Cours d'informatique (niveau L2)

Robert Alessi
CNRS UMR 8167 Orient et Méditerranée – Université de Poitiers

document créé le: 28 novembre 2010
à partir du fichier: cours_info_litteraire.tex

Sommaire

Liste des tableaux

Table des figures

Introduction

Introduction reste à faire.

Présenter les objectifs de ce séminaire.
Travail sous Linux. D'abord notions sur l'environnement et la ligne de commande, puis notions sur \LaTeX.

Chapitre 1

L'environnement Linux

1.1 La ligne de commande

Présentation générale

La *ligne de commande* est avant tout une interface de communication avec l'ordinateur, tout comme le sont les interfaces graphiques auxquelles nous sommes habitués. Les interfaces graphiques telles que le *Bureau* sous Windows ou bien le *Finder* sous Mac OS sont en réalité comme une couche qui se superpose à la ligne de commande, qui continue toujours à exister. Tout en masquant à l'utilisateur la ligne de commande, elles traduisent néanmoins en ligne de commande les opérations qui sont faites à l'aide de la souris. Car les systèmes d'exploitation ne connaissent que la ligne de commande.

Exemple Supposons que je veuille créer un répertoire [1] nommé *travail* sur le bureau, puis déplacer dans ce réperoire le fichier *exemple.pdf* que je viens de créer. À l'aide de la souris, je dois à peu près effectuer les opérations suivantes :

1. faire un simple-clic du bouton droit de la souris, et choisir `Dossier --> Nouveau` dans le menu contextuel qui s'affiche ;
2. dans la fenêtre qui s'affiche, saisir le nom du nouveau répertoire : `travail`, puis cliquer sur le bouton `Ok` ;
3. dernière opération, le déplacement du fichier *exemple.pdf* : à l'aide du bouton gauche de la souris, faire glisser le fichier *exemple.pdf* sur l'icône du répertoire *travail*, et relâcher le bouton à ce moment.

Pour réaliser les mêmes opérations à la ligne de commande, il aurait fallu saisir les lignes suivantes :

1. création du dossier *travail* :
   `mkdir travail`
2. déplacement du fichier *exemple.pdf* dans le dossier travail :
   `mv exemple.pdf travail`

1. C'est ansi que l'on appellera ce qui, sous Windows et Mac OS X, se nomme *dossier*.

Commentaire Il faut bien comprendre, comme on l'a dit plus haut, que le système d'exploitation n'interprète que les lignes de commande. Cela signifie le rôle de l'interface graphique n'est que de traduire en lignes de commande les opérations que nous effectuons à l'aide de la souris. Reprenons à présent les deux dernières lignes de commande pour mieux les comprendre :

1. dans « `mkdir travail` », `mkdir` est le nom d'un programme fait pour créer des répertoires ; `mkdir` est en effet pour l'anglais *make directory*. Quant à *travail*, c'est tout simplement le nom du répertoire qu'on veut faire créer par le programme `mkdir`. La terminologie est la suivante : `mkdir` est le nom du programme, et `travail` est *l'argument* que l'on passe à `mkdir`. Remarquez que l'on doit séparer l'argument du nom du programme par un espace. Pour terminer, on appuie sur la touche *Entrée* pour commander l'exécution du programme.
2. dans « `mv exemple.pdf travail` », le nom du programme est `mv`, pour l'anglais *move* ; sa fonction est de déplacer des fichiers ou des répertoires. Comme son comportement, par rapport au programme `mkdir`, est différent, il accepte non pas un, mais deux arguments, chacun séparé par des espaces. Observez de nouveau cette ligne de commande : tandis que le premier argument est le nom du fichier que l'on souhaite déplacer, le deuxième est le nom du répertoire de destination de ce fichier. Pour terminer, de la même manière que précédemment, on appuie sur la touche *Entrée* pour commander le déplacement du fichier.

Par cet exemple, on espère faire comprendre que si la syntaxe de la ligne de commande peut paraître au premier abord difficile à maîtriser, elle permet aussi, par sa sobriété même, de réaliser de manière bien plus rapide et bien plus sûre les opérations que l'on fait à l'aide de la souris. En voici les principales raisons :

- l'interface graphique est une surcouche logicielle ; elle ralentit donc le système d'exploitation ;
- l'interface graphique, comme tout logiciel très complexe, comporte des erreurs de programmation. Ces *bugs* peuvent aller jusqu'à bloquer complètement le système d'exploitation ;
- à l'aide de l'interface graphique, on ne peut réaliser que les équivalents en ligne de commande qui ont été prévus par les programmeurs. En se privant de la ligne de commande, l'utilisateur se prive donc aussi de pouvoir réaliser les opérations qui ont été laissées de côté [2] ;
- les lignes de commande peuvent être chaînées. Ainsi, par la simple ligne
  `mkdir travail ; mv exemple.pdf travail`
  on peut réaliser en une seule fois toutes les opérations décrites précédemment. Il suffit, comme on le voit ici, de séparer les commandes par un point-virgule (`;`) ;
- les lignes de commande acceptent des caractères appelés *jokers* à l'aide desquels on peut déclencher des opérations complexes, portant sur un très grand nombre de fichiers. Par exemple, le caractère `*` peut se substituer à n'importe quelle chaîne de caractères. Ainsi, pour reprendre ce qui précède, la commande

2. C'est d'ailleurs ainsi, bien souvent, que les techniciens compétents dépannent les ordinateurs : en réalisant des commandes auxquelles l'interface graphique ne permet pas d'accéder.

`mv *.pdf travail`
aura pour effet de déplacer automatiquement tous les fichiers au format `PDF` dans le répertoire *travail*.

En d'autres termes, en passant par la ligne de commande, l'utilisateur gagne en sécurité, en rapidité et en maîtrise du système ce qu'il perd en ergonomie.

Les chemins d'accès

L'interpréteur de commande sous Linux s'appelle le *terminal* ou la *console*. Il en existe un grand nombre. Les plus courants sont `xterm`, `rxvt`, `konsole` (sous KDE), `gnome-terminal` (sous Gnome), ou encore `terminal` (sous xfce4).

Nous utiliserons dans ce cours l'interpréteur `xterm`. Vous le trouverez normalement quelque part dans l'arborescence de vos menus.

Les commandes portent le plus souvent sur des fichiers. Il est donc important, pour savoir où se trouvent les fichiers que vous voulez traiter, de connaître leur *chemin d'accès*.

Le home directory Dans les systèmes Linux, tous vos fichiers se trouvent dans votre répertoire personnel, appelé le *home directory*. Le nom de votre répertoire personnel est le même que celui de l'identifiant sous lequel vous vous êtes connecté. Par ailleurs, tous les répertoires des différents utilisateurs sont situés à la racine du disque dur dans un répertoire fondamental appelé `home`.

Supposons que votre identifiant soit `jacques` ; votre répertoire personnel sera donc :

```
/home/jacques
```

Obervez attentivement cette ligne. Vous remarquez que les noms des répertoires sont séparés par le caractère `/`. Cela veut dire que le signe `/` est utilisé pour indiquer que l'on passe d'un répertoire donné à l'un de ses sous-répertoires. Dans notre exemple, le répertoire `jacques` est donc inclus dans le répertoire `home`.

Remarquez encore le `/` qui est placé *devant* `home` : comme il n'est lui-même précédé de rien, il indique que le répertoire `home` est placé *à la racine du disque dur*.

Definition : chemins absolus, chemins relatifs Un chemin d'accès est dit *absolu* quand il est donné à partir de la racine du disque dur. Il est *relatif* quand il est donné à partir de tout autre endroit du disque dur. Soit par exemple le répertoire `travail` créé par l'utilisateur `jacques` dans son répertoire personnel. À partir de ce répertoire, le chemin d'accès absolu sera

```
/home/jacques/travail/
```

tandis que le chemin relatif sera

```
travail/
```

Corrolaire : tout chemin d'accès absolu commence nécessairement par le caractère `/` ; quand ce n'est pas le cas, le chemin d'accès est nécessairement relatif.

Conventions Il existe un grand nombre de raccourcis ou de signes conventionnels qui sont utilisés dans la ligne de commande. On en retiendra trois pour le moment :

- *home directory* : depuis tout endroit du disque dur, tout utilisateur peut accéder à son répertoire personnel par le raccourci
  `~/`
  Ainsi, pour l'utilisateur `jacques`, `~/travail` est l'équivalent de `/home/jacques/travail`.
- répertoire parent : quel que soit le répetoire dans lequel on se trouve, la séquence `..` désigne le *répertoire parent*, c'est-à-dire le répertoire qui le contient, ou bien qui est situé au niveau supérieur dans l'arborescence du disque dur. Par exemple, à partir du répertoire `/home/jacques/travail`, `..` désigne le répertoire `/home/jacques`.
- répertoire courant : Quant au signe « `.` », il désigne tout simplement le répertoire dans lequel on se trouve.

