l'obtention d'une liste de paquets qui sont disponibles pour l'installation ou qui sont déjà installés, de même que pour rechercher la section dans laquelle se trouve un paquet, sa priorité, sa description, etc.

Mais... notre but ici est d'apprendre comment utiliser le vrai `apt`. Donc comment pouvez-vous retrouver le nom d'un paquet que vous voulez installer ?

Nous avons un bon nombre de recours pour cette tâche. Nous commencerons avec `apt-cache`. Ce programme est utilisé par le système `apt` pour le maintien de sa base de données. Nous jetterons un bref coup d'oeil à ses applications les plus pratiques.

8.5.1 Recherche de noms de paquets

Supposons, par exemple, que vous voulez vous rappeler les bons vieux jours de l'Atari 2600. Vous voulez utiliser `apt` pour installer un émulateur Atari, puis télécharger quelques jeux. Vous pouvez faire :

```
# apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

Nous trouvons différents paquets relatifs à ce que nous cherchons, accompagnés d'une brève description. Pour avoir des informations sur un paquet spécifique, je peux alors utiliser :

```
# apt-cache show stella
Package: stella
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
```

```
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

Depuis cette sortie vous avez plein de détails sur le paquet que vous voulez (ou ne voulez pas) installer, accompagnés de la description complète du paquet. Si le paquet est déjà installé sur votre système et qu'il y a une nouvelle version, vous verrez les informations des deux versions. Par exemple :

```
[root]@[/] # apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text
 screen)
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
Section: base
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text
 screen)
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Notez que le premier de la liste est le paquet disponible et le second est celui déjà installé. Pour plus d'informations générales sur un paquet, vous pouvez utiliser :

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-
1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_main_b
```

```
inary-i386_Packages)(/var/lib/dpkg/status)

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0)
 libsdl1.1 (0 (null)) zlib1g (2 1:1.1.3)
Provides:
1.4.5-1 -
Reverse Provides:
```

Et pour rechercher de quels paquets il dépend :

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

En résumé, nous avons une série d'armes que nous pouvons utiliser pour rechercher le nom d'un paquet que nous voulons.

8.5.2 Utilisation de dpkg pour trouver le nom des paquets

Une des solutions pour localiser le nom d'un paquet est de connaître le nom d'un fichier important trouvé dans le paquet. Par exemple, pour trouver le paquet qui fournit un fichier « `.h` » particulier dont vous avez besoin pour compiler, vous pouvez lancer :

```
# dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

ou :

```
# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

Pour trouver le nom de paquets installés sur votre système, ce qui est utile, par exemple, si vous prévoyez de nettoyer votre disque dur, vous pouvez lancer :

```
# dpkg -l | grep mozilla
ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

Le problème avec cette commande est qu'elle peut « casser » le nom des paquets. Dans l'exemple ci-dessus, le nom complet du paquet est `mozilla-browser`. Pour arranger cela, vous pouvez utiliser la variable d'environnement `COLUMNS` comme ceci :

```
[kov]@[couve] $ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser             0.9.6-7                      Mozilla Web
Browser - core and browser
```

ou la description ou une partie de celle-ci comme cela :

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

8.5.3 Comment installer des paquets « à la demande » ?

Vous compilez un programme et, tout à coup : patatras ! Il y a une erreur parce qu'il faut un fichier `.h` que vous n'avez pas. Le programme `auto-apt` peut vous sauver de ces scénarios. Il vous demande d'installer des paquets nécessaires, en stoppant le processus, et en continuant une fois que le paquet est installé.

Ce que vous avez simplement à faire est de lancer :

```
# auto-apt run commande
```

Où « commande » est la commande à exécuter qui devrait nécessiter des fichiers qui ne sont pas disponibles. Par exemple :

```
# auto-apt run ./configure
```

Il vous demandera d'installer les paquets nécessaires et appellera `apt-get` automatiquement. Si vous êtes sous X, une interface graphique remplacera l'interface texte par défaut.

Pour être efficace, auto-apt conserve des bases de données qui doivent être à jour. C'est fait en appelant les commandes `auto-apt update`, `auto-apt updatedb` et `auto-apt update-locale`.

8.5.4 Comment savoir à quel paquet appartient un fichier ?

Si vous voulez installer un paquet et que vous ne parvenez pas à découvrir quel est son nom en cherchant avec `apt-cache`, mais que vous connaissez le nom du programme lui-même, ou quelque autre fichier appartenant à ce paquet, vous pouvez alors utiliser `apt-file` pour retrouver le nom de ce paquet. Ceci est fait comme suit :

```
$ apt-file search nomdufichier
```

Il fonctionne comme `dpkg -S`, mais il vous affichera aussi les paquets non installés qui contiennent ce fichier. Il peut aussi être utilisé pour trouver quel paquet contient les fichiers d'en-tête manquant à la compilation d'un programme, bien que `auto-apt` soit une meilleure solution pour résoudre ce problème, voir Comment installer des paquets « à la demande » ?, Section 8.5.3.

Vous pouvez lister le contenu d'un paquet en exécutant :

```
$ apt-file list nomdupaquet
```

`apt-file` récupère une base de données des fichiers contenus dans tous les paquets, comme le fait `auto-apt` et elle doit être à jour. C'est fait en exécutant :

```
# apt-file update
```

Par défaut, `apt-file` et `auto-apt` utilisent la même base de données, voir Comment installer des paquets « à la demande » ?, Section 8.5.3.

8.5.5 Comment rester informé sur les changements dans les paquets ?

Tous les paquets installent dans leur répertoire de documentation un fichier appelé `changelog.Debian.gz` qui contient la liste des changements faits sur le paquet depuis la dernière version. Vous pouvez lire ces fichiers à l'aide de `zless`, par exemple ; mais ce n'est pas quelque chose de très simple, après une mise à niveau du système complète, de consulter les changelog de tous les paquets mis à niveau.

Il y a une solution pour automatiser cette tâche au moyen d'un outil appelé `apt-listchanges`. Pour commencer, vous avez besoin d'installer le paquet `apt-listchanges`. Pendant l'installation du paquet, Debconf le configurera. Certaines questions peuvent ne pas vous être affichées car cela dépend de la priorité fixée dans Debconf. Répondez aux questions comme vous le voulez.

La première question vous demande si vous souhaitez qu'`apt-listchanges` affiche les changements. Vous pouvez vous les faire envoyer par courriel, ce qui est une bonne idée pour les mises à niveau automatiques ou vous pouvez simplement demander de les afficher à l'aide d'un outil de défilement tel que `less`, vous pourrez donc inspecter les changements avant de permettre la poursuite de la mise à niveau. Si vous ne voulez pas qu'`apt-listchanges` soit lancé automatiquement lors des mises à niveau, vous pouvez répondre `none`.

Après que apt-listchanges a été installé, aussitôt que des paquets sont téléchargés (ou récupérés sur un cédérom ou un disque monté) par `apt`, il affichera la liste des changements faits sur ces paquets avant de les installer.

8.6 Travailler avec des paquets sources

8.6.1 Télécharger un paquet source

Il est courant dans le monde du logiciel libre d'étudier le code source ou éventuellement de faire des corrections sur du code bogué. Pour le faire, vous aurez besoin de télécharger les sources du programme. Le système `apt` fournit une solution facile pour obtenir le code source des nombreux programmes contenus dans la distribution, en incluant tous les fichiers pour créer un `.deb` pour le programme.

Un autre usage courant des sources Debian est d'adapter une version d'un programme plus récente, de la distribution unstable, par exemple, pour l'utiliser dans la distribution stable. Compiler un paquet avec *stable* générera des .deb avec des dépendances ajustées pour que ce paquet puisse être disponible dans cette distribution.

Pour accomplir ceci, l'entrée `deb-src` dans votre `/etc/apt/sources.list` doit pointer vers unstable. Elle doit être aussi permise (décommentée). Voir la section Le fichier /etc/apt/sources.list, Section 8.2.1.

Pour télécharger un paquet source, vous devez utiliser la commande suivante :

