Université Pierre et Marie Curie

Module lm204 de la licence math-info

Apprentissage et pratique de LaTeX

Manuel Pégourié-Gonnard

1er semestre 2008–2009

Préambule

LaTeX est un système de préparation de documents qui occupe une position dominante parmi les mathématiciens pour la réalisation de cours, feuilles d'exercices, notes de travail, articles de recherche… Les raisons de son omniprésence sont, outre sa capacité à mettre convenablement en forme les formules mathématiques les plus compliquées, la qualité professionnelle du résultat (tant pour le texte que les mathématiques) permettant une publication directe, mais surtout sa façon de concevoir un document structuré, en séparant son fond (le sens du texte) de sa forme (la mise en pages) et en déchargeant l'auteur de tâches fastidieuses, lui permettant ainsi d'atteindre une grande productivité.

En dehors du milieu mathématique universitaire, LaTeX a aussi sa place. Il est utilisé par des scientifiques de toutes disciplines et certains enseignants mathématiques du secondaire, pour ses performances dans la composition des mathématiques. Certains linguistes et « littéraires » l'apprécient pour son excellente gestion de document complexes, munis par exemple de lourds appareils de notes. Quelques éditeurs généralistes l'utilisent de préférence à des outils de PAO pour produire plus rapidement des documents structurés ; des banques et grandes compagnies l'exploitent pour mettre en forme automatiquement et avec une bonne qualité des documents issus de bases de données.

Malgré ces nombreux succès, LaTeX reste trop souvent perçu comme un outil de spécialiste. Une des raisons est sa difficulté d'apprentissage. Celle-ci est en partie réelle : d'une part, sa méthode de préparation des documents, séparant code source et résultat, n'est pas intuitive pour qui n'a pas une certaine culture informatique, et elle se prête peu à un auto-apprentissage sans document de référence. Enfin, de part son histoire, LaTeX ne forme pas un tout cohérent, et son univers peuplé de modules et de programmes auxiliaires, souvent encore trop éloigné des standards actuels, peut dérouter.

Mais une partie des difficultés d'apprentissage est plus environnementale qu'intrinsèque. En effet, on « apprend » souvent LaTeX sur le tas, à la va-vite, pour produire un mémoire, forcément à rendre pour hier, en recopiant des recettes trouvées ça et là, chez un collègue ou sur internet. Sur ce point, LaTeX ne se distingue pas des autres outils ou disciplines : on l'apprend mieux et plus vite en suivant un cours structuré et en prenant le temps de progresser étape par étape, que tout seul et dans l'urgence.

Par ailleurs, LaTeX a beaucoup évolué au cours de ces dernières années. Il est par exemple capable de produire des documents PDF exploitant les diverses possibilités de ce format. De nombreux modules généralistes ou spécialisés fournissent des solutions simples à des problèmes autrefois compliqués. Les (bonnes) pratiques évoluent, tenant compte de l'expérience accumulée. Apprendre LaTeX sur le tas en recopiant des exemples à droite à gauche ne permet pas de profiter de ces progrès : on se contente souvent de la première solution, fut-elle très sous-optimale, qui semble « marcher » sur le moment, et qui deviendra vite une (mauvaise) habitude. On risque ainsi de rencontrer plus tard certains problèmes, ou de tout simplement perdre en efficacité ou en qualité du résultat.

Pour toutes ces raisons, je suis extrêmement heureux d'avoir l'occasion cette année, grâce notamment au dynamisme et à la disponibilité de certains collègues, de donner ce cours, que j'espère structuré et bien informé, dans le cadre d'un module la deuxième année de licence. De telles initiatives sont à ma connaissance rares en France, et pourtant me semblent très utiles, tant la capacité à communiquer, notamment par le biais de documents électroniques soignés, est importante aujourd'hui dans le milieu scientifique (mais pas seulement).

L'outil informatique, censé faciliter de nombreuses tâches, est déroutant ou rebutant pour certains. Dans le domaine de la typographie, il a longtemps été une source d'appauvrissement et de baisse de qualité. Au contraire, LaTeX représente, dans son domaine, un exemple d'outil efficace, relevant le défi de rendre plus simple et plus rapide la production de documents, en conservant une qualité typographique à la hauteur de la tradition des compositeurs au plomb. J'espère vous en convaincre au long de ce cours.

Bien sûr, l'efficacité de LaTeX ne se comprend vraiment que par la pratique. Il est indispensable, en lisant ce cours, de mettre en application les éléments présentés. Pour cela, les exercices accompagnant le cours sont un bon point de départ. Ils sont en général accompagnés de corrigés détaillés, et comportent parfois des compléments de cours, qui ne seront pas toujours repris dans ce polycopié.

Mais les exercices sont souvent assez réducteurs : la seule façon de vraiment pratiquer LaTeX est de mettre en forme de vrais documents. En effet, les techniques présentées dans ce cours ne sont que des outils, dont le but est de servir votre document, et qu'il faut savoir utiliser à bon escient.

Je vous souhaite autant de plaisir à lire ce cours que j'en ai eu à l'écrire et à l'enseigner. Toutes vos remarques ou questions sont les bienvenues à l'adresse `mpg@math.jussieu.fr`.