Les premières commandes

pwd Signifie *print working directory*. Cette commande vous retourne tout simplement le chemin d'accès absolu du répertoire dans lequel vous vous trouvez. Très utile pour ne pas se perdre ! Exemple :

```
[robert@kiddo:~]$ pwd
/home/robert
```

> La séquence `[robert@kiddo:~]$` est *l'invite de commande* (anglais *prompt*). C'est à la suite de cette invite que l'on entre les commandes. Nous y reviendrons. Observez pour le moment quelques unes des informations données par cette invite : l'utilisateur `robert` est connecté sur l'ordinateur `kiddo` ; après les `:`, le signe `~` indique qu'il se trouve dans son *home directory*, ce que retourne en effet la commande `pwd` qui a été entrée ici.

mv Signifie *move*. Cette commande déplace les fichiers d'un endroit vers un autre. La syntaxe est la suivante :

`mv␣<source>␣<destination>`

> Par convention, le signe ␣ marque l'espace.

Exemple : déplacement du fichier `trachiniennes.pdf` dans le répertoire `travail` :

`[robert@kiddo:~]$␣mv␣trachiniennes.pdf␣travail/`

Déplacement du fichier `trachiniennes.pdf` depuis le répertoire `travail` vers le répertoire courant (désigné par le raccourci `.`) :

`[robert@kiddo:~]$␣mv␣travail/trachiniennes.pdf␣.`

Déplacement avec indication des chemins absolus :

`[robert@kiddo:~]$␣mv␣/home/robert/trachiniennes.pdf␣/home/robert/travail/`

Utilisation de raccourcis :

`[robert@kiddo:~]$␣mv␣~/trachiniennes.pdf␣~/travail/`

cp Signifie *copy*. Cette commande copie des fichiers depuis un endroit vers un autre. La syntaxe est comparable à celle de la séquence `mv`.

```
cp␣<source>␣<destination>
```

Exemple : copie du fichier `trachiniennes.pdf` dans le répertoire `travail` :

```
[robert@kiddo:~]$␣cp␣trachiniennes.pdf␣travail/
```

Copie du fichier `trachiniennes.pdf` depuis le répertoire `travail` vers le répertoire courant (désigné par le raccourci `.`) :

```
[robert@kiddo:~]$␣cp␣travail/trachiniennes.pdf␣.
```

Copie avec indication des chemins absolus :

```
[robert@kiddo:~]$␣cp␣/home/robert/trachiniennes.pdf␣/home/robert/travail/
```

Utilisation de raccourcis :

```
[robert@kiddo:~]$␣cp␣~/trachiniennes.pdf␣~/travail/
```

cd Signifie *change directory*. Permet de changer de répertoire courant, par exemple pour travailler sur les fichiers d'un répertoire différent de son *home directory*. La syntaxe est la suivante :

```
cd␣<chemin_d'accès_du_nouveau_répertoire>
```

Exemple : changement vers le répertoire /usr/bin :

```
[robert@kiddo:~]$␣cd␣/usr/bin
[robert@kiddo:/usr/bin]$
```

> Remarquez le changement de l'invite après l'exécution de la commande. L'invite nous donne l'indication du nouveau répertoire.

Confirmation par la commande `pwd` :

```
[robert@kiddo:/usr/bin]$␣pwd
/usr/bin
[robert@kiddo:/usr/bin]$
```

> NB : la commande `cd` seule fait revenir l'utilisateur directement dans son *home directory*.

ls Signifie *list*. Affiche à l'écran tous les fichiers et les répertoires contenus dans un répertoire donné. Si on ne précise pas le répertoire dont il faut lister les fichiers, la commande liste les fichiers du répertoire courant. Exemple : on vérifie que le fichier `trachiniennes.pdf` se trouve bien dans le répertoire travail :

```
[robert@kiddo:~]$␣ls␣travail/
trachiniennes.pdf
```

> Comme on le voit, la commande retourne le nom du seul fichier qui se trouve dans le répertoire `travail`.
> La commande `ls` est l'une des plus importantes ; elle admet de nombreuses options que nous détaillerons plus loin dans ce cours.

Chapitre 2

Présentation de LaTeX

LaTeX est le premier outil que nous aborderons dans ce séminaire. Il s'agit d'un système de composition et de mise en page extrêmement puissant. Pour ainsi dire, tout peut-être composé sous LaTeX : du texte, bien sûr, dans absolument toutes les langues connues, vivantes ou mortes, mais aussi des documents techniques ou scientifiques, des éditions de textes anciens accompagnées d'appareils critiques complexes, avec plusieurs étages de notes critiques, des partitions musicales, etc.

LaTeX dépasse donc de très loin les possibilités des traitements de texte ou des logiciels de PAO. Il produit aussi des documents de qualité typographique professionnelle. Il est capable de produire toutes sortes d'index, dans toutes les langues du monde et de générer des bibliographies scientifiques correspondant parfaitement aux normes internationales.

Comme un véritable typographe professionnel à qui l'on confierait un manuscrit, LaTeX connaît les règles de présentation et de typographie de tous les pays du monde. Du point de vue de la saisie du texte lui-même, LaTeX se distingue radicalement des traitements de textes ou des logiciels de PAO. En effet, quand on saisit du texte sous l'un de ces logiciels graphiques, le rendu visuel, à l'écran, est simultané. Dans ces conditions, l'on se préoccupe naturellement, lors de la saisie, tout autant de la saisie elle-même que des questions de mise en page, et cela, que la mise en page soit manuelle ou qu'on l'ait confiée à une feuille de style.

Sous LaTeX, en revanche, l'auteur saisit son texte « au kilomètre », sans avoir en aucune manière à se soucier des questions de typographie ou de mise en page. Il insère aussi dans son texte des *commandes de structuration* qui seront interprétées par LaTeX pour produire le document final, destiné à être imprimé.

LaTeX débarrasse donc l'auteur de tout souci de mise en page. Composer un texte sous LaTeX, c'est un peu comme d'abord l'écrire à la main, ou le taper sur une machine à écrire mécanique, pour ensuite porter au crayon à mine de plomb les recommandations que l'on destinait autrefois au typographe : « mettre en valeur cette citation ou ce paragraphe important, changer de chapitre, reprendre tel mot dans l'*index verborum*... »

Sous LaTeX, la différence fondamentale est donc que le rendu visuel n'est pas simultané. Saisie du texte et composition sont des opérations nettement séparées.

Quelques définitions Le texte saisi au kilomètre par l'auteur, accompagné des instructions de structuration, s'appelle le *source*. Il est saisi dans un logigiel que l'on appelle *éditeur de texte*, et que l'on évitera de confondre avec le *traitement de texte*[1]. Une fois le source (texte + commandes de structuration) composé, on le *compile* à l'aide de LaTeX ; LaTeX retourne alors un nouveau fichier qui ne peut plus être modifié directement. Ce fichier est par exemple seulement destiné à être visualisé sur un écran ou imprimé sur papier. Par suite, si l'on veut corriger des erreurs telles que des fautes de frappe, on doit corriger le source, puis le recompiler[2].

Le fichier source doit se terminer par l'extension `.tex` ; après compilation, le fichier produit par LaTeX pour visualisation ou impression porte l'extension `.dvi`[3] pour *device independent*. La chaîne de production est donc la suivante :

`fichier.tex` $\overset{\text{LaTeX}}{\rightarrow}$ `fichier.dvi`

Après compilation, le fichier `.dvi` peut être visualisé ou projeté à l'écran à l'aide de programmes tiers, dont le plus connu est `xdvi` sous les environnements de type Unix/Linux. D'autres commandes servent ensuite soit pour imprimer le fichier soit pour le convertir sous divers formats tels que PostScript ou `PDF`. D'autres commandes encore, que nous étudierons plus tard, permettent de convertir le source dans des formats tels que HTML, *Oasis Open Document*, ou même `RTF`. Il est donc parfaitement possible, à partir de la saisie d'un seul code source, d'obtenir des sorties extrêmement variées.

Les documents `HTML` produits à partir de sources LaTeX sont toujours parfaitement formatés. Comme ils sont conformes au standard unicode, ils permettent de faire passer très facilement sur le *web* des documents multilingues. Enfin, la conversion en des formats tels de `RTF` ou *Oasis Open Document* permet d'envoyer aux éditeurs qui le réclament des fichiers qui peuvent être repris dans les traitements de texte ou dans les programmes de `PAO`.

2.1 Les éléments fondamentaux du fichier source

Tout fichier source LaTeX comprend deux parties fondamentales :
- le *préambule* ;
- le *corps du document* lui-même.

Dans le préambule, on saisit essentiellement des instructions générales de formatage ou bien l'on charge des extensions complémentaires pour modifier le comportement par défaut de LaTeX : ces extensions sont appelées *packages*. Le préambule n'est donc ni visible dans le fichier de sortie `.dvi` ni imprimé.