```
$ apt-get source nomdupaquet
```

Cela téléchargera trois fichiers : un `.orig.tar.gz`, un `.dsc` et un `.diff.gz`. Dans le cas où les paquet sont faits spécialement pour Debian, le dernier de ceux-ci n'est pas téléchargé et le premier n'a généralement pas « `orig` » dans le nom.

Le fichier `.dsc` est utilisé par dpkg-source pour dépaqueter le paquet source dans le répertoire *nomdupaquet-version*. Avec chaque paquet source téléchargé, il y a un répertoire `debian/` qui contient les fichiers nécessaires pour la création d'un paquet `.deb`.

Pour construire automatiquement le paquet lorsqu'il est téléchargé, ajoutez seulement `-b` à la ligne de commande, comme ceci :

```
$ apt-get -b source nomdupaquet
```

Si vous décidez de ne pas créer de .deb lorsque vous le téléchargez, vous pouvez le créer plus tard en exécutant :

```
$ dpkg-buildpackage -rfakeroot -uc -b
```

à l'intérieur du répertoire créé pour le paquet après le téléchargement. Pour installer le paquet construit par la commande ci-dessus, vous devez directement utiliser le gestionnaire de paquets, comme ceci :

```
# dpkg -i fichier.deb
```

Il y a une différence entre les méthodes `apt-get source` et ses autres méthodes. La méthode `source` peut être utilisée par un utilisateur normal, sans avoir besoin des droits root. Les fichiers sont téléchargés dans le répertoire à partir duquel la commande `apt-get source paquet` a été appelée.

8.6.2 Paquets nécessaires pour la compilation d'un paquet source

Normalement, des en-têtes et des bibliothèques dynamiques spécifiques doivent être présentes pour qu'un paquet source puisse être compilé. Tous les paquets source ont un champ dans leur fichier contrôle appelé « Builds-Depends : » qui indique quels paquets supplémentaires sont nécessaires pour la construction à partir des sources.

`apt` possède une solution simple pour télécharger ces paquets. Exécutez juste `apt-get build-dep paquet`, où « paquet » est le nom du paquet que vous allez construire. Par exemple :

```
# apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev libgnorba-
dev
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

Les paquets qui seront installés sont les paquets nécessaires pour que `gmc` soit compilé correctement. C'est important de noter que cette commande ne cherche pas le paquet source du programme à compiler. Vous devrez par conséquent exécuter `apt-get source` séparément pour le récupérer.

Si tout ce que vous voulez est de vérifier quels paquets sont nécessaires pour construire un paquet donné, il existe une variante de la commande `apt-cache show` (voir Avoir des informations sur les paquets, 9.5) qui affichera, parmi toutes les informations disponibles, la ligne `Build-Depends` qui liste ces paquets.

```
# apt-cache showsrc paquet
```

8.7 Comment traiter les erreurs ?

8.7.1 Erreurs courantes

Des erreurs arriveront souvent, la plupart d'entre elles causées par les utilisateurs qui ne font pas attention. Ce qui suit est une liste des erreurs les plus fréquemment rapportées et comment les traiter.

Si vous recevez un message qui ressemble à celui ci-dessous quand vous essayez d'exécuter `apt-get install paquet`...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/
 Packages'
  (/var/state/apt/lists/people.debian.org_%7ekov_debian_unstable_Packages) -
stat
  (2 No such file or directory)
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

vous avez oublié d'exécuter `apt-get update` après vos derniers changements dans le fichier `/etc/apt/sources.list`.

Si l'erreur ressemble à :

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission
denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you
root?
```

quand vous essayez n'importe quelle méthode `apt-get` autre que `source`, vous n'avez pas les permissions root, parce que vous l'exécutez en utilisateur normal.

Il y a une erreur similaire à celle ci-dessus qui arrive lorsque vous exécutez deux copies d'`apt-get` en même temps, ou éventuellement si vous essayez d'exécuter `apt-get` pendant qu'un processus `dpkg` est actif. La seule méthode qui peut être utilisée simultanément avec les autres est la méthode `source`.

Si une installation s'interrompt au milieu du processus et que vous ne pouvez plus installer ou supprimer de paquets, essayez d'exécuter ces deux commandes :

```
# apt-get -f install
# dpkg --configure -a
# dpkg --reconfigure
```

et ensuite essayez à nouveau. Il sera peut être nécessaire d'exécuter la seconde des commandes ci-dessus plus d'une fois. C'est une leçon importante pour ces aventuriers qui utilisent « unstable ».

Si vous rencontrez l'erreur « E: Dynamic MMap ran out of room » en lançant `apt-get update`, ajoutez la ligne suivante dans `/etc/apt/apt.conf` :