Remerciements

Trop brièvement, j'aimerais remercier les personnes suivantes.
Pour l'organisation du module : Omer Adelman, Dominique Le Brigand, Laurent Koelblen, Muriel Thicot. Pour LaTeX au CIES Jussieu : Céline Chevalier, Benjamin Collas, Ismaël Soudères, Michel Landau. Du plateau 7C ou assimilé : Jérôme Gärtner, Julien Cornebise, Julien Grivaux et bien d'autres. Les contributeurs de `fr.comp.text.tex`, en particulier Jean-Côme Charpentier et Denis Bitouzé. Pour leur réactivité et leur intérêt, je remercie tous les étudiants ayant suivi le module. Et bien sûr, pour avoir créé TeX et l'avoir donné au monde, Donald Knuth. Merci !

Table des matières

1 Prise de contact

1.1 Présentation du module

1.1.1 Organisation

Le module est organisé en 12 séances de cours-TP de deux heures. En moyenne, sur chaque séance, une heure sera consacrée au cours et l'autre à la pratique, sauf la dernière séance consacré à l'examen. Les exercices seront en général trop longs pour être terminés en une heure et sont destinés à être finis à la maison. Ils contiennent parfois des questions volontairement difficiles ou demandant un travail de recherche personnelle.

Le découpage en chapitres du présent polycopié correspond en première approximation au découpage du cours en séances ; parfois, l'ordre sera légèrement modifié pour clarifier l'exposition.

L'évaluation des connaissances consistera en deux parties obligatoires et une optionnelle.

1. Un document libre (rapport) à préparer à la maison, individuellement ou par binôme. Les modalités précises sont décrites dans un document séparé [1].
2. Un examen pratique sur machine, en temps limité.
3. Un devoir à la maison optionnel.

L'examen, comme le devoir à la maison et la plupart des exercices, consistent en un document à reproduire aussi fidèlement que possible (hormis le filigrane « à reproduire » lui-même).

La dernière version de ce polycopié, ainsi que les exercices, leurs corrigés, les supports de présentation utilisés en cours, et différents documents d'accompagnement (aide-mémoire, liste de symboles mathématiques, etc.) sont disponibles à l'adresse suivante :

`http://people.math.jussieu.fr/~mpg/lm204/files/`

Une page voisine [2] comporte par ailleurs la progression du cours séance par séance, avec chaque fois que c'est possible l'indication des parties correspondantes des différents documents de référence. Une bibliographie séparée à été distribuée.

1.1.2 Contenu

Le but du cours n'est en aucun cas de tout vous apprendre sur LaTeX : d'une part parce que c'est impossible, d'autre part parce que certaines des ses possibilités ne vous seront sans doute pas utiles immédiatement. En revanche, les objectifs sont les suivants :

– Vous fournir un socle solide de connaissances de bases, si possible exempt de mauvaises habitudes. Ce cours se veut tour d'horizon relativement neutre des possibilités de LaTeX.
– Vous fournir des pistes pour être en mesure par la suite de continuer seul votre apprentissage.

1. `http://people.math.jussieu.fr/~mpg/lm204/files/doc-eval-rapport.pdf`
2. `http://people.math.jussieu.fr/~mpg/prog.html`

Les cinq premiers chapitres du cours fournissent le minimum vital pour composer la plupart des documents scientifiques ne comportant pas d'autres éléments complexes que des formules mathématiques. Les chapitres suivants présentent soit des éléments plus spécifiques (figures, tableaux, *listings*, présentations) soit des éléments généralistes d'usage moins courant, ou approfondissent certains points évoqués précédemment.

1.2 Présentation de LaTeX

1.2.1 Possibilités

Même s'il est principalement connu pour ses possibilités mathématiques, qui sont en effet pratiquement illimitées, LaTeX est adapté à la plupart des types de documents. En particulier, il prend soin de beaucoup de détails concernant le texte : césure, justification, ligatures… Par ailleurs, il est capable de produire directement des graphiques sophistiqués (possibilité qui ne sera pas étudiée en cours, faute de temps).

Pour quelques exemples, je renvoie aux transparents [3]. Observez les effets de texte, la parfaite intégration des mathématiques au reste du texte, les possibilités d'arrangements complexes de formules… Pour d'autres exemples, ce cours lui-même, ainsi que les trois livres et deux documents électroniques donnés en référence sont bien sûr réalisés avec LaTeX.

Par ailleurs, même si nous n'aurons pas trop le temps d'approfondir cet aspect, LaTeX est aussi un langage de programmation complet. C'est ce qui lui confère une partie de sa puissance : d'une part parce que de nombreuses personnes ont ainsi écrit (et publié) des modules étendant ses possibilités, que nous pouvons utiliser, mais aussi parce qu'on peut programmer certains aspects du document pour les automatiser ou les rendre plus aisément modifiables.

1.2.2 Particularités

Il est important de réaliser que LaTeX est un système de préparation de documents qui n'a essentiellement rien à voir avec un traitement de texte (à part la finalité commune : produire un document mis en forme). Dans un traitement de texte de type Word® ou OpenOffice Writer, le texte est mis en forme *en direct* pendant que vous le saisissez. En LaTeX, le processus est asynchrone : vous saisissez dans un fichier votre texte accompagné d'instructions de mise en forme [4] et vous demandez de temps en temps à LaTeX d'exécuter ces instructions.