Quant au *corps du document*, il comprend à la fois le texte lui-même que l'on saisit et diverses instructions de présentation et de structuration. Par exemple, par rapport au texte, saisi au kilomètre, d'un roman divisé en chapitres, le titre du chapitre est introduit dans une commande de structuration.

Commentons le source suivant :

1. Les éditeurs de texte n'acceptent que des caractères alphanumériques, à l'exclusion de tout enrichissement typographique (police, taille des et famille des caractères, etc.) C'est par exemple dans les éditeurs de texte que les programmeurs composent leurs programmes avant de les compiler.
2. Que l'on se rassure, la compilation est une opération très rapide !
3. LaTeX produit aussi d'autres fichiers auxiliaires sur lesquels nous reviendrons.

```
\documentclass{article}


\begin{document}

Hello, world!

\end{document}
```

Le *corps du document*, dans cet exemple, est compris entre la commande `\begin{document}` et `\end{document}` ; après compilation, le fichier `.dvi` se présentera donc sous la forme d'une page blanche avec les seuls mots *Hello, world !*

Quant au *préambule*, il est compris entre la commande `\documentclass` et la commande `\begin{document}`.

Remarquez enfin la forme particulière des commandes LaTeX : elles commencent toutes par le caractère \, appelé *séquence d'échappement*. Toutes les commandes LaTeX se présentent sous cette forme. Ces commandes sont des instructions que l'auteur donne à LaTeX au cours de la saisie de son texte ; elles seront interprétées et exécutées, mais, bien entendu, elles ne sont ni visualisables ni imprimables.

> **Important :** tout document LaTeX commence obligatoirement par la commande `\documentclass`, et doit comporter les deux commandes `\begin{document}` et `\end{document}`.

Dernière remarque : l'espace situé entre les commandes `\begin` et `\end` se nomme *environnement*. Dans l'exemple ci-dessus, nous avons donc défini un environnement *document* qui est l'espace dans lequel est saisi le corps du texte à imprimer. Nous étudierons plus loin d'autres environnements plus limités, tels que ceux qui servent à présenter des citations ou bien des listes numérotées. Les environnements peuvent êtres emboîtés les uns dans les autres, comme des poupées russes. Mais il est formellement interdit de fermer un environnement donné s'il contient des environnements qui n'ont pas été eux-mêmes fermés. Voici un exemple d'inclusion valide :

```
[  [  [  ]  ]  ]
A  B  C  C  B  A
```

Tandis que celui-ci ne l'est pas :

```
[  [  [  ]  ]  ]
A  B  C  B  C  A
```

En effet, dans ce dernier exemple, l'environnement `B` a été refermé alors que l'environnement `C` était encore ouvert. Cette grave erreur de programmation s'appelle *overlapping*. Elle bloque généralement l'exécution de LaTeX.

Observons maintenant la première commande `\documentclass{article}`. Elle se présente sous la forme suivante :

$$\boxed{\backslash <\textit{nom de la commande}>\{\textit{argument}\}}$$

La séquence d'échappement \ est immédiatement suivie du nom de la commande elle-même. On place ensuite entre accolades l'argument que l'on souhaite

passer à la commande. Dans l'exemple ci-dessus, la commande `\documentclass` permet de choisir la *classe de document* que l'on souhaite utiliser. Une classe de document est un ensemble de règles de formatage relatives à un type de document donné. Ici, l'on a choisi de produire un document de type *article*, sans option particulière. Le document sortira en recto, et il pourra accepter les commandes de structuration qui sont en usage dans les articles. À la place de *article*, on aurait pu par exemple choisir la classe de document *book*. D'autres commandes de structuration auraient alors été disponibles [4].

Pour modifier le comportement par défaut d'une commande donnée, on peut enfin passer à la commande des *options*. Ainsi, si l'on souhaite imprimer un article en recto-verso, avec une police de base de taille 12 points et sur du papier au format A4, la commande sera la suivante :

```
\documentclass[twoside,12pt,a4paper]{article}
```

Comme on le voit, les *options* sont passées à la commande entre crochets droits, immédiatement après la commande, et juste avant l'argument. Par ailleurs, quand on souhaite passer plusieurs options, on les sépare les unes des autres par de simples virgules.

On apprendra plus loin quelles sont les principales commandes et les principales options. Il est temps de produire notre premier document.

2.2 Production d'un premier document élémentaire en français

Quand on saisit une commande sans option, le résultat produit sera celui qui a été prévu par défaut. Reprenons notre premier exemple :

```
\documentclass{article}


\begin{document}

Hello, world!

\end{document}
```

Comme la classe de document *article* a été choisie sans plus de précision, le document produit par défaut aura les caractéristiques suivantes :

- langue anglaise américaine,
- respect des règles de typographie américaines,
- papier au format *letter* (8,5 pouces x 11 pouces),
- police de base de taille 10 points,
- numérotation des pages centrée, au pied de chaque page,
- tirages *recto*.

Pour modifier ce comportement, nous introduisons dans le préambule les commandes supplémentaires suivantes :

4. Par exemple, un livre peut être divisé en chapitres, tandis qu'un article ne le peut pas.

```
\documentclass[a4paper,12pt]{article}

\usepackage[french]{babel}
\usepackage[latin1]{inputenc}

\begin{document}

Désormais, je peux saisir mon texte en français. Les accents sur les
lettres, de même que les signes diacritiques français, seront reconnus.

Remarquer l'alinéa automatique. Remarquez aussi la production
automatique des espaces fines et des ligatures, dans des mots tels
que: affiliation et fleuve.

\end{document}
```

Le choix de la classe de document (*article*), et des options (papier A4, police de base en 12 points) a déjà été commenté. En revanche, deux lignes importantes ont été ajoutées dans le préambule, auxquelles il faut prêter attention.

usepackage La commande `\usepackage` est très importante sous LaTeX. Elle permet de charger en mémoire des extensions au programme appelées *packages*. Les *packages* permettent soit de modifier le comportement par défaut de LaTeX, soit de lui ajouter des fonctionnalités supplémentaires. Il en existe des milliers, qu'il s'agisse d'extensions destinées à la saisie et au formatage, ou bien au dessin, à la production de tableaux complexes, aux mathématiques, etc.

Tous les *packages* portent un nom. La première extension que nous avons chargée s'appelle ainsi *babel*. *Babel* permet de saisir des textes dans toutes les langues du monde, vivantes ou mortes. L'option *french* qui a été passée à *babel* commandera à LaTeX de respecter automatiquement les règles de la typographie française. De même, LaTeX écrira en français les dates et des titres tels que *Sommaire*, *Bibliographie*, *Annexes*, etc.

La deuxième extension, *inputenc*, permet d'utiliser les caractères accentués des claviers français. Par défaut, en effet, seuls les caractères non-accentués du clavier anglo-américain sont reconnus. Quant à l'option *latin1*, elle correspond à la page de code comportant les caractères accentués des langues de l'Europe de l'Ouest dont le français fait partie.

Production du fichier DVI Pour obtenir, à partir du source que nous venons de saisir, un fichier `.dvi` susceptible d'être visualisé ou imprimé, il faut compiler ce source à l'aide de LaTeX. Dans l'exemple suivant, à la ligne de commande, on suppose que le source s'appelle `fichier.tex` :

```
[robert@kiddo:~]$ latex fichier.tex
This is pdfeTeX, Version 3.141592-1.21a-2.2 (Web2C 7.5.4)
entering extended mode
(./fichier.tex
LaTeX2e <2003/12/01>
Babel <v3.8d> and hyphenation patterns for american, french, german, ngerman, b
ahasa, basque, bulgarian, catalan, croatian, czech, danish, dutch, esperanto, e
```

```
stonian, finnish, greek, icelandic, irish, italian, latin, magyar, norsk, polis
h, portuges, romanian, russian, serbian, slovak, slovene, spanish, swedish, tur
kish, ukrainian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size12.clo))
(/usr/share/texmf/tex/generic/babel/babel.sty
(/usr/share/texmf/tex/generic/babel/frenchb.ldf
(/usr/share/texmf/tex/generic/babel/babel.def)
**********************************
* Local config file frenchb.cfg used
*
(/usr/share/texmf/tex/generic/babel/frenchb.cfg)))
(/usr/share/texmf/tex/latex/base/inputenc.sty
(/usr/share/texmf/tex/latex/base/latin1.def))
No file fichier.aux.
(/usr/share/texmf/tex/latex/lm/t1lmr.fd) [1] (./fichier.aux) )
Output written on fichier.dvi (1 page, 404 bytes).
Transcript written on fichier.log.
[robert@kiddo:~]$
```

Observons les messages renvoyés par le terminal. On a saisi à la ligne de commande :

```
[robert@kiddo:~]$ latex fichier.tex
```

Le programme a ensuite renvoyé une longue série de messages, avant de retourner, au moment de rendre la main :

```
Output written on fichier.dvi (1 page, 404 bytes).
```

Ce qui nous indique que la compilation a réussi : `fichier.dvi` est disponible.