```
APT::Cache-Limit 10000000;
```

8.7.2 Où puis-je trouver de l'aide ?

Si vous êtes tourmenté par le doute, consultez l'importante documentation disponible pour le système de gestion des paquets Debian. Les `--help` et les pages de manuel peuvent être une énorme aide pour vous, comme le peut la documentation contenue dans les répertoires `/usr/share/doc` comme `/usr/share/doc/apt`.

Si la documentation ne suffit pas à faire disparaître votre peur, essayez de rechercher la réponse sur les listes de discussions Debian. Vous pouvez obtenir plus d'informations des listes d'utilisateurs spécifiques sur le site Web Debian : http://www.debian.org.

Souvenez-vous que ces listes et ressources doivent être utilisées seulement par les utilisateurs

Debian ; les utilisateurs d'autres systèmes trouveront un meilleur support dans les communautés de leur propre distribution.

9 Connection à distance SSH

9.1 Connection SSH

9.1.1. Introduction et mise-en-garde

9.1.1.1. Qu'est-ce que SSH ?

SSH signifie *Secure SHell*. C'est un protocole qui permet de faire des connexions sécurisées (i.e. cryptées) entre un serveur et un client SSH. Nous allons utiliser le programme OpenSSH, qui est la version libre du client et du serveur SSH.

9.1.1.2. Mise en garde sur la sécurité

9.1.1.2.1. Nature du problème

Installer un serveur SSH permet aux utilisateurs d'accéder au système à distance, en rentrant leur login et leur mot de passe (ou avec un mécanisme de clés). Cela signifie aussi qu'un pirate peut essayer d'avoir un compte sur le système (pour accéder à des fichiers sur le système ou pour utiliser le système comme une passerelle pour attaquer d'autres systèmes) en essayant plein de mots de passes différents pour un même login (il peut le faire de manière automatique en s'aidant d'un dictionnaire électronique). On appelle ça une attaque *en force brute*.

Il y a donc trois contraintes majeures pour garder un système sécurisé après avoir installé un serveur SSH :

- avoir un serveur SSH à jour au niveau de la sécurité, ce qui doit être le cas si vous faites consciencieusement les mises à jour de sécurité en suivant les mises à jour par apt;
- que les mots de passes de *TOUS* les utilisateurs soient suffisamment complexes pour résister à une attaque en force brute ;
- surveiller les connexions en lisant régulièrement le fichier de log `/var/log/auth.log`.

9.1.1.2.2. Choisir des mots de passe complexes

Un mot de passe complexe est un mot de passe qui ne veut rien dire, qui n'est pas dans le dictionnaire et qui comporte au moins 8 caractères, de préférence avec un mélange de lettres minuscules, de lettres majuscules, de chiffres et de caractères de ponctuation.

Une bonne méthode pour obtenir un mot de passe complexe et facile à retenir consiste à choisir une phrase et à prendre la première lettre de chaque mot, avec quelques complications en plus.

Par exemple, la phrase "*Linux, moi j'y comprends rien de rien !*" donne le mot de passe **Lmjycr2r!**

9.1.1.2.3. Tester la complexité des mots de passe

Pour vérifier que les mots de passe des utilisateurs du système sont vraiment complexes, le root peut les soumettre à un cracker de mots de passe... et voir combien de temps ils résistent !

Les mots de passes des utilisateurs sont stockés dans le fichier `/etc/shadow`. Seul l'utilisateur root peut lire ce fichier. Pour tester la complexité des mots de passes, le root peut donc installer le programme john et le lancer sur le fichier `/etc/shadow` :

```
# apt-get install john
# john /etc/shadow
```

Quand *john* a trouvé un mot de passe, il l'affiche avec le login associée.

Attention, *john* utilisera le processeur à 100 % ! Il est donc conseillé de lui donner un priorité faible (commande **nice** ou **renice**) si la machine doit être utilisée pendant ce temps. Plus le nombre d'utilisateurs est grand, plus il faudra laisser tourner *john* longtemps pour que le test soit significatif.

9.1.2. Le système de clés de SSH

9.1.2.1. La théorie de la cryptographie asymétrique

SSH utilise la cryptographie asymétrique RSA ou DSA. En cryptographie asymétrique, chaque personne dispose d'un couple de clé : une clé publique et une clé privée. La clé publique peut être librement publiée tandis que la clé privée doit rester secrète. La connaissance de la clé publique ne permet pas d'en déduire la clé privée.

Si la personne A veut envoyer un message confidentiel à la personne B, A crypte le message avec la clé publique de B et l'envoie à B sur un canal qui n'est pas forcément sécurisé. Seul B pourra décrypter le message en utilisant sa clé privée.

9.1.2.2. La théorie de la cryptographie symétrique

SSH utilise également la cryptographie symétrique. Son principe est simple : si A veut envoyer un message confidentiel à B, A et B doivent d'abord posséder une même clé secrète. A crypte le message avec la clé secrète et l'envoie à B sur un canal qui n'est pas forcément sécurisé. B décrypte le message grâce à la clé secrète. Toute autre personne en possession de la clé secrète peut décrypter le message.

La cryptographie symétrique est beaucoup moins gourmande en ressources processeur que la cryptographie asymétrique... mais le gros problème est l'échange de la clé secrète entre A et B. Dans le protocole SSL, qui est utilisé par SSH et par les navigateurs Web, la cryptographie asymétrique est utilisée au début de la communication pour que A et B puissent s'échanger une clé secrète de manière sécurisée... puis la suite la communication est sécurisée grâce à la cryptographie symétrique en utilisant la clé secrète échangée.

Pour plus d'informations sur la cryptographie, je vous conseille la lecture du dossier consacré à ce sujet par le magazine pour la science dans son hors-série de Juillet-Octobre 2002.

9.1.2.3. L'établissement d'une connexion SSH

Un serveur SSH dispose d'un couple de clés RSA stocké dans le répertoire `/etc/ssh/` et généré lors de l'installation du serveur. Le fichier `ssh_host_rsa_key` contient la clé privée et a les permissions 600. Le fichier `ssh_host_rsa_key.pub` contient la clé publique et a les permissions 644.

Nous allons suivre par étapes l'établissement d'une connexion SSH :

1. Le serveur envoie sa clé publique au client.
2. Le client génère une clé secrète et l'envoie au serveur, en cryptant l'échange avec la clé publique du serveur (cryptographique asymétrique). Le serveur décrypte la clé secrète en utilisant sa clé privée, ce qui prouve qu'il est bien le vrai serveur.

3. Pour le prouver au client, il crypte un message standard avec la clé secrète et l'envoie au client. Si le client retrouve le message standard en utilisant la clé secrète, il a la preuve que le serveur est bien le vrai serveur.
4. Une fois la clé secrète échangée, le client et le serveur peuvent alors établir un canal sécurisé grâce à la clé secrète commune (cryptographie symétrique).
5. Une fois que le canal sécurisé est en place, le client va pouvoir envoyer au serveur le login et le mot de passe de l'utilisateur pour vérification. La canal sécurisé reste en place jusqu'à ce que l'utilisateur se déloggue.

La seule contrainte est de s'assurer que la clé publique présentée par le serveur est bien sa clé publique... sinon le client risque de se connecter à un faux serveur qui aurait pris l'adresse IP du vrai serveur (ou toute autre magouille). Une bonne méthode est par exemple de demander à l'administrateur du serveur quelle est le *fingerprint* de la clé publique du serveur avant de s'y connecter pour la première fois. Le *fingerprint* d'une clé publique est une chaîne de 32 caractères hexadécimaux unique pour chaque clé ; il s'obtient grâce à la commande **ssh-keygen -l**.

Installation et configuration de SSH

9.1.3.1. Installation du client et du serveur SSH

Le client et le serveur SSH sont dans le même package *ssh*. Ce package est installé dès la première utilisation de *dselect*.

Maintenant que votre système est à jour niveau sécurité, vous pouvez activer le serveur SSH, si vous le souhaitez. Pour cela, supprimez le fichier `/etc/ssh/sshd_not_to_be_run` et lancer SSH :

```
# rm /etc/ssh/sshd_not_to_be_run
# /etc/init.d/ssh start
Starting OpenBSD Secure Shell server: sshd.
```

9.1.3.2. Configuration du serveur SSH

Le fichier de configuration du serveur SSH est `/etc/ssh/sshd_config`. A ne pas confondre avec le fichier `/etc/ssh/ssh_config`, qui est le fichier de configuration du client SSH.

Nous allons vous commenter les lignes les plus importantes de ce fichier de configuration :

```
Port 22
```

- Signifie que le serveur SSH écoute sur le port 22, qui est le port par défaut de SSH. Vous pouvez le faire écouter sur un autre port en changeant cette ligne. Vous pouvez aussi le faire écouter sur plusieurs ports à la fois en rajoutant des lignes similaires.