Un peu de vocabulaire : le fichier dans lequel vous écrivez votre texte et les instructions de mise en forme, dont l'extension est `.tex`, est appelé le *fichier source* ou, par élision, *le source*. Le processus qui le transforme en un document visualisable ou imprimable (en général au format PDF) est appelé *compilation*, par analogie avec des langages informatiques comme le C.

Si vous n'avez pas une certaine habitude des langages informatiques compilés, il est essentiel de bien comprendre la distinction entre le source et le document final. Ces deux fichiers sont complémentaires : le PDF est la forme distribuable, le source est la forme modifiable. La compilation, qui transforme le source LaTeX en document PDF, n'est *pas* un processus réversible. Il est comparable à l'impression d'un document Word® : le document imprimé est lisible sans ordinateur, mais n'est plus modifiable comme l'est le fichier `.doc`. Cette séparation est heureuse car elle permet de bien séparer les rôles : pour l'auteur, le `.tex`, pour les lecteurs, le `.pdf`.

3. `http://people.math.jussieu.fr/~mpg/lm204/files/01-intro.pdf`
4. On dit que LaTeX est un langage de balisage. Il est sur certains points comparables au couple HTML & CSS.

Interrompons un moment cette présentation un peu théorique pour regarder comme exemple un extrait d'un document LaTeX (on verra bientôt à quoi ressemble ou source complet). On voit sur l'exemple 1.1 comment le texte est entremêlé d'instructions de mise en forme comme `\textit` pour mettre en italique, ou `$...$` pour délimiter les fragments de formule, `^` pour mettre en exposant, de commandes pour obtenir des symboles particuliers comme un point centré.

```
Des \textit{maths} ici :
$2^2 = 2 \cdot 2 = 2 + 2$.
```

Des *maths* ici : $2^2 = 2 \cdot 2 = 2 + 2$.

Exemple 1.1 – Extrait de source.

Un autre point sur lequel LaTeX distingue plus les rôles qu'un traitement de texte ordinaire est que les tâches de saisie des instructions, et d'exécution de ces instructions, sont séparées. Le programme nommé `latex` (ou plus précisément, `pdflatex`) lit le fichier `.tex` et produit le fichier `.pdf`, mais il faut un autre programme, appelé *éditeur de texte*, pour produire le source `.tex`. Sur l'exemple 1.1, LaTeX assure la transformation de la partie de gauche en la partie de droite, mais il faut un logiciel distinct pour produire la partie de gauche.

Ainsi, contrairement à un traitement de texte qui fait tout (saisie et exécution des instructions), un environnement de travail LaTeX est divisé en au moins deux programmes. En fait, c'est encore pire : le programme `pdflatex` lui-même a besoin de nombreux fichiers supplémentaires (classes et modules, comme on le verra dans un instant) pour fonctionner, et parfois d'autres programmes auxiliaires (par exemple pour produire un index ou une bibliographie). Tous ces fichiers et programmes sont généralement réunis au sein d'une *distribution* TeX. Pour travailler avec LaTeX, il faut vous donc installer une distribution TeX *et* un éditeur adapté à LaTeX (voir sec. 1.3 pour les détails).

Une partie du développement de LaTeX, dont témoigne la nombre impressionnant de modules disponibles, est rendu possible par son statut de logiciel libre [5] : chacun est libre d'étudier son fonctionnement en détail, de l'améliorer, de le redistribuer. Tous les logiciels recommandés dans ce cours sont des logiciels libres ; vous n'avez pas besoin de payer pour les utiliser le plus légalement du monde.

Enfin, concluons cette section sur les particularités de LaTeX par l'énoncé de quelques un de ses défauts.

- LaTeX est plus difficile à apprendre seul qu'un traitement de texte, et il faut plus de temps pour se sentir capable de faire ce que l'on veut avec.
- Il est parfois difficile à installer, une peu hétéroclite, et l'intégration entre les différents modules n'est pas toujours parfaite.
- Pendant la phase de compilation, si votre source comporte des erreurs de syntaxe, les messages d'erreurs sont parfois délicats à comprendre.
- Pour certains types particuliers de documents, le manque de retour visuel immédiat peut être gênant.

Le dernier point est intrinsèque, mais largement compensé par le fait que pour la plupart des types de documents, le mode de travail « source & compilation » est au contraire un avantage. Quant aux trois premiers points, j'espère que ce cours vous aidera à les surmonter.

5. http://www.april.org/fr/articles/intro

1.3 Installation

Comme expliqué précédemment, pour travailler avec LaTeX, on a besoin de deux éléments logiciels distincts :

1. une distribution TeX [6] ;
2. un éditeur de texte.

L'éditeur de texte peut être extrêmement basique (le `notepad` de Windows® suffit en théorie) mais il est préférable d'en utiliser un spécialement adapté à LaTeX, qui proposera par exemple une mise en évidence des commandes, la possibilité de lancer la compilation et la visualisation du document sans quitter l'éditeur, un filtrage des messages d'erreur, etc. On appelle parfois IDE [7] un tel éditeur.

Si l'éditeur est un programme relativement simple, la distribution est au contraire un ensemble important de programmes, occupant beaucoup d'espace disque. Pour cette raison, elles existent parfois en plusieurs versions de *minimale* à *complète*. Une version présentée comme minimale n'est en aucun cas suffisante pour tout ce qui sera vu dans ce cours. Il est recommandé de toujours installer la version complète.