La visualisation à l'écran de `fichier.dvi` se fait à l'aide d'un programme externe, appelé `xdvi`. La commande est la suivante :

```
[robert@kiddo:~]$ xdvi fichier.dvi
```

Une nouvelle fenêtre apparaît alors (voir figure 2.1 page ci-contre). Cette illustration permet déjà d'observer que LaTeX a pris soin sans qu'on ait eu à s'en préoccuper de la présentation typographique du texte. Par exemple, bien qu'il n'y ait pas d'espace avant les deux-points dans le source, LaTeX a automatiquement ajouté une espace fine, parce que l'espace fine est obligatoire, en typographie française, avant tout signe de ponctuation double. LaTeX a aussi produit des ligatures dans les séquences « ffi » et « fl », la césure des mots en fin de ligne, ainsi que les retraits d'alinéa.

Impression du document Plusieurs solutions existent pour imprimer le fichier `fichier.dvi`. Nous en présenterons ici simplement deux.

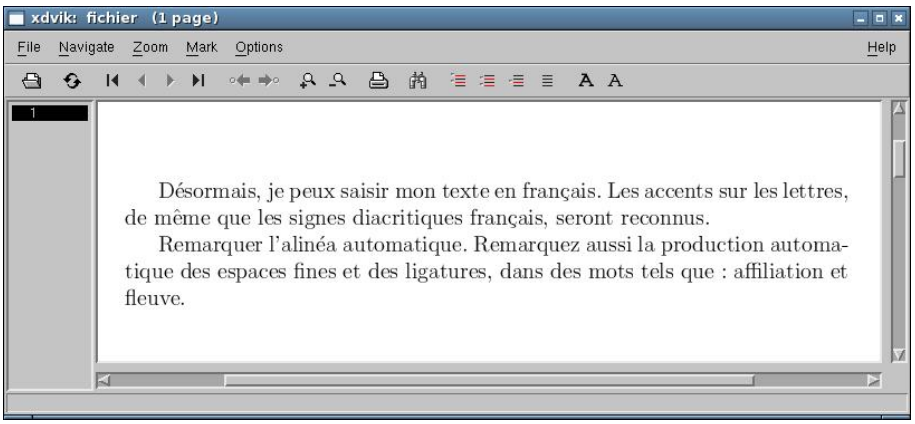

FIGURE 2.1: visualisation de `fichier.dvi`

PostScript *PostScript* est un langage de description de page qui permet de dessiner de manière vectorielle aussi bien des images que des polices de caractères. La plupart des imprimantes laser modernes interprètent nativement le format PostScript. Pour imprimer un tel fichier, il suffira de le transférer directement à l'imprimante. C'est le programme `dvips` qui se charge de convertir le fichier `fichier.dvi` en `fichier.ps` :

```
robert@kiddo:~$ dvips -o fichier.ps fichier.dvi
This is dvips(k) 5.95a Copyright 2005 Radical Eye Software (www.radicaleye.com)
' TeX output 2007.12.12:1234' -> fichier.ps
<tex.pro><texps.pro>. <cmr12.pfb>[1]
robert@kiddo:~$
```

Sur la ligne de commande, la séquence `-o␣fichier.ps` est une option qui a été passée au programme `dvips`, dans laquelle `-o` est pour *output*. En d'autres termes, cette option permet de donner un nom au fichier PostScript que le programme doit écrire. Quant à `fichier.dvi`, c'est l'*argument* passé à `dvips` : il s'agit simplement du nom du fichier `.dvi` qu'il faut traiter. Après validation de la ligne de commande, le programme retourne diverses informations, parmi lesquelles on remarquera le nom `fichier.ps`.

Que faire de `fichier.ps` ? Trois possibilités à ce niveau :

- si l'on dispose d'une imprimante PostScript, l'envoyer directement à l'imprimante, laquelle sort immédiatement la page ;
- à défaut d'imprimante PostScript, confier `fichier.ps` à un pilote d'imprimante capable de transformer le langage PostScript en un langage interprétable par l'imprimante. Sous Linux, c'est *GhostScript* qui se chargera de cette besogne. Nous y reviendrons plus tard ;
- confier `fichier.ps` à un imprimeur disposant d'imprimantes PostScript ;
- visualiser à l'écran `fichier.ps` à l'aide d'un programme spécialisé tel que *Ghostview* sous Linux. Dans ce cas, la commande à entrer est :
  `gv fichier.ps`
  Voir le résultat sur la figure 2.2 page suivante.

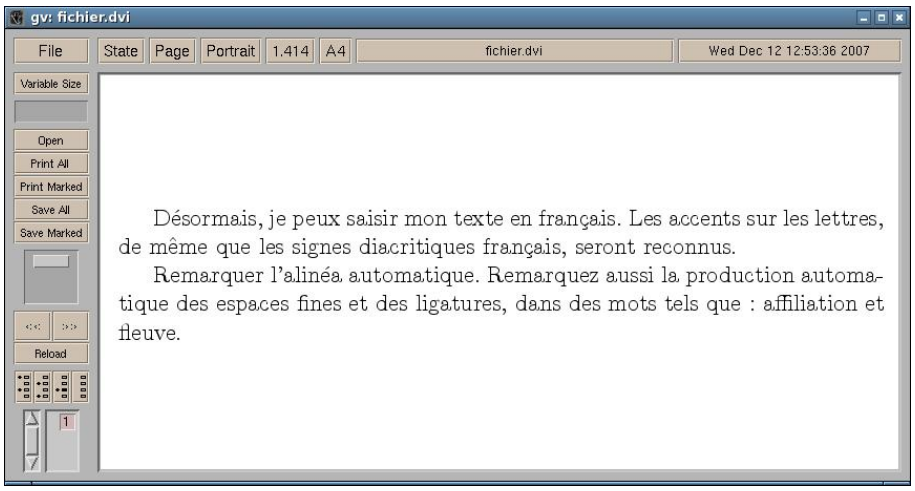

FIGURE 2.2: commande `gv fichier.ps`

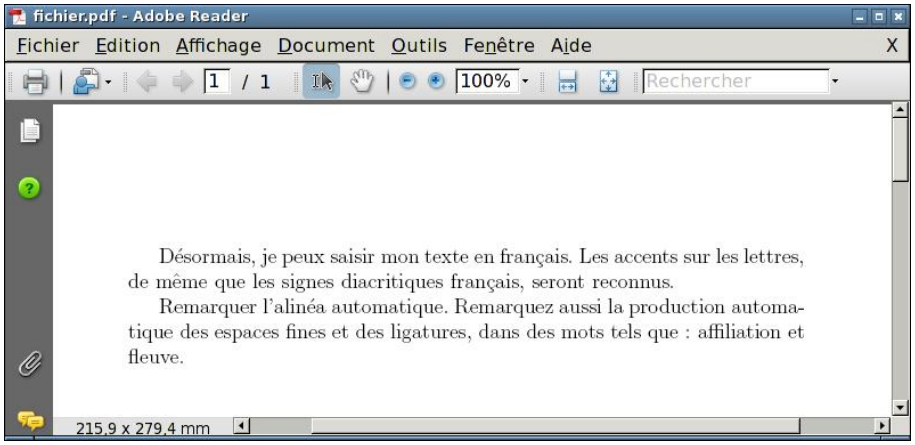

FIGURE 2.3: visualisation de `fichier.pdf`

PDF Il est aussi très facile de faire passer les fichiers `.dvi` au format *portable document format*, dit PDF. Le fichier `.pdf` obtenu pourra ensuite très facilement être visualisé sur tout ordinateur et imprimé sur toute imprimante à l'aide de logiciels spécialisés dont le plus connu est *Adobe Acrobat Reader*. La commande est la suivante :

```
robert@kiddo:~$ dvipdf fichier.dvi
robert@kiddo:~$
```

Comme on le voit, la commande ne retourne aucune information. Mais elle produit le fichier `fichier.pdf` que l'on ouvrira à l'aide d'*Acrobat Reader* (voir la figure 2.3 de la présente page). Il sera ensuite facile d'imprimer ou de transmettre le document.

2.3 Formatage élémentaire d'un document long

Nous pouvons à présent passer en revue les principales commandes. Dans cette section, nous présenterons les commandes de formatage qui sont les plus courantes dans les documents longs, à savoir :
- production d'une page de titre ;
- division du travail en parties, chapitres, sections, etc.
- notes de bas de page ;
- génération d'une table des matières.

