Petit Guide à l'Usage des Débutants UNIX

"P'tit GUDU"

v. 1.1.0

Thibaut Varène

Le 5 avril 2007

Table des matières

Introduction

Avertissement

Attention, lecteur aventureux ! Le présent document que tu t'apprêtes à parcourir n'est pas (encore) un document de référence. Pour reprendre une formule qui n'est pas de moi, je dirais qu'il *"n'engage que son lecteur (toi), et donc pas son auteur (moi)"*. J'espère qu'il ne comporte pas trop d'erreurs, mais si tu en découvrais, n'hésite pas à me contacter !

Ce document n'est pas totalement abouti, loin s'en faut. Cependant je pense (et j'espère) qu'il te sera utile, car il s'efforce de regrouper un maximum de réponses aux questions les plus fréquemment posées par les nouveaux arrivants au Groupe ESIEE, (qui regroupe, rappelons-le pour ne pas faire de jaloux [1], l'ESIEE, l'ESTE et l'ISTM), tous niveaux confondus en matière de compétences informatiques.

Présentation

Les objectifs de ce guide sont simples et vastes à la fois. Il s'agit de combler un vide en matière de documentation au Groupe ESIEE. En effet, jusqu'à présent, il n'existait pas de document réellement complet regroupant la plupart des informations dont les nouveaux élèves qui découvrent le parc informatique *UNIX* de l'école ont besoin. En outre, il se veut accessible quelque soit le niveau (ou presque). Un débutant complet, même si certaines notions lui échappent, doit pouvoir manipuler les machines de l'école en lisant ce guide.

Tout à commencé un jour de 2003 où un professeur m'a demandé de donner un TD à des élèves de première année pour leur expliquer les bases du système informatique de l'école, afin qu'ils ne perdent pas de temps avec ce qui devrait être de la routine, et puissent travailler efficacement avec le matériel mis à leur disposition.

J'espère que le résultat est à la hauteur de ces espérances, et qu'il sera utile à de nombreux étudiants au sein du Groupe ESIEE.

Copyright et licence

[1] Note bien que dans la suite, j'emploierai toujours le terme "ESIEE" au sens large.

Chapitre 1

Premier contact

Bienvenue à toi Ô lecteur téméraire ! Tu as passé l'épreuve de la couverture et de l'introduction, tu es donc prêt à affronter le monde obscur des systèmes *UNIX*. N'aie nulle crainte, je vais guider tes premiers pas grâce au guide que tu tiens entre tes mains. Inutile de te dire que tu ne dois jamais t'en séparer, c'est la clef de ta survie dans cette jungle impitoyable !

1.1 Prérequis

Pour simplifier les choses, voici quelques hypothèses de départ :
- On est à l'ESIEE, pas à Tombouctou, donc on part du principe qu'on utilise les machines qui s'y trouvent.
- On appellera "*PC*" les ordinateurs situés à peu près partout, fonctionnant sous *Linux* (ou *Windows*®©™, mais cela ne nous regarde pas) et utilisant l'environnement *KDE*.
- Si tu veux utiliser un *PC* sous *Linux* alors qu'il fonctionne sous *Windows*®©™, il faut le redémarrer. C'est dit, on n'y reviendra plus.
- Bien entendu tu as lu et signé la *Charte Informatique* et récupéré ton *login* et *password*.
- Enfin, comme on ne peux pas faire un guide généraliste sur l'usage des ordinateurs, (ça pèserait trop lourd) il est possible que certains termes nécessitent quelques explications... N'hésite pas à demander autour de toi (étudiants d'années supérieures et profs) si tu as des questions.

1.2 Ouvrir une session

Commençons par le commencement. La première chose à faire pour entamer le dialogue avec une machine sous *Linux*, c'est d'*ouvrir une session*, on dit aussi parfois *se logger* [1], d'après l'anglais *to log in*.

1.2.1 La manip'

Si tout se passe bien l'écran doit afficher une jolie boite avec un message du style : "**Bienvenue sur ...**", et un champ pour taper le *login* et le *password*. Sinon, secoue la souris, ça devrait s'arranger. Si ça ne s'arrange pas allume l'écran, et s'il ne se passe toujours rien, allume l'ordinateur !

La manip' à proprement parler est simple : entre ton *login* (alias *nom d'utilisateur*) et ton *password* (alias *mot de passe*) et valide en appuyant sur `[Entrée]`.

[1] Prononcer "*loguer*". Par convention j'écrirai ce mot qui n'existe pas dans la langue française avec deux "g" pour ne pas confondre avec le verbe "*se loger*".

Et voila ! Si tout va bien tu as maintenant un nouvel affichage, avec une barre en bas de l'écran. Sinon la machine a dû te dire que tu t'es planté, alors recommence !

1.2.2 Explication

Il y a beaucoup de monde à l'ESIEE, et chaque ordinateur peut être utilisé par n'importe qui. Cependant il ne faut pas que monsieur X puisse accéder aux données de mademoiselle Y, qui n'a pas forcément envie d'avoir la même organisation que le professeur Z pour ranger ses documents par exemple. Alors pour arranger les choses, on doit *s'identifier* sur l'ordinateur sur lequel on veut travailler, *via* l'opération précédente, afin de :

1. Prouver que tu es bien toi-même (si si, ça a l'air bête comme ça et pourtant...)
2. Permettre à la machine de récupérer toutes les informations te concernant : tes *préférences*, mais aussi tous tes documents, en bref, ton *environnement de travail*.

Il faut bien comprendre le lien qui existe entre ton *login/password*, et ton *compte d'utilisateur*, c'est-à-dire l'ensemble des ressources (voir chapitre 3 page 27) dont tu disposes sur toutes les machines de l'école : espace disque et son contenu, accès aux services d'impression, etc... C'est l'association des deux premiers qui permet à la machine de retrouver le second.

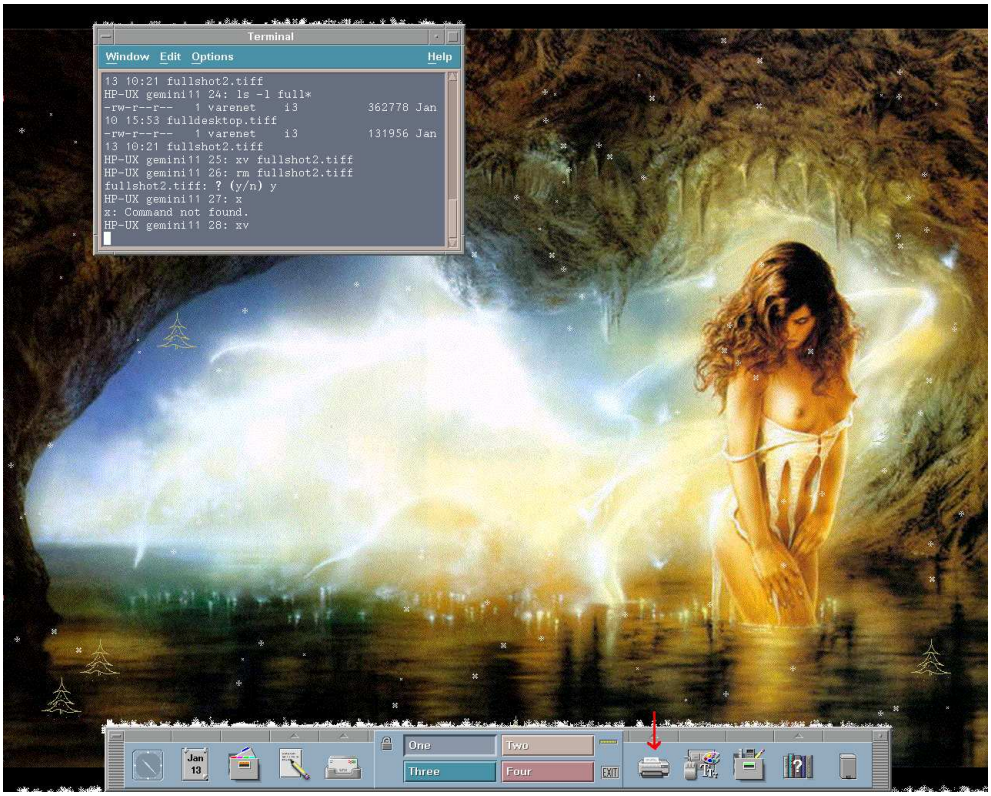
1.3 L'interface graphique

C'est un bon début : te voilà enfin "loggé" sur la machine. Un monde nouveau s'offre à toi ! Ce que tu vois maintenant est ton *bureau*, au sens informatique du terme. Tu peux y effectuer différents réglages, que je ne détaillerai pas ici.

Note cependant que les boutons de la souris ont tous une action différente, en particulier, lorsque tu sélectionnes du texte, le bouton du milieu peut le "*coller*" n'importe où. C'est une forme rapide et pratique du *copier/coller*. Le bouton droit, lui, affiche généralement un menu contextuel [2].

Voici un exemple de ce à quoi peut ressembler un *bureau* dans l'environnement graphique *CDE* de *HP-UX*, qu'on utilisait encore à l'époque reculée où ce guide a été écrit, et que je laisse ici pour le plaisir des yeux :

[2] Il s'agit d'un petit menu dont le contenu varie en fonction du *contexte*.



Il est bien évident qu'au départ le tiens ne ressemblera pas à ça [3].

Enfin et pour finir, savoir se "*délogger*" est une bonne chose. Tu vas voir, ce n'est pas trop dur : sous *Linux* il te suffit de cliquer sur le gros "K" blanc situé tout à gauche de la barre du bas (voir l'image page 9), puis sur le choix "Quitter l'environnement" (parfois "Logout") qui apparaît alors. Et voila !

[3] Ne t'inquiète pas, je vais te donner quelques astuces pour parvenir à un résultat similaire en annexe A page 43.

Chapitre 2

Affronter le Term

Jusqu'ici tout allait bien (ou presque), tout était facile, des images et des icônes partout, et une souris pour tout piloter. Oui mais voilà, c'était bien trop beau pour durer. Il va maintenant falloir mettre les mains dans le cambouis !

2.1 Qu'est-ce que le Term

Le *Term* est un barbarisme désignant le *terminal* présent sur tout système *UNIX*. Dans ce *terminal* s'exécute un *shell*[1]. Pour simplifier on confondra les deux. C'est grâce à lui que tu vas pouvoir réellement piloter l'ordinateur jusque dans les moindres détails. Il est donc très important que tu en comprennes le mieux possible le fonctionnement, et les principales commandes. Je vais tout t'expliquer ici.

2.1.1 Ouvrir un terminal

Il existe beaucoup de possibilités pour ouvrir un *terminal* dans l'environnement graphique *KDE*. Voici une des plus simples, je te laisse libre de découvrir les autres !

Pour démarrer un *terminal* avec *KDE*, il te suffit de cliquer sur l'icône représentant une fenêtre noire avec un coquillage dessus, située à gauche dans la barre d'outils qui se trouve en bas de l'écran, comme sur l'image qui suit.

