Chapitre premier

Présentation

LaTeX est un outil de compilation de documents. Ce n'est pas un traitement de texte au sens classique du terme. Il te permettra, à l'aide de mots clefs spécifiques, de dire comment tu veux voir apparaître ton document, la logique qui le caractérise, tes conventions de mise en page, etc., pour qu'il puisse, tout seul, produire un document le plus convenable possible.

LaTeX est particulièrement étudié pour les mathématiques, il permet la mise en forme de rapports, de thèses, de polys... d'une qualité exceptionnelle. On peut, sans trop se tromper, dire que personne ne vient lui faire d'ombre pour tout ce qui touche à l'édition d'équations, et que ses possibilités sont grandes pour tout le reste.

Cependant, LaTeX n'est pas fait pour produire des affiches et ne peut par conséquent pas réaliser simplement des travaux d'édition de texte tordus. Il y a pour cela d'autres outils puissants et gratuits, dont `gimp`, `tcm`, `xfig` et sans doute d'innombrables autres.

LaTeX compile des fichiers ascii standards, avec l'extension `.tex`. Il reconnaît ses commandes au fait que celles-ci commencent par un « backslash[1] » (\), et reconnaît certains caractères comme étant spéciaux.

LaTeX s'inscrit dans la longue tradition des langages à balises, tradition que l'on peut faire remonter aux débuts de la typographie, quand les éditeurs annotaient les manuscrits des auteurs pour indiquer au typographe la mise en page à effectuer : telle marque indiquait de mettre le texte en italique, telle autre d'aligner différemment, etc. En cela, il ressemble assez à d'autres choses plus modernes, comme le HTML ou le XML, mais avec des contraintes différentes (LaTeX est destiné à produire des documents imprimés, avec un souci d'excellence, là où HTML est destiné à produire des documents éphémères à l'écran, avec un souci de simplicité) et des possibilités plus étendues.

[1] Ce symbole, le \, ne porte pas de nom précis et classique en français, en anglais non plus d'ailleurs puisqu'il ne correspond à rien en typographie. Cependant, son symétrique, /, est appelé en anglais, un slash, et en français, un divisé. D'où l'appellation anglo-saxonne usuelle de backslash, parfois francisée en anti-slash, voire pour certains puristes en anti-divisé. Mais moi, anti-divisé, je trouve que c'est pas beau. Alors dans ce livre je parlerai de backslash, voire d'anti-slash, ou aussi de caractère d'échappement, puisque c'est le rôle qu'il joue pour TeX. En effet, en informatique, lorsqu'un caractère permet de différencier une commande des données normales, on l'appelle caractère d'échappement.

Chapitre premier
Présentation

1 Pré-requis : compiler un document

Souvent, j'ai rencontré des gens qui voulaient utiliser LaTeX parce qu'ils voyaient bien l'intérêt de la chose : des professeurs de mathématiques qui ne voulaient pas remettre des énoncés manuscrits, des amateurs de langues orientales qui voulaient mêler le français et leur langue d'étude (japonais, coréen, araméen, hébreux biblique ou moderne, etc.).

L'idée d'utiliser un outil complexe, demandant efforts et attention pour obtenir un résultat de bonne qualité ne les rebutait pas, mais ils avaient absolument besoin d'aide pour qu'on leur mette le pied à l'étrier.

Ça peut sembler simple, si tu es comme moi informaticien, mais les utilisateurs dont je viens de parler vont buter sur des obstacles pourtant bien faciles à contourner.

Voilà les manipulations élémentaires qu'il faut absolument savoir faire avec son ordinateur, et les quelques notions qu'il faut maîtriser pour pouvoir utiliser LaTeX.

1.1 Notions élémentaires

Il faut un minimum de vocabulaire, ce qui n'est pas absolument évident quand on a jamais croisé un ordinateur. Je ne suis pas le mieux placé pour te le donner, et des auteurs plus doués que moi dans ce domaine ont écrit des ouvrages pour débutants.

Il faut savoir ce qu'est un répertoire, avec la notion de ce qu'est le répertoire courant, savoir ce qu'est un fichier, savoir ce qu'est un programme, idéalement faire la différence entre une application et un programme[2].

Pour comprendre LaTeX, il vaut mieux savoir ce qu'est un langage informatique, mais cette notion-là, contrairement aux précédentes, je me sens capable de te l'expliquer. Un langage, de manière générale, c'est une langue très formelle, dans laquelle on donne des instructions à l'ordinateur, pour qu'il les exécute. Ce qui différencie les différents langages, ce sont les ordres qu'ils arrivent à modéliser, et la forme que l'on donne à ces ordres.

Certains langages sont faits pour piloter le déplacement des robots ; dans ces langages, le concept « faire un quart de tour avec le moteur numéro 3 » a un sens. D'autres sont faits pour faire des dessins à l'écran, et n'ont pas ce concept, mais auront plus probablement le concept « tracer le cercle qui passe par les points A, B et C ».

Il ne faut pas confondre un langage et les outils qui sont proposés pour le manipuler. Ce n'est pas parce qu'on change de marque de stylo qu'on devient incapable d'écrire une langue. Il en va de même en informatique : quand on sait utiliser un langage sur un ordinateur, on sait l'utiliser sur un autre, moyennant le temps de retrouver ses marques et de localiser les outils. La majorité des gens, lorsqu'ils se retrouvent face à un de mes ordinateurs, se sentent perdus. Ils mettent souvent plusieurs minutes à comprendre pourquoi : la souris n'est pas du côté habituel. Changer d'ordinateur pour utiliser le même langage, c'est la même chose : retrouver ses habitudes.

[2] Une application, c'est un usage de l'ordinateur, un programme, c'est une tâche qui s'exécute sur la machine, une application *peut* utiliser plusieurs programmes différents, par exemple l'un pour le calcul et l'autre pour la mise en forme.

Pourquoi je te raconte tout ça ? Parce que LaTeX est un langage de mise en forme de documents, tout simplement.

1.2 Éditer ton document

Un document LaTeX est avant tout un texte, pas seulement le texte que l'on veut dans le document, mais également toutes les commandes indiquant à LaTeX comment il doit le mettre en forme.

La première phase de préparation de ton document, ce sera de l'éditer. Pour cela, il te faut un éditeur de texte[3].

Je te propose comme premier document le document suivant :

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

\usepackage[T1]{fontenc}
\usepackage{babel}

\begin{document}
Bonjour, Monde!
\end{document}
```

Sauve-le sous un nom finissant par `.tex`, mettons `premier.tex`. Ça y est, ton premier document existe.

1.3 Compiler ton document

Ce premier document, il te reste à le compiler. La manip est simple, très simple, mais savoir laquelle, ça dépend de ta machine. Je vais essayer de te guider en te donnant des indices :

- Dans les environnements graphiques, un double clic sur l'icône du fichier `premier.tex` suffira souvent. Parfois un drag'n'drop[4] vers une application du nom de LaTeX ou LaTeX 2_ε sera préféré au double clic.
- Si tu as accès à une ligne de commande (un terminal pour les unixiens, une fenêtre DOS pour les windowsiens, et quelque chose qui ressemble pour les autres), alors le logiciel à appeler est normalement `latex`, mais parfois aussi `latex2e`. Normalement, la syntaxe d'appel est celle-là :
  `latex␣premier`
  où ␣ dénote un espace.
- Si tu as un éditeur de texte un peu futé, sur Mac par exemple, la compilation doit être appelable depuis un menu ou par une commande simple.

À ce stade plusieurs fichiers ont été créés :

[3] Son nom dépend des habitudes chez toi et de la machine. Les habitués d'UNIX peuvent avoir affaire à des choses comme `vi`, sur tous les UNIX, `ved` sur HP, `textedit` sur SUNOS, ou encore `Xedit` ou `xcoral` un peu partout. Les utilisateurs de PC sous DOS/windows auront plutôt recours à `edit` ou au bloc notes. Les pros du Mac auront normalement la chance de trouver `alpha` avec leur version de LaTeX. Pour débuter, n'importe quel éditeur fera l'affaire, mais pour une utilisation un peu intensive, un éditeur de grande qualité commme `emacs` ou des versions modernes de `vi`, ou dédié à LaTeX comme `alpha`, est à recommander. Ce type d'éditeur offre l'avantage de simplifier la saisie avec des automatismes ou même des mises en valeur des mots clefs, avec une aide à la détection des fautes !

[4] Glisser-lacher en français, si c'est une manipulation que tu ne connais pas c'est qu'elle n'est pas classique sur ta machine et que ça ne doit pas être ça.

`premier.log` : c'est la transcription détaillée de tout ce qui s'est passé à la compilation, en particulier des éventuelles erreurs.
`premier.aux` : c'est un fichier auxiliaire (parmi plusieurs autres pour certains documents plus complexes) très précieux à LaTeX.
`premier.dvi` : c'est le résultat de la compilation, celui que tu visualiseras.

1.4 Regarder ton document

Pour visualiser, c'est comme pour compiler : simple mais pas partout pareil. Souvent, en mode graphique, un double clique sur le fichier `premier.dvi` suffira. Parfois un drag'n'drop vers le logiciel de visualisation suffira. Souvent, il sera possible de lancer le visualiseur comme tout autre programme, puis de lui demander de charger le fichier `premier.dvi`.

Sous UNIX, le visualiseur le plus classique est `xdvi`, sous Windows c'est souvent `windvi`. Sous DOS il arrive que l'on doive lancer

```
view␣premier
```

qui cache souvent un appel à `dvi2scr` ou à `dvi2vga`.

1.5 Imprimer ton document

Si, satisfait de ton travail, tu souhaites l'imprimer, là encore la marche à suivre dépend de ta machine. Mais, pire, cela dépendra souvent aussi de ton imprimante. Et pour un tandem ordinateur/imprimante donné, il peut également y avoir plusieurs solutions possibles.

On distinguera quelques grandes pistes.

La première, c'est le cas le plus favorable à l'utilisateur débutant : un des logiciels que tu as déjà croisé te propose dans un de ses menus d'imprimer. Par exemple un éditeur spécialisé, à côté du menu pour lancer la compilation et de celui pour visualiser le document, en proposera un pour imprimer. Ou encore le logiciel de visualisation pourra proposer d'imprimer.

La seconde est un peu moins favorable : quel que soit le système que tu utilises, tu peux y installer le programme `ghostview`, qui permet de visualiser et d'imprimer du PostScript. Et il se trouve que le programme `dvips` permet de convertir un document LaTeX compilé (un fichier `.dvi`) en PostScript. On peut donc, via une conversion en PostScript, imprimer depuis ce logiciel. La syntaxe d'appel de `dvips` est :

```
dvips premier.dvi -o premier.ps
```

La troisième est la pire des solutions. Le seul environnement où il me soit arrivé d'en venir à ces extrêmes, ce sont les systèmes UNIX mal configurés. En effet, quand ils sont bien configurés, le fichier PostScript est directement imprimable, sans autre manipulation : le système se chargera de le convertir pour l'envoyer vers l'imprimante. Sur les systèmes mal (ou pas) configurés, il faut faire cette conversion à la main, par exemple par un appel à :

```
gs -q -dNOPAUSE -sDEVICE=deskjet -sOutputFile=premier.pcl premier.ps
```

Attention pour les unixiens : quand un système UNIX est bien configuré, comme je viens de le dire, il est capable d'imprimer directement un fichier PostScript. Certains systèmes sont parfois *trop* bien configurés (ou croient l'être) : ils proposent d'imprimer directement les fichiers `.dvi`. C'est un très mauvaise idée. En effet, il arrive que ça ne marche pas.

Un fichier PostScript (surtout ceux générés par `dvips`) est autonome : il n'a besoin d'aucun fichier annexe pour être imprimé. Par contre les fichiers `dvi` ne le sont pas : ils n'incluent pas directement les images, mais simplement le nom (relatif au répertoire courant)

du fichier contenant chaque image. Ainsi, lorsque ce système trop bien configuré voudra imprimer le fichier `dvi`, il en fera une copie dans la file d'attente de l'imprimante, et, quand son tour viendra, il le convertira pour l'imprimer. Mais cette conversion aura lieu dans la file d'attente de l'imprimante, pas dans le répertoire courant de la compilation : il ne trouvera pas les images.

1.6 Dernière fois

C'est le seul endroit de ce livre où j'évoque l'interaction entre ton ordinateur et toi. Cela ne t'a probablement rien apporté de très constructif, puisque je ne sais pas sur quel système tu travailles, et cela t'a donné l'impression fausse d'avoir affaire à un système complexe.

Chapitre premier
Présentation

2 Structuration du document

Un document LaTeX ressemble par de nombreux aspects à un programme, pour ceux qui connaissent. Globalement, on distingue généralement 3 grandes parties. La première, où l'on indique à LaTeX ce dont on aura besoin comme outils dans le document, est obligatoire. La seconde, où l'on explique comment fabriquer les outils qui manqueraient encore, est optionnelle, à tel point que je n'en parlerai quasiment pas avant le chapitre neuvième page 571 consacré à la programmation. Enfin, la dernière est le document lui-même.

2.1 Les classes

Les classes de LaTeX 2_ε les plus courantes sont référencées dans le tableau 2.1 page suivante.

Une classe définit la structure du document, son découpage logique, ainsi que la majorité de son apparence globale (format de page, mise en valeur des différents titres, etc.). Par exemple pour ce livre j'ai écrit une classe qui fonctionne avec un environnement `chapter` au lieu de la commande habituelle `\chapter` pour commencer un chapitre, et qui produit la mise en page que tu vois.

La commande `\documentclass`, que tu as vue dans le document `premier.tex`, indique à LaTeX à quelle classe appartient le document et doit donc apparaître en premier, et une seule fois.

Les classes sont, sur ton disque, des fichiers portant l'extension `.cls`, par exemple `report.cls`. Les deux principales à retenir sont **article** pour les documents sans chapitre (quelques pages) et **report** pour ceux, plus gros, qui ont plusieurs chapitres.

2.2 Les options

Les classes de document admettent des options pour effectuer certains réglages. Les options classiques des trois classes communément utilisées sont listées dans le tableau 2.2 page 8.

Conformément aux habitudes sous LaTeX, les options sont placées entre crochets juste après le nom de la commande. Ainsi pour charger la classe de document **report** en 12 points, on tapera en tête de document :