```
Protocol 2
```

- Signifie que votre serveur SSH accepte uniquement la version 2 du protocole SSH. C'est une version plus sécurisée que la version 1 du protocole. Seuls certains vieux clients SSH ne savent faire que du SSH version 1. Si vous voulez que le serveur accepte les deux protocoles, changez la ligne en :

```
Protocol 2,1
```

```
PermitRootLogin yes
```

- Signifie que vous pouvez vous logguer en root par SSH. Vous pouvez changer et mettre "*no*", ce qui signifie que pour vous connecter en root à distance, vous devrez d'abord vous connecter par SSH en tant que simple utilisateur, puis utiliser la commande **su** pour devenir root. C'est une sorte de double protection.

```
X11Forwarding yes
```

- Signifie que vous allez pouvoir travailler en export display par SSH. Ce sera expliqué plus tard, dans la partie suivante 9.2 *Faire de l'export display*.

Si vous avez modifié le fichier de configuration du serveur, il faut lui dire de relire son fichier de configuration :

```
# /etc/init.d/ssh reload
Reloading OpenBSD Secure Shell server's configuration.
```

9.1.4. Se logguer par SSH

9.1.4.1. Authentification par mot de passe

C'est la méthode la plus simple. Depuis la machine cliente, tapez :

```
% ssh login@nom_DNS_du_serveur_SSH
```

- Si c'est la première connexion SSH depuis ce client vers ce serveur, il vous demande si le fingerprint de la clé publique présentée par le serveur est bien le bon. Pour être sûr que vous vous connectez au bon serveur, vous devez connaître de façon certaine le fingerprint de sa clé publique et la comparer à celle qu'il vous affiche. Si les deux fingerprints sont identiques, répondez *yes*, et la clé publique du serveur est alors rajoutée au fichier `~/.ssh/known_hosts`.
- Si vous vous êtes déjà connecté depuis ce client vers le serveur, sa clé publique est déjà dans le fichier `~/.ssh/known_hosts` et il ne vous demande donc rien.

Ensuite, entrez votre mot de passe... et vous verrez apparaître le prompt, comme si vous vous êtiez loggué en local sur la machine.

9.1.4.2. Authentification par clé

Au lieu de s'authentifier par mot de passe, les utilisateurs peuvent s'authentifier grâce à la cryptographie asymétrique et son couple de clés privée/publique, comme le fait le serveur SSH auprès du client SSH.

9.1.4.2.1. Générer ses clés

Pour générer un couple de clés DSA, tapez :

```
% ssh-keygen -t dsa
```

Les clés générées ont par défaut une longueur de 1024 bits, ce qui est aujourd'hui considéré comme suffisant pour une bonne protection.

Par défaut (il demande confirmation lors du processus de création), la clé privée est stockée dans le

fichier `~/.ssh/id_dsa` avec les permissions 600 et la clé publique est stockée dans le fichier `~/.ssh/id_dsa.pub` avec les permissions 644.

Lors de la création, il vous demande une *pass phrase* qui est un mot de passe pour protéger la clé privée. Cette *pass phrase* sert à crypter la clé privée. La *pass phrase* vous sera alors demandée à chaque utilisation de la clé privée, c'est à dire à chaque fois que vous vous logguerez en utilisant cette méthode d'autentification. Un mécanisme appelé *ssh-agent* permet de ne pas rentrer le mot de passe à chaque fois... comme nous le verrons un peu plus loin dans ce chapitre.

Vous pouvez à tout moment changer la *pass phrase* qui protège votre clé privée avec la commande **ssh-keygen -p**.

9.1.4.2.2. Autoriser votre clé publique

Pour cela, il suffit de copier votre clé publique dans le fichier `~/.ssh/authorized_keys` de la machine sur laquelle vous voulez vous logguer à distance. La commande suivante permet de réaliser cette opération via SSH :

```
% ssh-copy-id -i ~/.ssh/id_dsa.pub login@nom_DNS_du_serveur
```

et entrez le mot de passe de votre compte sur le serveur.

9.1.4.2.3. Se logguer

La commande est la même que pour une autentification par mot de passe.

9.1.5. Transfert de fichiers par SSH

9.1.5.1. En console

Le transfert de fichiers par SSH est possible avec **scp** (comme Ssh CoPy), qui s'utilise la même manière que la commande **cp.**

Encore une fois, vous pouvez utiliser la méthode d'autentification par mot de passe ou par clés, l'utilisation est la même.

Pour illustrer la syntaxe, je vais donner quelques exemples :

- pour transférer le fichier `test1.txt` situé dans le répertoire courant vers le home du compte *toto* de la machine *ordi1.exemple.org* sur laquelle tourne un serveur SSH :

```
% scp test1.txt toto@ordi1.exemple.org:
```

- pour récupérer le fichier `test2.txt` situé le home de l'utilisateur *toto* de la machine *ordi2.exemple.org* et l'écrire dans le répertoire courant :

```
% scp toto@ordi2.exemple.org:test2.txt .
```

- pour récupérer tous les fichiers ayant l'extension `.txt` situés dans le répertoire `/usr/local` de la machine *ordi2.exemple.org* et l'écrire dans le sous-répertoire `test-scp` du répertoire courant :

```
% scp toto@ordi2.exemple.org:/usr/local/*.txt test-scp
```

- pour transférer l'intégralité du sous-répertoire `test-scp` du répertoire courant vers le sous répertoire `incoming` du home de l'utilisateur *toto* de la machine *ordi1.exemple.org* :

```
% scp -r test-scp toto@ordi1.exemple.org:incoming
```

9.1.5.2. En graphique

gFTP, fait également office de client SFTP.

Lançez *gFTP* avec la commande **gftp**. Ensuite, allez dans le menu *FTP / Options*, sélectionnez l'onglet *SSH*, mettez le paramètre *Chemin sftp-server SSH2* à `/usr/lib/` et cliquez sur *Enregistrez*.

Pour vous connecter, entrez le nom DNS du serveur ainsi que le login et le mot de passe, sélectionnez *SSH2* à la place de *FTP* dans la liste déroulante et tapez **Entrée**.

9.1.6. Se logguer par SSH sans taper de mot de passe

9.1.6.1. Le principe

Cette section s'adresse à ceux qui utilisent un couple de clés publiques / privées, et qui ont crypté leur clé privée avec une *pass phrase* (c'est la configuration la plus sûre). Par conséquent, le client SSH demande la *pass phrase* à chaque utilisation des clés pour s'autentifier.

Pour éviter d'avoir à taper systématiquement sa *pass phrase*, il faut utiliser *ssh-agent* : ce programme tourne en tâche de fond et garde la clef en mémoire. La commande *ssh-add* permet de donner sa clé à *ssh-agent*. Ensuite, quand vous utilisez le client SSH, il contacte *ssh-agent* pour qu'il lui donne la clé.

9.1.6.2. La pratique

9.1.6.2.1. en console

Dans une console, ouvrez un screen avec *ssh-agent* en tâche de fond :