2.1.2 Anatomie d'un terminal

La fenêtre qui doit normalement se présenter à toi présente quelques particularités qu'il te faut connaître. En l'occurrence, il y a deux parties à différencier : le *prompt* (on dit parfois *invite* en français), et la zone de saisie.

Reconnaître les deux n'est pas très sorcier : le *prompt*, c'est le petit texte que t'affiche le *terminal* lorsque tu le lances. Par exemple :

```
idefix 1:
```

[1] En fait, le *terminal* est l'application dans laquelle s'exécute un *shell* quelconque. Pour simplifier, le *terminal*, c'est la fenêtre, et le *shell*, son contenu, qu'on appelle aussi en français *interpréteur de commandes*

Son contenu est réglable [2], donc ne t'inquiète pas si le tiens ne ressemble pas à l'exemple. La zone de saisie, quant à elle, s'étend après le *prompt*.
Dans la suite, mon *prompt* ressemblera à cela :

```
[varenet@idefix ~]>
```

Dans toute la suite du document, tu taperas les **commandes** indiquées dans la zone de saisie, suivies de la touche `[Entrée]` qui permet de les valider. Après chaque commande tapée, validée et terminée, une nouvelle ligne avec un *prompt* doit apparaître.

☞	Il existe une fonctionnalité du *terminal* qui est *très* pratique. Il s'agit de ce que l'on appelle en jargon technique *l'autocomplétion*, la possibilité de finir automatiquement une commande ou un nom de fichier par exemple. Pour bien comprendre ce que c'est, facile : tape les premières lettres d'une commande ou d'un nom de fichier et appuie sur la touche `[Tab]`. Magie ! Le *terminal* finit le mot à ta place. A condition qu'il n'y ai pas *ambiguïté*, c'est-à-dire pas plusieurs commandes ou noms qui commencent pareil.

2.1.3 Fermer un terminal

Fermer un *terminal* est encore plus simple que l'ouvrir : il suffit de taper la commande "**exit**", ou encore d'utiliser la combinaison de touches `[Ctrl-D]`.

[2] A l'usage tu verras que tu peux obtenir des information très utiles dans le prompt. L'exemple donné indique ici l'utilisateur, le nom de la machine et le répertoire courant, voir Annexe B page 47.

2.2 Les principales commandes

Je vais détailler ici les commandes "vitales" qu'il faut connaître pour pouvoir gérer son *compte utilisateur*. Petits prérequis : tout d'abord, toute commande *UNIX* est de la forme **commande -option(s) autre(s)_argument(s)**.

ATTENTION!

Toutes les commandes sont sensibles à la cAsSe (majuscules/minuscules), que ce soit pour le nom de la commande elle-même, ou ses arguments.

☞ En outre, les accents ne sont pour ainsi dire *pas* gérés correctement, donc il vaut mieux les éviter au maximum, en particulier dans les noms de fichiers.

Quelques définitions :
- Un *argument* est une combinaison de caractères formant parfois un mot (et oui !) tapée après le nom de la commande, et séparée par un espace. Une seule commande peut avoir plusieurs *arguments*.
- Une *option* est un argument d'un genre particulier. Elle commence généralement par un tiret "**-**" (ou deux : "—"), et sert à modifier le comportement d'une commande.
  En notation explicative pour une commande dans la suite de ce guide, les options sont représentées entre crochets `[ ]`, et les autres arguments (comme les noms de fichier) entre crocodiles simples `< >`. Il ne faut bien entendu pas taper ces crochets ou crocodiles lorsque tu utilises un argument. Tu peux en général combiner plusieurs *options* d'affilées. Par exemple, "**-s -k**" devient "**-sk**".
- Un *chemin* est une liste successive de nom de répertoires [3].

Enfin sache qu'un bon nombre des commandes qui suivent partagent des options identiques, mais pour plus de clarté je les indiquerai à chaque fois.

2.2.1 man

C'est probablement la première commande à connaître, tant elle est utile. Son nom est l'abréviation de l'anglais *"MANual"*. Il s'agit en effet d'un utilitaire permettant d'afficher le manuel de n'importe quelle commande du système. Donc, c'est la toute première source d'information à consulter, avant d'aller poser une question à ton voisin d'en face ! De même, si tu cherches d'avantage d'informations que celles fournies dans ce guide, essaye d'abord la commande **man**.

Usage

man <commande>
Son usage est très simple : on tape **man** suivi du nom de la commande sur laquelle on cherche de l'aide. Pour faire défiler ligne à ligne le texte, tu peux utiliser la touche `[Entrée]`, ou pour faire défiler page par page, la touche `[Espace]`. Pour quitter **man**, tape simplement `[Q]`.

Exemple :

```
[varenet@Tatooine ~]$ man man
Reformatting man(1), please wait...

man(1)                          Manual pager utils                          man(1)

NAME
       man - an interface to the on-line reference manuals
```

[3] Les répertoires sont les "dossiers" dans lesquels tu peux ranger tes documents. J'y reviens au paragraphe 2.2.7 page 15.

```
SYNOPSIS
       man  [-c|-w|-tZHT  device]  [-adhu7V]  [-i|-I]  [-m  system[,...]]  [-L
       locale] [-p string] [-M path] [-P pager]  [-r  prompt]  [-S  list]  [-e
       extension] [[section] page ...] ...
```

2.2.2 ls

L'outil **ls**, comme *"LiSt"* en anglais, est un un outil permettant de lister le contenu de répertoires.

Usage

ls [options] <élément>
Il est pourvu d'un certain nombre d'options, parmi lesquelles les plus courantes sont :

`-l` : (*long*) Affiche les *attributs*[4] des éléments. (comme les droits d'accès).

`-k` : (*kilo*) Affiche les tailles des éléments en Ko (par défaut la taille est affichée en octets).

`-h` : (*human*) Affiche les tailles sous forme facilement compréhensible (Méga-octet, Kilo-octet, selon le besoin...).

`-t` : (*time*) Trie par date de modification (les plus récents en premier)

`-a` : (*all*) Liste les éléments dont le nom commence par un point (éléments dits *invisibles*)

`-d` : (*directory*) Les répertoires sont affichés comme des fichiers (leur contenu n'est pas listé) et les liens symboliques[5] ne sont pas suivis (utile pour afficher leurs *attributs*).

`-r` : (*reverse*) Ordre lexicographique inverse. Selon le critère de tri : ordre alphabétique inverse ou chronologique inverse.

Enfin il faut savoir que le délimiteur de nom de répertoire est le "`/`" (*slash*), et que "`.`" représente le répertoire actuel, et "`..`" représente le répertoire supérieur. On peut cascader les "`../`" pour remonter plusieurs répertoires à la fois.

```
[varenet@Tatooine ~/ESIEE]$ ls
IN301          miniguide.dvi  miniguide.tex
miniguide.aux  miniguide.log  miniguide.toc
[varenet@Tatooine ~/ESIEE]$ ls -l IN301/
total 60
drwxr-xr-x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxr-xr-x    1 varenet  users        1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Dec 25 18:07 TP4
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-r--r--    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
[varenet@Tatooine ~/ESIEE]$ ls -ld IN301/
drwxr-xr-x    4 varenet  users        2048 Dec 25 17:03 IN301/
[varenet@Tatooine ~/ESIEE/IN301]$ ls -lh rapport-tp1.tex
-rw-r--r--    1 varenet  users         26K Dec  1 22:36 rapport-tp1.tex
```

Attention!

Tu l'auras peut-être remarqué, tu ne peux pas afficher la taille occupée par tout un dossier avec **ls**. Il existe un utilitaire pour calculer les tailles, c'est **du** (voir paragraphe 2.2.10 page 16).

[4] Tu y verras plus clair à la section 2.3 page 23 consacrée aux permissions d'accès.
[5] Expliqués paragraphe 2.2.11 page 17.

2.2.3 cd

Cette commande permet de changer de répertoire. Ce sont les initiales de *"Change (working) Directory"*. Par contre, elle ne permet pas de lire un compact disque ! Sans argument, cette commande retourne au répertoire de départ (*home directory*).

Usage

cd <répertoire>
C'est une des commandes les plus basiques qui soient :

```
[varenet@Tatooine ~/ESIEE]$ cd IN301/
[varenet@Tatooine ~/ESIEE/IN301]$
```

pwd

Complémentaire de **cd**, **pwd**, alias *"Print Working Directory"*, affiche le *chemin* complet du répertoire courant :

```
[varenet@Tatooine ~/ESIEE/IN301]$ pwd
/home/varenet/ESIEE/IN301
```

2.2.4 cp

Cet outil permet d'effectuer des copies de fichier. C'est l'abréviation du mot anglais *"CoPy"*.

Usage

cp [options] <original> <copie>
Parmi ses options :

`-r` : (*recursive*) Permet de copier des répertoires récursivement.

`-i` : (*interactive*) Demande confirmation de toutes les actions incertaines (potentiellement dangereuses).

`-f` : (*force*) Force les actions (méfie-toi, c'est dangereux).

`-v` : (*verbose*) Détaille chacune des actions effectuées.

Si on passe un nom de répertoire en argument `<copie>`, l'original est copié dans ce répertoire avec son nom d'origine. On peut aussi passer un nom de répertoire suivi d'un nouveau nom, à ce moment là c'est ce nouveau nom qui sera utilisé pour la copie dans le répertoire cible. Voici quelques exemples simples :

Copie simple d'un fichier :

```
[varenet@Tatooine ~/ESIEE/IN301]$ cp rapport-tp1.tex toto
[varenet@Tatooine ~/ESIEE/IN301]$ ls
CVS  IN301TP3.sh  TP4  rapport-tp1.tex  rapport-tp2.tex  rapport-tp3.tex
toto
```

Copie de tout un répertoire :

```
[varenet@Tatooine ~/ESIEE/IN301]$ ls ../
IN301          miniguide.dvi  miniguide.tex
miniguide.aux  miniguide.log  miniguide.toc
[varenet@Tatooine ~/ESIEE/IN301]$ cp -r TP4 ../TP-sauvegarde
```

```
[varenet@Tatooine ~/ESIEE/IN301]$ ls ../
IN301          miniguide.aux  miniguide.log  miniguide.toc
TP-sauvegarde  miniguide.dvi  miniguide.tex
```

2.2.5 mv

Cet outil permet de déplacer/renommer des fichiers. C'est l'abréviation du mot anglais *"MoVe"*.

Usage

mv [options] <original> <copie>

`-i` : (*interactive*) Demande confirmation de toutes les actions incertaines (potentiellement dangereuses).

`-f` : (*force*) Force les actions (méfie-toi, c'est dangereux).

`-v` : (*verbose*) Détaille chacune des actions effectuées.