Choix de la classe de document

Comme on l'a vu plus haut (2.1 page 9), tout document LaTeX commence obligatoirement par la commande

```
\documentclass
```

qui permet de choisir la classe du document que l'on souhaite produire. Dans cette section, on travaillera avec les éléments suivants :

```
\documentclass[12pt,a4paper]{book}
```

La classe *book* produit par défaut des documents imprimables en *recto-verso*. Elle permet d'introduire dans le document les divisions suivantes :
- partie ;
- chapitre ;
- section ;
- sous-section ;
- sous-sous-section ;
- paragraphe ;
- sous-paragraphe.

Par défaut, le titre de chaque partie apparaît seul sur une belle page. La page suivante est toujours laissée blanche. Les chapitres commencent eux aussi sur une belle page. LaTeX ajoutera si nécessaire une fausse page blanche avant le début du chapitre.

Nous passons enfin à la classe de document *book* deux options qui commandent respectivement le choix de la taille de la police de base (12 points) et du papier (format `A4`).

La page de titre

La composition d'une page de titre est normalement une tâche d'infographiste professionnel. Elle met en œuvre des logiciels spécialisés.

Il est cependant parfaitement possible de produire en langage LaTeX des pages de titre complexes. Nous nous bornerons ici à quelques éléments de base. Nous verrons plus loin comment produire des pages de titres élaborées.

Pour produire une page de titre simple, LaTeX a besoin de connaître, outre le titre lui-même, le nom de l'auteur de l'ouvrage. Ces renseignements sont habituellement placés dans le préambule. Les commandes sont les suivantes :

```
\title{Mon Premier document \LaTeX}
\author{Sophie Fonsec}
```

Il faut ensuite, dans le corps du document, indiquer à quel endroit on souhaite insérer la page de titre. Généralement, ce sera immédiatement après la commande `\begin{document}`. La commande à employer est :

```
\maketitle
```

Nous avons donc au total le code suivant :

Mon Premier document LaTeX

Sophie Fonsec

28 novembre 2010

```
\documentclass[12pt,a4paper]{book}

\title{Mon Premier document \LaTeX}
\author{Sophie Fonsec}

\begin{document}
\maketitle

[...]

\end{document}
```

[...]

Remarquez que LaTeX a automatiquement ajouté la date du jour. On peut cependant modifier ce comportement en ajoutant, toujours dans le préambule, la commande `\date`. Voici deux exemples possibles :

`\date{2 avril 2004}`	Pour une date fixe, par exemple le 2 avril 2004.
`\date{}`	Pour supprimer l'affichage de la date, placer un argument vide.

Les parties du document

La classe de document *book* que nous avons décidé d'utiliser autorise les découpages suivants :

Nom	**Commande** [1]
Partie	`\part{Nom}`
Chapitre	`\chapter{Nom}`
Section	`\section{Nom}`
Sous-section	`\subsection{Nom}`
Sous-sous-section	`\subsubsection{Nom}`
Paragraphe	`\paragraph{Nom}`
Sous-paragraphe	`\subparagraph{Nom}`

Le titre d'une *partie* apparaît seul, centré au milieu d'une belle page. Il est suivi d'une fausse page blanche. Les *chapitres*, quant à eux, démarrent toujours sur une belle page, à la différence des autres parties qui apparaissent simplement sur la page en cours.

Nous verrons plus loin comment affiner cette façon de diviser les documents, notamment quand nous chercherons à produire un *avant-propos*, au début du document, ou bien, à la fin du document, des annexes, des index, des bibliographies, des listes de tables ou de figures.

Par défaut, les parties du document sont numérotées par LaTeX lors de la compilation. Pour insérer la table des matières du document, il suffira de saisir, à l'endroit où l'on souhaite que la table soit générée, la commande suivante (généralement à la fin du document) :

1. Dans le document, on remplacera *Nom* ci-dessous par le nom réel de la partie du document.

Sommaire

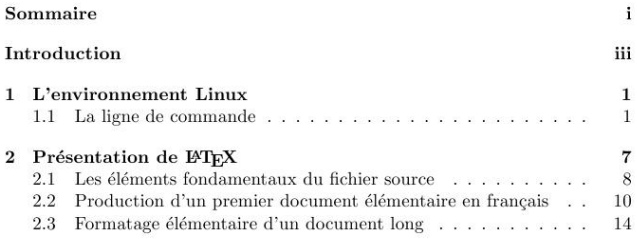

FIGURE 2.4: Exemple de table des matières

`\tableofcontents`

LaTeX insérera automatiquement dans la table le titre *Table des matières* (en français, grâce à l'extension *babel*) les numéros des parties, les points de suite et la pagination.

Il est aussi parfaitement possible de ne pas numéroter les parties du document. On dispose pour cela, face à chaque commande de sectionnement, d'une variante dite « étoilée ». Par exemple, tandis que

```
\chapter{Nom_du_chapitre}
```

sera numéroté, tandis que

```
\chapter*{Nom_du_chapitre}
```

ne le sera pas. Mais pour cette même raison, le chapitre ne sera pas mentionné dans la table des matières, à moins qu'on ne demande explicitement à LaTeX de le faire. La syntaxe est alors la suivante :

```
\chqapter*{Nom_du_chapitre} \addcontentsline{toc}{chapter}{Nom_du_chapitre}
```

C'est la deuxième ligne qui commande l'insertion dans la table des matières. Comme on le voit, cette commande prend trois arguments : le premier, `toc`, est pour *table of contents*. Il renvoie au nom de la table dans laquelle il faut insérer la ligne. Le deuxième, `chapter`, indique le niveau de profondeur auquel il faut insérer la ligne. Comme, dans notre exemple, il s'agit d'un chapitre, nous avons saisi *chapter*. Enfin, dans le dernier argument, on saisit le nom réel du chapitre en question. On trouvera en figure 2.4 de la présente page un exemple de table des matières, avec des éléments non-numérotés à côté d'éléments numérotés.

Les notes de bas de page

C'est le dernier élément que nous aborderons dans ce rapide tour d'horizon.

La production de notes de bas de page est chose très aisée sous LaTeX. Il suffit, en cours de saisie du texte, d'insérer la commande `\footnote` de la manière suivante :

Effet souhaité	Commande	Résultat
Mise en valeur	`\emph{Exemple}`	*Exemple*
Forte mise en valeur	`\textbf{Exemple}`	**Exemple**
Texte *Machine à écrire*	`\texttt{Exemple}`	`Exemple`
Petites capitales	`\textsc{Exemple}`	Exemple
Texte « penché »	`\textsl{Exemple}`	*Exemple*
Texte sans sérif	`\textsf{Exemple}`	Exemple

Table 2.3: Mise en valeur de mots

The Hippocratic physicians did not make any particular distinction between veins and arteries, and ignored the arterial pulse. [a]

a. Apart from palpitations or throbbing at the temples which were well known as pathological symptoms, for instance in the group of the *Epidemics* treatises.

```
The Hippocratic physicians did not
make any particular distinction
between veins and arteries, and
ignored the arterial
pulse.\footnote{Apart from palpitations
or throbbing at the temples which were
well known as pathological symptoms,
for instance in the group of the
\emph{Epidemics} treatises.}
```

Le texte de la note a été saisi entre accolades, juste après la commande `\footnote`. Comme on le voit, LaTeX se charge de produire automatiquement l'appel de note exactement au point ou la commande `\footnote` a été saisie ; il place au bas de la page le texte de la note qui est saisi à la suite entre accolades, sous un filet séparateur. Le texte de la note apparaît aussi dans un corps de police plus petit [2].

2.4 Mise en forme du texte

Maintenant que nous connaissons quelques commandes de structuration du texte, nous pouvons aborder très rapidement les principales commandes de mise en forme. La logique de LaTeX est toujours la même : l'auteur du texte indique à LaTeX qu'il souhaite mettre en valeur un mot, un membre de phrase, ou bien plusieurs lignes. LaTeX se chargera des choix typographiques dans le respect des règles nationales.

Mise en valeur des mots

Le tableau 2.3 de la présente page présente les commandes les plus usuelles, et les effets qu'elles produisent.

Dans la présentation des travaux de recherche en lettres et sciences humaines, la commande la plus utilisée est évidemment `\emph`, pour l'anglais *emphasize*, « mettre en valeur ». Vous remarquerez qu'il s'agit d'une commande sémantique : l'auteur du texte laisse à LaTeX le soin de choisir la police appropriée en fonction de l'environnement dans lequel le mot à mettre en valeur se

2. Dans l'exemple ci-dessus, l'appel de la note est en lettres italiques. Mais par défaut, LaTeX numérote les notes de bas de page en chiffres arabes. La numérotation redémarre à chaque nouveau chapitre.

trouve. Par exemple, dans un paragraphe en italiques, LaTeX présentera le mot en caractère romains, et *vice versa*.

Mise en valeur de paragraphes

Nous nous intéressons ici non pas à la mise en valeur de mots, pas de paragraphes entiers. Il est d'usage, en effet, de présenter sous la forme de paragraphes séparés du texte principal les citations de plus de deux ou trois lignes, autrement appelée « citations longues ». Les deux environnements les plus utilisés à ce titre sont `quote` et `quotation`[3].