Il existe essentiellement deux ou trois distributions TeX, et un grand nombre d'éditeurs. Voyons maintenant quelques choix possibles et comment les installer suivant les plateformes.

1.3.1 Windows

On a le choix parmi les distributions TeX Live [8] et MiKTeX [9]. Pour les éditeurs, le plus classique est TeXnicCenter [10], mais TeXmaker [11] est aussi un très bon choix. (Deux autres bons éditeurs sont disponibles, mais ne sont pas libres : ils s'agit de LeD et WinEDT.)

Un document séparé [12] décrit en détails l'installation d'un environnement MiKTeX & TeXnicCenter sous windows. Ce choix n'est pas forcément meilleur que les autres mais correspond à la configuration utilisée en TP. Pour l'installation de cette configuration, il est essentiel de suivre ce document au moins jusqu'à la section 3.2 incluse.

1.3.2 Mac OS X

Bien qu'il y ait en principe plusieurs choix, l'un se distingue ici très clairement : il s'agit de la distribution MacTeX [13], dérivée de TeX Live et munie d'un installateur spécifique pour Mac OS. En outre, elle intègre un éditeur, TeXShop, et constitue donc à elle seule un environnement complet de travail.

D'autres éditeurs intéressants, ainsi qu'un peu plus de documentation, sont fournis dans le paquet supplémentaire MacTeXtras [14], qui peut donc être intéressant à installer.

6. LaTeX est en fait lui-même une extension d'un autre système, appelé TeX. Une distribution contient en général le programme TeX et tous ses dérivés.
7. Pour *integrated development environment*, soit environnement de développement intégré.
8. `http://tug.org/texlive/`
9. `http://ww.miktex.org/`
10. `http://www.toolscenter.org/`
11. `http://www.xm1math.net/texmaker/index_fr.html`
12. `http://people.math.jussieu.fr/~mpg/lm204/files/doc-install-miktex+txc.pdf`
13. `http://tug.org/mactex/`
14. `http://tug.org/mactex/mactextras.html`

Signalons de suite un point technique mais facile à régler : il est conseillé, dès la première utilisation de TeXShop, d'aller dans le menu TeXShop, préférences, onglet document, et dans la liste déroulante « encodage », de choisir latin1 (ou son synonyme iso-8859-1) ou éventuellement UTF-8. Si vous choisissez cette dernière option, il vous faudra changer l'option passée à `inputenc`, comme expliqué dans la note 16, par rapport à certains exemples.

1.3.3 Linux

La distribution de référence est ici TEX Live (il faut mieux oublier teTEX qui se fait vraiment vieille). Installez-la de préférence depuis le gestionnaire de votre distribution Linux. Elle est souvent présentée sous la forme de plusieurs paquets : si un paquet nommé `texlive-full` existe, installez-le. Sinon, installez au moins tous les paquets dont le nom contient `latex` ou `recommended` ; parfois certains paquets particuliers ne comportent pas le mot `texlive` dans leur nom même s'ils en font partie : c'est par exemple le cas de `latex-xcolor`, `latex-beamer`, `cm-super` et `lmodern` sous Debian et Ubuntu.

Concernant l'éditeur, le plus courant et sans doute un des meilleurs est Kile, mais vous pouvez aussi utiliser TeXmaker, également à installer depuis le gestionnaire de paquets de votre distribution Linux.

Sous la plupart des distributions, l'encodage par défaut des éditeurs sera l'UTF-8. Si vous conservez cet encodage, il vous faudra penser à changer l'option passée à `inputenc`, comme expliqué dans la note 16, par rapport à certains exemples. Sinon, vous pouvez sélectionner latin1 ou son synonyme iso-8859-1 comme encodage par défaut, généralement dans le menu édition, sous-menu préférences, de votre éditeur.

1.4 Premiers documents

Voyons maintenant à quoi ressemble un source LATEX complet. Le source 1.1 est insuffisant pour un usage réel, mais juste assez complet pour compiler correctement. Il produit un document d'une page qui comporte le seul texte « *Hello, world!* ».

```
\documentclass{minimal}
\begin{document}
Hello, world!
\end{document}
```

Source 1.1 – Source minimum compilable.

On voit ici que les mots précédés de `\` sont des *commandes* qui n'apparaissent pas dans le document mais sont interprétées par LATEX. Les mots entourés d'accolades qui suivent une commande sont des *arguments* de cette commande. La structure d'un source comporte quelques contraintes :

1. La première ligne doit toujours être une déclaration de classe de document, c'est-à-dire utiliser la commande `\documentclass` avec un argument.
2. Tout le texte devant apparaître dans le document produit doit être contenu entre les balises `\begin{document}` et `\end{document}`.

La partie précédent le `\begin{document}`, ici réduite à la première ligne, est appelée *préambule* du document ; celle qui suit jusqu'au `\end{document}` constitue le *corps* du document ; enfin tout ce qui suit le `\end{document}` est ignoré.

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{amsmath, amssymb}
\usepackage[french]{babel}
\newcommand\abs[1]{\lvert #1\rvert} % valeur absolue
\begin{document}
Un document plus réaliste, avec du texte pour commencer.