```
\documentclass[12pt]{report}
```

Il a aussi été constaté que des versions différentes survivaient selon les endroits où on les cherchait. Ainsi, en recevant un document d'une autre université, il pouvait supposer une version différente de celle actuellement utilisée. Pour cela, on peut spécifier une date de version à partir de laquelle on accepte le fichier :

```
\documentclass[12pt]{report}[1994/06/01]
```

permet de demander à LaTeX 2_ε d'utiliser la classe **report** avec l'option `12pt` dans une version datant d'après le premier juin 1994. Si une version plus ancienne est trouvée, un message te le signalera lors de la compilation.

Classe	Description
article(*)	Définit un article comme on en voit dans les revues scientifiques avec sections, sous-sections, etc. Un niveau de sectionnement « partie », non numéroté, permet d'indiquer l'articulation globale d'un article ayant beaucoup de sections, ou n'en ayant pas, comme dans les quotidiens par exemple.
report(*)	Ajoute un niveau « chapitre » à la classe `article`. La majorité des numérotations sont remises à zéro pour chaque chapitre (notes en bas de page, figures, tableaux, équations, etc.). Chaque chapitre commence une nouvelle page. Les parties sont numérotées.
book(*)	Ajoute un découpage en trois (`\frontmatter`, `\mainmatter`, `\backmatter`) à la classe **report** permettant d'isoler le texte principal des fioritures (tables diverses, intro et préfaces au début, biblio, index et épilogue à la fin).
letter(*)	Permet la mise en page d'une lettre, mais en utilisant les conventions américaines.
lettre	Permet la mise en page d'une lettre, en utilisant les conventions françaises en la matière.
slides(*)	Remplaçant et successeur de SliTEX, pour faire des transparents. Très peu utilisé, il existe plusieurs extensions plus puissantes et plus intéressantes.
refman	Classe permettant la mise en page de manuels de référence, du même type que les pages `man` d'UNIX. Rarement utilisé.
refman-s	Comme **refman** mais en plus petit.
proc(*)	Classe spéciale pour les compte rendus de conférences (proceedings).
ltxguide(*)	Classe spéciale utilisée par l'équipe de développement de LaTeX 2_ε pour écrire un mode d'emploi.
ltxdoc(*)	Classe de documents utilisée pour fournir un exemplaire très lisible et bien documenté des sources de LaTeX 2_ε.
ltnews(*)	Classe utilisée pour écrire la lettre d'information diffusée avec LaTeX 2_ε.

(*) Ces classes sont fournies en standard avec LaTeX, les autres peuvent ne pas être installées par défaut sur ton système.

TAB. 2.1: Classes utilisables par LaTeX 2_ε

Option	Action
10pt, 11pt, 12pt	Change la taille de la fonte principale du document, ainsi que toutes les fontes annexes (pour les titres) et les dimensions liées (indentation, espace avant et après les équations ...). 10pt est pris par défaut.
fleqn	Les équations sont mises à `\mathindent` de la marge gauche au lieu d'être centrées.
leqno	Les numéros des équations se trouveront à gauche (à l'intérieur pour les ouvrages recto-verso) au lieu de la droite par défaut.
twoside	Le document est prévu pour le recto-verso, avec alternance des marges et des en-têtes, ouverture des parties (pour les rapports, des chapitres aussi pour les livres) sur les pages impaires (de droite), etc.
oneside	Contraire de la précédente. Permet d'indiquer un document recto pour une classe qui prend du recto-verso par défaut comme **book**.
openright	Dans un document recto-verso, même les chapitres doivent commencer sur de belles pages (de droite). Option prise par défaut pour les livres.
openany	Le contraire.
a4paper, a5paper	Indique la taille du papier. Attention, cela n'a aucune influence sur les marges.
letterpaper, legalpaper	Pareil, mais pour des formats plus classiques aux États-Unis.
landscape	Échange hauteur et largeur du papier, en n'influant toujours pas sur les marges.
draft	Indique que le document est encore un brouillon. L'effet est de ne pas inclure les images (pour accélérer les impressions) et de montrer clairement les textes qui débordent dans la marge, entre autres.
final	Contraire de la précédente. Par défaut.
titlepage	Indique que le titre fait l'objet d'une pleine page. Option par défaut pour les rapports.
notitlepage	Le contraire. Option par défaut des articles.
twocolumn	Le document sera en deux colonnes. Utilisé pour certains articles.
openbib	Change l'apparence de la biliographie.

TAB. 2.2: Liste des options classiques de `\documentclass`.

On utilise très rarement cette fonctionnalité, mais s'il t'arrive qu'un de tes documents ne marche pas sur certaines versions, tu pourras ainsi le signaler.

2.3 Les packages

Les packages de macro-commandes à charger avant la compilation sont désormais légion sous LaTeX. Une commande spéciale a donc été prévue.

Ainsi, si tu souhaites utiliser les macros du fichier `a4.sty`, tu placeras cette ligne entre le `\documentclass` et le `\begin{document}`[5] :

```
\usepackage{a4}
```

Les packages prennent des options et peuvent avoir des dates de version à respecter.

Pour éviter d'avoir à répéter sans cesse cette commande, il est possible de donner une liste de packages à charger, simplement en séparant leur nom par des virgules. De plus, toutes les options positionnées dans un `\usepackage` sont passées à tous les packages lus par cette commande.

Enfin, les options déclarées dans le `\documentclass` et qui ne sont pas comprises par la classe sont globales et inchangeables, elles sont donc passées à tous les packages sans les répéter. Si ni la classe ni les packages utilisés ne font usage d'une telle option, généralement cela indique une faute de frappe dans son nom, LaTeX te le signale donc par un petit message à l'écran.

Par exemple, le préambule actuel de ce document pourrait être[6] :

```
\documentclass[frenchb,10pt]{book}
\usepackage[german,english]{babel}
```

Une option du `\documentclass` qui n'est pas reconnue par la classe n'est pas forcément une erreur : cela peut être une option globale du document, à passer à tous les packages, comme le frenchb[7] dans l'exemple précédent. Au contraire, une option non-reconnue dans un `\usepackage` est une erreur, puisque personne d'autre n'aura l'occasion d'utiliser cette option. Un message d'erreur est donc émis dans ce cas là.

L'option frenchb a été placée dans les options du `\documentclass` car tout mon document est en français, et je ne souhaite pas avoir à la répéter sans cesse pour les packages concernés.

Différents packages ainsi que les options qu'ils reconnaissent sont décrits tout au long de ce livre.

2.4 Le mode compatibilité

Les anciennes versions de LaTeX (avant 1994) ne fonctionnaient pas de la même façon. On y utilisait `\documentstyle` au lieu de `\documentclass`, et les noms des packages étaient

[5] Comme son nom l'indique, cette commande marque le début du document. Il ne doit pas y avoir de texte avant.

[6] En fait, le préambule de ce document mériterait a lui seul un chapitre, d'abord parce qu'une classe de document spécifique a été définie, mais aussi parce que document charge plusieurs dizaines de packages, et que certains de ces packages sont incompatibles entre eux.

[7] Toi, lecteur angliciste, tu es à deux doigts de me soupconner d'une faute de frappe. Il n'en est rien. Bien qu'en anglais, on dise *french*, pour « français », l'option que j'appelle ici est bien frenchb. Il existe plusieurs implémentations du français pour LaTeX. L'option frenchb demande explicitement celle de `babel`. L'option french existe mais est plus dangereuse : s'il existe un package `french`, il est utilisé, sinon c'est l'implémentation de `babel` qui est prise. Ainsi, en précisant french, tu ne sais jamais vraiment ce qui sera utilisé. Il vaut mieux éviter.

donnés en option de `\documentstyle`. Les packages ne prenaient pas d'options, et étaient plus rares.

Les auteurs de LaTeX 2$_\varepsilon$ étant des gens sérieux, une compatibilité avec l'ancienne version de LaTeX a été prévue. Ainsi, si ton document commence avec l'ancien `\documentstyle` au lieu du nouveau `\documentclass`, LaTeX 2$_\varepsilon$ passera tout seul en mode « compatibilité 2.09 », c'est-à-dire qu'il fera comme si on compilait avec LaTeX 2.09.

Quelques remarques rapides :

- Aucune des nouvelles possibilités n'est accessible depuis ce mode.
- La compatibilité n'est garantie que pour l'utilisation des commandes de haut niveau. En particulier pour la gestion des fontes, il peut s'avérer utile de charger le package `rawfonts` si certaines de tes macros font appel à des macros « bas niveau » pour les fontes.

  Ce cas étant difficile à détecter, pose la question autour de toi avant de dire que le mode compatibilité ne marche pas.

  Certaines extensions faisaient appel à des commandes bas niveau et ne sont donc plus tolérées. C'est pour cela que je recommande parfois de garder les deux versions quand on a de très vieux documents en stock.
- Il est fortement recommandé de ne pas utiliser ce mode pour de nouveaux documents car il n'est là que pour éviter les pertes et ne sera probablement plus dans LaTeX3.
- Si le mode compatibilité échoue lamentablement, essaie de passer en LaTeX 2$_\varepsilon$ pur en corrigeant l'en-tête et les changements de fonte (`\bf`, `\it` ...) puisque ces deux changements sont les deux plus grosses évolutions d'une version à l'autre.
- La majorité des extensions pour LaTeX 2.09 fonctionnent encore avec LaTeX 2$_\varepsilon$.

2.5 L'environnement `filecontents`

Certains documents ont tendance à utiliser des fichiers de style qui ne sont pas standards, comme par exemple ton fichier de macro à toi qui contient tes macros habituelles.

Donc si tu expédies ton joli document par mail (par exemple), ou si tu le publies dans les news, il y a fort à parier que ton correspondant n'aura pas ces fichiers.

On a donc la possibilité de placer

```
\begin{filecontents}{mesmacros.sty}
Contenu du fichier tel qu'il doit etre
(copier-coller).
\end{filecontents}
```

en tout début de document (avant même le `\documentclass`) pour que, si le fichier est inaccessible, il soit re-généré. C'est pratique et permet de rester rigoureux dans la portabilité.

Chapitre premier
Présentation

3 Documents types

Je ne te donnerai ici que trois documents types. Le premier est un rapport bidon, dont le seul intéret est de te montrer comment cela se présente normalement. Je n'y utilise pas le système d'inclusion de fichiers, mais il est recommandé de le faire en respectant le principe qui dit « un chapitre par fichier ».

3.1 Rapport type

Ce document est disponible sur Internet à l'URL :

`http://jmpl.fr.eu.org/JMPL/rapport.type.tex`

Pour chaque partie de ce document type, je commenterai les éléments à retenir, et tu trouveras des renvois vers les passages décrivant plus avant les commandes intéressantes.

3.1.1 En-tête du document

```
\documentclass[francais,twoside,openright]{report}

\usepackage[T1]{fontenc}
\usepackage{babel}

\begin{document}
```

À noter :
- l'utilisation du package `fontenc` qui doit se faire comme cela pour permettre la césure convenable de notre belle langue ou de toute langue ayant des caractères latins accentués ;
- l'utilisation du package `babel` :
  - c'est lui qui se chargera de faire les traductions utiles (par exemple « chapter » en « chapitre »),
  - lors de cet appel, il reçoit (même si ça n'est pas évident au premier abord) l'option francais,
  - pour plus de détails sur `babel`, voir section 5 page 26 ;
- l'option francais qui est positionnée comme étant globale alors que ce n'est pas une option de **report**. Simplement, elle sera passée à tous les packages qui en auront besoin : c'est-à-dire tous les packages qui sont suceptibles de produire du texte et qui sont bien écrits ;
- l'option twoside qui demande à ce que l'on travaille en recto-verso ; ça a deux conséquences :
  - la première est que les parties commencent sur des pages impaires,

  - la seconde est que les marges — et éventuellement les en-têtes de pages — seront alternés selon que l'on est sur une page de droite (recto) ou de gauche (verso) ;
- l'option `openright`, qui n'a de sens que si la précédente est là, indique que même les chapitres doivent commencer sur des pages impaires (de droite).

3.1.2 Le titre

```
\begin{document}

\title{Rapport bidon}
\author{Benjamin \textsc{Bayart}
  \and Moi \textsc{M\^eme}
  \and Moi \textsc{Aussi}
  \and Personne \textsc{d'Autre}}
\date{Le \today}

\maketitle

\strut\thispagestyle{empty}
\vfill\pagebreak

\setcounter{page}{1}
```

À noter :
- le titre, la date, l'auteur et la composition de la page de titre, sans commentaires, pour plus de détails, voir section 11.1 page 44 ;
- la commande `\textsc` qui est l'une des nombreuses qui permettent de changer de fonte, celle-ci passe en petites capitales (small caps en anglais), voir section 7 page 32 pour plus de détails ;
- on peut faire une page de titre plus personnalisée grâce à l'environnement `titlepage`, par exemple :
  ```
  \begin{titlepage}
   \vspace*{3cm}
   \begin{center}\Large\textbf{Le titre}
   \end{center}
   \vspace*{3cm}
   \begin{center}
    \includegraphics{LeDessin.ps}
   \end{center}
  \end{titlepage}
  ```
  la meilleure méthode pour apprendre étant alors d'essayer soi-même ;
- juste après, c'est-à-dire pile à l'endroit où se trouvent les références ISBN[8] et autres formalités pour les livres :
  - je mets un truc invisible (`\strut`),
  - je demande à ce que *cette* page-là ne soit pas numérotée,
  - je la remplis de blanc (`\vfill`),
  - enfin, je change de page ;

  le but de toute cette manœuvre est d'insérer une page blanche au dos de la couverture pour que lorsque l'on imprime directement en recto-verso, tout ne soit pas décalé d'une

[8] ISBN : International Standard Book Number, numéro de référence unique attribué à chaque livre.

page sous prétexte que la couverture est en recto simple ;
- il convient de penser à ramener le compteur de page à 1 sinon toute la manœuvre précédente n'a servi à rien : LaTeX devine si une page est recto ou verso d'après sa parité (page impaire = recto, page paire = verso)[9].

3.1.3 Le début

```
\tableofcontents

\chapter*{Introduction}
\addcontentsline{toc}{chapter}{Introduction}
\markboth{\uppercase{Introduction}}
         {\uppercase{Introduction}}

Voici l'introduction de mon rapport, elle est
tr\'es jolie et tout et tout. Tu noteras que
dans un rapport en \LaTeX{}, il n'y a que ce qui
se trouve \textsc{avant} le
\verb"\begin{document}" qui est d\'epandant de
la machine utilis\'ee.
```

À noter :
- la table des matières (`\tableofcontents`) ;
- l'introduction : cette construction complexe sert à produire un chapitre non-numéroté et recensé dans la table des matières, une commande toute faite est donnée plus loin (section 11.3 page 45), que tu peux reprendre pour te simplifier la vie.

3.1.4 Le corps du document

Attention : *l'exemple réel contient un texte, mais je n'ai pas souhaité le reproduire ici pour ne pas perdre de place.*

```
\part{\'Etude pr\'eliminaire}

\chapter{On a commenc\'e}

\section{Le commencement}

C'est par l\'a qu'on a commenc\'e, comme d'habitude.