De même que pour **cp**, les règles concernant l'argument `<copie>` s'appliquent aussi.

```
[varenet@Tatooine ~/ESIEE/IN301]$ mv toto TP4/
[varenet@Tatooine ~/ESIEE/IN301]$ ls
CVS  IN301TP3.sh  TP4  rapport-tp1.tex  rapport-tp2.tex  rapport-tp3.tex
[varenet@Tatooine ~/ESIEE/IN301]$ ls TP4/
CVS       ascii.1  include                 rapport-tp4.tex  toto
Makefile  ascii.c  preprocesseur-exemple.c  sources          xxx.c
```

2.2.6 touch

touch permet de créer un fichier vide (de 0 octet). Il permet aussi de mettre à jour la date de dernière modification d'un fichier sans l'ouvrir.

Usage

touch <fichier>

```
[varenet@Tatooine ~/ESIEE/IN301]$ ls
CVS  IN301TP3.sh  TP4  TP5  rapport-tp1.tex  rapport-tp2.tex  rapport-tp3.tex
[varenet@Tatooine ~/ESIEE/IN301]$ touch toto
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 62
drwxr-xr-x    2 varenet  users        2048 Jan 11 17:24 CVS
-rwxrwxr-x    1 varenet  palinux      1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Jan 11 17:24 TP4
drwxr-xr-x    3 varenet  users        2048 Jan 11 17:24 TP5
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-------    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
-rw-r--r--    1 varenet  users           0 Jan 12 16:47 toto
```

2.2.7 mkdir

On se sert de **mkdir** pour créer des répertoires. C'est l'abréviation de *"MaKe DIRectory"*. Les répertoires, par oppostion aux fichiers de données par exemple, sont des "conteneurs". Tu peux y regrouper différents documents. C'est bien pratique pour les trier !

Usage

mkdir [options] <répertoire>

`-p` : (*parent*) Crée les répertoires intermédiaires non existants passés en argument.

`-v` : (*verbose*) Détaille chacune des actions effectuées.

ATTENTION!

Si les répertoires intermédiaires n'existent pas et que tu n'utilises pas l'option "`-p`", **mkdir** échoue lamentablement.

☞ Sous *Linux* il existe un équivalent direct de **mkdir -p** : c'est **mkdirhier**.

```
[varenet@Tatooine ~/ESIEE/IN301]$ mkdir -p TP5/rapport/bidon
[varenet@Tatooine ~/ESIEE/IN301]$ ls -RF TP5
TP5:
rapport/

TP5/rapport:
bidon/

TP5/rapport/bidon:
```

2.2.8 rm

Cet outil efface un élément. C'est l'abréviation de *"ReMove"*.

Usage

rm [options] <éléments>
rm dispose de différentes options, parmi lesquelles :

`-r` : (*recursive*) Permet de supprimer des répertoires récursivement (avec leur contenu).

`-i` : (*interactive*) Demande confirmation de toutes les actions incertaines (potentiellement dangereuses).

`-f` : (*force*) Force les actions (méfie-toi, c'est dangereux).

`-v` : (*verbose*) Détaille chacune des actions effectuées.

```
[varenet@Tatooine ~/ESIEE/IN301]$ rm toto
[varenet@Tatooine ~/ESIEE/IN301]$ rm -rf ../TP-sauvegarde/
[varenet@Tatooine ~/ESIEE/IN301]$ ls . ..
.:
CVS  IN301TP3.sh  TP4  TP5  rapport-tp1.tex  rapport-tp2.tex  rapport-tp3.tex

..:
IN301          miniguide.dvi  miniguide.tex
miniguide.aux  miniguide.log  miniguide.toc
```

2.2.9 rmdir

Je ne t'étonnerai sans doute pas si je te dis que cette commande supprime des répertoires ; mais attention : elle ne fonctionne que sur des répertoires **vides**. C'est l'abréviation de *"ReMove DIRectory"*.

Usage

rmdir [options] <répertoire>

`-p` : (*parent*) Supprime les répertoires intermédiaires existants passés en argument.

`-v` : (*verbose*) Détaille chacune des actions effectuées.

De même que **mkdir**, il existe l'option "`-p`" qui supprime tous les répertoires intermédiaires s'ils sont vides.

```
[varenet@Tatooine ~/ESIEE/IN301]$ rmdir TP5/rapport/bidon/
[varenet@Tatooine ~/ESIEE/IN301]$ ls TP5/rapport/
[varenet@Tatooine ~/ESIEE/IN301]$ rmdir -p TP5/rapport/
[varenet@Tatooine ~/ESIEE/IN301]$ ls
CVS  IN301TP3.sh  TP4  rapport-tp1.tex  rapport-tp2.tex  rapport-tp3.tex
```

2.2.10 du

du, comme *"Disk Usage"*, permet de calculer la place occupée sur un *système de fichier* [6] par un ou plusieurs éléments. C'est évidemment une commande très utile !

Usage

du [options] <chemin>
Par défaut **du** indique les tailles de tous les éléments du répertoire courant. Les options les plus utiles sont :

`-s` : (*silent*) N'affiche que le total final.

`-k` : (*kilo*) Affiche les tailles en Ko.

`-h` : (*human*) Affiche les tailles dans un format facilement compréhensible. (voir la même option pour **ls** : page 12).

```
[varenet@Tatooine ~/ESIEE/IN301]$ du TP4/
8 TP4/CVS
8 TP4/include/CVS
12 TP4/include
8 TP4/sources/CVS
12 TP4/sources
84 TP4
[varenet@Tatooine ~/ESIEE/IN301]$ du -sh TP4/
84K TP4
```

[6] Pour simplifier, disons qu'un système de fichier représente n'importe quel endroit de l'ordinateur où se trouvent des documents.

2.2.11 ln

Cette commande permet de créer des liens. C'est l'abréviation de *"LiNk"*. Il existe deux types de liens : les liens symboliques, qui peuvent être vus comme des "pointeurs" vers d'autres fichiers (comme les "*Raccourcis*" de *Windows*®©^{TM}, ou les "*Alias*" de *MacOS*®) ; et les liens physiques, qui sont des relations établies directement au niveau du *système de fichier*, avec un certain nombre d'implications : un lien physique n'est pas différenciable de son original, tout changement sur le lien ou l'original se répercute immédiatement sur l'autre. *Tout se passe comme si l'original et son lien physique étaient en fait un seul et même fichier, accessibles sous différents noms.* Il est bien entendu possible de créer plusieurs liens (symboliques ou physiques) pour le même original. En général on utilise plutôt des liens symboliques.

Usage

ln [options] <original> <lien>

`-s` : (*symbolic*) Crée des liens symboliques.

`-i` : (*interactive*) Demande confirmation de toutes les actions incertaines (potentiellement dangereuses).

`-f` : (*force*) Force les actions (méfie-toi, c'est dangereux).

`-v` : (*verbose*) Détaille chacune des actions effectuées.

> ATTENTION!
>
> Par défaut **ln** crée des liens physiques.

```
[varenet@Tatooine ~/ESIEE/IN301]$ ls -lF TP4/
total 56
drwxr-xr-x    2 varenet  users        2048 Dec 25 18:07 CVS/
-rw-r--r--    1 varenet  users         538 Dec 20 12:40 Makefile
-rw-r--r--    1 varenet  users        1102 Dec 20 12:27 ascii.1
-rw-r--r--    1 varenet  users        1863 Dec 20 12:27 ascii.c
drwxr-xr-x    3 varenet  users        2048 Dec 25 17:03 include/
-rw-r--r--    1 varenet  users         248 Dec 20 12:27 preprocesseur-exemple.c
-rw-r--r--    1 varenet  users        8990 Dec 25 18:06 rapport-tp4.tex
drwxr-xr-x    3 varenet  users        2048 Dec 25 17:06 sources/
-rw-r--r--    1 varenet  users       26648 Dec 26 00:45 toto
-rw-r--r--    1 varenet  users         707 Dec 20 12:40 xxx.c
[varenet@Tatooine ~/ESIEE/IN301]$ ln -s TP4/toto riri
[varenet@Tatooine ~/ESIEE/IN301]$ ln TP4/toto fifi
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 90
drwxr-xr-x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxr-xr-x    1 varenet  users        1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Dec 26 00:58 TP4
-rw-r--r--    2 varenet  users       26648 Dec 26 00:45 fifi
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-r--r--    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
lrwxrwxrwx    1 varenet  users           8 Dec 26 01:09 riri -> TP4/toto
```

On voit bien ici la différence entre un lien symbolique et un lien physique. Les liens symboliques sont représentés très clairement avec une petite flèche indiquant le chemin vers l'original... Le lien physique a lui tous les *attributs* de son original, notamment taille et date de création.

☞	Les liens symboliques vers des répertoires ne sont pas des répertoires eux-mêmes. On utilise donc **rm** pour les supprimer.

2.2.12 find

Celui-ci, tu vas très vite apprendre à ne pas l'oublier, tellement il va te rendre des services ! Cet utilitaire très puissant permet en effet d'effectuer des recherches sur un système de fichier. Il dispose de tant d'options et de possibilités d'utilisation qu'on pourrait écrire un livre à son sujet. Je vais te donner ici celles qui sont probablement les plus utiles, mais je te recommande vivement d'aller jeter un œil à la page de **man** correspondante.

Usage

find <chemin> [options]

ATTENTION!

Contrairement à la plupart des commandes, **find** reçoit ses options *après* l'argument du chemin dans lequel chercher.

Voici quelques exemples d'options de recherche courantes :

`-name` : Permet d'effectuer des recherches sur le nom.

`-[ac]newer`, `-[ac]time`, `-[ac]min` : Permettent d'effectuer des recherches sur les dates des fichiers.

`-exec` : Permet d'exécuter une commande sur les éléments trouvés.

`-fstype` : Permet de se limiter à un *système de fichier* donné.

`-perm` : Effectue des recherches basées sur les permissions, etc.

La page **man** de find est réputée pour être une des plus riches qui soient en options.

☞	**find** accepte les expressions régulières [7] dans ses options.

```
[varenet@Tatooine ~/ESIEE/IN301]$ find . -name "rapport*"
./TP4/rapport-tp4.tex
./rapport-tp1.tex
./rapport-tp2.tex
./rapport-tp3.tex
```

☞	Ici le *chemin* est ".".

2.2.13 chmod/chown/chgrp

Ces trois utilitaires sont complémentaires : ils permettent de modifier les permissions et possesseurs de fichiers. **chmod** s'occupe des permissions d'accès, **chown** et **chgrp** changent respectivement le possesseur et le groupe des fichiers.

Usage

chmod [options] <permissions> <éléments>
chown [options] <possesseur> <éléments>
chgrp [options] <groupe> <éléments>

[7] Une expression régulière, ou *regexp*, indique un motif récurrent à rechercher. Elle est composée de caractères spéciaux. **man regex** t'en apprendra plus.

Dans certaines versions récentes de **chown**, on peut regrouper les deux dernières opérations en une seule : **chown <possesseur>.<groupe> <éléments>**. Quelques options utiles :

`-R` : (*recursive*) Applique les changements récursivement (à tous les éléments et sous répertoires).

`-f` : (*force*) Force les actions (méfie-toi, c'est dangereux).

`-v` : (*verbose*) Détaille chacune des actions effectuées.