Puis un deuxième paragraphe avec une équation $\abs{x} = r_0$
à l'intérieur.
Et un dernier paragraphe ?
\end{document}
```

SOURCE 1.2 – Tout petit document en français.

Présentons maintenant un exemple un peu plus réaliste de très court document LaTeX écrit en français. Le source 1.2 illustre un certain nombre de points de la syntaxe de LaTeX. Vous n'avez pas besoin de tout comprendre pour le moment, mais voici toutefois quelques points à retenir.

1. Il existe plusieurs classes de documents, dont la plus courante pour de courts document est `article` : c'est celle que nous utiliseront presque tout le temps pour les exemples et les exercices.
2. La classe de document peut prendre des options entre crochets, comme ici `a4paper`. On utilisera systématiquement cette option, la taille de papier par défaut étant sinon le format `letter` américain.
3. On peut, dans le préambule uniquement, charger des modules [15] qui étendent les possibilités de LaTeX ou modifient son comportement, avec la commande `\usepackage`.
4. Pour pouvoir saisir des caractères accentués, on a besoin du module `inputenc`. L'option à lui passer dépend de l'éditeur utilisé et de son réglage. Pour simplifier, j'écrirai toujours `latin1` : c'est le seul réglage reconnu par TeXnicCenter, l'éditeur utilisé en cours. Si on utilise TeXShop, on prendra soin de le régler comme expliqué en sec. 1.3.2 ; pour TeXmaker ou Kile, voir sec. 1.3.3. [16]
5. Pour que les caractères accentués utilisés en Français apparaissent correctement dans le PDF et que LaTeX puisse calculer automatiquement les coupures de mots, il faut charger le module `fontenc` avec l'option `T1`.
6. On peut charger d'autres modules spécifiques, comme ici `amsmath` et `amssymb` qui étendent les possibilités mathématiques de LaTeX.

15. Aussi appelés extensions, paquets ou en anglais *packages*.

16. Cependant, afin de laisser le choix libre, je propose toutes les solutions des exercices en latin1 et en utf8. Si vous réglez votre éditeur sur UTF-8 (ou que c'est son réglage par défaut), pensez à remplacer `latin1` par `utf8` dans chaque exemple ou corrigé qui fera appel à `inputenc` et à utiliser les fichiers de solution dont le nom contient `utf8`.

7. On utilise enfin le module `babel` avec l'option `french` pour annoncer que le document produit est en Français, ce qui permet à LaTeX de traduire certains titres produits automatiquement, de respecter certaines règles typographiques françaises, etc.
8. On peut en outre définir des nouvelles commandes, comme ici `\abs`, que l'on peut ensuite utiliser dans le document comme toute autre commande standard de LaTeX.
9. Le caractère `%` introduit un *commentaire* : tout ce qui le suit jusqu'à la fin de la ligne est totalement ignoré par LaTeX. Il sert soit à expliquer ce qui est fait dans le source, soit à masquer temporairement certains éléments, sans pour autant les effacer du source.
10. Enfin, les coupures de ligne au sein du source ne correspondent pas aux coupures de ligne dans le document final : LaTeX calcule celles-ci automatiquement pour bien remplir la zone de texte.

Comme annoncé, vous n'avez pas besoin de tout comprendre de ce document maintenant. Par contre, retenez absolument que certaines options et certains modules sont *obligatoires* pour chaque document. Un source minimal est disponible en ligne [17] et je vous conseille *fortement* de vous en servir comme base de départ pour tous les documents en Français que vous produirez. (Ne pas utiliser l'option `a4paper` et les trois modules chargés par cette base sera considéré comme une faute.)

```
\usepackage{hyperref}
Le \href{http://tug.org}{groupe
  d'utilisateurs de \TeX}.
```

Le groupe d'utilisateurs de TeX.

Exemple 1.2 – Exemple d'exemple.

Par ailleurs, dans ce polycopié, tous les exemples supposeront que vous utilisez ce source de base, et ne signaleront que les modules supplémentaires à charger. De plus, pour économiser la place, le `\begin{document}` ne sera pas montré : tout est supposé être dans le corps du document, sauf les lignes précédées d'un filet vertical, qui sont à ajouter au préambule de base. Ainsi, l'exemple 1.2 représente en fait [18] le source complet 1.3.

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{hyperref}
\begin{document}
Le \href{http://tug.org}{groupe d'utilisateurs de \TeX}.
\end{document}
```

Source 1.3 – Source développé de l'exemple 1.2

Si vous voulez essayer les exemples du cours (ce qui est recommandé), vous devrez donc à chaque fois faire cette substitution qui devrait vite devenir un réflexe.

17. `http://people.math.jussieu.fr/~mpg/lm204/files/doc-source-minimal.tex`
18. Pour être exact, il faut éventuellement remplacer `latin1` par `utf8`, selon le réglage de votre éditeur.

Il est maintenant recommandé d'essayer de compiler les sources 1.1 et 1.2. Vous pouvez les copier-coller depuis la version PDF de présent polycopié.

Si vous observez attentivement le résultat des la compilation de ces deux exemples, vous constaterez le numéro « 1 » en bas de page du 1.2 : c'est le numéro de page. LaTeX (ou plus précisément, la classe de document) gère automatiquement pour vous les détails comme la numérotation des pages. Essayez de changer la classe `minimal` en `article` dans le 1.1, vous verrez apparaître alors le numéro de page.