\section{Juste apr\`es}

Ben on a continu\'e de commencer. Mais plus fort.

\section{Enfin}

.....................................
[tr\`es long rapport tr\`es intelligent]
```

[9] Convention typographique plus que classique. À tel point que les pages impaires sont dites « belles pages » et les pages paires sont dites « fausses pages ».

```
.........................................

\chapter{Sieste au bureau}

\chapter{Sieste \`a la Kfet}

\section{Et souvent en plus}
```

À noter :
- les commandes de sectionnement ;
- c'est principalement cette partie là qu'explique le reste du livre.

3.1.5 La fin du rapport

```
\chapter*{Conclusion}
\addcontentsline{toc}{chapter}{Conclusion}
\markboth{\uppercase{Conclusion}}
         {\uppercase{Conclusion}}

Heureusement que le projet il durait pas
plus longtemps parce que \c{c}a co\^ute
vachement cher de glander \`a la Kfet
toute la journ\'ee.

\part{Annexes}

\appendix

\chapter{R\'esultats chiffr\'es}

\begin{tabular}{|l|r|r|r|}  \hline
          & D\'epenses & Recette & Bilan    \\ \hline
Caf\'e    & 3658,25 F  & 1,75 F  & -3656,50 \\ \hline
Croissant & 1548,12 F  & 0,00 F  & -1548,12 \\ \hline
Beignet   &  925,30 F  & 8,25 F  &  -917,05 \\ \hline
                                             \hline
Total     & 6131,67 F  & 10,00 F & \textbf{-6121,67} \\ \hline
\end{tabular}

\chapter{R\'esultats humains}

Ben je m'est beaucoup amus\'e. Pas vous?

\end{document}
```

À noter :
- la conclusion qui marche comme l'introduction, on utilisera volontiers le même système pour les remerciements et la préface, pour la bibliographie c'est pas la peine, c'est prévu par LaTeX ;
- l'exemple le plus simple de création de tableau ;

– pour passer du corps du document aux annexes, on indique simplement le mot clef `\appendix` ; note qu'il ne produit pas de page avec écrit en gros **Annexes**, c'est à toi de le faire.
  Cette commande sert juste à indiquer à LaTeX que dorénavent les chapitres doivent être numérotés différemment ;
– les deux chapitres viendront en fait créer l'« **Annexe A** » et l'« **Annexe B** », c'est le choix fait automatiquement par LaTeX ;
– cet exemple date un peu, du coup il est en francs, comme autrefois, pour le traduire en euro, voir la section 2.35 page 133.

3.2 Lettre type — modèle anglais

La lettre type est donnée un peu plus loin. Avant, je vais exposer comment on utilise la classe **letter**.

Il s'agit d'une classe de document que j'utilise plus que rarement, aussi me bornerai-je à te donner les commandes usuelles en te rappelant que la mise en page choisie est celle « à l'américaine ». Cette classe de document a été écrite par Leslie Lamport (l'auteur de LaTeX à l'origine), puis reprise par Frank Mittelbach et Rainer Schöpf [116], elle est maintenue par l'équipe du projet LaTeX3.

Les cinq déclarations globales :
– `\name` pour le nom de l'expéditeur ;
– `\signature` pour sa signature ;
– `\address` pour son adresse ;
– `\location` pour un ajout de précision à l'adresse, par exemple « bureau 1452 » ;
– `\telephone`, y faut vraiment te faire un dessin ?

Chaque lettre est en fait un environnement `letter`.

Il est jugé comme essentiel que le nom du destinataire soit la première ligne de l'argument, et que chaque ligne de celui-ci soit séparée des autres par un `\\`.

Voilà l'exacte étendue de mes connaissances sur le sujet.

Voilà ce que peut être une lettre type. On trouvera le document ci-dessous à l'URL :

`http://jmpl.fr.eu.org/JMPL/lettre.type.tex`

Le source est le suivant et le résultat est illustré figure 3.1 page 17.

```
\documentclass[french,12pt]{letter}

\usepackage[T1]{fontenc}
\usepackage{babel}

\begin{document}

\name{Benjamin \textsc{Bayart}}
\signature{}
\address{1, rue de Beaumont\\
95 560 Maffliers}
\location{}
\telephone{01~34~69~88~17}

\begin{letter}{Beno\^it Joly\\
3, rue du Duc de Dantzig\\
```

```
97 340 Gala-plage}
\opening{Tr\‘es cher Beno\^it,}

Depuis ces presque cinq semaines que nous ne
nous sommes vus, bien des choses se sont
produites. Bien des \’ev\’enements ont
\’et\’e c\’el\’ebr\’es.

Il faudra que je te raconte combien
int\’eressant fut ce projet, que je te dise
\‘a quel point il est d\’elicat de travailler
sur d’aussi complexes sujets sans avoir de
solides bases th\’eoriques auxquelles se
raccrocher.

Comme convenu je pense que nous nous
retrouverons tr\‘es bient\^ot pour discuter
de tout cela et du reste.

\closing{Tr\‘es affectueusement,}
\end{letter}
\end{document}
```

3.3 CV type

Nous devons ce paragraphe à Pierre Le Maguet *qui par son insistance folle a réussi à me faire écrire un petit package répondant au doux nom de* `ESIEEcv` *et qui sert à produire des CVs comme il les aime.*

Un CV type, écrit avec ce package est disponible sur Internet à l'URL suivante :

`http://jmpl.fr.eu.org/JMPL/cv.type.tex`

Libre à toi, lecteur feignant, de recopier ce CV et de le modifier pour faire le tien. Le résultat produit par ce CV fait l'objet de la figure 3.2 page 18.

`ESIEEcv` est un package, même si cela aurait probablement dû être une classe de document. Le seul intérêt est de pouvoir inclure un CV dans un document autre, dont la mise en page sera régie par la classe de document elle-même. Ce package fait appel à `tabularx` pour la mise en forme (le CV est une suite de tableaux), et à `stmaryrd`, pour dessiner le petit triangle.

Le CV y est articulé en différentes rubriques (formation, expériences, etc), chacune comprenant des sous-rubriques (chaque diplôme, école, emploi, etc.).

3.3.1 La Rubrique

La rubrique a un titre, censé résumer son thème, on utilisera donc les commandes suivantes :

```
\begin{rubrique}{titre de la rubrique}
...
\end{rubrique}
```

1, rue de Beaumont
95 560 Maffliers

8 mars 2004

Benoît Joly
3, rue du Duc de Dantzig
97 340 Gala-plage

Très cher Benoît,

Depuis ces presque cinq semaines que nous ne nous sommes vus, bien des choses se sont produites. Bien des événements ont été célébrés.

Il faudra que je te raconte combien intéressant fut ce projet, que je te dise à quel point il est délicat de travailler sur d'aussi complexes sujets sans avoir de solides bases théoriques auxquelles se raccrocher.

Comme convenu je pense que nous nous retrouverons très bientôt pour discuter de tout cela et du reste.

Très affectueusement,

Benjamin BAYART

FIG. 3.1: Résultat de la compilation de la lettre type

Benjamin BAYART
50, rue de Chambéry
97 123 LOIN
Tél : 09 12 11 16 10
Mail : bayartb@edgard.fdn.fr

Né le 24 octobre 1973 (24 ans)
Nationalité française
Célibataire sans enfant
Sursitaire à l'incorporation

Formation initiale

1990-1991 LYCÉE NOTRE-DAME PROVIDENCE D'ENGHIEN-LES-BAINS (95). BACCALAURÉAT SÉRIE C (MATHÉMATIQUES ET SCIENCES PHYSIQUES).

1991-1996 (5 ans) INGÉNIEUR ESIEE (ÉCOLE SUPÉRIEURS D'INGÉNIEURS EN ÉLECTRONIQUE ET ÉLECTROTECHNIQUE).
Spécialisation en informatique.
▷ *Programmation système.* ▷ *Conception et programmation objet.* ▷ *Théorie des langages.* ▷ *Langages interprétés.* ▷ *Programmation logique.* ▷ *Programmation des interfaces graphiques.* ▷ *SGBD.*

1995-1996 (1 an) UNIVERSITÉ DE MARNE-LA-VALLÉE. DIPLÔME D'ÉTUDES APPROFONDIES.
Informatique Fondamentale et Applications
▷ *Théorie des automates.* ▷ *Programmation logique avancée.* ▷ *Théorie des partitions d'entiers.* ▷ *Calcul combinatoire.* ▷ *Algorithmique du texte.*

1996- (3 ans) UNIVERSITÉ DE MARNE-LA-VALLÉE. THÈSE DE DOCTORAT.
Nouvelles pistes pour une typographie électronique de qualité

Expériences

1996 (6 mois) LABORATOIRE D'ÉLECTRONIQUE PHILIPS. SIMULATION DE PROCESSEURS POUR LA DÉMODULATION NUMÉRIQUE.
Écriture d'une plateforme de développement pour un processeur massivement parallèle en developpement.
▷ *Projet de type industriel.* ▷ *Travail dans un département de R&D.* ▷ *Approche des problèmes d'architecture des processeurs dédiés.* ▷ *Approche du micro-parallèlisme.*

1995 (2 mois) GROUPE ESIEE. RECONNAISSANCE DE PHONÈMES PAR CARTES DE KOHONEN.
Utilisation de cartes auto-organisante de Kohonen pour la reconnaissance et l'étiquettage de phonèmes après apprentissage non supervisé.
▷ *Travail en traitement automatisé du signal.* ▷ *Étude et utilisation des réseaux de neurones.* ▷ *Première approche de travaux de recherche.*

1995 (6 semaines) GROUPE ESIEE. SEGMENTATION D'IMAGES PAR HYPER-CARTES DE KOHONEN.
Approche des techniques multi-résolution.

1994 (1 an) GALA ESIEE 94.
Responsabilités diverses dans l'équipe d'organisation d'un grand évènement estudiantin (6000 personnes). En particulier infographie, communication, imprimerie, et logistique finale.
▷ *Travail sur la durée dans une équipe très soudée avec un projet directeur fort et ambitieux.*

1996 (2 mois) GALA ESIEE 96.
Participation à l'organisation finale, à la conception technique de la communication, au fonctionnement de la trésorerie, et à la logistique finale.
▷ *Apprentissage de la gestion d'équipe et des ressources humaines.*

1997 (2 mois) GALA ESIEE 97.
Conseil technique du bureau d'organisation, établissement de partenariats relationels, aide à la gestion de la sécurité et de la trésorerie
▷ *Gestion de la motivation des personnes impliquées.* ▷ *Prise en compte de graves retards organisationels.*

Langues et divers

Anglais — Lu, écrit, parlé. Anglais technique courrant.
Espagnol — Niveau scolaire
Passe-temps : philatélie, typographie, gravure, programmation, cinéma...

FIG. 3.2: CV type réalisé avec le package `ESIEEcv`

avec un résultat dont la magnificence nous échappe encore :

Titre de la rubrique

...

3.3.2 La sous-rubrique

À l'intérieur de la rubrique, on trouvera plusieurs sous-rubriques, que nous avons classées comme suit :

- Les sous-rubriques de type « expérience située dans le temps, l'espace, la fonction, etc ».
- Les sous-rubriques du type « compétence » telles que les langues, les outils divers ou les langages informatiques.
- Les autres sous-rubriques, auxquelles nous n'avons pas pensé. Ces sous-rubriques n'ont pas été implémentées, nous n'en parlerons donc pas.

En langage LaTeX, la sous-rubrique est, comme la rubrique, un environnement. On la déclare donc de la même manière, au titre près :