`-h` : Change les liens symboliques passés en arguments (avec `-R`).

☞	Il n'est en général pas possible de changer le possesseur d'un fichier t'appartenant pour le faire appartenir à un de tes copains. **chown** n'est en principe utilisé que par `root`. De même, pour pouvoir changer le groupe d'un fichier, il faut que ton utilisateur fasse partie du groupe choisi [8].

Au final, à l'ESIEE, tu n'utiliseras pratiquement que **chmod** !

```
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 60
drwxr-xr-x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxr-xr-x    1 varenet  users        1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Dec 26 00:58 TP4
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-r--r--    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
[varenet@Tatooine ~/ESIEE/IN301]$ chgrp palinux IN301TP3.sh
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 60
drwxr-xr-x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxr-xr-x    1 varenet  palinux      1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Dec 26 00:58 TP4
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-r--r--    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
[varenet@Tatooine ~/ESIEE/IN301]$ chmod g+w IN301TP3.sh
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 60
drwxr-xr-x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxrwxr-x    1 varenet  palinux      1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Dec 26 00:58 TP4
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-r--r--    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
[varenet@Tatooine ~/ESIEE/IN301]$ chmod 600 rapport-tp2.tex
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 60
drwxr-xr-x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxrwxr-x    1 varenet  palinux      1010 Dec 11 23:54 IN301TP3.sh
drwxr-xr-x    5 varenet  users        2048 Dec 26 00:58 TP4
-rw-r--r--    1 varenet  users       26648 Dec  1 22:36 rapport-tp1.tex
-rw-------    1 varenet  users       15845 Nov 29 09:28 rapport-tp2.tex
-rw-r--r--    1 varenet  users        7638 Dec 11 23:54 rapport-tp3.tex
```

[8] Pour d'évidentes raisons de sécurité,il n'est pas possible d'attribuer un fichier à un autre utilisateur que toi (cela influencerait son *quota*), ou à un groupe dont tu ne fais pas partie. Tu en sauras plus section 2.3 page 23.

2.2.14 quota

La consultation du *quota* (voir page 28) se fait grace à la commande **quota**.

Usage

quota
Cette commande affiche les données sur ton *quota* :

```
[varenet@idefix ~]> quota
Disk quotas for user varenet (uid 4021):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
yfiler:/vol/user
                 16240  204800  204800            1525  102400  102400
```

On voit ici que j'occupe 16 240 kilo-octets sur les 204 800 qui me sont alloués, et que j'ai 1 525 fichiers sur mon compte, sur un maximum autorisé de 102 400.

2.2.15 ps

J'aborde ici succinctement quelques notions de gestion des *processus*. Disons pour simplifier qu'à chaque fois que tu lances un programme sur une machine *UNIX*, l'ordinateur considère qu'il s'agit d'un nouveau *processus*, c'est-à-dire une tâche à accomplir. La commande **ps** te permet d'afficher la liste des *processus* qui sont en train de s'exécuter sur la machine que tu utilises au moment où tu tapes cette commande. Elle peut t'indiquer entre autres le *possesseur* de chaque processus, la date début, le temps CPU [9] et le nom de chaque *processus*.

Usage

ps [options]

`-e` : (*extended*) Demande la liste de toutes les tâches.

`-f` : (*full*) Demande l'affichage de toutes les infos sur chaque tâche.

`-w` : (*wide*) Permet de poursuivre les lignes trop longues sur la ligne suivante.

Tu utiliseras généralement la combinaison des deux options "`-e`" et "`-f`".

```
[varenet@Tatooine ~]$ ps -efw
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0  2002 ?        00:00:10 init
root         7     1  0  2002 ?        00:00:00 [scsi_eh_0]
root        11     1  0  2002 ?        00:00:20 [kjournald]
root        25     1  0  2002 ?        00:00:05 /sbin/devfsd /dev
daemon     123     1  0  2002 ?        00:00:00 /sbin/portmap
root       266     1  0  2002 ?        00:00:02 /usr/sbin/nmbd -D
root       268     1  0  2002 ?        00:00:00 /usr/sbin/smbd -D
bin        271     1  0  2002 ?        00:02:39 /usr/sbin/snmpd -s -l /dev/null
root       279     1  0  2002 ?        00:00:00 /usr/sbin/sshd
root       290     1  0  2002 ?        00:00:02 /usr/sbin/ntpd
root       293     1  0  2002 ?        00:00:00 /usr/sbin/rpc.nfsd
root       295     1  0  2002 ?        00:00:00 /usr/sbin/rpc.mountd
nobody     302     1  0  2002 ?        00:00:00 proftpd (accepting connections)
```

[9] Le temps CPU correspond au temps que la machine à effectivement passé à exécuter une tâche.

```
root       309     1  0  2002 ?        00:00:00 /usr/sbin/cron
root       318     1  0  2002 ?        00:00:00 /usr/bin/kdm
root       328   318  2  2002 ?        04:54:14 /usr/X11R6/bin/X -dpi 100
-nolisten tcp vt7 -auth /var/lib/kdm/authfiles/A:0-biT30X
varenet    370   341  0  2002 ?        00:00:03 /usr/bin/gnome-session
varenet    416     1  0  2002 ?        00:03:39 sawfish --sm-client-id
11c0a8450300010066395490000000005500002 --sm-prefix a8v3swhutc
varenet    429     1  0  2002 ?        00:07:15 xchat --sm-config-prefix
/xchat-mOYnoj/ --sm-client-id 11c0a845030001036847358000000
varenet   7072     1 95  2002 ?        8-19:25:27 ./setiathome -nice 20
varenet  10669     1  0  2002 ?        00:00:09 gv JoliManuelPourLaTeX.ps
varenet   3658     1  0  2002 ?        00:03:09 /usr/lib/mozilla/mozilla-bin
http://slashdot.org/article.pl?sid=02/12/31/149202
varenet   5457     1  0  2002 ?        00:00:19 multi-gnome-terminal
--use-factory --start-factory-server
varenet   5461  5457  0  2002 pts/0    00:00:00 -bash
varenet  25228  5457  0 19:02 pts/3    00:00:00 -bash
varenet  25261 25228  0 19:02 pts/3    00:00:01 ssh mkhppa02
varenet  27923  5457  0 22:28 pts/4    00:00:00 -bash
root     28395     1  0 23:00 tty1     00:00:00 /sbin/getty -f
/etc/issue.linuxlogo 38400 tty1
varenet  28583  5461  0 23:13 pts/0    00:00:00 xdvi.bin -name xdvi gudu
varenet  28680  5457  0 23:19 pts/5    00:00:00 -bash
varenet  28963 27923  2 23:41 pts/4    00:00:25 nedit gudu.tex
varenet  29172 28680  0 23:56 pts/5    00:00:00 ps -efw
```

Et hop ! (Voila comment on remplit une page aussi !) Les titres des colonnes te renseignent sur les informations fournies. Je ne vais pas trop insister là dessus, mais si ça t'intéresse, le manuel est ton ami !

Enfin, sache qu'il existe également la commande **top**, qui te donne sensiblement les mêmes informations, mais en temps réel avec une mise à jour régulière. De même que pour **man**, on utilise `[Q]` pour quitter **top**.

2.2.16 kill

C'est bien de pouvoir lister les *processus*, mais encore faudrait-il savoir comment en "tuer" un s'il correspond par exemple à un programme que tu as créé et que tu ne peux plus contrôler. Pour cela, rien de plus simple. À l'aide de la commande **ps**, que nous venons de voir, récupère le `PID` du *processus* que tu veux tuer. Ensuite, tape **kill <PID>**.

> **Attention!**
>
> Tu ne peux tuer que les processus qui t'appartiennent, c'est-à-dire ceux dont le possesseur (`UID`) correspond à ton *login*.

Usage

kill [signal] <PID>
L'option `[signal]` permet d'envoyer au programme désigné par `<PID>` un signal autre que `TERM` ("terminaison", c'est-à-dire "fin de tâche"). En particulier pour les programmes vraiment récalcitrants, on peut utiliser **kill -9 <PID>**. Ici on envoie le signal `KILL`, qui demande directement à l'ordinateur de mettre fin au *processus* concerné, sans préavis. Autant dire que ce n'est pas sans risques.

Attention!

L'option `[signal]` est à manipuler avec précaution.

Enfin, **man signal** et **man kill** t'en apprendront plus. Petit exemple : si, dans la liste qui précède, si je veux tuer **setiathome**, je vais faire :

```
[varenet@Tatooine ~]$ kill 7072
```

Pour finir, il faut que tu connaisses l'existence de **xkill**. C'est une sorte de **kill** graphique, c'est-à-dire que lorsque tu tapes **xkill** dans un *terminal*, le curseur de ta souris va se transformer en une petite cible, et le *processus* correspondant à la première fenêtre sur laquelle tu vas cliquer sera tué.

2.3 Les permissions d'accès

Il est important de ne jamais perdre de vue que tu travailles sur un *réseau*, et que donc toutes tes données sont accessibles de n'importe où dans l'école. Il faut donc apporter un soin tout particulier dans le choix de ce que tu permets ou ne permets pas aux autres utilisateurs du réseau de faire avec tes fichiers.

2.3.1 Les notions de base

Il faut que tu te familiarises rapidement avec les notions de bases en matière de permissions : Il y a trois catégories pour différencier les utilisateurs sur *Linux* : on parle de *user* (`u`) (le possesseur), de *group* (`g`) (un groupe d'utilisateurs déterminé) et des "*others*" (`o`) (les autres !)

> **Attention!**
>
> Il est très important que tu comprennes bien ces notions pour ne pas avoir de mauvaises surprises dans le futur.

user désigne donc le *propriétaire* de l'élément concerné. C'est celui qui a le droit de modifier les permissions sur l'élément. C'est normalement également celui qui a le plus d'autorisations sur l'élément.

group correspond au groupe d'utilisateurs pour lequel on peut choisir des permissions spécifiques, par opposition à

others qui englobe tous les utilisateurs ne correspondant pas aux deux premières catégories.

Il est donc possible d'attribuer trois niveaux d'autorisations/restrictions à chaque élément.

2.3.2 Un cas d'école

À chacune des catégories mentionnées correspondent certains champs de statut des éléments, tels qu'on peut les voir dans le résultat de la commande **ls -l** :

```
[varenet@Tatooine ~/ESIEE/IN301]$ ls -l
total 60
drwxr-x--x    2 varenet  users        2048 Dec 25 17:03 CVS
-rwxrwx---    1 varenet  palinux      1010 Dec 11 23:54 IN301TP3.sh
```

Le premier champ à observer est celui composé de dix caractères, situé au tout début de chaque ligne : `drwxr-xr-x`. Chaque caractère a une signification particulière, en fonction de sa position et de sa valeur.