```
% ssh-agent screen
```

Puis donnez votre clé à l'agent :

```
% ssh-add
```

Il vous demande alors votre *pass phrase*. Maintenant que votre clé a été transmise à l'agent, vous pouvez vous connecter sans entrer de mot de passe à toutes les machines pour lesquelles vous avez mis votre clé publique dans le fichier `~/.ssh/authorized_keys`.

9.1.6.2.2. en mode graphique

Démarrez le serveur graphique avec la commande :

```
% ssh-agent startx
```

Puis ouvrez un *xterm* et tapez :

```
% ssh-add
```

L'agent sera alors actif pour toutes les applications que vous utiliserez en mode graphique, et notamment tous les *xterm* ouverts ou que vous ouvrirez.

9.1.6.2.3. avec GDM

Si vous utilisez GDM, l'agent SSH a déjà été lançé par GDM. Vous n'avez donc plus qu'à exécuter **ssh-add** une fois que vous êtes loggué.

9.1.7. Faire des tunnels SSH

Faire un tunnel SSH est un moyen simple de crypter n'importe quelle communication TCP entre votre machine et une machine sur laquelle vous avez un accès SSH.

Par exemple, pour établir un tunnel SSH pour une connexion HTTP vers la machine *serveur.exemple.org* :

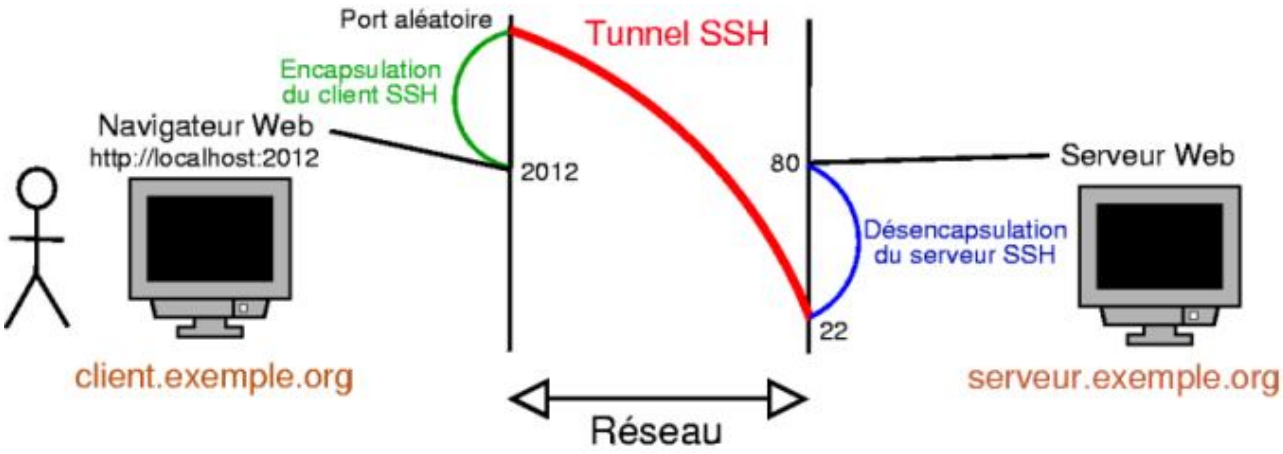
```
% ssh -L 2012:serveur.exemple.org:80 toto@serveur.exemple.org
```

où *2012* est le port sur la machine cliente à partir duquel la connexion entre dans le tunnel SSH (le port doit être supérieur à 1024 si on ne veut pas avoir à lançer le tunnel en tant que *root*).

Ensuite, il suffit de lançer un navigateur Web en lui demandant de se connecter en local sur ce port :

```
% w3m http://localhost:2012
```

Figure 1-2. Exemple de tunnel SSH



9.1.8. Et le bon vieux Telnet... ?

9.1.8.1. Qu'est-ce que Telnet ?

Telnet, c'est comme SSH... mais en moins bien ! Telnet est un protocole qui permet d'accéder à distance à une machine, mais la connexion n'est pas sécurisée : le mot de passe et les données sont transférés en clair ! Telnet ne permet pas de faire des transferts de fichiers. Il est donc conseillé de

ne pas utiliser Telnet mais uniquement SSH.

9.1.8.2. Client et Serveur Telnet

Le client Telnet se trouve dans le package *telnet*. Ce package est installé par défaut.

Le serveur Telnet se trouve dans le package *telnetd*. Il n'y a aucune configuration à faire.

Pour se connecter à un serveur Telnet, tapez :

```
% telnet nom_DNS_du_serveur_telnet
```

et ensuite rentrez votre login et votre mot de passe quand il vous le demande.

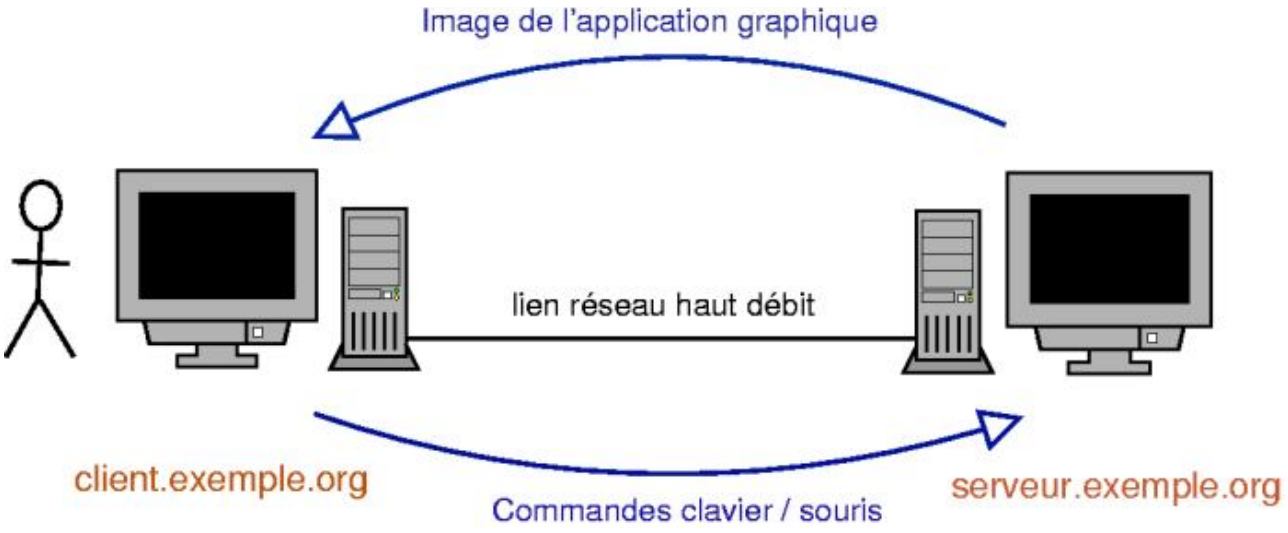
9.2 Faire de l'export display (X Forwarding)

9.2.1 Qu'est-ce que l'export display ?

L'*export display* consiste à se logguer à distance en mode graphique, comme on le fait avec un client et un serveur SSH en mode texte. On peut alors exécuter des applications graphiques sur le serveur distant : la fenêtre graphique de l'application et son contenu seront envoyés par le réseau vers la machine cliente ; les données du clavier et de la souris de la machine cliente sont envoyées vers le serveur.

L'export display nécessite une bonne connexion réseau entre le client et le serveur puisque le serveur envoie des images de l'écran au client...

Figure 2-1. Le principe de l'export display



9.2.2 Se connecter à un Unix/Linux à distance...

9.2.2.1. depuis une machine Unix/Linux

Il y a trois possibilitées de connexion, sachant que seule la première est entièrement cryptée.

9.2.2.1.1. Export display par SSH

SSH possède une fonction d'export display. Il faut que le serveur SSH distant ait autorisé la fonction d'export display, comme expliqué précédemment. Pour l'utiliser, tapez dans un *xterm* :