```
\begin{sousrubrique}
...
\end{sousrubrique}
```

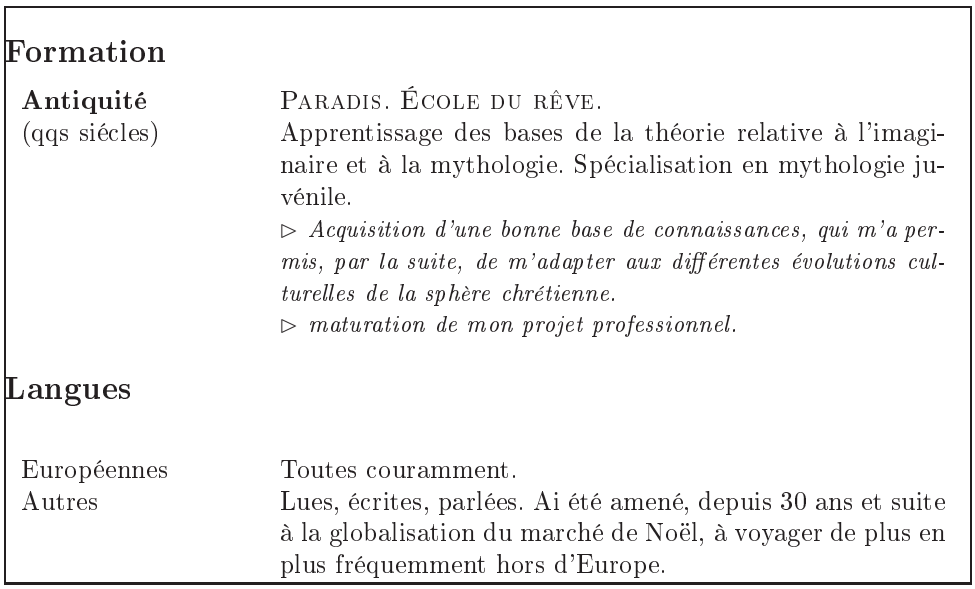
Le titre a disparu. En effet, le titre, de même que les autres champs, est optionnel. On trouve huit champs dans une sous-rubrique.

- six champs utiles si la sous-rubrique sert à décrire une expérience :
  - la date de l'expérience, introduite par la commande `\Date`,
  - la durée de l'expérience, introduite par la commande `\Duree`,
  - le lieu de l'expérience, introduit par la commande `\Lieu`,
  - le titre de l'expérience, introduit par la commande `\Titre`,
  - une description, introduite par la commande `\Descr`,
  - le champ « Apport », qui permet de mettre en style télégraphique ce qui manque ; contrairement aux autres champs, « Apport » présente la particularité d'être répétable. Il correspond à la commande `\Apport`.
- deux champs utiles pour la description de compétences. Le champ « Compétence » (commande `\Competence`), qui sera associé avec le champ « Description » (`\Descr`), déjà présent plus haut.

3.3.3 Exemples

Voici, par exemple, un extrait du CV du père Noël :

```
\begin{rubrique}{Formation}
\begin{sousrubrique}
\Titre {\'Ecole du r\^eve}
\Date {Antiquit\'e}
\Duree {qqs si\'ecles}
\Lieu {Paradis}
\Descr {Apprentissage des bases de la
  th\'eorie relative \`a l'imaginaire
  et \`a la mythologie.
  Sp\'ecialisation  en mythologie
  juv\'enile.}
\Apport {Acquisition d'une bonne base
  de connaissances, qui m'a permis,
  par la suite, de m'adapter aux
  diff\'erentes \'evolutions
  culturelles de la sph\`ere
  chr\'etienne}
\Apport {Maturation de mon projet
  professionnel}
\end{sousrubrique}
\end{rubrique}
\begin{rubrique}{Langues}
\begin{sousrubrique}
```



Formation

Antiquité (qqs siécles)	PARADIS. ÉCOLE DU RÊVE. Apprentissage des bases de la théorie relative à l'imaginaire et à la mythologie. Spécialisation en mythologie juvénile. ▷ *Acquisition d'une bonne base de connaissances, qui m'a permis, par la suite, de m'adapter aux différentes évolutions culturelles de la sphère chrétienne.* ▷ *maturation de mon projet professionnel.*

Langues

Européennes	Toutes couramment.
Autres	Lues, écrites, parlées. Ai été amené, depuis 30 ans et suite à la globalisation du marché de Noël, à voyager de plus en plus fréquemment hors d'Europe.

FIG. 3.3: Exemple de CV fait par le package `ESIEEcv`

```
\Competence {Europ\'eennes}
\Descr {Toutes couramment.}
\end{sousrubrique}
\begin{sousrubrique}
\Competence {Autres}
\Descr {Lues, \'ecrites, parl\'ees.
  Ai \'et\'e amen\'e, depuis 30 ans
  et suite \`a la globalisation du
  march\'e de No\"el, \`a voyager de
  plus en plus fr\'equemment hors
  d'Europe.}
\end{sousrubrique}
\end{rubrique}
```

qui, après compilation, donne le résultat de la figure 3.3.

N.B. : Il est possible de remplir une rubrique sans passer par les sous-rubriques. La rubrique sera alors remplie comme un tableau normal. Ainsi,

```
\begin{rubrique}
{Essai: rubrique sans sous-rubrique}
Mais ce n'est pas tr\'es & propre,\\
donc ne dites pas & \`a Benjamin que
c'est moi qui vous en ai parl\'e\\
\end{rubrique}
```

donne la figure 3.4 page suivante.

3.3.4 Paramètrage

Tu auras compris qu'en fait, chaque rubrique crée un tableau bi-colonne, et chaque sous-rubrique se contente de remplir une ligne de ce tableau. Or, pour obtenir une jolie mise en page, il faut que la première colonne de chacun de ces tableaux ait la même largeur que ses petites copines des tableaux des autres rubriques.

On a tout d'abord pensé à faire un système qui prendrait la plus grande de ces largeurs, et l'imposerait aux autres. Mais il n'y a pas de commandes LaTeX permettant de faire cela simplement. La plus grande largeur est en effet connue après la compilation, et LaTeX n'est

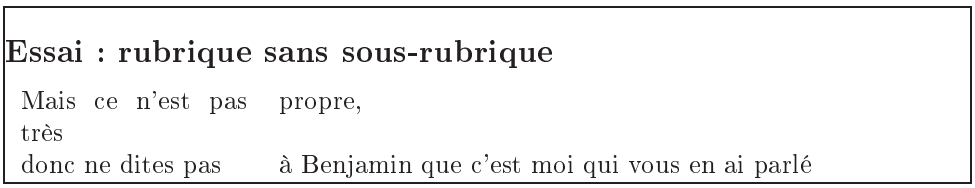


FIG. 3.4: Exemple de rubrique de CV

pas vraiment d'accord pour recommencer la compilation. Tu es donc obligé, si tu ne veux pas de la largeur par défaut (3 cm), de redéfinir toi-même cette largeur.

Elle est rangée dans la variable `\largeurcolonne`. Tu devras donc — en début de document — écrire quelque-chose dans ce goût-là :

```
\setlength {\largeurcolonne}{1.2cm}
ou
\settowidth{\largeurcolonne}{texte de r\'ef\'erence}
```

Voilà. Si tu n'as pas envie d'en savoir plus, tu peux te débrouiller. Mais pour ceux que cela intéresse, je propose que nous voyons comment changer la mise en page.

Le seul point que j'aborderai encore est celui de la redéfinition des 8 commandes qui sont définies au début du fichier `ESIEEcv.sty`.

Les deux premières, `\PreApport` et `\PostApport` correspondent à ce qui est mis avant et après chaque « Apport » dans une liste d'apports. Par défaut, leurs définitions sont les suivantes :

```
\newcommand{\PreApport}{\ensuremath{\triangleright} }
\newcommand{\PostApport}{.\newline}
```

On pourra obtenir un CV plus condensé en redéfinissant `\PostApport` pour qu'il ne change pas de ligne.

Les deux suivantes, `\PrePreApports` et `\PostPostApports` correspondent à ce qui est mis avant et après la liste des apports. Par défaut leurs définitions sont les suivantes :

```
\newcommand{\PrePreApports}{\newline}
\newcommand{\PostPostApports}{}
```

Enfin les quatre dernières, `\FonteApport`, `\TailleApport`, `\FonteLieu`, et `\FonteTitre` permettent de changer la taille et la fonte des apports, lieux et titres facilement. Leurs définitions par défaut sont :

```
\newcommand{\FonteApport}{\itshape}
\newcommand{\TailleApport}{\small}
\newcommand{\FonteLieu}{\scshape}
\newcommand{\FonteTitre}{\scshape}
```

Pour redéfinir une commande on utilisera la commande `\renewcommand`. Par exemple, pour que les apports soient écrits en gras au lieu de l'italique par défaut, il suffira de faire :

```
\renewcommand\FonteApport{\bfseries}
```

C'est aussi simple que ça.

Bon CV !

Chapitre premier
Présentation

4 Mise en page

LaTeX gère chaque page de ton document selon un grand nombre de règles, qui sont toutes paramètrables.

Certaines sont *absolument* globales. Il est normalement interdit de les changer en cours de document. Plus exactement, si tu les changes, tu prends le risque d'avoir un document catastrophique.

Les autres sont plus ou moins locales : locales à la page en cours, locales « jusqu'à nouvel ordre », etc.

4.1 Les paramètres globaux

Les plus évidents des paramètres globaux sont ceux qui touchent à la géométrie du document : la taille de la page, la taille des marges, etc.

4.1.1 Les marges

Tout ce dont je parlerai ici est représenté graphiquement par la figure 4.1 page ci-contre.

Les méthodes proposées par LaTeX pour règler ces différents paramètres sont assez austères, je te recommande vivement d'utiliser des extensions comme `geometry` (section 4.16 page 258) ou `vmargin` (section 4.17 page 270) pour faire ces réglages plus simplement.

LaTeX n'a pas véritablement la notion de taille de la feuille de papier : cette grandeur-là ne l'intéresse pas.

LaTeX écrit de gauche à droite, et de haut en bas, en temps normal.

Ce qui l'intéresse, c'est de savoir quel blanc il faut laisser à gauche avant d'écrire (la marge gauche) et sur quelle largeur il doit écrire (la largeur du texte). La largeur de la page ne l'intéresse pas.

De même, il s'intéresse à la taille du blanc à laisser en haut de la page (marge du haut), et à la hauteur du texte à remplir, mais pas à la hauteur de ta feuille.

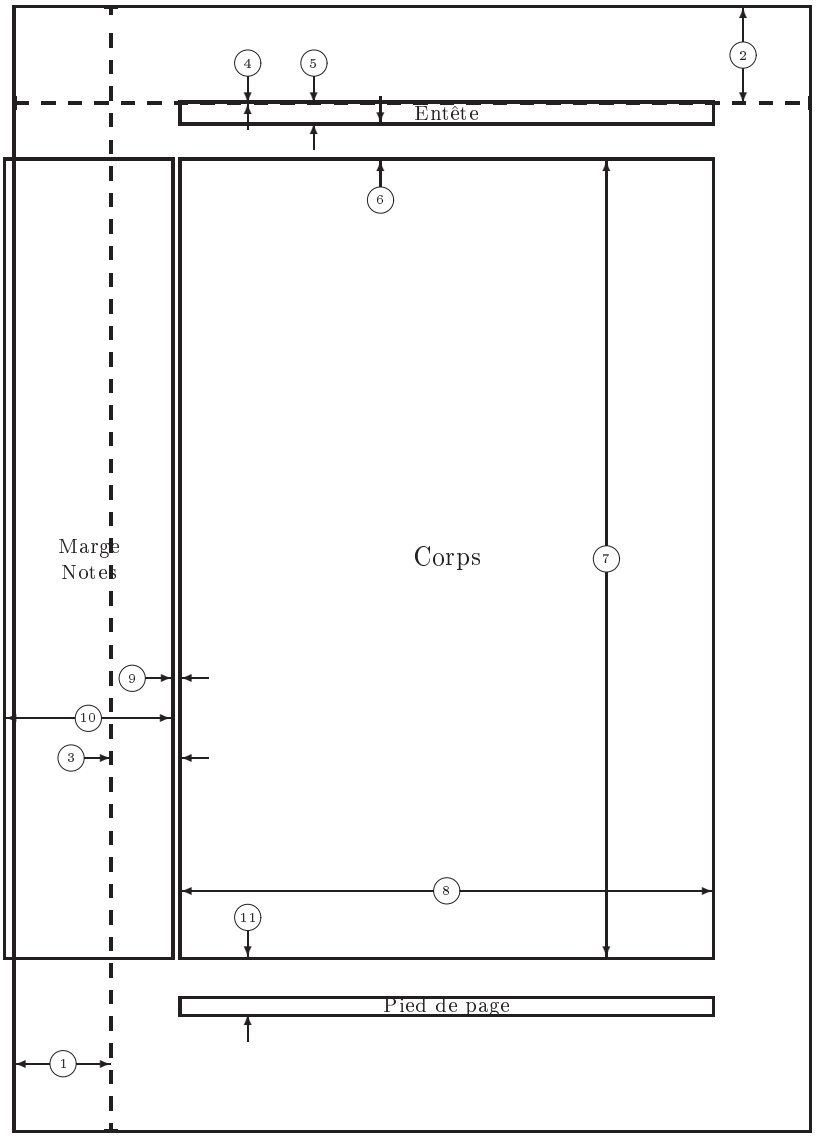
Grosso-modo, LaTeX va utiliser les grandeurs suivantes concernant la largeur de la page : `\textwidth` est la largeur de la zone principale de texte et `\leftmargin` est la marge de gauche. `\columnwidth` est la largeur de la colonne en cours : en mode normal, c'est pareil que `\textwidth`, en mode deux colonnes, c'est un peu moins de la moitié. `\linewidth` est la largeur de la ligne en cours : en temps normal, c'est `\columnwidth`, mais c'est parfois un peu moins, par exemple dans une liste, le texte est resserré.

Tu peux à tout moment changer ces grandeurs, mais il n'est raisonnable de le faire qu'au début du document :