- Le premier caractère (ici "`d`") indique le type de l'élément listé : parmi les plus courants : "`-`" représente un *fichier régulier*, tel que tout document, bref, un fichier contenant des données lisibles ; "`d`" représente tu l'auras deviné un répertoire (`d` comme *directory*) ; "`l`" (*link*) représente un lien symbolique...
- Les neuf autres caractères se lisent par groupe de trois : "`rwx`" et fonctionnent chacuns selon la logique "tout ou rien". En clair, soit la lettre est présente, et l'autorisation correspondante est donnée (`r` : read/lecture, `w` : write/écriture, `x` : execute/exécution) ; soit elle ne l'est pas (remplacée par "`-`"), et l'autorisation est alors refusée. Dans certains cas bien particuliers, ces lettres peuvent prendre d'autres valeurs, jette un œil sur **man chmod** pour plus de détails. Le premier groupe de trois caractères correspond aux droits de l'*user*, le deuxième aux droits du *group* et le dernier aux droits pour *others* (les autres).

Ensuite, tu ne t'attarderas pas sur le deuxième champ, et tu regarderas les troisième et quatrième, qui t'indiquent respectivement l'*user* et le *group* de l'élément concerné.

Prenons le cas de nos deux éléments mentionnés plus haut : la première ligne nous indique que nous avons affaire à un répertoire (`d`), que l'*user* (ici `varenet`) a droit de lire (`r`), écrire (`w`) – ce qui signifie notamment pouvoir y ajouter des éléments, mais aussi l'effacer s'il est vide – et exécuter (`x`) (pour un répertoire cela signifie "pouvoir le parcourir", ou encore faire "**cd <ce répertoire>**", mais pas nécessairement *voir* son contenu (avec **ls**) – il faut le droit `r` pour ça). On voit également que les membres du groupe `users` ne peuvent que lire et parcourir ce répertoire, et que tous les autres utilisateurs ne peuvent que le parcourir.

La deuxième ligne est tout aussi intéressante : `IN301TP3.sh` est un *fichier régulier* lisible, inscriptible et exécutable par le possesseur `varenet` et tous les membres du groupe `palinux`, tandis que le reste des utilisateurs ne peut rien faire de ce fichier.

☞	Il est important de noter qu'il existe une sorte d'héritage : si un dossier appartient à `spirou`, avec les droits suivant : `drwxr-xr-x`, et contient des fichiers appartenant à `fantasio`, celui-ci pourra modifier/supprimer ces fichiers, mais ne pourra pas en créer de nouveaux dans ce répertoire, à moins que des dispositions spéciales soient prises par rapport au groupe ou au reste des utilisateurs.

ATTENTION!

Il est fortement déconseillé de donner le droit d'écriture à tous les autres utilisateurs (`o`)...

2.3.3 Les notations de droits

Il existe différents moyens d'exprimer les droits, en particulier pour les différentes commandes qui gèrent les permissions (telles que **umask**, **chmod**).

Notation symbolique

On représente dans cette notation les actions sur les autorisations de façon assez transparente : `<qui><action><quoi>` ; `<qui>` étant soit `u` (*user*), `g` (*group*), `o` (*others*) ou `a` (les trois à la fois) ; `<action>` pouvant être "`+`" (ajouter) ou "`-`" (retirer) et `<quoi>` tout ou partie de `rwx`. Par exemple **chmod go-rw** supprime les droits de lecture et écriture pour *group* et *others*.

Notation numérique

Il s'agit ici de représenter chaque permission par un bit. La combinaison *qui*/*quoi* se fait par addition de ces bits :

`4` correspond au droit de lecture

`2` correspond au droit d'écriture

`1` correspond au droit d'exécution

Pour définir le *qui*, il suffit de modifier la position des bits correspondant dans le triplet final : `751` équivaut à `rwxr-x--x` en notation symbolique. Le résultat est appelé *mode*. Un petit coup d'œil à **man umask** peut t'aider !

ATTENTION!

Prends garde ! La commande **umask** est traîtresse !

Elle permet de régler le *mask*, qui, comme son nom l'indique, est un masque, c'est-à-dire le complément du *mode* : mathématiquement, on a :

$$mode = 666 \ \& \sim umask$$

(vraiment, un coup d'œil à la manpage de **umask** te sera très utile, crois moi !). Pour faire simple (ou du moins essayer), **umask 022** va avoir pour conséquence que tous les fichiers créés après cette commande auront pour mode `644`. Il faut faire bien attention à cela, car un mauvais masque (tel que `002`, qu'on peut trouver parfois), a pour conséquence que tous tes fichiers seront inscriptibles (et donc effaçables...) par tous tes camarades de promo (à l'ESIEE, ton groupe correspond à ta promo (`i3` dans mon cas). Ce qui s'avère souvent pratique !)

Chapitre 3

Ressources partagées

3.1 Notions de base

Comme je te l'ai déjà dit tout à l'heure (voir page 6), lorsque tu travailles à l'école, tu utilises des *ressources partagées*. C'est très simple à comprendre : tu n'es pas tout seul dans l'établissement, et les ordinateurs sont accessibles par tout le monde (du moins ceux des salles banalisées). Ils sont *partagés*.

En outre, tu as dû te rendre compte que quelque soit l'ordinateur que tu utilises, tu retrouves toujours l'ensemble de tes réglages, ainsi que tous les documents que tu as sauvegardés (que ce soit sous *Linux* ou *Windows*®©™ d'ailleurs). Ceci est la conséquence du fait qu'aucune de ces machines n'a la moindre donnée t'appartenant sur son *disque dur local*. En d'autres termes, ces données *ne se trouvent pas directement sur la machine*.

En fait, ce qui se passe c'est que l'intégralité de ton *compte utilisateur* [1] se trouve sauvegardé sur un serveur dédié à cette tâche, équipé pour stocker tous les comptes de tous les utilisateurs de l'école. Lorsque tu te "logges", l'ordinateur sur lequel tu te trouves va aller demander à ce serveur de lui fournir l'accès aux données de ton compte. Du coup, lorsque tu lis ou écris ces données, ce n'est pas sur le disque dur de la machine que tu le fais, mais *via* le réseau, directement sur ce serveur.

C'est la raison pour laquelle, lorsque tu travailles sur un gros document, ou pendant un TP, il est préférable que tu fasses une copie locale des documents concernés dans le répertoire temporaire prévu pour ça (`/tmp` sous *Linux* ; `\C:\Temp` sous *Windows*®©™ pour autant que je me souvienne), pour éviter d'une part de surcharger le réseau, et d'autre part pour ton confort personnel, en effet, les temps d'accès au disque local sont bien évidemment beaucoup plus rapides que ceux du réseau. En fait, quand je dis *préférable*, c'est un euphémisme. C'est vraiment *nécessaire* !

En outre il est indispensable de prendre l'habitude de sauvegarder régulièrement ton travail (toutes les dix minutes par exemple), comme ça s'il y a un problème, tu ne perds que dix minutes de ton dur labeur dans le pire des cas, et pas quatre heures de TP comme c'est déjà arrivé à certains dont je tairai les noms, par égard pour leurs familles !

☞	Petit détail à noter : si chez toi tu as un ordinateur, tu as peut-être l'habitude de l'éteindre une fois que tu n'en n'as plus besoin. Et bien ici, à l'ESIEE, ce n'est pas la peine. Je dirai même plus, il **ne** faut **pas** le faire ! Donc lorsque tu as fini de te servir d'une machine, tu te délogges, et tu n'éteins rien, ni écran ni ordinateur.

[1] Si tu as oublié de quoi il s'agit, je te renvoie à nouveau page 6 !

3.2 Quota

Une des premières choses que tu doives savoir, et dont tu vas de toute façon très vite te rendre compte, c'est que l'espace de stockage dont tu disposes pour tes données personnelles n'est pas illimité, loin de là. C'est une des conséquences de ce que je viens de t'expliquer : nous sommes très nombreux à utiliser les ressources de l'école, les données de tous les utilisateurs sont enregistrées sur un seul et même serveur, dont la capacité n'est pas illimitée.

Pour cette raison, et pour éviter d'éventuels abus, une limite est imposée en ce qui concerne la taille de ton compte. Selon ta promo, elle est de l'ordre de quelques dizaines de Méga-octets.

ATTENTION!

Si tu dépasses cette limite, tu ne pourras plus ouvrir de session [2] par exemple.

Tu peux la vérifier, ainsi que l'usage que tu en fais, grâce à la commande **quota** que nous avons vue en 2.2.14 page 20.

3.2.1 Les cores

Un "*core*" (pas au pied). Toute personne qui utilise une machine *UNIX* entend ce mot un jour ou l'autre. Et généralement pas en bien. On en parle comme de la peste, on a peur *"d'avoir un core sur son compte..."*. Bref, c'est l'angoisse.

Alors tout d'abord qu'est-ce que c'est ? Si j'en parle ici, dans la section consacrée au *quota*, ce n'est pas par hasard. Il s'agit *grosso modo* d'un gros (parfois très gros) fichier généré automatiquement lorsqu'une application *plante*, c'est-à-dire qu'elle ne se termine pas normalement. Comme un *core* peut prendre beaucoup de place [3], il faut en empêcher l'écriture sur ton compte, pour ne pas risquer de dépasser ton *quota*.

Il existe une unique bonne méthode, infaillible, qui marche à tous les coups. Si tu n'as pas changé ton *shell* par défaut (si tu ne sais pas ce que c'est alors il n'y a pas de problème), tu utilises donc **tcsh**, le *shell* fétiche de l'école. Pour te protéger, c'est très simple, il te suffit de taper exactement les commandes suivantes suivies de `[Entrée]` :

```
1  [varenet@idefix ~]> cd
2  [varenet@idefix ~]> echo "limit coredumpsize 0" >> .cshrc
```

Ceci n'affichera rien d'autre et c'est normal !

Ouvre un nouveau *terminal*, et vérifie que la commande **limit** t'indique bien :

```
coredumpsize  0 kbytes
```

Au cas (peu probable) où tu utilises **bash**, la marche à suivre est sensiblement la même :

```
1  [varenet@idefix ~]> cd
2  [varenet@idefix ~]> echo "ulimit -c 0" >> .bashrc
3  ulimit -c 0
```

Ouvre un nouveau *terminal*, et vérifie que la commande **ulimit -a** t'indique bien :

```
core file size        (blocks, -c) 0
```

[2] Voir 1.2 page 5.
[3] Plusieurs dizaines, voir centaines de Mega-octets.

3.3 Imprimer

Dans le même ordre d'idées, l'*impression* est une autre ressource partagée. Pour l'instant, la seule imprimante disponible pour l'ensemble des élèves de l'ESIEE est la **d640** située au sous-sol de l'épi 5.

Pour imprimer, c'est très simple : depuis un *terminal*, si ton document est au format *PostScript* (son nom se termine par `.ps`), c'est tout bon : tu tapes **lp <document>.ps** et hop, c'est parti. Sinon, il faut te servir de **a2ps** pour convertir toutes sortes de documents au format adéquat. **a2ps** est capable de faire beaucoup de choses, notamment de mettre joliment en forme du code... Va jeter un œil sur la page de manuel correspondante (**man a2ps**).