```
% ssh -X login@serveur.exemple.org
```

puis lancez l'application graphique de votre choix...

9.2.2.1.2. Export display classique

Le serveur graphique possède une fonction d'export display, mais qui n'est pas cryptée comme avec SSH. Il faut d'abord autoriser les connexions en provenance du serveur X, puis demander au serveur X distant de renvoyer le display vers le serveur X de la machine cliente :

```
% xhost + serveur.exemple.org
% ssh login@serveur.exemple.org
% export DISPLAY=client.exemple.org:0.0
```

Pour que cela marche, il faut que le serveur X de la machine cliente ne soit pas lançé avec l'option **-nolisten tcp**, ce qui est le cas par défaut !

Si vous utilisez la commande **startx**, enlevez cette option du fichier `/etc/X11/xinit/xserverrc` :

```
#!/bin/sh
exec /usr/bin/X11/X -dpi 100
```

et redémarrez le serveur X.

Si vous utilisez GDM, enlevez cette option du paramètre *command=* de la section *server-Standard* du fichier `/etc/gdm/gdm.conf` :

```
[server-Standard]
name=Standard server
command=/usr/bin/X11/X -deferglyphs 16
```

et redémarrez GDM.

9.2.2.1.3. Export display avec XDMCP

Le protocole XDMCP permet de se logguer au serveur graphique d'une machine distante, et de faire exactement comme si on était loggué sous X en local sur la machine.

Ce protocole fait tout transiter en clair sur le réseau, y compris les mots de passe !

Toujours sur le serveur, éditez le fichier `/etc/gdm/gdm.conf` et modifiez la section *xdmcp* :

```
[xdmcp]
Enable=true
```

Redémarrez GDM :

```
# /etc/init.d/gdm restart

Stopping GNOME Display Manager: gdm.
Starting GNOME Display Manager: gdm.
```

Sur le client, vous pouvez alors vous logguer au serveur graphique distant. En console, tapez :

```
% X -query serveur.exemple.org
```

et vous devriez voir la fenêtre GDM du serveur :

Figure 2-2. Export display par XDMCP

Si vous avez déjà un serveur graphique de lançé sur le client et que vous ne pouvez pas le fermer, vous pouvez lancer un deuxième serveur graphique depuis une console :

```
% X :1 -query serveur.exemple.org
```

Le deuxième serveur X est alors présent sur la console n°13 ; pour y accéder depuis une autre console, vous pouvez vous mettre sur la console n°1 et faire **Alt-Flèche Gauche**.

9.2.2.2. depuis une machine Windows

Il suffit d'installer sur la machine Windows un programme qui sait se connecter à un serveur X, comme Cygwin par exemple. Cygwin est un environnement Unix libre pour Windows.

9.2.2.2.1. Installer Cygwin

Allez sur www.cygwin.com et cliquez sur l'icône *Install Cygwin now* qui se trouve en haut à droite de la page. Téléchargez le fichier `setup.exe` et exécutez-le. Le fichier contient le programme d'installation, mais pas Cygwin en lui-même.

La procédure d'installation démarre alors :

1. Une fois passé l'écran d'accueil, sélectionnez *Install from Internet*.
2. Sélectionnez le répertoire d'installation.
3. Sélectionnez un répertoire dans lequel il va écrire les fichiers qu'il va télécharger.
4. Si vous devez passer par un proxy pour accéder à Internet, entrez ses paramètres. Sinon, sélectionnez *Direct Connection*.
5. Sélectionnez un miroir dans la liste. Si vous êtes connecté au réseau VIA, entrez l'adresse *ftp://ftp.via.ecp.fr/pub/cygwin/* et cliquez sur *Add*.
6. Ensuite vient l'étape de sélection des packages. Sélectionnez deux packages supplémentaires par rapport à la configuration par défaut :
   - *openssh* dans la section *Net*,
   - *XFree86-base* dans la section *XFree86*.

7. Il va ensuite télécharger les packages sélectionnés et les installer.
8. Après la dernière étape, il lance les scripts de configuration-après-installation et ajoute l'icône *Cygwin* sur le bureau.

9.2.2.2.2. Utiliser Cygwin

Double-cliquez sur l'icône *Cygwin* ; une console apparaît.

Vous pouvez alors vous servir de tous les outils Unix disponibles avec Cygwin... comme si vous étiez sous Linux !

Si vous voulez lancer un serveur X, commencez par éditez le fichier `/usr/X11R6/bin/startxwin.bat` et rajoutez à la fin du fichier la ligne suivante :

```
run setxkbmap -layout fr
```

ce qui vous permettra d'avoir un clavier français sous X. Vous pouvez maintenant lancer le serveur graphique :

```
Administrateur@CLIENT
$ startxwin.bat
```

Depuis la console Cygwin, vous pouvez également lancer un serveur graphique vers une autre machine, comme si vous étiez sur une console Linux :

```
Administrateur@CLIENT
$ X -query serveur.exemple.org
```

9.2.3 Se connecter à un Windows à distance depuis un Linux

Il y a plusieurs possibilités :

- par VNC vers un Windows sur lequel tourne un serveur VNC : installez le package **xvncviewer** qui contient un client VNC puis lancez la commande suivante pour vous connecter au serveur VNC :

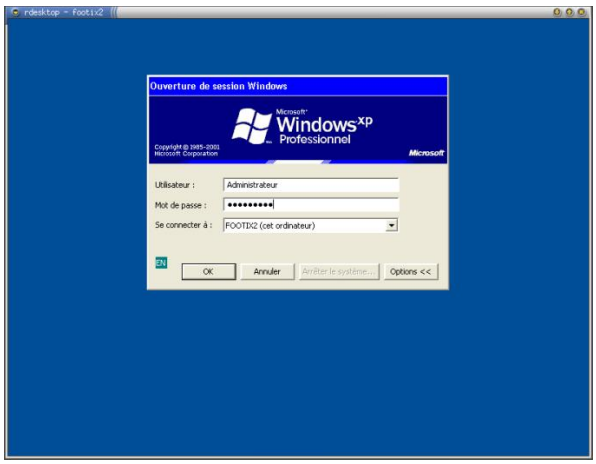
```
% xvncviewer serveur.exemple.org
```

- par Terminal Serveur vers un Windows 2000 Server ou Remote Desktop vers un Windows XP Pro (le protocole est le même) : installez le package **rdesktop** qui contient un client RDP (Remote Desktop Protocol) et lancez la commande suivante pour vous connecter au Windows distant :