```
\textwidth=14cm
\leftmargin=1.5cm
```

Pour le positionnement vertical, ça se complique terriblement parce qu'il y a l'en-tête et le pied de la page. LaTeX enchaîne donc ces longueurs-là :

– `\topmargin` est le blanc laissé en haut de la page.



1	un pouce + \hoffset	2	un pouce + \voffset
3	\evensidemargin = 53pt	4	\topmargin = 0pt
5	\headheight = 15pt	6	\headsep = 28pt
7	\textheight = 600pt	8	\textwidth = 400pt
9	\marginparsep = 7pt	10	\marginparwidth = 125pt
11	\footskip = 43pt		\marginparpush = 5pt (non affiché)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 598pt		\paperheight = 845pt

FIG. 4.1: Construction de la page

– `\headheight` est la hauteur réservée à l'en-tête de la page.
– `\headsep` est le blanc entre l'en-tête et le corps de la page.
– `\textheight` est la hauteur du corps du texte, y compris les notes de pied de page.
– `\footskip` est la distance entre le bas du corps de la page et le bas du pied de page.

Ça peut sembler déroutant que l'en-tête et le pied de la page soient traités de manière différente, mais c'est techniquement plus simple pour LaTeX de faire comme ça.

4.1.2 Les notes de pied de page

D'autres grandeurs régissent le fonctionnement de LaTeX et peuvent être modifiées.

Commençons par ce qui touche aux notes de pied de page. Je te donne les infos, mais je veux te mettre en garde : **n'y touche pas**. Ou alors, tu assumes.

`\skip\footins` est le blanc à laisser entre le corps de la page et le séparateur des notes. Par défaut, c'est l'équivalent d'une ligne du corps du texte. Ça peut se modifier comme ça :

```
\skip\footins=5mm
```

`\footnoterule` est une commande qui est appelée entre le corps du texte et les notes de pied de page. Elle doit produire un résultat qui fait exactement 0,4 pt de haut. Il est donc très délicat de lui faire faire quoi que ce soit.

On peut cependant la redéfinir, par exemple comme ça :

```
\skip\footins=5\baselineskip
\def\footnoterule{%
  \vskip-2\baselineskip
  \hbox{Notes:}
  \vskip\baselineskip
  \vskip .4pt
}
```

4.1.3 Les notes en marge

La commande `\marginpar` permet de mettre des notes dans la marge. Il est recommandé d'y mettre un texte *court*.

`\marginparwidth` régit la largeur de ces notes. Il est de bon goût que ce soit moins que la taille de la marge elle-même.

Par défaut, cette largeur est convenable en mode une colonne, elle permet par exemple de placer un symbole et quelques chiffres, et est très réduite en mode deux colonnes (les marges y sont plus faibles).

`\marginparsep` est la taille du blanc entre la note en marge et le corps du texte.

`\marginparpush` est la taille du blanc entre deux notes en marge qui risqueraient de se toucher, par exemple s'il y en a deux sur la même ligne.

4.2 Les styles de page

Par défaut, LaTeX définit trois styles de page :
– empty, style dans lequel il n'y a ni en-tête ni pied de page, utilisé par exemple pour les pages blanches (quand on veut revenir à une page impaire pour commencer un chapitre, il y a une page blanche).
– plain, style « normal », de base, dans lequel il n'y a pas d'en-tête, et en pied de page uniquement le numéro de la page, centré horizontalement.

– headings, style dans lequel il n'y a rien en bas de la page, et en haut d'un côté le numéro de la page, de l'autre côté le titre courant (chapitre ou section, selon le cas).

On change de style de page par un appel à `\pagestyle`. Le nouveau style s'applique à partir de la page courante, et pour toutes les suivantes, jusqu'au prochain changement. Par exemple :

```
\pagestyle{empty}
```

On peut ne changer de style que pour une seule page. C'est par exemple ce que fait LaTeX spontanément pour un début de chapitre : quel que soit le style de page adopté pour ton document, il repasse en plain pour le chapitre. À la base, c'est pour qu'il n'y ait pas d'en-tête au-dessus du titre de chapitre. La commande pour faire ça est `\thispagestyle`. Elle ne change que le style de la page courante.

Par exemple, pour dire que *cette* page est en empty alors que les autres sont en plain, on peut tout à fait dire ça :

```
\thispagestyle{empty}
\pagestyle{plain}
```

Ces possibilités sont étendues par le package `fancyhdr`, dont je parle à la section 4.3 page 200 du chapitre troisième.

Chapitre premier
Présentation

5 Gestion des langues — LaTeX polyglotte

LaTeX permet d'utiliser tout plein de langues sans se fouler trop. Plus de problèmes d'options tordues dans le choix du compilateur ou de trucs dans ce genre, comme ça a pu être le cas pour les vieilles versions (avant 1994). Le compilateur LaTeX 2$_\varepsilon$ standard comprend déjà plusieurs langues, s'il est bien paramétré. Le tableau 5.1 page ci-contre liste les langues connues de `babel`.

La gestion d'une langue se décompose en trois parties :

1. La gestion de la césure pour cette langue. Elle doit être configurée de manière globale pour une installation donnée, en indiquant dans un fichier de paramètres quelles sont les langues à prendre en compte. Ce fichier indique, pour toutes les langues que tu souhaites utiliser sur ton système, dans quel fichier sont décrites les règles de césure à appliquer.

   Les informations de césure pour un grand nombre de langues existent sur Internet et sont librement disponibles. Elles ne font pas forcément partie de ton installation.

   Pour le français, rassure-toi, c'est une des langues les plus répandues.

2. La gestion typographique, qui est régie pour chaque document par l'appel au package `babel`. C'est dans cette partie là qu'on indique qu'en français il faut mettre une espace fine insécable devant les ponctuations doubles, ou que les listes se font avec des tirets, et non avec des puces.

3. La gestion du texte : tous les textes que LaTeX produit tout seul doivent être traduits. Ceux produits par LaTeX lui-même sont traduits par le package `babel` (par exemple « list of figures » est traduit par « liste des figures »). Ceux produits par des packages annexes doivent être traduits par ces packages.

Les modèles de césures intégrés dans le noyau LaTeX sont réglés par un fichier `language.dat`, tu pourras changer les langues installées en corrigeant ce fichier et en recompilant le noyau LaTeX[10].

Plusieurs méthodes existent pour exploiter les changements de langue sous LaTeX. Celle que j'ai retenue pour le noyau LaTeX 2$_\varepsilon$ chez moi est connue sous le nom de `babel`. C'est la plus courante sur le plan international, mais l'une des plus contestées en France. On lui préfère parfois ce qui est connu sous le nom de « style `french` ». Je ne l'ai pas retenu parce qu'il n'offre pas la même souplesse au niveau des changements de langue et de la reconnaissance au plan international. Ce choix est contestable, et a été contesté, principalement par Bernard Gaulle, auteur du « style `french` ».

Depuis quelques années, le « style `french` » n'est plus dans le domaine public, ou plus exactement, pas sa version complète. Il est donc vivement recommandé de *ne pas s'en servir* et de convertir tous les anciens documents qui s'en seraient servis.

[10] Cette opération est décrite au chapitre huitième page 547 sur la configuration et l'installation de LaTeX.
Dans la majorité des installations modernes de LaTeX cette manipulation est simplifiée par un petit outil qui permet de choisir les langues et qui recompile tout seul ce qu'il y a à recompiler. Renseigne-toi sur ton installation.

Langue	Option	Langue	Option
Allemand	german ou germanb	Hongrois	hungarian ou magyar
Anglais (USA)	american	Icelandais	icelandic
Anglais (GB)	english	Irelandais	irish
Autrichien	austrian	Italien	italian
Basque	basque	Latin	latin
Brésilien	brazil	Néerlandais	dutsh
Breton	breton	Norvégien	norsk
Bulgare	bulgarian	Polonais	polish
Catalan	catalan	Portugais	portuges
Croate	croatian	Roumain	romanian
Danois	danish	Russe	russianb
Écossais	scottish	Serbe	serbian
Espagnol	spanish	Slovaque	slovak
Espéranto	esperanto	Slovène	slovene
Estonien	estonian	Bas Sorabe	lsorbian
Finnois	finnish	Haut Sorabe	usorbian
Français	francais ou french[a]	Suédois	swedish
Galicien	galician	Tchèque	czech
Greek	greek	Turque	turkish
Hébreu	hebrew	Ukrainien	ukraineb

[a] Le français peut être demandé à `babel` par trois options différentes. Les deux premières, francais et frenchb demandent explicitement le français de `babel`. La troisième, french est plus dangereuse : s'il existe un package `french`, elle l'utilisera, en supposant que tout va bien se passer, et sinon, elle va se rabattre sur le français de `babel`. À la base, c'est fait pour être compatible avec le package `french` de Bernard Gaulle. Ce package n'est pas gratuit, n'est pas dans le domaine public, et a été à plusieurs reprises incompatible avec `babel`. Mon conseil : éviter. Donc, au choix, frenchb ou francais, et c'est tout.

Tab. 5.1: Langues disponibles sous LaTeX 2_ε

Donc, à retenir, j'utilise `babel`, un point c'est tout. Le `french` de B. Gaulle est de toute façon livré avec une excellente documentation en français qui n'a pas besoin d'être paraphrasée.

5.1 Le français

Pour l'utilisation des langues sous LaTeX 2_ε tu devras faire appel au package `babel`, que nous devons à Johannes Braams [24], avec comme option les langues que tu souhaites utiliser, par exemple :

```
\usepackage[francais]{babel}
```

Mais tu peux aussi faire comme moi, déclarer les langues[11] dans les options principales et ne pas les rappeler. D'autres packages que `babel` se posent la question de la langue : c'est normalement le cas de tous les packages qui produisent du texte et qui, s'ils sont bien écrits, doivent se demander dans quelle langue le produire.

Pour sélectionner la langue courante, il te faudra ensuite utiliser :

```
\selectlanguage{francais}
```

[11] Un document peut en effet être totalement polyglotte.

Les spécificités du français avec `babel` :

- Les espacements français sont gérés (une espace fine avant le point-virgule, et une espace pleine après, par exemple) ;
- il n'est plus utile de taper `\^{\i}` pour obtenir î, un simple `\^i` devrait suffire ;
- pour le tréma, c'est idem : `\"i` produira ï ;
- la date est réadaptée automatiquement de telle sorte que l'habituelle commande `\today` produise : 5 mars 2005.

Un tas de petites choses ont été rajoutées, comme notre bien aimé degré (°) qui s'obtient par `\degre`.

Enfin `babel` offre également des facilités pour nos numérotations étranges et assez incomprises des anglo-saxons et autres non-francophones. Il est agréable de pouvoir obtenir :
1° C'est un essai
2° en deuxième
3° en troisième
4° en quatrième.

J'ai pour cela utilisé `\primo`, `\secundo`, `\tertio`, `\quarto`. Et pour aller à 5 ? Facile ! Tu fais :

```
\FrenchEnumerate5,
\FrenchEnumerate{12345}.
```

pour obtenir : 5° , 12345° .

Enfin, et toujours dans la série des numérotations, il y a les 1°) , 2°) , 3°) et 4°) qui sont obtenus par `\fprimo)`, `\fsecundo)`, `\ftertio)` et `\fquarto)` et que l'on peut étendre par :

```
\FrenchPopularEnumerate5,
\FrenchPopularEnumerate{42}
```

pour obtenir : 5°) , 42°)

C'est quand même agréable.

5.2 L'allemand

Pour les germanistes, `babel` et l'option `german` offrent de belles possibilités. Les europhiles seront heureux, ils peuvent enfin pondre des documents en anglais, français et allemand mélangés sans avoir trop de problèmes de typographie ou de césure.

L'histoire de l'implémentation de l'allemand dans LaTeX est assez chaotique (tout comme celle du français) ce qui fait que l'on arrive à des choses bizarres. Il semblerait que le plus fréquent dans les universités allemandes soit d'utiliser une installation spécifique datant en majeure partie de LaTeX 2.09 (fin des années 80) et qui était très poussée et très avancée. Cependant, mon but n'est pas d'obtenir que l'Allemand soit géré parfaitement par une version vieillissante de LaTeX à l'exclusion de toute autre langue. Aussi, je me suis restreint à ce que prévoit le package `babel`, et je te conseille de faire de même.

L'allemand pose de gros problèmes à LaTeX pour de nombreuses raisons, en particulier pour la césure. En anglais, ou en français, quand on veut césurer un mot (mettons « bonjour ») on rajoute un trait d'union à l'emplacement de la césure (« bon-jour »), alors qu'en allemand, les lettres peuvent changer. Par exemple « ck » ne se césurera pas « c-k » mais « k-k », et « ff » viendra se césurer « ff-f ». De plus les germanophones utilisent ce joli petit caractère : ß de manière assez courante, ainsi que l'accent tréma.

En allemand, sous LaTeX 2_ε, le guillemet (ou double-quote) devient un caractère spécial, comme le backslash.

Le `"a` donne ä. Le `"s` donne ß, `"S` donne SS[12] pour être utilisé comme étant le caractère associé au précédant en majuscule. `"‘` et `"’` produiront „ et “. De même, en Allemand, des guillemets « à la française » sont accessibles à partir des commandes `"<` et `">` qui viennent produire « et ».

Les césures spéciales devront être indiquées au cas par cas à l'aide de ce même guillemet : `"ck` et `"ff`. Pas forcément évident à exploiter, enfin c'est ce qu'il me semble, à moi qui n'ai jamais fait d'allemand.

Pour sélectionner le style allemand, il faut compiler en chargeant le package `babel` avec l'option `german` ou `germanb`. Pourquoi les deux ? Parce qu'en Allemagne, `german` c'est le style datant de LaTeX 2.09 qui tend à se remettre à jour, et que `germanb` c'est le style lié à `babel`.

5.3 Les autres langues

Les autres langues peuvent être appelées de façon similaire, comme étant des options du package `\babel`. Les noms courants sont donnés par LaTeX sitôt qu'on le lance. En effet, il affiche la liste des langues qu'il connaît, chez moi, ça donne ça :