Sinon tu peux également imprimer depuis la plupart des applications *graphiques*, via la commande Print (ou Imprimer) généralement située dans le menu File (ou Fichier).

Pour suivre l'état des impressions, sous *Linux* on utilise **lpq**. Pour supprimer toutes les impressions que tu viens de lancer, tape depuis la machine où tu les as lancées, la commande **lprm** sous *Linux*.

Chapitre 4

Quelques outils

Je vais maintenant m'efforcer de te donner quelques pistes pour profiter au mieux des possibilités d'utilisation des machines sous *Linux* à l'école.

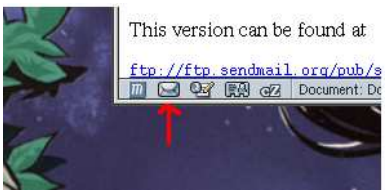
4.1 Mozilla

Commençons par ce qui va constituer la plus grande partie de ton usage des ordinateurs de l'école : le butinage sur Internet. Pour cela, l'outil que le gentil SMIG [1] met à ta disposition s'appelle *Mozilla*. C'est un navigateur Internet très performant, comparable à *Netscape Communicator*^{TM} dont il reprend bien des fonctions, ou encore à *Internet Explorer*^{TM}.

Je ne vais pas m'étendre sur le paramétrage de base de la chose, tu as dû (si tout fonctionne comme avant dans notre belle école) recevoir un poly t'expliquant les réglages propres à l'ESIEE.

Mozilla est en vérité une véritable usine à gaz, capable de faire beaucoup de choses dans sa version complète (surfer sur le *web* évidemment, mais aussi lire tes mails, créer des pages *web*, gérer ton carnet d'adresses, ou encore "chatter" sur *IRC*). Je ne vais me préoccuper ici que des deux premières fonctionnalités, et te laisse libre de découvrir les autres.

Petite information avant de commencer : toutes les fonctions de *Mozilla* sont accessibles directement par les boutons situés en bas à gauche de toutes les fenêtres de l'application. Selon les réglages, et les fonctionnalités installées, ces boutons ressemblent à ça :



De gauche à droite, tu trouves : le Navigateur, le Mail (fléché en "rouge" ici), le Composeur (de pages *web*), le Carnet d'adresse, et enfin l'IRC.

4.1.1 Navigateur Internet

En ce qui concerne la partie "Navigation sur Internet", qu'on peut démarrer très facilement depuis un *terminal* par la commande **mozilla**, quelques petits détails pratiques à connaître :

[1] Service des Moyens Informatiques et Généraux, c'est lui qui a la charge de l'ensemble du parc informatique du Groupe ESIEE.

Tabbed Browsing *Navigation Par Onglets* en français dans le texte. Il s'agit de pouvoir ouvrir plusieurs sites simultanément dans la même fenêtre du navigateur. Inutile de dire combien c'est pratique... Par défaut, pour ouvrir un nouveau site dans une fenêtre existante, il faut commencer par créer une nouvelle "*tab*" (un nouvel *"onglet"*) : Menu File➔New➔Navigator Tab, ou encore avec un raccourci clavier bien pratique : `[Ctrl-T]`. Cependant, il existe quelques options très utiles : dans Edit➔Preferences : Navigator : Tabbed Browsing, j'ai personnellement coutume de cocher les quatre options, elles sont diablement pratiques !

Internet Search Il s'agit du moteur de recherche utilisé par *Mozilla*. Par défaut c'est *Netscape Search*, mais je te recommande d'utiliser *Google* à la place, car c'est un bien meilleur moteur. Pour cela, dans Edit➔Preferences : Navigator : Internet Search, choisis "Search using : Google". Les autres options dépendent de tes goûts.

HTML Pipelining Là c'est du peaufinage, pour améliorer l'efficacité du "surf". Le principe est assez simple : en temps normal, le navigateur crée une connexion au serveur *web* pour chaque élément à télécharger (notamment les images). Avec cette option activée, il fait toutes ses requêtes dans la même connexion. D'où un certain gain de temps. Pour l'activer : Edit➔Preferences : Advanced : HTTP Networking, case "Enable Pipelining".

4.1.2 Client Mail

Après avoir vu la partie rigolade et ballade ludique sur la toile, attachons-nous un peu à ce qui constitue le deuxième usage principal de *Mozilla* sous *Linux* : la consultation des *e-mails*. Là encore, la partie configuration basique a été réglée par le superbe polycopié que tu as dû (et je n'en doute pas) recevoir (et lire) en début d'année. Cependant, bien qu'il soit d'excellente facture, ce document ne t'a pas averti de certaines "spécificités" du client mail de *Mozilla*. Qu'à cela ne tienne, je vais arranger ça !

Petit détail d'importance, pour démarrer le client mail de *Mozilla* directement, ouvre un *terminal* (comme indiqué paragraphe 2.1.1 page 9), et tape la commande suivante : **mozilla -mail**. Sinon, clique sur le bouton Mail comme vu sur l'image page précédente.

Un souci fréquemment rencontré avec *Mozilla Mail* est celui de l'occupation de l'espace disque [2]. En effet, si tu n'y prends pas garde, il se pourrait qu'un beau matin, tu ne puisses plus te "logger", et tu te demanderas pourquoi, puisque *"tu as toujours bien fait attention à ne pas surcharger ton compte avec des gros fichiers !"*. Et pourtant... Un petit **du** [3] sur ton répertoire `.mozilla` t'apprendra que celui-ci "pèse" plusieurs dizaines de Méga-octets. *"Comment est-ce possible ?"* demanderas-tu, toi qui prends toujours soin de mettre tes messages lus à la corbeille. Et bien malheureusement pour toi, *Mozilla Mail* est un petit filou, qui, lorsque tu mets un message à la corbeille, ne le supprime pas vraiment, à moins que tu ne lui demandes de faire autrement. C'est ce que je vais t'expliquer dans ce qui suit.

Compact Folders C'est la toute première option à surveiller : Edit➔Preferences : Offline & Disk Space, cocher la case : "Compact folders when it will save over **100** KB". Cela va réduire drastiquement les prétentions de *Mozilla* en matière d'espace de stockage.

Empty Trash C'est la seconde option importante et nécessaire pour être relativement à l'abri des mauvaises surprises. Son rôle est très simple : vider réellement la corbeille à la fermeture du client. Pour l'activer : Edit➔Mail & Newsgroup Account Settings : <compte> : Server Settings, case "Empty Trash on Exit".

Emoticons Pour ce dernier réglage, il s'agit plus de "tape à l'œil" qu'autre chose. Le dialogue de préférence est pour le moins explicite sur le résultat obtenu, va voir toi-même : Edit➔Preferences : Mail & Newsgroup : Message Display, case "Display emoticons as graphics".

☞	Quoi qu'il en soit, en matière d'espace disque, prend bien garde aux contenus "multimédia", tels que images, vidéos et autres animations. Ce sont de véritables petits gouffres dévoreurs de Kilo-octets !

[2] Ton espace disque disponible est limité par le *quota* qui t'est alloué. Voir section 3.2 page 28.
[3] Voir 2.2.10 page 16.

4.2 Nedit^{TM}

Après le *web*, le texte. Commençons par ce qui est jusqu'à maintenant la *Rolls* des éditeurs de texte sur *station*, j'ai nommé *Nedit*^{TM} ! Ce merveilleux éditeur *graphique* (par opposition aux éditeurs en *mode texte*, dont *Vim* – présenté ci-après – est un brillant représentant), qui n'est pas tout jeune, et pas *libre*[4] non plus, t'offre une foule de fonctionnalités très pratiques, en particulier pour éditer du code.

Toutes sortes de codes. Il connaît en effet pas moins d'une trentaine de langages différents (diablement efficace en C/C++, mais aussi pour éditer du code LaTeX – voir page 36). Il est capable d'en faire ressortir la syntaxe en couleur. Il gère bien sûr l'auto-indentation du code, les retours à la ligne, il surveille la bonne fermeture des parenthèses ouvertes, est capable d'exécuter du code dans un shell à la volée, numérote les lignes, j'en passe et des meilleures. Bref, il s'agit d'un éditeur graphique, son paramétrage se fait de façon relativement simple, je ne m'étendrai donc pas d'avantage sur le sujet.

☞	Il est bon de noter que la plupart des options ne sont pas activées par défaut. On peut les essayer *via* le menu Preferences, mais pour les rendre définitives, il faut les régler dans Preferences➔Default Settings, et ensuite faire un Save Defaults....

4.3 Vim

Ah ! Enfin, un de mes morceaux favoris. *Vim*, alias *Vi IMproved*, pour les intimes, est probablement l'éditeur de texte *en mode texte* par excellence. Je ne rentrerai pas dans la guéguerre culturelle entre *VIManiacs* et *Emacs-addicts*, je connais mal *Emacs*, ma première tentative a été décourageante et ne m'a pas incité à poursuivre la rencontre. Cependant, si tu connais bien *Emacs* et veux contribuer à ce guide, n'hésite pas à me contacter !

Bref, revenons-en à *Vim*, puisque c'est lui qui m'intéresse ici. C'est réellement un outil très puissant, dont même après plusieurs années d'usage il est rare de maîtriser correctement toutes les fonctionnalités. Tu l'auras donc bien compris, je ne vais pas faire une documentation exhaustive ici.

Vim, qu'on lance tout simplement par la commande **vim**, sait faire tout ce que *Nedit*^{TM} fait et bien plus encore. Mais *Attention !* **vim** et **vi** sont très différents. En général, **vi** lance *Vim* dans un mode limité visant à imiter le comportement du *Vi* original.

4.3.1 Modes

Vim dispose de plusieurs modes de fonctionnement, le principal étant le mode `Normal`. On passe de l'un à l'autre soit par combinaison de touches (actions ou commandes, voir page 35), soit dans certains cas par la souris (voir le paramétrage page suivante).

mode Normal En mode `Normal`, tu peux taper toutes les commandes usuelles de l'éditeur. C'est le mode par défaut lors du démarrage de *Vim* – sauf si tu en as décidé autrement dans ton fichier de configuration (voir page suivante) avec l'option `insertmode` qui te placera alors en mode `Insert` au démarrage. On l'appelle aussi le mode `command`.

mode Visual Ce mode est sensiblement identique au mode `Normal`, à ceci près que toutes les commandes autres que celles de déplacement affectent uniquement la zone sélectionnée. (D'où le terme "*visual*").

mode Select Ce mode ressemble beaucoup au fonctionnement habituel de certains éditeurs de texte classiques : si tu tapes un caractère, la zone sélectionnée est supprimée et tu passes en mode `Insert`.

[4] La notion de logiciel libre est un peu compliquée. Si cela t'interesse, va voir les site de la FSF (`http://www.fsf.org/`), et du Debian Project (`http://www.debian.org/`).

mode Insert Dans ce mode, le texte que tu tapes est inséré au fur et à mesure, comme dans tout traitement de texte traditionnel.

mode Command-line Également connu sous le doux acronyme de mode `Cmdline`, il s'agit en fait du mode dans lequel tu te trouves lorsque tu commences à taper une commande débutant par "`:`", "`?`", "`/`" ou "`!`" en mode `Normal`. Une ligne apparaît alors en bas de la fenêtre dans laquelle ta commande s'affiche.