```
% rdesktop -u login serveur.exemple.org
```

Si vous suivez la méthode Woody, ajoutez l'option **-k 40c** pour mettre le clavier en AZERTY français (ne mettez pas cette option si vous suivez la méthode Sid). Vous pouvez utiliser en plus l'option **-d nom_du_domaine** pour préciser un domaine.

Figure 2-3. rdesktop


rdesktop - Footix2
Ouverture de session Windows
Microsoft
Windows xp
Professionnel
Copyright © 1985-2001
Microsoft Corporation
Microsoft
Utilisateur :
Administrateur
Mot de passe :
Se connecter à :
FOOTIX2 (cet ordinateur)
OK
Annuler
Arrêter le système...
Options <<

10. Le partage de fichiers

10.1. NFS côté client

10.1.1. Configuration nécessaire

Tout d'abord, il faut avoir compilé le module du noyau *NFS File system support* avec en plus l'option *Provide NFSv3 client support*. Ensuite, il faut le package *nfs-common* qui est normalement installé par défaut.

10.1.2. Monter un répertoire partagé par NFS

Pour monter le répertoire `/home/ftp/` partagé par la machine dont le nom DNS est *ordi1.exemple.org* dans le répertoire `/mnt/test` déjà crée, utilisez la commande **mount** :

```
# mount -t nfs ordi1.exemple.org:/home/ftp /mnt/test
```

Une fois que vous n'avez plus besoin de ce partage, vous pouvez le démonter :

```
# umount /mnt/test
```

Pour que ce répertoire soit monté à chaque démarrage, rajoutez la ligne suivante dans le fichier de configuration `/etc/fstab` :

```
ordi1.exemple.org:/home/ftp  /mnt/test   nfs
soft,timeo=5,intr,rsize=8192,wsize=8192  0  0
```

Pour comprendre les options, regardez leur description dans **man mount**.

10.2. NFS côté serveur

10.2.1. Configuration nécessaire

Tout d'abord, il faut avoir compilé le module du noyau *NFS Server support* avec en plus l'option *Provide NFSv3 server support* .

Il faut également le package *nfs-kernel-server* :

```
# apt-get install nfs-kernel-server
```

10.2.2. Partager un répertoire

Editez le fichier `/etc/exports` et rajoutez la ligne suivante pour partager le répertoire `/home/test/` à la machine *ordi2.exemple.org* :

```
/home/test      ordi2.exemple.org(rw,root_squash)
```

L'option **rw** permet d'exporter en *lecture-écriture (utiliser **ro** pour exporter en lecture seule). L'option **root_squash** spécifie que le root de la machine ordi2.exemple.org n'a pas les droits de root sur le répertoire partagé (l'option **no_root_squash** spécifie que le root de la machine sur laquelle le répertoire est monté a les droits de root sur le répertoire). L'option root_squash est*

l'option par défaut.

L'option **rw** signifie en réalité que l'utilisateur dont l'ID est 1001 (par exemple...) sur le client NFS a les droits d'écriture sur les fichiers et les répertoires qui appartiennent à l'utilisateur dont l'ID est 1001 sur le serveur NFS. Attention, ces utilisateurs n'ont pas forcément le même nom de compte Unix et ne correspondent pas forcément aux mêmes personnes !

Enfin, demandez à *nfs-kernel-server* de démarrer :

```
# /etc/init.d/nfs-kernel-server start
Exporting directories for NFS kernel daemon...done.
Starting NFS kernel daemon: nfsd mountd.
```

Par la suite, il suffira de lui dire de relire son fichier de configuration après chaque modification :