1.5 Bases théoriques

1.5.1 Commandes, arguments, environnements

Reprenons maintenant de façon un peu plus méthodique certains éléments vus ci-dessus. LaTeX est un langage de balisage : dans le source sont entremêlés le texte réel de votre document et des commandes pour le mettre en forme, sauf au tout début du document, dans le préambule, qui ne comporte que des instruction, comme le chargement de modules.

Pour pouvoir donner des instructions à LaTeX, certains caractères ont une signification spéciale. En voici la liste complète :

```
\ { } $ & # ^ _ ~ %
```

Vous connaissez déjà certains d'entre eux, les autres seront présentés petit à petit. Ce sont les dix caractères qui n'apparaîtront pas tel quel dans le document si vous les saisissez simplement dans le source mais provoqueront plutôt des actions spéciales. Nous verrons au prochain chapitre comment les inclure dans un document.

Le caractère `\` sert à indiquer le début d'une commande. Les noms de commandes en LaTeX sont de deux types :

1. Les commandes-mot sont les plus courantes : leur nom est constitué d'une ou plusieurs lettres.
2. Les commandes-caractère : leur nom est constitué d'un unique caractère qui n'est pas une lettre.

Dans ce contexte, *lettre* signifie l'un des 26 caractères non accentués de l'alphabet, et la casse (majuscule ou minuscule) compte : `\abc` et `\Abc` sont deux commandes distinctes. Un exemple de commande-caractère est la commande `\%` qui permet d'obtenir le caractère %.

Une commande peut prendre zéro, un, ou plusieurs arguments, normalement délimités par des accolades `{...}`. Certains arguments, comme les options de la commande `\documentclass`, sont optionnels : il peuvent être omis. Dans ce cas, ils sont délimités par des crochets carrés `[...]`. Dans tous les autres cas, le nombre d'argument d'une commande doit être respecté : ne pas passer d'argument à une commande qui en attend provoquera des erreurs, parfois très étranges. Il faut aussi bien distinguer `[...]` et `{...}` qui ne sont pas interchangeables.

Dans tout ce cours, on présentera ainsi les nouvelles commandes :

`\documentclass[`⟨*options*⟩`]{`⟨*classe*⟩`}`

Il est entendu que les arguments entre crochets peuvent être omis. Les éléments comme ⟨*options*⟩ ne sont pas à saisir tels que, mais à remplacer par une valeur appropriée.

Deux commandes sont particulières : \begin et \end. Utilisées conjointement sous la forme

\begin{⟨env⟩}
⟨contenu⟩
\end{⟨env⟩}

elles définissent un *environnement* dont le nom est ⟨*env*⟩ et le contenu ⟨*contenu*⟩. Ainsi, on peut dire que le corps du document est le contenu de l'environnement {document}. À chaque fois que je ferais référence ainsi à un nom entre accolades, il s'agira du nom d'un environnement [19].

Ceci constitue l'essentiel de la syntaxe « régulière » de LaTeX. Il existe quelques autres éléments syntaxiques un peu moins cohérents, que nous aurons le temps de découvrir plus tard.

1.5.2 Espaces et fins de ligne

Des règles particulières régissent les espaces en LaTeX. Parfois surprenantes au début, elles sont en fait très pratiques à l'usage (sauf peut-être la première). Les voici.

1. Les espaces suivant une commande-mot sont ignorés.
2. Les espaces en début d'une ligne sont ignorés.
3. Plusieurs espaces successifs sont équivalents à un seul espace.
4. Un retour à la ligne est équivalent à un espace.

La première règle demande un peu d'attention. Par exemple [20] pour écrire que « LaTeX c'est bien », il faut saisir par exemple \LaTeX{} c'est bien, car la version naïve \LaTeX c'est bien donne le mauvais résultat « LaTeXc'est bien ». Ici, les accolades vides ne produisent rien, mais assurent que l'espace qui les suit n'est plus « mangé » par la commande-mot qui précède : je parlerai souvent de « protéger » l'espace pour désigner ce procédé. Les trois autres règles sont en fait tellement commodes qu'on les oublie vite.

Comme on l'a vu, un retour à la ligne seul est équivalent à un espace. Deux retours à la ligne successifs (c'est-à-dire une ligne vide dans le source) sont considérés comme délimitant un paragraphe [21]. Plusieurs lignes vides successives sont équivalentes à un seule ligne vide.

Comme un retour à la ligne dans le source est interprété comme un espace (ce qui est très pratique en fait), pour provoquer un retour à la ligne dans le document, le plus simple est de changer de paragraphe en laissant une ligne vide dans le source. Ceci insère aussi un retrait d'alinéa au début du paragraphe suivant.

On verra au prochain chapitre comment provoquer un retour à la ligne qui ne soit pas un changement de paragraphe. On y apprendra aussi les commandes servant à laisser des espaces vides horizontalement ou verticalement : pour l'instant, retenez qu'il est vain de chercher à atteindre ce résultat en laissant plusieurs espaces ou plusieurs lignes vides dans le source.

19. La réciproque est peut-être fausse : il m'arrivera d'omettre les accolades quand le contexte indique assez clairement qu'il s'agit d'un environnement.