Il existe six autres modes, dont cinq sont des variantes de ceux indiqués ici. Je n'entrerai pas dans le détail, l'aide est là pour ça !

4.3.2 Paramétrage

Là encore, il n'est pas question de faire le tour de tous les réglages de *Vim*, mais de te donner quelques clefs bien utiles. À toi d'aller chercher davantage d'informations ailleurs (je te recommande pour cela l'excellent Vim HOWTO [5]). Voici un extrait d'une partie de mon fichier `.vimrc`, présent dans mon *home directory* :

```
set hlsearch
set ignorecase
colorscheme blue
syntax enable
set autoindent
set history=100
set ruler
set showcmd
set selectmode=mouse
set tabstop=8
set shiftwidth=8
set mouse=a
set magic
set showmode
```

Explications

set hlsearch Lorsque qu'une recherche est lancée, surligne toutes les instances du résultat trouvé.

set ignorecase Indique que les recherches de chaînes de caractères sont insensibles à la casse.

colorscheme blue Établi le *thème* par défaut. Il en existe différents : `default, blue, darkblue, elflord, evening, koehler, morning, murphy`... Tous ne sont pas forcément installés, mais tu peux faire défiler ceux disponibles en lançant **vim**, et en tapant la touche `[Esc]`, puis `:colorscheme` et en appuyant sur la touche `[Tab]` plusieurs fois de suite. Comme nous le verrons plus loin, *Vim* gère l'auto-complétion, en particulier des noms de commandes.

syntax enable Complémentaire de l'option précédente, celle-ci active la coloration de la syntaxe du code édité.

set autoindent Active l'indentation automatique. En fonction du code que tu édites (C, PHP, Java, etc), *Vim* va gérer tout seul les tabulations, lorsque tu ouvres et refermes des accolades par exemple.

set history=100 Indique à *Vim* qu'il doit conserver les 100 dernières commandes tapées (accessibles en mode `Cmdline` – comme expliqué page précédente – avec les flèches haut et bas du clavier).

set ruler Active la ligne d'information en bas de la fenêtre de *Vim* (position actuelle dans le fichier : ligne, colonne, pourcentage parcouru, décalage de la prochaine commande, etc).

[5] Que tu peux trouver sur `http://dogma.esiee.fr/`, rubrique "HOWTO".

set showcmd Indique les informations sur les commandes "partielles", telles que le nombre d'éléments affectés par la commande en cours de préparation, ou bien encore le nombre d'éléments sélectionnés en mode `Visual` par exemple.

set selectmode=mouse Cette option est question de goût personnel : elle indique que toute selection de la souris passe en mode `Select`, ce qui a pour conséquence en particulier d'empêcher l'usage normal du copier/coller à la souris [6].

set tabstop=8 Indique combien de caractères de large fait une tabulation. 8 est le défaut communément utilisé et préconisé par les *Coding Styles* du noyau *Linux*.

set shiftwidth=8 Détermine le nombre de caractères de décalage pour l'auto-indentation. En général il est d'usage de mettre la même valeur que pour `tabstop`.

set mouse=a Active la possibilité d'utiliser la souris. Ne fonctionne pas avec tous les *terminaux*. l'argument de cette option détermine les modes dans lesquels la souris fonctionne : "`n`" pour `Normal`, "`v`" pour `Visual`, "`i`" pour `Insert`, "`c`" pour `Cmdline`, et enfin "`a`" pour tous ces modes à la fois.

set magic Active le "*magic*", c'est-à-dire que certains caractères qui peuvent avoir des significations spéciales (comme '$', '*') sont reconnus et interprétés pendant la saisie. Sinon il faut les "*escaper*" avec un backslash ("\") pour qu'ils soient reconnus. Pour bien comprendre l'intérêt et l'usage de cette fonction, il faut avoir déjà pas mal bidouillé avec *Vim*. L'aide est bien utile : `:help magic`.

set showmode Affiche le mode en cours d'utilisation (`Insert, Replace, Visual`...). Le mode `Normal` n'est jamais indiqué.

4.3.3 Commandes et actions

Voici donc quelques commandes et actions de base pour utiliser *Vim*. Toutes les commandes sont accessibles via leur nom complet (comme `:quit`), mais elles ont également pour la plupart un "raccourci" (comme `:q`). À chaque fois que possible je mentionnerai plutôt le raccourci, car il est plus facile à mémoriser et plus rapide à taper. Toutes les commandes qui suivent, ainsi que la plupart des actions, se tapent en mode `Normal`, c'est-à-dire lorsqu'on n'est pas en train d'éditer le texte (comme en mode `Insert` ou `Replace`). Dans le cas contraire je le préciserai toujours.

Les commandes sont précédées de "`:`", les actions se tapent directement sans préfixe particulier. Dans certains cas on peut cependant les faire précéder d'un nombre, qui précisera le nombre de caractères, lignes ou autres éléments affectés par l'action.

> **Attention!**
>
> Dans *Vim*, toutes les commandes et actions sont sensibles à la cAsSe. `p` n'a pas le même effet que `P`.

`:w` (*write*) Permet d'enregistrer le document en cours.

`:q` (*quit*) Termine le programme.

`:wq` Constitue la combinaison des deux opérations précédentes (enregistre et quitte).

`:help` Pour avoir de l'aide sur à peu près tout et n'importe quoi. L'aide de *Vim* est extrêmement complète, et bien que parfois un peu indigeste, elle te sera très utile !

`:s/<origine>/<remplacement>/g` Remplace dans la ligne active toutes les occurrences de `<origine>` par `<remplacement>`.

`/<chaine>` Recherche `<chaine>` dans tout le texte suivant le point d'insertion.

`?<chaine>` Recherche `<chaine>` dans tout le texte précédant le point d'insertion.

Les actions suivantes qui peuvent être précédées d'une valeur numérique correspondant au nombre d'éléments à modifier, sont indiquées par la formule "*un(e) ou plusieurs*".

[6] Souviens toi de la section 1.3 page 6.

`Y` Copie une ou plusieurs ligne(s) en dessous du point d'insertion.

`p` Colle les lignes copiées en dessous du point d'insertion.

`P` Colle les lignes copiées au dessus du point d'insertion.

`x` Coupe un ou plusieurs caractère(s) après le point d'insertion.

`dd` Coupe une ou plusieurs ligne(s) en dessous du point d'insertion.

4.3.4 Plugins

Vim peut utiliser des *plugins* que tu peux installer dans ton compte. Il en existe de très nombreux, et ce n'est pas le but de ce guide que de tous les lister. En voici cependant qui peuvent te rendre bien des services, je te recommande d'y jeter un œil [7] :

Tag Explorer Ce plugin, basé sur les *ctags* (voir plus loin), est particulièrement pratique lorsque tu écris du code. Il te permet notamment la navigation dans les répertoires, mais en l'améliorant : tu peux passer d'un fichier à un autre pour retrouver où est défini tel ou tel symbole par exemple...

Spell Checker Il s'agit d'un correcteur orthographique très puissant.

LaTeX-Suite (aussi connu sous le nom de *Vim-LaTeX*). C'est un petit concentré de bonnes idées, qui te permet de créer tes documents LaTeX de A à Z sans jamais quitter *Vim*.

Enfin, il faut connaître l'existence du système des *ctags*, qui n'est pas un plugin à proprement parler, mais une fonctionnalité avancée de *Vim*. Il s'agit, pour simplifier, d'avoir la possibilité lorsque tu édites du code source (C, C++, Perl, Pascal...), de pouvoir avoir la complétion automatique des noms de fonctions et l'accès permanent à leur prototypes. Si tu ne programmes pas, ce que je viens de dire doit ressembler à du chinois pour toi, mais dans le cas contraire tu as dû comprendre l'intérêt de la chose !

Pour plus d'info, je t'invite à regarder **man ctags**.

4.4 LaTeX

Est-il encore besoin de présenter ce formidable outil, érigé au rang de culte dans notre belle école ? Au cas où tu ne t'en serais pas encore rendu compte, le précieux document que tu tiens entre tes petites main potelées a été entièrement réalisé avec LaTeX.

Il s'agit donc, tu l'auras deviné, d'un logiciel extrêmement puissant et efficace de formatage de texte. Sache qu'il est de coutume de dire que tout rapport *"tapé en LaTeX"* est généralement gratifié d'un bonus !

Trêve de plaisanteries, en ce qui me concerne j'ai eu un peu de mal à m'y mettre, mais une fois lancé j'avoue ne plus avoir pu m'arrêter, tellement cet outil est **génial !** D'autre part, vu qu'il n'est pas question d'écrire une doc exhaustive de LaTeX ici, je te renvoie à l'excellent *"LaTeX par la pratique"*, édité chez O'REILLY, ainsi qu'au non moins excellent guide que nous devons à un ancien de l'ESIEE, j'ai nommé le *"Joli Manuel Pour LaTeX"*, également connu sous le nom de *"JMPL"*. À l'heure actuelle, tu peux le trouver (en interne à l'école) sur le ftp anonyme suivant : `ftp://dogma.esiee.fr/`. Inutile de préciser que ces deux ouvrages m'ont beaucoup aidé dans la conception et la réalisation du présent guide.

[7] Tu trouveras la plupart de ces plugins sur Internet. Cherche sur *Google* : `http://www.google.fr/`, ou directement sur `http://vim.sourceforge.net/`.

4.5 Open Office

Un rapide petit mot à propos de cette suite logicielle qui se veut l'équivalent libre de *Microsoft*® *Office*^{TM}. Il faut savoir que tu n'es pas obligé d'utiliser cette dernière pour pouvoir lire, imprimer et éditer des documents *Word*^{TM} ou *Excel*^{TM} par exemple. *Open Office* est capable de lire et enregistrer des documents dans les formats habituels de *Microsoft*® ; et même si dans certains cas un peu compliqués le résultat peut ne pas être parfait, il est en général tout à fait satisfaisant.

Donc tu n'as plus d'excuse pour justifier le fait de redémarrer sous *Windows*®©^{TM} une machine que tu trouves sous *Linux*. À part peut-être *Counter Strike* !

Chapitre 5

Exercices pratiques

Voici maintenant quelques idées d'exercices pour t'entraîner à manipuler les machines sous *Linux*. Soit tranquille : au commencement, ton répertoire est vide. Tu ne peux donc rien casser, alors fais des essais l'esprit détendu !