```
Babel <v3.7h> and hyphenation patterns for
american, french, german, ngerman, dutch,
greek, irish, polish, portuges, russian,
spanish, ukrainian, nohyphenation, loaded.
```

Si tu demandes une langue autre et que LaTeX ne te jette pas comme un malpropre, c'est qu'il a compris et qu'il ne fera pas de césure dans ton texte puisqu'il ne maîtrise pas la langue.

En effet, quand tu demandes à `babel` une langue qu'il connaît mais pour laquelle les césures ne sont pas prévues dans ton installation de LaTeX, il produit un message d'erreur comme :

```
Package babel Warning: No hyphenation patterns were loaded for
(babel)                the language ‘Catalan’
(babel)                I will use the patterns loaded for \language=0 instead.
```

Si ton installation de LaTeX est bien faite, la langue numéro zéro est une langue dans laquelle il n'y a pas de césure. Si elle est mal faite, c'est l'anglais (et des résultats plus ennuyeux sont à redouter).

[12] En effet, il n'existe pas de ß en majuscule ; or dans certains titres de chapitre ou de section, les allemands aiment à pouvoir utiliser ce joli caractère, mais ces titres sont reportés en haut des pages dans certains cas, et ils sont alors convertis en majuscule. C'est pour que cette conversion se passe bien que le `"S` a été créé.

Chapitre premier
Présentation

6 Saisie de texte

Comme tu dois commencer à t'en rendre compte, un document LaTeX est avant tout un document texte, avec des balises indiquant la mise en forme : fonte, marge, tableaux, listes, etc.

Ce que tu n'as pas encore vu, ce sont les règles de saisie typographique permettant d'obtenir un texte joli presque à tous les coups. Je vais te donner ici quelques règles, certes un peu contraignantes, mais qui t'aideront à avoir un document plus propre.

6.1 Changement de paragraphe

Un changement de paragraphe ne peut s'obtenir que de deux façons. L'une, implicite et propre, est de laisser une ligne blanche dans ton texte (sauter une ligne). L'autre est d'appeler la commande `\par` qui termine un paragraphe s'il y en avait un en cours.

Toute autre méthode est à proscrire.

En particulier, la commande `\\`, que tu croiseras parfois, est à proscrire. Elle ne sert qu'à changer de ligne dans le paragraphe en cours (ce qui n'est pas la même chose que de changer de paragraphe) ou à changer de ligne dans un tableau.

6.2 Espaces insécables

La notion d'espace insécable est une notion que de plus en plus de gens connaissent, à force de manipuler des traitements de texte ou des environnements HTML. Il s'agit d'une[13] espace qui produit bien un blanc dans la ligne, mais où l'on n'a pas le droit de changer de ligne. Pour LaTeX, il s'écrit `~`.

Par exemple :

```
M.~Dugomier est bien sot.
```

En français, il faut une espace insécable (et plus petite que la normale, parfois) dans un certain nombre de contextes : devant une ponctuation double, ou autour des guillemets. Il est recommandé de ne pas saisir l'espace devant la ponctuation, mais d'utiliser le package `babel` convenablement paramétré. Par contre, pour ce qui est des « guillemets », il faut les saisir comme ça[14] :

```
est des <<~guillemets~>>, il...
```

[13] Oui, ça surprend, mais, en typographie, l'espace est au féminin. À ne pas confondre avec un espace, par exemple inter-sidéral en astrophysique, ou vectoriel en mathématiques. Ou un espace, utilisé pour désigner « un caractère de code ascii 32 », et qui est une notion d'informatique.

On retiendra que, sur les traitements de texte habituels, quand on saisit *un* espace, ça ajoute *une* espace au document.

[14] Avec `babel` et `frenchb`, on peut utiliser les deux commande `\og` (ouvrez les guillemets) et `\fg` (fermez les guillemets) à la place des guillemets, ces commandes produiront d'elles-mêmes les espaces insécables, d'une taille très légèrement inférieure à l'espace standard. Par exemple : `\og guillemets\fg` produira « guillemets » au lieu de « guillemets ».

Rappelons quelques règles pour les espaces insécables. Il en faut pour séparer une quantité d'une unité. On dit 12 km (`12~km`), ou 8 h (`8~h`). Il en faut pour rattacher le titre (civilité) et le nom (notre M. Dugomier de tout à l'heure), lorsque le titre est abrégé. Il en faut lorsque l'on souhaite marquer les milliers par des espaces : 12 000 000 de personnes (`12~000~000`). L'option `frenchb` de `babel` te simplifie un peu la vie en proposant : `\nombre{12000000}`. Je ne suis pas convaincu de l'utilité de la chose (c'est plus long à taper, et le résultat n'est guère différent).

6.3 Espaces ignorés

Certains espaces sont ignorés par LaTeX, en particulier :
– Ceux qui suivent immédiatement le nom d'une commande.
– Ceux qui sont en début de ligne.

C'est important de le savoir, et d'en tenir compte. Ainsi, lorsque je parle de LaTeX, j'utilise les artifices de saisie suivants :

```
\LaTeX{} est un bel...
\LaTeX\ est un bel...
\LaTeX, c'est beau.
\LaTeX , c'est beau.
```

Dans le premier cas, ce qui suit immédiatement la commande, c'est un groupe d'accolade (`{}`). L'espace ne suit donc pas *immédiatement* la commande, il est pris en compte et produira une espace mot normale.

Dans le second cas, ce qui suit immédiatement la commande `\LaTeX` c'est une commande un peu particulière, (`\␣`) qui produit une espace mot normale.

Enfin, dans le troisième cas, je ne souhaite pas mettre de blanc, puisqu'il y a une virgule juste après.

Le troisième et le quatrième cas donnent le *même* résultat.

6.4 Espaces ajoutés

LaTeX, en temps normal, traite les changements de ligne comme des espaces. C'est ce qui fait que dans un texte « normal », le dernier mot d'une ligne ne se retrouve pas collé au premier mot de la suivante.

Dans certains contextes, ça peut poser problème.

Par exemple :

```
\textit{Ceci, \textbf{
cela}.
```

Dans cet exemple, on obtiendra *deux* espaces entre la virgule et « cela », ce qui n'est pas forcément le résultat recherché.

Pour indiquer à LaTeX de ne pas prendre en compte le changement de ligne, il suffit de le mettre en commentaire, avec un `%`. Ça donne :

```
\textit{Ceci, \textbf{%
cela}.
```

Chapitre premier
Présentation

7 Gestion des fontes

7.1 Comment changer de fonte

L'une des premières choses que l'on cherchera à faire avec un traitement de texte sera de changer de fonte pour mettre en évidence certains passages. Par exemple en mettant quelques mots en italique ou en gras. Le principe est simple. Pour obtenir **toto** on tape :

```
... obtenir \textbf{toto} on ta...
```

On peut bien évidemment passer plusieurs mots comme paramètre à la commande `\textbf`. Par contre on évitera de lui passer plusieurs paragraphes ou plusieurs pages : d'abord, c'est interdit ; ensuite ça devient beaucoup plus difficile de compter les accolades ; et enfin ça complique la vie de LaTeX : il doit tout lire jusqu'à l'accolade fermante avant de commencer son travail.

D'autres changements que le gras peuvent être utilisés. Ils sont exhaustivement[15] recensés dans le tableau 7.1 page ci-contre.

Certains changements de fonte nécessitent une explication. Par exemple, `\textmd` sert à repasser en normal lorsque l'on est en gras. Deux exemple pour bien comprendre :

```
\textbf{Mot en \textmd{textmd} dans du gras.}
\textit{Mot en \textmd{textmd} dans de l'italique.}
```

Et ça donne ces deux lignes là :

Mot en textmd **dans du gras.**
Mot en textmd dans de l'italique.

Note, au passage, que le choix de la graisse n'a pas d'influence sur l'italicisation (plus de détails sur ce thème par la suite).

De même, `\textup` sert à « annuler » un `\textit`, un `\textsl` ou un `\textsc`.

La commande `\emph` a un usage très particulier. Elle met en évidence un passage dans son contexte. Dans un texte « normal » il passe en italique, dans un passage italique, il remet en « `\textup` ». C'est assez pratique : si on décide de passer tout un paragraphe en italique, les mots mis en évidence par ce moyen le restent.

7.2 Changements à longue portée

Lorsque l'on souhaite changer de fonte sur une grande portion de texte (plusieurs pages, voire tout le document), il est très désagréable de devoir se trimbaler des accolades partout. Il serait plus simple de dire « c'est du gras » et puis c'est tout.

Il existe des commandes pour cela, dont la liste fait l'objet du tableau 7.2 page suivante.

Écrire

[15] Ne sont recensés que ceux proposés par LaTeX en standard, d'autres peuvent être ajoutés par des packages.

Command	Résultat
`\textrm`	Normal (romain)
`\textbf`	**Gras** (bold face)
`\textit`	*Italique*
`\textsc`	Petites Capitales (Small Caps)
`\textsf`	Sans sérifications
`\textsl`	*Penché* (Slanted)
`\texttt`	`Machine à écrire` (Tele Type)
`\textmd`	Graisse normale (Medium)
`\textup`	Droit
`\textnormal`	Fonte par défaut
`\emph`	Mise en *évidence* (emphase)

Tab. 7.1: Changements de fonte standards

Changement de fonte	
à long terme	**à court terme**
`\rmfamily`	`\textrm`
`\sffamily`	`\textsf`
`\ttfamily`	`\texttt`
`\bfseries`	`\textbf`
`\mdseries`	`\textmd`
`\itshape`	`\textit`
`\slshape`	`\textsl`
`\scshape`	`\textsc`
`\upshape`	`\textup`
`\normalfont`	`\textnormal`
`\em`	`\emph`

Tab. 7.2: Changements de fonte

```
{\itshape toto}
```

est équivalent[16] à écrire

```
\textit{toto}
```

et presque équivalent à

```
\itshape toto\upshape
```

En effet, la troisième forme ne ramène pas forcément à la fonte de départ, par exemple si l'on est déjà en italique au début, ou si l'on est en PETITES CAPITALES.

`\normalfont` (comme `\textnormal`) change à la fois la famille, la série, et la forme[17] pour te ramener à la fonte par défaut qui est, a priori, du romain (`\textrm`) en graisse moyenne (`\textmd`) et en forme droite (`\textup`). C'est la seule commande dans le lot à affecter plus d'un paramètre à la fois.

Les commandes du tableau 7.2 page précédente peuvent également être utilisées comme des environnements, évitant ainsi l'accolade fermante qui se promène toute seule à la fin. Ça donnera par exemple ça :

```
\begin{itshape}
Tout un passage du document en italique.
\end{itshape}
```

7.3 Le club des cinq

LaTeX choisit la fonte qu'il va utiliser en fonctions de 5 paramètres indépendants les uns des autres :

1. La famille : c'est ce qui dit si on travaille en caractère romain (le texte normal), en télétype (comme ça), en sans-serif (comme ça), etc.
2. La série : c'est ce qui correspond à la graisse.
3. La forme : c'est italique, petites capitales, penché, droit, etc.
4. L'encodage : les deux classiques sont OT1, qui est le codage des fontes historiques de TeX, et T1 qui est le codage définit lors de la conférence de Cork pour les alphabets latins (compatible avec la norme ISO 8859-1). On peut aussi trouver la famille des T2[18] pour le cyrillique.
5. La taille.

C'est ce partage qui fait qu'il n'y a pas, de manière native, dans LaTeX, de petites capitales italiques, par exemple.

---

[16] À un microscopique détail près : dans le premier cas, la correction d'espacement due au fait que l'on sort de l'italique pour retourner, par exemple, à du caractère droit n'est pas ajoutée, ce qui vient créer des espacements disgracieux. Par exemple,
`V{\itshape V}V`
produira V *V*V alors que
`V\textit{V}V`
produira V *V* V. Je t'accorde que cette succession de lettres n'est pas la plus fréquente, par contre, si dans « l'Homme » je mets le « l » en italique pour insister sur son unicité, la mauvaise méthode donne « *l*'Homme » là où la bonne (avec `\textit`) donne « *l*'Homme ». Et comme ça ne t'aura pas échappé, dans le premier cas, l'apostrophe est trop proche du « *l* ».

[17] Shape, en anglais, pour ceux qui ne connaissent pas bien le grand-breton.

[18] Selon la langue que tu veux écrire en cyrillique, tu pourras être amené à utiliser T2A, T2B ou T2C.

8 Gestion des tailles

Ça fonctionne (tout simplement) comme les changements de fonte « à longue portée » (section 7.2 page 32). En effet, on ne change normalement pas de taille pour un mot en plein milieu d'une phrase.

Les commandes de changement de taille font l'objet du tableau 8.1.

On pourra également envisager les changements de taille comme des « environnements », c'est-à-dire comme ceci :

```
\begin{small}
Passage du texte \'ecrit plus petit que le reste.
\end{small}
```

Pour obtenir :

Passage du texte écrit plus petit que le reste.

Commande	Exemple
`\tiny`	Exemple
`\scriptsize`	Exemple
`\footnotesize`	Exemple
`\small`	Exemple
`\normalsize`	Exemple
`\large`	Exemple
`\Large`	Exemple
`\LARGE`	Exemple
`\huge`	Exemple

TAB. 8.1: Changements de taille

Chapitre premier
Présentation

9 Accents, caractères spéciaux et symboles

À retenir avant tout : si tu ajoutes la ligne suivante à ton document, et que tu as une machine normale[19], tu pourras taper directement tes accents au clavier. Au moins ceux que tu y trouveras.