20. Comme en exercice, j'introduis parfois en exemple des nouvelles commandes, comme ici celle servant à composer le logo LaTeX. Quand elles sont suffisamment simples, elles sont alors considérées comme connues pour la suite, même si elles ne sont jamais formellement présentées ailleurs. C'est le cas ici.

21. On peut aussi séparer les paragraphes par la commande \par ; je le ferais souvent dans les exemples pour gagner de la place, mais dans la pratique une ligne vide est bien plus courante.

1.6 Exercices

Il n'y a pas d'exercice formel pour ce chapitre : votre travail pour cette semaine consiste à installer LaTeX sur votre ordinateur personnel si vous en possédez un, ou à vous familiariser avec l'installation LaTeX disponible dans les salles libre-service de l'université. Vous devez savoir compiler et visualiser un document simple. Vous êtes encouragés à partir de l'exemple donné en cours et à le modifier.

De façon générale, une attitude active, consistant à essayer de « jouer » avec LaTeX pour voir ce que vous pouvez en faire, au-delà de la simple résolution des exercices proposés, ne pourra vous être que bénéfique.

2 Le mode texte

2.1 Caractères et symboles particuliers

2.1.1 Caractères réservés

On a déjà présenté (section 1.5.1) les dix caractères réservés par LaTeX pour des usages particuliers, et promis d'expliquer comment les obtenir dans le document. Il est temps de tenir promesse : voici donc les commandes.

Caractère	Usage	LaTeX	textcomp	Math
\	commandes		\textbackslash	\backslash
{	argument ou	\{	\textbraceleft	\{
}	groupe	\}	\textbraceright	\}
$	mode math	\$		\$
&	alignements	\&		
#	définitions	\#		
^	exposant	\^{}		
_	indice	_		
~	espace insécable	\~{}	\textasciitilde	
%	commentaire	\%		

Comme on le constate, LaTeX ne propose pas de commande pour obtenir certains caractères en mode texte. On doit faire appel au module `textcomp` pour disposer des commandes de l'avant-dernière colonne, et être en mode mathématique pour utiliser celles de la dernière colonne (les modes mathématiques seront vus au chapitre 4).

Je rappelle que faire appel au module `textcomp` signifie placer la ligne

```
\usepackage{textcomp}
```

dans le préambule du document. Si vous oubliez de le faire et essayez d'utiliser quand même une des commandes fournies par ce module, vous obtiendrez le message d'erreur *undefined control sequence*, qui signifie en gros « commande non définie ». En général, vous obtenez ce message d'erreur soit quand vous avec oublié de charger un module, soit quand vous avez fait une faute de frappe dans le nom d'une commande.

2.1.2 Accents et symboles

À part les caractères réservés que nous venons de voir, vous pouvez saisir tous les autres caractères directement dans le source. Tous ? Non ! Une poignée d'irréductibles résiste, essentiellement pour deux types de raison :

1. Ils n'existent pas dans l'encodage d'entrée [1] utilisé.

1. C'est celui qu'on passe en paramètre à `inputenc`.

2. Vous ne savez pas les saisir sur votre clavier.

Dans le premier cas, on trouve par exemple le symbole de l'euro qui n'existe pas en `latin1` [2]. Dans le deuxième cas… ça dépend de vous et de votre clavier.

Dans les deux cas, la stratégie reste la même : utiliser des commandes pour obtenir ces symboles. Il y a tout d'abord des commandes d'accent, généralistes, pour obtenir tous les caractères accentués. Ces commandes, ainsi que celles servant à obtenir les ligatures orthographiques usuelles, ont des noms assez faciles à retenir (voire deviner). L'exemple 2.1 présente les plus importantes d'entre elles.

```
\'A, \`A, \^A, \"A, \dots\par
\'E, \'I, \'i, \'n, \dots\par
\c Ca me fend le c\oe ur !
Tch\"u\ss!
```

Á, À, Â, Ä, …
É, Í, í, ń, …
Ça me fend le cœur ! Tchüß!

Exemple 2.1 – Quelques accents, cédilles et ligatures.

Quelques commentaires sur cet exemple : observez l'utilisation de `\par` à la fin des deux première lignes pour changer de paragraphe (et donc revenir à la ligne) dans le document. Remarquez aussi la commande `\dots`, qui produit un résultat plus élégant que trois points à la suite : « … » est plus lisible que « ... ».

Enfin, des détails syntaxiques : dans `\'A`, on a une commande `\'` et son argument `A`. Comme l'argument est réduit à une lettre, on n'est pas obligé de l'entourer d'accolades, mais si on le désire on peut écrire `\'{A}`. C'est d'ailleurs sans doute plus clair pour `\c{C}a`. Ici, il faut faire attention à ne pas écrire `\cCa` sans espace, car LaTeX y verrait une commande nommée `cCa`. C'est dans des exemples comme `c\oe ur` que la règle disant que les espaces sont avalés est, pour une fois, pratique.

D'autres symboles ne dérivent pas d'une lettre. Ils sont alors obtenus par des commandes spécifiques. Par exemple, pour le symbole euro, le module `textcomp` fournit la commande `\texteuro`. On peut obtenir un symbole euro d'apparence différente (celui fournit avec les polices par défaut de LaTeX n'est pas très joli) avec la commande `\EUR` du [3] module `marvosym`.