1. Commence par t'entraîner à te *logger* et te *délogger* proprement. (*cf.* 1.2 page 5)
2. Familiarise-toi avec l'environnement graphique. L'action des différents boutons de la souris, les différents menus à ta disposition... (*cf.* 1.3 page 6)
3. Lance un *term*. (*cf.* 2.1.1 page 9)
4. Regarde quelques pages de **man** ! (*cf.* 2.2.1 page 11)
5. Liste le contenu de ton répertoire (y compris les fichiers invisibles). (*cf.* 2.2.2 page 12)
6. Essaye toi à la manipulation de fichiers : crée des répertoires, des fichiers, déplace-les, renomme-les, supprime-les... (*cf.* 2.2.5, 2.2.7, 2.2.8 page 15)
7. Surveille la taille de différents éléments avec **du**. (*cf.* 2.2.10 page 16)
8. Vérifie ton *quota* (*cf.* 2.2.14 page 20)
9. Entraîne toi à lister les processus et à en tuer certains (*cf.* 2.2.15, 2.2.16 page 21)
10. Teste avec quelqu'un l'effet des changements de permissions sur un répertoire de ton choix... (*cf.* 2.2.13 page 18)
11. Démarre *Mozilla*, fais tes premiers pas sur Internet, regarde tes mails. (*cf.* 4.1 page 31)
12. Tente d'imprimer une page *web* (ton emploi du temps de la semaine, par exemple – c'est très pratique de l'avoir à portée de main le matin au réveil !). (*cf.* 3.3 page 29)
13. Enfin tu peux également à l'occasion essayer *Nedit*^{TM}, *Vim* voire même LaTeX !

Remerciements

L'auteur (moi-même, votre humble serviteur) voudrait remercier les quelques personnes qui suivent pour leur précieuse aide, sans laquelle le présent guide n'aurait probablement pas vu le jour. Enfin... si quand même. Mais pas aussi bien que ça disons !

Hall Of Fame

Les heureux élus sont par ordre alphabétique :

– Elysabeth Bastien
– Frank Bonnet
– Régis Boudin
– Monique Piogé
– Serge Renaux
– Thierry Simonnet
– Thibaut Varène

Annexe A

Personnalisation de KDE

Ha ha ! Petit malin, l'image de l'introduction t'a mis l'eau à la bouche et tu voudrais bien en savoir plus, n'est-ce pas ? Et bien soit, je vais te donner ici quelques tuyaux qui circulent depuis l'aube des temps chez les élèves de l'ESIEE un peu rodés avec *Linux*.

Une chose est certaine, et tu vas vite t'en apercevoir tout seul : *KDE* est très riche en réglages de personnalisation. Tu peux en particulier t'en rendre compte en furetant dans le menu Preferences accessible depuis le gros "K" en bas à gauche de ton écran.

A.1 Fond d'écran

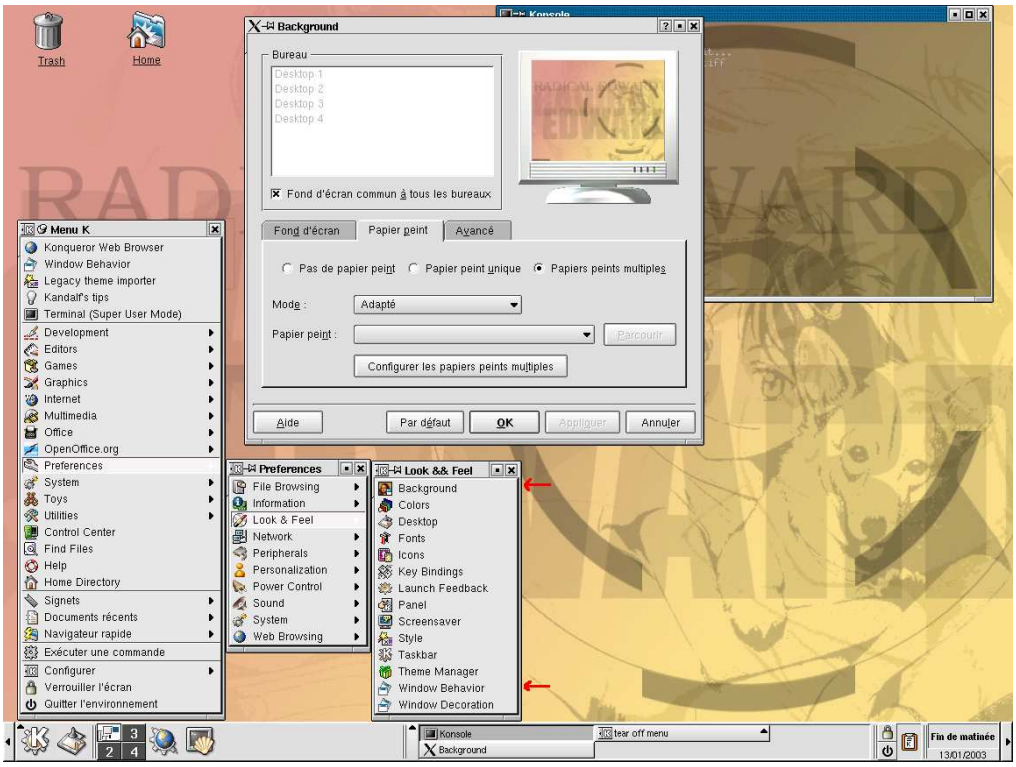
Tu voudrais bien changer le fond d'écran pas vraiment original ni très sympa qui orne ton bureau. Voici comment faire.

Tout d'abord, supposons que tu aies sur ton compte une image quelconque, nommée `fond.jpg` (par exemple). Pour pouvoir en faire ton fond d'écran, le réglage est beaucoup plus simple que sous *CDE* :

Soit tu cliques-droit [1] sur l'image de fond, et tu choisis dans le menu qui apparaît l'item Configure Background...

Soit tu cliques sur le "K" et tu déroules les menus suivants : Preferences➔Look & Feel➔Background comme tu peux le voir sur l'image page suivante.

[1] C'est-à-dire faire un clic avec le bouton droit de la souris.



A.1.1 Réglages

Une nouvelle fenêtre s'ouvre alors dans laquelle tu peux choisir soit un *motif* de fond (onglet Fond d'écran, soit une image (onglet Papier Peint. Tu cliques sur ce dernier onglet, comme le montre l'image, et tu peux choisir soit Papier peint unique (pour avoir un unique papier peint en permanence), soit Papiers peints multiples (qui permet de faire changer automatiquement à intervalles de temps réguliers l'image de fond).

Une fois les réglages terminés, cliques sur Appliquer, puis sur OK. Et voila le travail !

☞	Petite astuce : le *terminal* transparent (comme sur la copie d'écran) s'obtient en faisant un clic-droit sur le *terminal*, et en choisissant Modèle➔MC Transparent.

A.2 Focus souris

Le réglage du *focus* de la souris est possible avec *KDE*. Sur l'image qui précède la deuxième petite flèche rouge t'indique le menu à cliquer pour accéder aux réglages du *focus*, il s'agit de Window Behavior.

A.2.1 Réglages

Tu disposes de différentes options, parmi elles :

"Cliquer pour avoir le focus" C'est le comportement normal, tu dois cliquer pour *activer* les fenêtres.

"Le focus suit la souris" C'est un mode très agréable : lorsque ton curseur passe sur une fenêtre, celle-ci est activée, et elle le reste jusqu'à ce que le curseur passe sur **une autre** fenêtre. Par exemple, si tu mets le curseur sur le bureau, la fenêtre restera activée.

"Le focus est sous la souris" Lorsque tu passes sur une fenêtre avec le curseur, elle est *activée*, tu en ressors elle est *désactivée*.

Annexe B

Personnalisation du prompt

Voici un des points les plus intéressants pour celui qui veut tirer le meilleur des machines *Linux*, et bénéficier d'un environnement de travail pratique et efficace. Je vais te donner quelques informations sur les arcanes de la configuration du *prompt*, dont je t'ai déjà un peu parlé au chapitre 2 page 9.

L'idée de départ est assez simple. Par défaut, et tu t'en es peut-être rendu compte, le *prompt* de ton *terminal* ne donne que très peu d'informations (en l'occurrence le nom de la machine sur laquelle tu te trouves, et le numéro de commande). Je fais partie de ceux qui trouvent que ça n'est pas franchement passionnant, et je vais te montrer comment obtenir un *prompt* plus sympa (comme celui que je donne dans mes exemples, qui est déjà un poil plus évolué, mais on peut faire beaucoup mieux !).

B.1 Avec tcsh

A l'heure actuelle, le *shell* [1] par défaut de l'ESIEE est le **tcsh** [2], donc je commence par celui-ci.

Ce que je vais détailler ici n'est pas beaucoup plus qu'une traduction de **man tcsh** (ou **man csh**), donc tu sais ce qu'il te reste à faire...

Le principe est le suivant, dans ton `.cshrc` tu dois modifier le réglage de la *variable d'environnement* `prompt`, qui est définie généralement au début du fichier par une ligne commençant par `set prompt = "..."`. Pour modifier ton prompt tu dois remplacer le contenu situé entre les guillemets. En effet, tout ce qui se trouve entre guillemets est affiché (après interprétation) comme *prompt* sur ton *terminal*. Je dis *"après interprétation"*, car il existe des caractères spéciaux qui sont remplacés par une valeur particulière. En voici quelques uns parmi les plus utilisés :

`%/` : Répertoire de travail actuel en format complet (comme avec **pwd**, *c.f.* 2.2.3 page 13).

`%~` : Comme le précédent, mais cette fois le chemin part du *home directory* quand c'est possible ("`/nfs/user/eleve/i3/varenet`" est remplacé par "`~`" dans mon cas).

`%c[n]` : Comme pour `%~`, sauf qu'en remplaçant `[n]` par un chiffre, on peut limiter le nombre de sous répertoires affichés. Très pratique.

`%!` : Numéro de commande actuelle.

`%m` : Nom de la machine.

`%n` : Nom d'utilisateur.

`%?` : Code de retour de la commande précédente.

[1] Voir page 9.

[2] *TC-Shell* de son petit nom, qui ne s'appelle pas comme ça par hasard, puisque sa syntaxe étendue se base sur celle du C.

Il y en a beaucoup d'autres, je ne donne que les plus courants. Les possibilités de réglage sont très nombreuses, à toi d'explorer le **man** !

Par exemple dans mon cas j'utilise : "`[%n@%m %c3]%#` "

Attention de ne pas oublier l'espace final...

B.2 Avec bash

Tout comme pour **tcsh**, le *prompt* de **bash** (un des *shell* les plus répandus) est très finement paramétrable. À la différence de **tcsh** en revanche, c'est la *variable* `PS1` qu'il te faut changer (ou définir si elle ne l'est pas) dans ton `.bashrc`. Voici les principaux caractères spéciaux :

`\h` : Nom de la machine.

`\u` : Nom d'utilisateur.

`\w` : Comme `%~` pour **tcsh**.

`\W` : Comme `%/` pour **tcsh**.

`\#` : Numéro de commande actuelle.

Et comme pour **tcsh**, je t'invite à regarder **man bash**, rubrique "PROMPTING".

Dans mon cas j'utilise : "`[\u@\h \w]\$` "

Voilà, il n'y a pas grand chose d'autre à rajouter, sinon j'écrirai sur chacun des *shell* un volume aussi épais (voire même plus) que celui que tu tiens en ce moment...