Les environnements, sous LaTeX, prennent place entre des commandes `\begin` et `\end`. Voici un exemple de citation correctement formatée :

> Moreover, which is to me the strongest evidence, one find in this *katástasis* of Perinthus some unquestionable usages of the first person singular which may very well mean that this doctor, though he arrived in Perinthus accompanied by other doctors and/or disciples, was the only one taking on the responsibility for his statements and observations. The following two examples from this *katástasis* will show it clearly. – Robert Alessi

Pour obtenir ce formatage, nous avons utilisé l'environnement suivant :

```
\begin{quote}\footnotesize
  Moreover, which is to me the strongest evidence...
\end{quote}
```

Il faut ici commenter la commande `\footnotesize`, placée juste après l'ouverture de l'environnement `quote` : cette commande a pour effet de basculer la taille de police de caractères de tout le texte de la citation dans la taille de caractères des notes de bas de page. Mais comme la commande a été saisie *à l'intérieur* de l'environnement `quote`, elle cesse d'agir au moment de la fermeture de l'environnement.

Dernière remarque : comme on peut le voir après compilation, le paragraphe de la citation n'a pas d'alinéa. Si l'on veut le respect de l'alinéa, il faut utiliser à la place l'environnement `quotation`.

Listes

Les environnements de liste permettent de mettre en valeur des séries d'élements. Ils peuvent éventuellement être ordonnés dans des listes numérotées. Nous présenterons ici les trois environnements fondamentaux les plus utilisés.

Itemize

L'environnement `itemize` permet de présenter simplement des listes non numérotées. À l'intérieur de l'environnement, chaque élément devant être précédé d'un tiret doit être précédé de la commande `\item` suivie d'un espace, de la manière suivante :

3. *To quote*, en anglais, veut dire « citer » ; *quotation* veut dire « citation ».

Liste du matériel de base pour la collation d'un manuscrit :
– feuilles de papier blanc ;
– crayon à mine de plomb ;
– taille crayon ;
– compte-fils.

```
Liste␣du␣matériel␣de␣base
pour␣la␣collation
d'un␣manuscrit:
\begin{itemize}
\item␣feuilles␣de␣papier␣blanc;
\item␣crayon␣à␣mine␣de␣plomb;
\item␣taille␣crayon;
\item␣compte-fils.
\end{itemize}
```

Il est très facile de créer, à l'intérieur d'une liste donnée, une sous-liste d'éléments : il suffit pour cela de créer au bon endroit, à l'intérieur de l'environnement `itemize` un second environnement `itemize` imbriqué, comme dans l'exemple suivant :

Liste du matériel de base pour la collation d'un manuscrit :
– feuilles de papier blanc ;
– instruments d'écriture :
  – crayon à mine de plomb ;
  – taille crayon.
– compte-fils.

```
Liste du matériel de base
pour la collation
d'un manuscrit:
\begin{itemize}
  \item feuilles de papier blanc;
  \item instruments d'écriture:
    \begin{itemize}
       \item crayon à mine de plomb;
       \item taille crayon.
    \end{itemize}
  \item compte-fils.
\end{itemize}
```

Enumerate

L'environnement `enumerate` permet de créer automatiquement des listes numérotées. Les règles sont les mêmes que celles de l'environnement `itemize`. Reprenons ici de cette façon les éléments de la liste précédente.

Liste simple

Liste du matériel de base pour la collation d'un manuscrit :

1. feuilles de papier blanc ;
2. crayon à mine de plomb ;
3. taille crayon ;
4. compte-fils.

```
Liste␣du␣matériel␣de␣base
pour␣la␣collation
d'un␣manuscrit:
\begin{enumerate}
\item␣feuilles␣de␣papier␣blanc;
\item␣crayon␣à␣mine␣de␣plomb;
\item␣taille␣crayon;
\item␣compte-fils.
\end{enumerate}
```

Listes imbriquées

Liste du matériel de base pour la collation d'un manuscrit :

1. feuilles de papier blanc ;
2. instruments d'écriture :
   a) crayon à mine de plomb ;
   b) taille crayon.
3. compte-fils.

```
Liste du matériel de base
pour la collation
d'un manuscrit:
\begin{enumerate}
  \item feuilles de papier blanc;
  \item instruments d'écriture:
    \begin{enumerate}
       \item crayon à mine de plomb;
       \item taille crayon.
    \end{enumerate}
  \item compte-fils.
\end{enumerate}
```

Dans ce dernier exemple, on voit que LaTeX redémarre la numérotation de la sous-liste, tout en la présentant automatiquement de façon différente.

Description

`description` est un environnement permettant de mettre en valeur des éléments dont on donne une descrption ou une définition. La syntaxe est à peine plus complexe que celle des autres environnements de liste : il faut passer en option à la commande `\item`, entre crochets droits, le nom de l'élément à décrire ou à définir, de la manière suivante :

Conspectus siglorum :

M *Marcianus gr. 269* ;

I *Parisinus gr. 2140* ;

H *Parisinus gr. 2142* ;

R *Vaticanus gr. 277*.

```
  Conspectus siglorum:
\begin{description}
\item[M] \emph{Marcianus gr. 269};
\item[I] \emph{Parisinus gr. 2140};
\item[H] \emph{Parisinus gr. 2142};
\item[R] \emph{Vaticanus gr. 277}.
\end{description}
```

2.5 Les tableaux

LaTeX permet de mettre en page des tableaux extrêmement complexes. On s'en tiendra ici à quelques notions de base. On apprendra comment déterminer le nombre de colonnes, formater les cellules, tracer les lignes, joindre les cellules, présenter des tableaux sur plus d'une page.

Observons le code suivant avant de le commenter :

```
 1  \begin{tabular}{|l|l|l|}
 2    \hline
 3    Sigle & Cote & Bibliothèque\\
 4    \hline\hline
 5    M & \emph{Marcianus gr. 269} & Venise \\ \hline
 6    V & \emph{Vaticanus gr. 276} & Vatican \\ \hline
 7    I & \emph{Parisinus gr. 2140} & Paris \\ \hline
 8    R & \emph{Vaticanus gr. 277} & Vatican\\ \hline
 9    H & \emph{Parinisus gr. 2142}& Paris \\ \hline
10  \end{tabular}
```

Commentaire Le tableau est défini dans un environnement `tabular` (lignes 1 et 10). Remarquez que la commande `\begin{tabular}` admet un deuxième paramètre dans lequel on définit le nombre de colonnes du tableau. Dans notre exemple, trois colonnes justifiées à gauche, et séparées par un filet vertical : la lettre `l`, pour *left*, signifie « insérez une colonne justifiée à gauche », tandis que le caractère *pipe* (`|`, appelé par `Alt-Gr-6` sur les claviers de type PC) définit l'endroit où doit se trouver le filet vertical séparateur. Pour obtenir une colonne de texte centré, il aurait fallu saisir `c` à la place de `l`, et pour une colonne justifiée à droite, la lettre `r` pour l'anglais *right*.

Le tableau proprement dit commence dès la ligne 2. Le code que nous avons saisi permet d'ailleurs de « lire » le tableau :

ligne 2 production d'une ligne horizontale, la ligne supérieure du tableau, par la commande simple `\hline` pour l'anglais *horizontal line* ;

ligne 3 première ligne du tableau ; remarquez que les cellules sont séparées par le signe `&`. À la fin de la ligne, le passage à la ligne suivante est commandé par le double *backslash* (`\\`) ;

ligne 5 production de deux lignes horizontales, pour détacher la première ligne du tableau qui est une ligne de titre ;

lignes 5 à 9 remplissage du tableau avec les données, ligne après ligne, suivant la syntaxe définie ci-dessus ; à la fin de chaque ligne, production du retour à la ligne et d'une ligne horizontale ;

ligne 10 fermeture de l'environnement `tabular`.

Le résulat est le suivant après compilation :

Sigle	Cote	Bibliothèque
M	*Marcianus gr. 269*	Venise
V	*Vaticanus gr. 276*	Vatican
I	*Parisinus gr. 2140*	Paris
R	*Vaticanus gr. 277*	Vatican
H	*Parinisus gr. 2142*	Paris

Comme on le voit, LaTeX a calculé automatiquement la largeur des colonnes en fonction du contenu. La ligne de titre du tableau, séparée de la zone consacrée aux données par une double ligne, se détache nettement.