```
\usepackage[latin1]{inputenc}
```

Les options du package `inputenc` sont recensées dans le tableau 9.1 page suivante.

Ce package nous est fourni par Alan JEFFREY et Frank MITTELBACH [148] ; il est maintenu par l'équipe du projet LaTeX3 ; il est livré avec LaTeX. Tu dois donc l'avoir sur ta machine si ton installation de LaTeX a moins de 10 ans.

9.1 Les accents

Si, comme moi, tu as appris l'informatique sur des machines UNIX d'il y a quelques années, tu as expérimenté les machines sans accents au clavier.

Même si tu as un ordinateur moderne, il te manque toujours au clavier un bon nombre de caractères accentués ou assimilés (les majuscules accentuées, le « œ », etc.).

Cela pose problème pour des textes en français[20].

Et si jamais tu essayes de taper autre chose que du français, tu vas très vite être ennuyé : pas de trace du ß ou du Å sur les claviers AZERTY[21] qu'on trouve en France.

De plus LaTeX a été prévu pour fonctionner sur tous les ordinateurs et dans toutes les configurations, donc il sait produire les accents s'ils n'existent pas sur la machine où il tourne. Il y a des commandes exprès pour cela. Elles sont dans le tableau 9.2 page 38.

Note qu'elles fonctionnent de manière très générale, c'est-à-dire que R-cédille (Ŗ) s'obtient le plus naturellement du monde : `\c{R}`.

9.2 Caractères non-américains

La plupart des caractères non-américains utilisés dans les langues européennes ont été prévus. Les commandes permettant de les obtenir font l'objet du tableau 9.3 page 38.

[19] Une machine normale, c'est-à-dire une machine dont le codage de caractères dans les fichiers texte répond à la norme ISO numéro 8859, alinéa 1. C'est le cas de tous les UNIX modernes, et de Windows, le plus souvent. Ce n'est pas le cas des machines Apple avant Mac OS X, et même sous Mac OS X, ce n'est le cas que quand le logiciel est configuré pour. Quant aux diverses variétés de Windows, ils ont encore tendance à mal coder un certain nombre de caractères. En particulier l'apostrophe.

[20] Rappelons, parce qu'il y a parfois confusion, que les accents relèvent de l'orthographe, et qu'ils sont donc rigoureusement obligatoires, y compris sur les majuscules. Qu'entends-je ? À la petite école, à toi aussi, on a dit le contraire ? Hé bien, on t'a raconté des sornettes. Dans l'écriture cursive (les lettres rondes qu'on apprend à l'école héritées de l'écriture à la plume) on omet parfois les accents sur les majuscules à la condition expresse qu'il n'y ait aucun risque d'ambiguïté.

Donc, en résumé : les accents sont obligatoires, y compris sur les majuscules.

[21] AZERTY parce que ce sont les premières lettres de la première ligne du clavier. Aux États-Unis, c'est QWERTY. En Allemagne QWERTZ.

Option	Codage
ascii	Mode de fonctionnement par défaut de TEX (et donc de LATEX).
ansinew	Tentative de normalisation ANSI (organisme de standardisation américain) pour des caractères européens, peu (pas ?) utilisé.
latin1	Norme ISO 8859-1 (dite latin 1), caractères des alphabets latins des langues occidentales.
latin2	ISO 8859-2. Langues d'europe centrale ou orientale à alphabet latin (polonais, etc).
latin3	ISO 8859-3. Langues d'europe mériodionale à alphabet latin (espagnol, italien, etc).
latin4	ISO 8859-4. Langues d'europe septentrionale à alphabet latin (Pays Baltes, le lapon, groënlandais, etc)
non prévu	ISO 8859-5. Langues européennes à alphabet cyrillique.
non prévu	ISO 8859-6. Arabe.
non prévu	ISO 8859-7. Grec.
non prévu	ISO 8859-8. Langues hébraïques.
latin5	ISO 8859-9. Comme ISO 8859-1, avec l'ajout de lettres islandaises.
non prévu	ISO 8859-10 (devrait être latin 6). Tentative, avortée, d'amélioration d'ISO 8859-4, codage inutilisé.
non prévu	ISO 8859-11. Thaï.
non prévu	ISO 8859-12. Indien.
non prévu	ISO 8859-13 (devrait être latin 7). Ajout à ISO 8859-10 pour les langues baltes.
non prévu	ISO 8859-14 (devrait être latin 8). Langues celtiques (irlandais, gallois).
latin9	ISO 8859-15. Comme latin1 mais en ajoutant quelques caractères oubliés.
next	Codage propre aux machines Next.
decmulti	Codage propre aux machines DEC.
applemac	Codage des systèmes Apple avant Mac OS X, ou des Mac OS X configurés hors-norme.
non prévu	Extension de Microsoft sur ISO 8859-13 (langues baltes) (devrait être cp1257).
cp1252	Codage utilisé classiquement par Windows, sur-ensemble d'ISO 8859-1 remplaçant les caractères de contrôles par des caractères jugés plus utiles par Microsoft.
non prévu	Extension Microsoft sur ISO 8859-5 (cyrillique) (devrait être cp1251).
cp1250	Extension de Microsoft sur ISO 8859-2.
cp437	Codage (ancien) d'IBM, utilisé autrefois par certains systèmes MS-DOS.
cp437de	Idem en Allemagne.
cp850	Mêmes origines et utilisations que cp437.
cp852	Idem.
cp865	Idem.

TAB. 9.1: Codages reconnus (ou non) par `inputenc`

Accent	Commande
á	\'a
à	\`a
â	\^a
ä	\"a
ā	\=a
ñ	\~n
ȧ	\.a

Accent	Commande
ă	\u{a}
a̲	\b{a}
ǎ	\v{a}
ç	\c{c}
a͡	\t{a}
a̋	\H{a}
ạ	\d{a}

TAB. 9.2: Accents en mode texte

Caractère	Commande
œ/Œ	\oe/\OE
æ/Æ	\ae/\AE
ß/SS	\ss/\SS
å/Å	\aa/\AA
ð/Ð	\dh/\DH
đ/Đ	\dj/\DJ
ŋ/Ŋ	\ng/\NG
þ/Þ	\th/\TH
ø/Ø	\o/\O
ł/Ł	\l/\Ł
ı	\i
ȷ	\j

Caractère	Commande
£	\pounds
¡	\textexclamdown
¿	\textquestiondown
«	<<
»	>>
«	\guillemotleft
»	\guillemotright
‹	\guilsinglleft
›	\guilsinglright
‚	\quotesinglbase
„	\quotedblbase
“	\textquotedblleft
”	\textquotedblright

TAB. 9.3: Caractères non-américains

Caractère	Commande
&	\&
%	\%
_	_
{	\{
~	$\sim$
\	\textbackslash
$	\$
}	\}

TAB. 9.4: Caractères spéciaux

9.3 Caractères spéciaux

Certains caractères sont interprétés de manière spéciale par LaTeX, comme tu t'en seras sûrement rendu compte.

On relèvera :

& utilisé dans les tableaux ;
~ espace insécable entre deux mots ;
% commentaire jusqu'à la fin de la ligne ;
\ début d'un nom de commande ;
_ indice en mode mathématiques[22] ;
$ début ou fin de mode mathématiques ;
{} délimiteurs.

Pour les obtenir dans du texte, il n'y a que deux solutions. La première est celle que j'utilise pour te montrer les noms de commandes comme `\texttt` : `\verb"\texttt"`. Le premier caractère non lettre qui suit cette commande sert de délimiteur pour marquer la zone sur laquelle elle s'applique. Cette commande indique simplement à LaTeX qu'il ne doit pas traiter la zone de texte concernée.

La méthode plus élégante et moins bourrine est d'utiliser les commandes du tableau 9.4. En effet, écrire 12% c'est plus joli que 12%.

9.4 Ponctuation

Le français a des règles d'espacement de la ponctuation[23] qui sont assez strictes (voir à ce sujet [153]) et que très peu de traitements de texte respectent. Par exemple, avant un point d'exclamation, il faut une espace fine insécable. LaTeX respectera ces règles, pour peu que tu aies indiqué que ton document est en français. En général tu indiques ça en utilisant le package `babel` avec l'option francais (ou frenchb).

Notons au passage que les points de suspension (...) s'obtiennent à l'aide de la commande `\dots`, et pas autrement. En particulier pas en tapant trois points. Ceci pour l'espacement soit le bon (...) au lieu de (...). Précisons clairement que les points de suspension (ou ellipse) sont au nombre de trois. Jamais moins, en aucun cas plus, comme le précise d'ailleurs [153]. Ils remplacent du texte, pas de la ponctuation : on conserve la ponctuation logique (virgules, points d'interrogations, etc.) sauf dans le cas du point.

[22] Reporte-toi au chapitre deuxième page 49 pour plus de précisions sur le mode maths en général et sur l'usage de l'indice en particulier.

[23] L'anglais aussi, mais ce ne sont pas les mêmes.

Attention : le package `babel`, en français, corrige la commande `\dots`, pour qu'elle produise le bon résultat (plus resséré qu'en anglais). Ainsi, si pour une raison ou une autre tu es amené à écrire en français sans pouvoir utiliser `babel`, il vaut mieux trois points que le `\dots` anglais.

9.5 Paragraphe

Contrairement aux ignominies que j'ai entendues proférer des centaines de fois un peu partout, il n'y a qu'une seule et unique bonne manière d'indiquer la fin d'un paragraphe : laisser une ligne blanche dans le fichier source.

On peut régler facilement l'alinéa en donnant une nouvelle valeur à la variable `\parindent` :

```
\parindent=15pt
```

La typographie française tolère de 0,5 à 2 fois le corps de la fonte en cours d'utilisation, en préférant une valeur entre 1 et 1,5 fois sa taille (dixit [153]), c'est-à-dire

```
\parindent=15pt
```

pour un document en 12pt. Note au passage que le corps par défaut sous LaTeX est 10pt (le plus courant en typographie) et que l'on peut le passer à 11pt ou 12pt en spécifiant `11pt` ou `12pt` en option au `\documentclass`.

De plus il existe une unité spéciale correspondant pile poil au corps en cours d'utilisation, c'est `em`. On pourra donc faire :

```
\parindent=1.25 em
```

pour obtenir une indentation de 1,25 fois la taille de fonte[24].

[24] L'unité `em` indique, en théorie, la largeur d'un M, ce qui est en règle générale la taille du texte utilisé : en 10 points, le M fait à peu près 10 points de large.

Chapitre premier
Présentation

10 Constructions simples

10.1 Environnements

Tu as déjà vu certains environnements à la section 8 page 35, ceux qui sont utilisés pour les changements de taille.

Eh bien, il en existe plein d'autres ! Les plus courants sont ceux permettant d'écrire du texte centré (`center`), au fer à droite[25] (`flushright`), ou au fer à gauche (`flushleft`). Par exemple[26] :

```
\begin{flushleft}
Ceci est un paragraphe d'exemple de
texte mis au fer \`a gauche avec
l'environnement \textenv{flushleft}.
\end{flushleft}
```

produira :

Ceci est un paragraphe d'exemple de texte mis au fer à gauche avec l'environnement `flushleft`.

De même que pour les changements de taille il existe une autre commande pour faire la même chose, ces trois environnements-là ont des commandes équivalentes : `\centering` pour centrer du texte, `\raggedright` pour mettre au fer à gauche (si, si, à gauche) et `\raggedleft` pour mettre au fer à droite. En effet, pour l'exemple ci-dessus il eut été équivalent d'écrire :

```
{\raggedright
Ceci est un paragraphe d'exemple de
texte mis au fer \`a gauche avec
la commande \Cde{raggedright}.}
```

la preuve, ça produit :

Ceci est un paragraphe d'exemple de texte mis au fer à gauche avec la commande `\raggedright`.

On s'en servira, par exemple, dans les tableaux ; comme c'est le cas dans la section 9.2 page 424.

10.2 Les listes

Trois autres environnements sont bons à retenir, ceux qui permettent d'obtenir des listes : `itemize` pour des listes normales, `enumerate` pour des listes numérotées, et `description` pour des listes type dictionnaire. Un exemple vaut toujours mieux qu'un long discours :

[25] « Au fer à droite », en typographie, ça veut dire « tout à droite », plus précisément « poussé jusqu'en butée vers le fer à droite ». Cela nous vient de l'héroïque époque du plomb. De même, composer « en drapeau à gauche » signifie composer « au fer à droite », cela vient de la forme de drapeau qu'évoque le côté gauche du texte dans ce cas.

[26] La commande `\textenv` n'est pas une commande LaTeX standard, elle a été définie spécialement pour la rédaction de ce manuel. De même, `\Cde`, que tu verras dans l'exemple suivant, est une commande définie pour ce manuel, qui me permet d'écrire le nom d'une commande, comme son nom l'indique.

```
\begin{itemize}
\item premier truc \‘a savoir,
\item deuxi\‘eme truc \‘a savoir,
\item troisi\‘eme truc \‘a savoir.
\end{itemize}
```

produira :

- premier truc à savoir,
- deuxième truc à savoir,
- troisième truc à savoir.