```
# /etc/init.d/nfs-kernel-server reload
Re-exporting directories for NFS kernel daemon...done.
```

11. Internet en ligne de commande

11.1 Lynx

11.1.1 Présentation

Lynx est un client World Wide Web (WWW) complet pour les utilisateurs utilisant des périphériques adressables via le curseur, à affichage orienté caractère (p.ex. les terminaux vt100, les émulateurs vt100 tournant sur Windows 95/NT ou Macintosh, ou n'importe quel affichage « orienté-curses »). Il affichera des documents HTML contenant des liens vers des fichiers résidant dans le système local, aussi bien que des fichiers résidant sur des systèmes distants exécutant des serveurs Gopher, HTTP, FTP, WAIS et NNTP. Les versions actuelles de *Lynx* tournent sur Unix, VMS, Windows 95/NT, 386DOS et OS/2 EMX.

Lynx peut être utilisé pour accéder à des informations présentes sur le World Wide Web, ou pour construire des systèmes d'information ayant pour vocation principale de servir à des accès locaux. Par exemple, *Lynx* a été utilisé pour construire plusieurs systèmes d'information de campus (Campus Wide Information Systems, CWIS). De plus, *Lynx* peut être utilisé pour construire des systèmes isolés à l'intérieur d'un même LAN.

11.1.2 Lancement

Au démarrage, *Lynx* chargera tout fichier local ou toute URL distante spécifié(e) sur la ligne de commandes. Pour obtenir de l'aide sur les URL, appuyez sur « **?** » ou « **H** » quand vous êtes dans *Lynx*. Suivez ensuite le lien intitulé « Help on URLs. » (Aide sur les URL).

11.1.3 Variables d'environnement

PROTOCOLE_proxy
: *Lynx* supporte l'utilisation de serveurs proxy qui peuvent agir comme des passerelles garde-barrières et des serveurs de cache. Les protocoles utilisés par *Lynx* (http, ftp, gopher, etc) trouvent leur représentants respectifs en définissant des variables d'environnement de la forme PROTOCOLE_proxy (à savoir : http_proxy, ftp_proxy, gopher_proxy, etc.), en « http://some.server.dom:port/ ».

WWW_HOME
: Cette variable, si elle est définie, surchargera l'URL de démarrage par défaut spécifiée dans l'un des fichiers de configuration de *Lynx*.

Pour toutes les autres variables d'environnement, voyez le man.

11.1.3 Commandes de base et naviguation :

Principe :
: Le curseur se déplace de lien en lien

Flêche bas : Passer au lien suivant, à défaut écran suivant.
Flêche haut : Revenir au lien précédent, à défaut remonte d'un écran.
Flêche à droite : Suivre le lien en cours
Flêche à gauche : Retour à la page précédente
Barre d'espacement : Ecran suivant
1 : Fin du document
3 : Ecran suivant

7 : Début du document
9 : Ecran précédent
Formulaires :

Les boutons sont traîtés par Lynx comme des liens (flêche à droite ou Entrée). Pour le bouton d'envoi, on peut aussi actionner **"x"** ou **"d"**. Les cases à choisir sont entre **"[ ]"**, les boutons radio entre **"( )"**. Dans les premières, le choix fait apparaître un **"x"**, dans les seconds, un **" * "**. Lynx sait faire apparaître les listes d'options ; pour les zones de saisie, jouer du clavier (flêche haut-bas, Tabulation ou Entrée), le curseur ne passera pas de lui-même à une autre zone.

Sortir de Lynx :

"q" puis **"y"**.

Entrer un URL pour se connecter à un serveur, ou afficher un fichier ou répertoire :

"g" puis entrer l'URL. Flêche haut/Flêche bas font réafficher les précédents URL utilisés.

Réafficher la page en cours

Ctrl+R

Interrompre la commande en cours :

"[Ctrl] + c".

Liste de toutes les commandes :

"k", donne la "keymap", table des options clavier.

Afficher l'aide :

"?" ou **"h"** donne le mode-opératoire.

Récupérer / Imprimer un fichier

"d" (download) sur le lien qui pointe vers ce fichier. On peut choisir ensuite entre sauvegarder le fichier et l'afficher. **"p"** (print) donne seulement le choix entre sauvegarder et envoyer par mail la page en cours, car Lynx sous Windows n'autorise pas l'impression directe.

Rechercher une chaîne de caractères :

"/" puis entrer la chaîne de caractères recherchée.

Continuer la recherche de chaîne de caractères :

"n" pour rechercher plus loin ; et **"N"** pour poursuivre la recherche en arrière.

Enregistrer une page dans les signets :

"a" puis **"d"** (document en cours) ou **"l"** (lien en cours). Entrer ensuite un libellé.

Utiliser les signets :

"v" puis choisir dans la liste. Les signets sont dans le fichier lynx_bookmarks.htm, qui peut être édité manuellement.

Afficher la liste des URL déjà visités :

"V" (majuscule)

Masquer/afficher les "[Inline]"

" [" (crochet ouvert) : cette touche permet de faire disparaître (ou alternativement réapparaître) les fâcheux INLINE. Ceux-ci correspondent aux images (*tag* <img>) sans attribut "alt", qui recouvrent beaucoup de pages web. En l'occurence le fichier de configuration doit contenir la valeur : MAKE_PSEUDO_ALTS_FOR_INLINE: TRUE. Voir aussi le paramètre-pseudo_inline.

Afficher la liste des liens ou "références" de la page :

"l" (list), option utile, on trouve des pages avec des liens CACHES (sans donnée affichée).

Afficher la source :

"\" l'antislash permet de passer alternativement de l'affichage normal à l'affichage du fichier source avec son balisage html.

Gérer l'accès aux Cookies :

"[Ctrl] + k". La liste des domaines (exemple : ad.doubleclick.net) ayant fait l'objet d'une

action particulière apparaît et le statut de chacun d'entre eux peut être modifié (**A**=toujours accepter, **V**=toujours refuser, **P**=demander confirmation, **D**=supprimer de la liste, **C**=annuler). Voir dans *lynx.cfg* les différentes options de gestion des cookies.

Menu d'options de paramétrage :

"o" :

Afficher la "bannière"

" # " : fera remonter à la barre d'outil correspondant aux balises LINK avec attributs REL ou REV (partie HEAD d'une page HTML).

11.2 wget

Extraits du man : (man wget):

wget - Un récupérateur réseau non interactif

11.2.1 SYNOPSIS

wget [*option*]... [*URL*]...

11.2.2 DESCRIPTION

GNU Wget est un programme non interactif de téléchargement de fichiers depuis le Web. C'est un logiciel libre. Il supporte les protocoles HTTP, HTTPS et FTP ainsi que le téléchargement au travers des proxies HTTP.

Wget est non interactif c'est-à-dire qu'il peut travailler en arrière-plan, sans intervention de l'utilisateur. Ceci vous permet de lancer un téléchargement et de vous déconnecter du système, laissant Wget finir le travail. En revanche, la plupart des navigateurs Web requièrent la présence constante de l'utilisateur, ce qui est particulièrement pénible lorsqu'on transfère beaucoup de données.

Wget peut suivre les liens des pages HTML et XHTML et créer une copie locale de sites web distants, en récréant complètement la structure du site original. Ceci est parfois désigné sous le nom de « téléchargement récursif ». En faisant cela, Wget respecte le standard d'exclusion de robots (*/robots.txt*). Wget peut aussi convertir les liens dans les fichiers HTML téléchargés pour la consultation locale.

Wget a été conçu pour être robuste en dépit des connexions réseaux lentes ou instables : si un téléchargement échoue suite à un problème réseau, il réessayera jusqu'à ce que l'intégralité du fichier soit téléchargée. Si le serveur supporte la reprise, il lui demandera de reprendre là où le téléchargement s'est interrompu.

11.2.3 Utilisation simple

Si vous voulez juste télécharger une URL. Tapez juste :

```
wget http://fly.srk.fer.hr/
```

Mais que se passera-t-il si la connexion est lente et le fichier gros ? La connexion sera probablement interrompue avant que l'intégralité du fichier ne soit rapatriée. Dans ce cas, Wget essayera de re-télécharger le fichier jusqu'à ce qu'il y arrive ou qu'il dépasse le nombre de tentatives par défaut (c'est-à-dire 20). Il est facile de changer le nombre de tentatives par 45, pour s'assurer que l'intégralité du fichier arrivera :

```
wget --tries=45 http://fly.srk.fer.hr/jpg/flyweb.jpg
```

.

Maintenant laissons Wget travailler en arrière-plan, et écrire sa progression dans le fichier de log. C'est fatigant de taper **--tries**, aussi nous utiliserons **-t**.

```
wget -t 45 -o log http://fly.srk.fer.hr/jpg/flyweb.jpg &
```

L'esperluette à la fin de la ligne envoie Wget à l'arrière plan. Pour avoir un nombre infini de tentatives, utilisez **-t inf**.

.

L'usage du FTP est aussi simple. Wget tiendra compte du login et du mot de passe.

```
wget ftp://gnjilux.srk.fer.hr/welcome.msg
```

Si vous spécifiez un répertoire, Wget téléchargera l'inventaire du répertoire et le transformera en document HTML. Essayez :

```
wget ftp://prep.ai.mit.edu/pub/gnu/ links index.html
```

12. Divers

Accéder aux partitions linux depuis windows

Explore2fs est un programme GPL qui permet de lire ses partitions Linux de type Ext2 et/ou Ext3 depuis Windows.

Voili voilou cette petite doc se termine ici. J'espère que sa lecture ne vous a pas donné de trop grosses migraines, mais surtout qu'elle vous sera utile!

Un grand merci particulier aux auteurs des nombreux sites où j'ai pompé ces articles, entres autres :

- http://www.linux-france.org/article/debutant/commentfaire/commentfaire.html#toc8
- www.lisa.univ-paris12.fr/linux/linux_commandes.htm
- http://dominique.guebey.club.fr/tekno/lynx/
- http://www.trustonme.net
- http://people.via.ecp.fr/%7Ealexis/formation-linux/formation-linux.html