À noter cependant : les tableaux de l'environnement `tabular` sont faits pour tenir sur une seule et même page. Si l'on souhaite présenter des tableaux plus longs, susceptibles de s'étendre sur plusieurs pages, il faut charger une extension supplémentaires. On présentera ici à titre d'exemple l'extension `longtable`. Il faut la charger dans le préambule du document à l'aide de la commande

```
\usepackage{longtable}
```

Ce package définit alors un nouvel environnement `longtable`, dont la syntaxe est la même que celle de `tabular`. Voici à titre comment se présenterait le tableau de l'exemple précédent dans cet environnement :

```
1   \begin{longtable}{|l|l|l|}
2     \hline
3     Sigle & Cote & Bibliothèque\\
```

```
4     \hline\hline
5     \endhead
6     M & \emph{Marcianus gr. 269} & Venise \\ \hline
7     V & \emph{Vaticanus gr. 276} & Vatican \\ \hline
8     I & \emph{Parisinus gr. 2140} & Paris \\ \hline
9     R & \emph{Vaticanus gr. 277} & Vatican\\ \hline
10    H & \emph{Parinisus gr. 2142}& Paris \\ \hline
11  \end{longtable}
```

Remarquez, à la ligne 5, la commande `\endhead` : elle permet de délimiter la zone d'en-tête du tableau : si le tableau vient à se poursuivre sur la page suivante, tout ce qui précède la commande `\endhead` sera repris sur la nouvelle page.

On terminera cette rapide présentation par deux éléments supplémentaires de formatage de cellules.

Les colonnes à largeur fixe servent à obtenir, dans les tableaux, des cellules dont la largeur ne dépende pas du contenu. Quand le texte atteint la limite fixée, LaTeX introduit passe à la ligne suivante, à l'intérieur de la même cellule. Cela permet de construire de petits paragraphes dans les tableaux. Le code suivant permet de construire un tableau avec une colonne de 5 cm de largeur :

```
\begin{tabular}{|l|p{5cm}|}
  \hline
Colonne variable & Colonne de largeur fixe. Quand le texte atteint la
limite, \LaTeX{} passe automatiquement à la ligne suivante. \\
\hline
\end{tabular}
```

Le résultat est le suivant :

Colonne variable	Colonne de largeur fixe. Quand le texte atteint la limite, LaTeX passe automatiquement à la ligne suivante.

La syntaxe de formatage est donc : `p{`*dimension*`}`, où *dimension* peut être exprimée en de nombreuses unités, dont les centimètres (`cm`), comme dans l'exemple fourni, et les pouces (`in`). Il peut être utile, cependant, de fixer la largeur des colonnes non pas de manière absolue, mais de manière relative, en référence à la longueur de la ligne de texte, notée `\textwidth` sous LaTeX. Ainsi, pour une largeur de colonne égale à 50 % de la longueur de la ligne de texte, la commande serait :

```
\p{0.5\textwidth}
```

Remarquez que le séparateur des décimales est le point (convention anglo-saxonne), et non la virgule.

Colonnes fusionnées La commande qui permet de joindre les cellules d'un tableau est la suivante :

```
\multicolumn{x}{<format>}{<contenu>}
```

où `x` est le nombre de cellules à joindre à partir de la cellule courante comprise, `<format>` la lettre caractérisant l'alignement, éventuellement entourée de filets séparateurs, comme nous l'avons vu plus haut page 23, et `<contenu>` le contenu de la cellule fusionnée. En voici un exemple :

```
\begin{tabular}{|l|l|l|}
  \hline
un & deux & trois \\ \hline
quatre & \multicolumn{2}{c|}{cinq} \\ \hline
six & sept & huit \\
\hline
\end{tabular}
```

Pour le résultat suivant :

un	deux	trois
quatre	cinq	
six	sept	huit

2.6 Objets flottants

Les *objets flottants* sont des éléments tels que des figures, des images scannées ou des tableaux que LaTeX devra placer à l'endroit le plus approprié possible. S'il s'agit d'une image de grande taille, par exemple, LaTeX pourra la placer directement sur une nouvelle page, voire en fin de section ou de chapitre. De cette façon, les pages seront toujours remplies de manière équilibrée, et l'on ne trouvera jamais, au bas de certaines pages, de vastes espaces laissés vides à chaque fois que les dimensions de l'objet que l'on souhaite insérer dépassent l'espace disponible sur la page entre le point d'insertion et lebas de la page.

Nous présenterons ici deux types d'objets flottants : les tables et les figures.

Tables

L'environnement `table` permet de donner une légende à chaque tableau d'un document, de le numéroter, et de répertorier tous les tableaux du document dans une *liste des tableaux* que l'on fera figurer à côté de la table des matières ou du sommaire. Observons le code suivant :

```
\begin{table}
\centering
\begin{tabular}{|l|l}
\hline
\textbf{Sigle} & \textbf{Bibliothèque} \\ \hline\hline
M & \emph{Marcianus gr. 269}\\ \hline
V & \emph{Parisinus gr. 2140}\hline
\end{tabular}
\caption{Liste des manuscrits}
\end{table}
```

À l'intérieur de l'environnement `table`, on trouve notamment les commandes suivantes :

Sigle	Bibliothèque
M	*Marcianus gr. 269*
V	*Parisinus gr. 2140*

Table 2.4: Liste des manuscrits

– `\centering` qui s'applique sur la totalité de l'environnement `table`. Commande facultative qui permet d'obtenir un tableau centré ;
– `\caption` qui permet d'affecter une légende au tableau. Le texte de la légende se retrouvera aussi bien au bas du tableau que dans la *liste des figures*.

Le résultat apparaît dans le tableau 2.4 de la présente page.

Figures

Comme son nom l'indique, l'environnement `figure` permet d'insérer des figures, des graphes, des dessins vectoriels ou des images scannées dans le document. Ces figures pourront recevoir une étiquette (*label*), et être répertoriées dans une *liste des figures*. Observons le code suivant :

```
\begin{figure}
\begin{center}
\includegraphics[width=\textwidth]{logo_umr-small}
\end{center}
\caption{Logo UMR 8167}
\label{fig:logoUMR}
\end{figure}
```

À l'intérieur de l'environnement `figure`, l'insertion de l'image se fait par la commande `\includegraphics` qui prend comme argument le nom du fichier graphique à insérer, ici `logo_umr-small`. Remarquez qu'il n'a pas été nécessaire de préciser l'extension du nom de ce fichier (en l'occurrence `.jpg`). À ce titre, quelques précisions sont nécessaires. En effet, pour pouvoir utiliser la commande `includegraphics`, il faut avoir chargé dans le préambule le package `graphicx` par la commande
`\usepackage{graphicx}`
Sans précision supplémentaire, LaTeX n'acceptera que des fichiers graphiques au format PostScript encapsulé (extension `.eps`). Si l'on souhaite insérer des fichiers aux formats *jpeg* ou *png*, plus courants, il faut passer au package `graphicx` l'option `pdftex`, sous la forme suivante :
`\usepackage[pdftex]{graphicx}`
Dès cet instant, LaTeX produira par défaut, lors de la compilation, un fichier au format `pdf`.

Dernière observation : dans le code ci-dessus, nous avons passé à la commande `\includegraphics` une option `[width=\textwidth]` qui est une commande de mise à l'échelle automatique, en fonction de la largeur (*width* en anglais) de la ligne de base du document, ici spécifiée par la valeur `\textwidth` que nous avons décrite page 24.

Quant aux commandes `\label` et `\caption`, elles ont été décrites au point 2.6 page précédente. Le résultat de cet ensemble de commandes apparaît dans la

FIGURE 2.5: Logo UMR 8167

figure 2.5 de la présente page. Comme on le voit, l'image a été mise à l'échelle, et occupe toute la largeur de la ligne de base.

Deux commandes, enfin, méritent d'être connues :
- `\listoftables` qui produit, à l'endroit du document où elle est insérée, une liste numérotée de toutes les tables insérées dans le document, avec indication de la page où elles se trouvent ;
- `\listoffigures` qui produit le même résultat sur les figures.

Il est d'usage de saisir ces commandes juste après le sommaire du document, ou bien juste avant la table des matières. Chaque table ou chaque figure sera décrite par les mots saisis dans les commandes `\caption` des environnements correspondants.

Chapitre 3

Structuration avancée

Le chapitre précédent portait sur la mise en forme d'éléments simples, tous susceptibles de tenir sur une même page. Nous allons à présent aborder des commandes de structuration plus complexes portant sur diverses parties du document, telles que les références croisées, l'insertion de données bibliographiques et les index.

3.1 Les références croisées

Les références croisées servent à renvoyer le lecteur à divers endroits du document, soit plus haut, soit plus bas. Elles sont indispensables dans tout document long correctement structuré.

Elles mettent en jeu deux éléments fondamentaux :

- *la source*, qui est l'endroit du document où l'on souhaite faire référence à un point déjà traité plus haut ou bien que l'on a l'intention de traiter plus bas ;
- *la cible*, qui est l'endroit du document où l'on souhaite renvoyer le lecteur.