```
\begin{enumerate}
\item En premier lieu;
\item en second lieu;
\item en dernier lieu.
\end{enumerate}
```

produira :

1. En premier lieu ;
2. en second lieu ;
3. en dernier lieu.

Et enfin

```
\begin{description}
\item[itemize] liste normale,
\item[enumerate] liste
     num\’erot\’ee,
\item[description] liste
     comme celle-ci.
\end{description}
```

produira :

itemize liste normale,

enumerate liste numérotée,

description liste comme celle-ci.

10.3 Tableaux faciles

Obtenir un tableau n'est pas une chose enfantine avec LaTeX. Mais, tu verras, c'est tout à fait faisable.

La première chose à retenir est qu'un tableau est un environnement `tabular` qui prend comme argument le type des différentes colonnes qui doivent apparaître dans le tableau. À l'intérieur de ce tableau, le texte des différentes colonnes est délimité par le caractère `&` et celui des différentes lignes par la commande `\tabularnewline`, que l'on pourra dans la majorité[27] des tableaux abréger en `\\`.

Les trois types de colonnes les plus classiques sont :

l colonne de type gauche (*left*)
r colonne de type droite (*right*)
c colonne de type centré (*center*)

Voyons tout de suite un premier exemple simple :

```
\begin{tabular}{lr}
Premier mot & Second mot \\
Troisi\‘eme mot & Quatri\‘eme mot
\end{tabular}
```

produira :

Premier mot	Second mot
Troisième mot	Quatrième mot

La largeur des colonnes sera automatiquement adaptée à la largeur du texte qu'elles contiennent. Cela peut parfois poser problème, par exemple lorsque le texte à mettre dans une case est un peu long. Pour régler ce problème précis, LaTeX prévoit un type de colonnes exprès, le type `p`, auquel on indique la largeur de la colonne à créer. On en verra un exemple tout à l'heure.

Le prochain problème à régler est de mettre des filets[28] entre les lignes et entre les colonnes

[27] En fait, principalement, il suffit de retenir que la commande `\tabularnewline` doit être utilisée dans le cas des colonnes de type `p`, ainsi que dans toutes les dérivées de ce type simple, sitôt que l'on change la justification pour un centrage. Dans la pratique, si tu utilises `\\` et que ça ne marche pas, alors essaye `\tabularnewline`, celle-là, elle doit marcher partout.

[28] Des traits, si tu préfères.

du tableau. Deux choses à retenir à cet effet : un | entre deux descripteurs de colonne insérera un filet entre deux colonnes, et un `\hline` entre deux lignes les séparera par un filet.

Voyons maintenant l'exemple promis :

```
\begin{center}
\begin{tabular}{|l|p{4cm}|}                    \hline
Descripteur & Sens                          \\ \hline
\texttt{l}  & Colonne de type gauche        \\ \hline
\texttt{c}  & Colonne de type centr\'e      \\ \hline
\texttt{r}  & Colonne de type droite        \\ \hline
\texttt{p}  & Colonne d'une largeur
donn\'ee \`a l'avance. De nombreuses unit\'es
peuvent \^etre utilis\'ees pour
sp\'ecifier des distances \`a \LaTeX{}:
\texttt{cm} (centim\`etre), \texttt{mm}
(millim\`etre), \texttt{in} (pouce),
\texttt{pt} (point typographique anglais,
72,27\texttt{pt}=1\texttt{in})\dots       \\ \hline
\end{tabular}
\end{center}
```

produira :

Descripteur	Sens
`l`	Colonne de type gauche
`c`	Colonne de type centré
`r`	Colonne de type droite
`p`	Colonne d'une largeur donnée à l'avance. De nombreuses unités peuvent être utilisées pour spécifier des distances à LaTeX : `cm` (centimètre), `mm` (millimètre), `in` (pouce), `pt` (point typographique anglais, 72,27`pt`=1`in`)...

Tu vois bien, c'était pas si terrible que ça, les tableaux. Cependant, un grand nombre de raffinements sont possibles, pour en savoir plus, reporte-toi à la section 9 page 423 du chapitre troisième.

Chapitre premier
Présentation

11 Chapitres et table des matières

11.1 Le titre

La première chose que tu vas taper dans ton document, c'est son titre. Donc LaTeX a prévu des commandes pour cela. Elles viennent généralement juste après

```
\begin{document}
```

Un exemple suffira :

```
\title{Reconnaissance de phon\'emes par
cartes de Kohonen sans apprentissage
supervis\'e}
\author{Benjamin \textsc{Bayart}
\\ I$_4$PASTI \and Pascal \textsc{Vincent}
\\ I$_4$PASTI}
\date{\today}
\maketitle
```

Selon que tu écris un **report** ou un **article**, LaTeX décidera s'il est nécessaire ou non de réserver une page entière au titre. Plus exactement, si tu écris un **report**, l'option titlepage est prise par défaut, alors que si tu écris un **article**, c'est l'option notitlepage qui est prise par défaut. Libre à toi d'outrepasser l'option par défaut en en spécifiant une explicitement.

De même, comme je te le disais déjà en 3.1.2 page 12, l'environnement `titlepage` te permet de faire un peu tout ce que tu veux. Il n'utilise pas les commandes `\title`, `\author` et `\date`.

11.2 Division du document

Un document classique se divise en plusieurs parties, chapitres, sections, etc. Chacun a un numéro qui lui est propre et une mise en page qui doit être toujours la même au long du document si l'on souhaite rester cohérent. C'est pour ces raisons que LaTeX prévoit des commandes. Elles sont recensées dans le tableau 11.1 page ci-contre.

Par exemple :

```
\chapter{Cahiers des charges}
```

commence un nouveau chapitre en le numérotant convenablement et en lui donnant le titre « Cahiers des charges ».

Notons au passage que le niveau de division[29] `\chapter` existe pour les rapports et pas pour les articles. Notons également que les chapitres sont numérotés continûment au travers des parties. C'est-à-dire que la première partie contiendra les chapitres 1 à i et la seconde les chapitres $i+1$ à n.

[29] Ou niveau de sectionnement.

Commande	Niveau de sectionnement
\part	1
\chapter	2
\section	3
\subsection	4
\subsubsection	5
\paragraph	6
\subparagraph	7

TAB. 11.1: Commandes de sectionnement

11.3 Chapitre d'introduction

L'introduction, dans un rapport, tout comme la conclusion d'ailleurs, est souvent un problème. En effet, elle a valeur de chapitre, mais généralement on ne la numérote pas.

Il te suffit de savoir que pour toutes les commandes de sectionnement, en leur ajoutant une * elles viennent créer des parties, sections... non-numérotées pour croire qu'un simple

```
\chapter*{Introduction}
```

est la solution idéale. Malheureusement tel n'est pas le cas. En effet, un simple `\chapter*` n'est pas recensé en table des matières et posera, pour ce cas précis et pour certaines mises en page, des problèmes.

La meilleure solution serait d'utiliser pour cela la commande `\introchapter` (qui n'est définie nulle part) :

```
\introchapter{Introduction}
```

Si tu veux l'utiliser, en voici la définition que tu peux copier dans ton document, ou encore dans un fichier `mesajouts.sty` qui deviendra alors le package `mesajouts` :

```
\newcomand\introchapter[1]{%
  \chapter*{#1}
  \addcontentsline{toc}{chapter}{#1}
  \markboth{\MakeUppercase{#1}}
           {\MakeUppercase{#1}}
}
```

Sinon sache simplement que cette commande est équivalente à

```
\chapter*{Introduction}
\addcontentsline{toc}{chapter}{Introduction}
\markboth{INTRODUCTION}{INTRODUCTION}
```

et ne pose surtout pas de question là-dessus sans avoir lu avant [115] et [149].

11.4 Tables diverses

Comme je te sais impatient de savoir comment on produit une table des matières, je vais te donner la bonne méthode pour en faire une. En premier lieu, sache qu'il existe plusieurs

Type	Commande
Table des matières	`\tableofcontents`
Liste des tableaux	`\listoftables`
Liste des figures	`\listoffigures`

TAB. 11.2: Déclarations des trois types de table les plus courants

sortes de table, et donc plusieurs façons de les produire. Les trois principales font l'objet du tableau 11.2.

Pour obtenir une table n'importe où dans le document, tu insères la commande correspondante. Tu peux donc sans difficulté aucune faire apparaître plusieurs fois la table des matières, si cela te fait plaisir.

Il est à noter que dans un **article** les trois tables figurent sur la même page que le texte qui les suit, alors que pour un **report** LaTeX change de page à la fin de chaque table.

11.5 Recompilation

Pour obtenir une table des matières à jour, il faudra au moins compiler deux fois ton document, parfois plus.

Cela vient du mode de fonctionnement de LaTeX. Pour tout un tas de bonnes raisons, LaTeX ne mémorise pas tout le document pendant la compilation, mais seulement la page en cours[30]. C'est ce qui lui permet de traiter des documents potentiellement très longs : quand il travaillera sur la beaucoutième page, il n'aura pas la mémoire encombrée par les innombrables pages précédentes.

Mais ça présente un inconvénient majeur : il ne peut pas savoir à quelle page commence le chapitre 3 avant d'en être arrivé là. Pour résoudre ce petit problème, il triche honteusement. Au début de la compilation, il ouvre un fichier annexe (le fameux fichier `.aux`) dans lequel il prendra des notes durant toute la compilation. Il y notera par exemple que le chapitre 3 commence à telle page et a tel titre.

Au début de la toute première compilation, ce fichier n'existe pas, alors, ne sachant pas ce qu'il faut mettre dans la table des matières, LaTeX te met une table des matières vide. Et durant toute cette compilation, il prendra des notes dans son fichier.

Ainsi, au début de la seconde compilation, il relit ses notes, et quand tu lui demandes la table des matières, il sait ce qu'il faut mettre dedans.

Comme je te le disais, parfois, il faut plus de deux compilations pour que le résultat soit juste. Le plus simple pour le comprendre est de prendre un exemple. Supposons un document assez volumineux (un livre, ou une thèse), sa table des matières fait plusieurs pages, mettons 4. Il est composé comme suit : une page de titre, numérotée 0, puis la table des matières, en page 1, puis le premier chapitre. Durant la première compilation, LaTeX ne sait pas quel contenu mettre dans la table des matières, alors il met une table vide, en page 1. Il commence donc le premier chapitre sur la page 2, et indique dans ses notes « *le chapitre 1 commence en page 2 et a tel titre* ». Durant la deuxième compilation, quand tu lui demandes la table des matières, il regarde ses notes, et il s'empresse de te composer une table des matières de la bonne longueur, puisqu'il a des notes sur l'ensemble du document. Cette table des matières fait donc 4 pages, elle commence en page 1 et se termine en page 4. Elle indique, comme il l'a noté, que le chapitre 1 commence en page 2. Puis il commence le chapitre 1, mais cette

[30] C'est un peu plus subtile que ça en réalité. Dans certains cas précis, il mémorise un peu plus que la page en cours, mais en tous cas pas tout le document.

fois-ci sur la page 5. La table des matières a donc la bonne taille, mais elle est fausse. Durant cette deuxième compilation, il prend aussi des notes (il prend toujours des notes, en fait), et cette fois-ci, il note « *le chapitre 1 commence en page 5 et a tel titre* ». Quand tu attaqueras la troisième compilation, la table des matières aura la bonne taille, mais en plus, bonheur, elle sera juste.

À retenir : pour un petit document, deux compilations sont suffisantes, pour un gros document, il en faudra au moins trois.

Au moins trois ? Malheureusement oui, il faut parfois encore d'avantage de compilations. Par exemple si ton ouvrage a un index, à la fin de la troisème compilation, tous les numéros de page sont justes, tu peux alors produire ton index. Mais il te faudra une *quatrième* compilation pour inclure cet index.

Tu pourrais être tenté par un raisonnement simplificateur du style « bon, bin, si c'est comme ça, je compile toujours 4 fois et puis c'est marre ». D'abord, ce sera souvent inutile, mais en plus il existe des cas pervers où ces 4 compilations ne suffiront pas. Il n'existe pas de règle absolue pour déterminer le nombre de compilations indispensables pour un document véritablement complexe, par exemple pour ce livre. On retiendra cependant :

- au moins trois compilations pour un ouvrage dont la table des matières est longue,
- s'il y a une bibliographie, il faut la compiler (avec BibTEX, par exemple) juste après la première compilation, et l'inclure : le remplacement des points d'interrogation que LATEX utilise par les vraies références bibliographiques pourra éventuellement décaler du texte, il faut donc trois compilations après la bibliographie,
- s'il y a des références à l'intérieur du document, il faut recompiler *jusqu'à ce que LATEX cesse de dire que les références ont changées*[31],
- maintenant que tous les numéros de page sont justes, tu peux inclure l'index et recompiler,
- si l'index n'est pas le dernier chapitre de ton ouvrage, le fait de le rajouter a dû changer les numéros de page des chapitres suivants, il faut donc recompiler une fois pour corriger la table des matières.

[31] En effet, aussi longtemps que les références ne sont pas justes, LATEX te le dit à la fin de la compilation.

Chapitre premier
Présentation

12 Inclusion de fichiers

Toi, lecteur programmeur, tu as l'habitude d'écrire tes programmes dans plusieurs fichiers, avec un système d'inclusion : un fichier principal qui fait appel à plusieurs petits fichiers annexes. C'est plus facile à utiliser pour des programmes, il en serait sûrement de même pour un document LaTeX, non ? Mais oui. Tu n'as qu'à le faire.

Pour cet usage LaTeX, qui est grand, prévoit deux commandes : `\input` et `\include`. Leur usage est expliqué ci-dessous.

12.1 Inclusion simple

`\input` permet d'importer purement et simplement un fichier : `\input toto` agira exactement comme si le contenu de `toto.tex` était dans ton texte.

Cela permet de se créer ses propres bibliothèques de macros, bien que l'écriture propre d'un package soit *fortement recommandée*.

12.2 Compilation partielle

`\include` marche différemment. Tout fichier importé par lui doit contenir des chapitres complets. Il créera un fichier `.aux` par fichier inclus. Il permet une compilation partielle des documents quand il est utilisé avec `\includeonly`.

Exemple :

```
\include{intro}
\include{chap1}
\include{chap2}
\include{chap3}
\include{chap4}
\include{concl}
```

lira les 6 fichiers comme il faut, puis si tu ajoutes *avant*

```
\includeonly{intro.tex,concl.tex}
```

LaTeX ne compilera que l'introduction et la conclusion, et en gardant les bons numéros de page, en conservant une table des matières complète et tout comme il faut. Toutefois, il faudra qu'au moins une compilation complète ait eu lieu pour qu'il puisse se repérer.

De plus aucun `\include` ne doit apparaître avant le `\begin{document}` ni dans les fichiers lus avec `\include`.