Pour générer une référence croisée, il faut donc d'abord repérer, dans le document, l'endroit vers lequel on souhaite renvoyer le lecteur (*cible*). Cet endroit doit être marqué par la commande

```
\label{nom_cible}
```

Label, en anglais, signifie « étiquette » ; quant au nom à donner à l'étiquette (`nom_cible` dans notre exemple), vous pouvez librement le choisir. Veillez cependant à respecter quelques règles de bon sens :

- donnez des noms à la fois courts et explicites ;
- n'insérez dans les étiquettes ni espacements ni caractères accentués ;
- faites précéder chaque type d'étiquette d'un préfixe qui vous permettra de les classer facilement. Par exemple
  - `sec:nom_section` pour une partie de document ;
  - `tab:nom_tableau` pour un tableau ;
  - `fig:nom_figure` pour une figure ;
  - `ref:nom_référence` pour une autre référence.

La deuxième étape consiste enfin à insérer la référence à l'endroit *source*, celui où doit se trouver la page où figure la cible. Les deux commandes les plus répandues sont les suivantes :

- `\pageref{nom_cible}`, qui permettra à LaTeX de générer automatiquement le numéro de la page où se trouve la cible ;
- `\ref{nom_cible}`, qui permettra à LaTeX de désigner la cible par son numéro de référence, c'est-à-dire, selon le cas, un numéro de chapitre, de section, ou encore un numéro de note de bas de page.

Remarque importante Pour que le système fonctionne, il faut évidemment que les deux étiquettes `nom_cible`, à la fois dans `\label` et dans `\ref` ou `\pageref` soient strictement identiques.

Exemple d'application Dans un document donné, on souhaite renvoyer le lecteur à la page où l'on a déjà traité de l'aoriste grec. On procédera de la manière suivante :

1. repérage, dans le document, de l'endroit où l'on a traité de l'aoriste ;
2. insertion, à cet endroit, de l'étiquette `\label{ref:aoriste}` ;
3. à l'endroit source, saisie d'une phrase telle que : « comme nous l'avons déjà vu dans notre exposé sur l'aoriste en `\ref{ref:aoriste}` page `\pageref{ref:aoriste}`, l'aoriste sigmatique est un temps athématique. »
4. LaTeX produira alors une phrase telle que « comme nous l'avons déjà vu dans notre exposé sur l'aoriste en 2.3 page 46, l'aoriste sigmatique est un temps athématique. »

Cet exemple montre qu'une fois saisie l'étiquette, on peut parfaitement combiner les commandes `\ref` et `\pageref` à l'endroit source pour obtenir respectivement le numéro de la section et le numéro de la page où se trouve la cible. Si la cible s'était trouvée dans une note de bas de page, `\ref` aurait produit le numéro d'appel de la note.

Le package *varioref*

Le package `varioref`, dont l'usage est conseillé, permet d'obtenir un formatage plus intelligent des références croisées. En fonction de l'endroit où se trouve la cible, il est capable de produire des références telles que : *page précédente*, *page suivante, page ci-contre*, etc.

Pour l'utiliser, il faut d'abord le charger avec l'option de langue française :

```
\usepackage[french]{varioref}
```

On retiendra surtout trois commandes utiles de `varioref` :

- `\vref{nom_cible}` : renvoie automatiquement une référence telle que : « 2.3 page 46 », c'est-à-dire à la fois le numéro de référence de la cible et sa page, précédée du mot « page » ;
- `\vpageref{nom_cible}` : renvoie seulement le numéro de la page, sans le numéro de référence. Ce pourra être aussi bien, selon le cas, « page 46, page précédente, page ci-contre, etc. »
- `\vpagerefrange{cible1}{cible2}` : renvoie à une suite de pages dont la première est celle de `cible1` et la deuxième celle de `cible2`. Avec cette commande, on pourra par exemple obtenir un texte tel que : « ce point a déjà été examiné pages 46-53 ». Dans ce cas, il faudra penser à placer non pas une, mais deux étiquettes dans le texte.

Exemple Le code suivant :

```
Pour savoir ce que signifie \verb|\textwidth|, reportez-vous au point
\vref{ref:textwidth}\ldots

La définition de \verb|\textwidth| se trouve
\vpageref{ref:textwidth}\ldots
```

Produit ceci :

> Pour savoir ce que signifie `\textwidth`, reportez-vous au point 2.5 page 24...
> La définition de `\textwidth` se trouve page 24...

3.2 La bibliographie

LaTeX permet très facilement de produire automatiquement des bibliographies très élégantes, et surtout parfaitement conformes avec les normes internationales. La génération de la bibliographie elle-même se fait à l'aide d'un logiciel externe qui se nomme BibTeX.

Il faudra bien sûr apprendre à construire la base de données bibliographique dans laquelle on puisera [1]. Cette base doit être saisie dans un fichier texte séparé dont l'extension doit être `.bib`.

Les étapes par lesquelles il faut passer pour produire une bibliographie dans un document LaTeX sont les suivantes :

- création d'un fichier source `.tex` dans lequel on saisit des références bibliographiques à l'aide de la commande `\cite{étiquette}` ;
- compilation de ce fichier à l'aide de LaTeX ; outre le fichier de sortie `.dvi`, LaTeX produit aussi un fichier dont l'extension est `.aux` ;
- création de la base de données bibliographique, sous la forme d'un fichier `.bib`, dans lequel chaque entrée porte une *étiquette* unique, celle-là même que l'on utilise dans la commande `\cite` ;
- compilation grâce à BibTeX du fichier `.aux`. La commande est de la forme :
  `bibtex␣fichier.aux` [2]
- nouvelle compilation du fichier LaTeX pour incorporer la bibliographie. Il est prudent, en fait, de compiler le fichier deux fois pour que toutes les références soient en place.

Dans le fichier LaTeX lui-même, on saisira les commandes suivantes :

- *dans le préambule*, chargement d'un package de formatage de bibliographie, puis choix d'un style de blibliographie. Choix conseillé :
  ```
  \usepackage{natbib}
  \bibliographystyle{apalike-fr}
  ```

1. Des éditeurs de texte tels qu'Emacs ou Vim offrent des séries de raccourcis qui permettent de saisir facilement des entrées BibTeX. Je conseille vivement l'utilisation de JabRef, logiciel open-source multiplateforme, que l'on peut se procurer à l'adresse suivante : `http://jabref.sourceforge.net` Documentation en ligne à l'adresse suivante : `http://jabref.sourceforge.net/help/fr/Contents.php`

2. En fait, de même qu'il n'est pas obligatoire de donner l'extension `.tex` quand on compile un fichier `.tex`, il n'est pas nécessaire de donner l'extension `.aux` après BibTeX. En pratique, on saisira donc tout simplement `latex fichier` puis `bibtex fichier`

– à la fin du document, à l'endroit où l'on souhaite intégrer la bibliographie elle-même, insertion de la commande `\bibliography`, qui prend comme argument le nom du fichier `.bib` de base de données bibliographiques que l'on a construit séparément. Attention : si ce fichier n'est pas dans votre répertoire courant, il faudra le faire précéder de son chemin d'accès [3]. Par exemple :
`\bibliography{biblio}` ou bien :
`\bibliography{/home/robert/travail/biblio}`
Observez qu'il n'est pas nécessaire de spécifier l'extension `.bib` de votre fichier de bibliographie.

Si l'on applique tous les conseils qui précèdent, le fichier LaTeX se présente donc à peu près sous la forme suivante :

```
\documentclass{article}
[...]
\usepackage{natbib}
\bibliographystyle{apalike-fr}
[...]
\begin{document}
[...]

\bibliography{/home/robert/travail/biblio}
\end{document}
```

À l'intérieur du document, il reste désormais à saisir les références bibliographiques à des ouvrages précis. Comme on l'a dit plus haut, chaque entrée bibliographique dans le fichier `.bib` est désignée par une étiquette unique.

Pour aller plus loin Deux commandes, dans le fichier LaTeX, sont particulièrement utiles. Elles sont intégrées dans le package *natbib*.

– `\citet{}`, pour des citations comportant le nom de l'auteur suivi de la date entre parenthèses ;
– `\citep{}`, pour que l'ensemble de la citation, nom de l'auteur et date, soit entre parenthèses.

Notez enfin que toute commande `\cite{}` peut recevoir une option entre crochets droits, qui sera utilisée pour donner la ou les pages auxquelles on veut faire allusion.

Soit le code suivant :

```
Voici quelques exemples de citations: sur ce point, lire
\citet[p.~234]{Jouanna1996}, \citet[p.~543]{Jouanna1992} et
\citet[p.~175, n.~3]{Jouanna1990}. Voir aussi \citet{Alessi2005} et encore
\citet[p.~56]{Alessi2005a}.
```

Cela donne :

Voici quelques exemples de citations : sur ce point, lire **?**, p. 234, **?**, p. 543 et **?**, p. 175, n. 3. Voir aussi **?** et encore **?**, p. 56.

3.3 L'indexation

3. Sur les notions de répertoire courant et de chemin d'accès, voir le point 1.1 page 3.