D'autres symboles sont disponibles, bien plus qu'il n'est possible de présenter ici. Le document de référence pour tous les symboles disponibles sous LaTeX est `symbols-a4.pdf` [4]. Il est important en utilisant ce document de prêter attention aux modules à charger pour disposer des commandes indiquées. Nous verrons tout bientôt (sec. 2.1.4) comment localiser des documents de référence.

Enfin, certains caractères particuliers s'obtiennent sans même faire appel à une commande : LaTeX peut automatiquement « mélanger » deux ou plusieurs caractères [5] pour en obtenir un autre. Par exemple, on obtient un tiret moyen « – » avec deux tirets à la suite `--` et un long « — » avec trois (mais ça ne va pas plus loin).

Une autre ligature utile est celle qui permet d'obtenir des guillemets "anglais" : on saisit pour cela ``` ``anglais'' ```. Par contre, en Français, on utilisera plutôt des guillemets « français » : ce qui

2. À ceux qui seraient tentés d'utiliser à la place l'option `utf8` pour éviter ce problème (ce qui est au demeurant une bonne idée), je rappelle qu'il ne suffit pas de changer l'option d'`inputenc`, mais qu'il faut aussi changer le réglage de son éditeur de texte, et que ceci n'est malheureusement pas possible sous TeXnicCenter.
3. Il faut donc écrire `\usepackage{marvosym}` dans le préambule pour avoir le droit d'utiliser cette commande.
4. `http://mirror.ctan.org/info/symbols/comprehensive/symbols-a4.pdf`
5. Un tel mélange s'appelle une *ligature*.

fournit une bonne transition vers la prochaine section.

2.1.3 Franchouillardises

En plus de traduire certains textes automatiques, d'assurer une césure correcte des mots, et d'aider au respect de quelques règles orthotypographiques propres au Français, l'option `french` [6] de `babel` fournit aussi quelques commandes utiles.

Exemple	Résultat
`\og guillemets \fg`	« guillemets »
`M\up{me}, D\up{r}`	M^{me}, D^{r}
`1\ier, 1\iere, 1\ieres`	1er, 1re, 1res
`2\ieme 4\iemes`	2^{e} 4es
`\No 1, \no 2`	N^{o} 1, n^{o} 2
`20~\degres C, 45\degres`	20 °C, 45°
`\bsc{M. Durand}`	M. Durand
`\today`	19 avril 2009

Comme on le sait, les commandes-mot comme `\ier` ont tendance à avaler les espaces qui les suivent, de sorte que `le 1\ier jour` donne « le 1erjour ». Il faut penser à protéger l'espace en saisissant `le 1\ier{} jour` pour obtenir le résultat correct « le 1er jour ».

Les commandes fournies par `french` permettent d'éviter cet inconvénient et de restaurer automatiquement l'espace lorsqu'il est nécessaire, à condition de charger le module `xspace` auparavant. Pour cela, décommentez [7] la ligne 8 du source minimal [8] proposé. (Ceci sera systématiquement utilisé dans les corrigés des exercices.)

2.1.4 Interlude : documentation

Je l'ai déjà dit, vous ne saurez jamais tout de LaTeX. Il est donc important de prendre l'habitude de consulter des documents de référence, et j'en citerai souvent. La plupart des documents que je cite sont disponible en ligne, mais aussi présents sur votre disque dur : ils font partie de la distribution TeX.

Pour les localiser, chaque distribution fournit un outil. Sous MiKTeX, il s'appelle `mthelp`, sous TeX Live c'est `texdoc`. Dans les deux cas, l'utilisation est la même : il s'agit de taper, en ligne de commande [9] l'une des commandes suivantes :

`mthelp` ⟨*fichier*⟩	`mthelp` ⟨*module*⟩
`texdoc` ⟨*fichier*⟩	`texdoc` ⟨*module*⟩

Vous pouvez ainsi rechercher soit un fichier dont vous connaissez le nom, comme `symbols-a4.pdf`, soit de la documentation sur un module dont vous connaissez le nom, comme `babel`. Par

6. Synonyme des options `francais` (sans cédille) et `frenchb` (pour *french babel*). J'utiliserai souvent `frenchb` par habitude, car les trois n'ont pas toujours été synonymes.
7. C'est-à-dire, retirez le signe `%` présent en début de ligne.
8. `http://people.math.jussieu.fr/~mpg/lm204/files/doc-source-minimal.tex`
9. Pour en ouvrir une sous Windows, enfoncez simultanément la touche Windows du clavier et la lettre R, puis saisissez `cmd` dans la boîte de dialogue. Sous Mac ou Linux, vous avez probablement une icône « terminal » quelque part.

exemple, pour rechercher la documentation de babel sous TEX Live, vous tapez `texdoc babel` dans une ligne de commande. Pour rechercher le fichier `symbols-a4.pdf` sous MiKTEX, tapez `mthelp symbols-a4`.

Si pour une raison ou une autre vous n'arrivez pas à localiser la documentation sur votre disque dur, vous pouvez essayer en ligne. Le site de référence pour tous les modules LATEX et la plupart des documents le concernant est le CTAN [10]. Il dispose d'une page de recherches [11]. Par ailleurs, chaque module LATEX est recensé dans un catalogue : des informations (dont parfois un lien vers la documentation) sur chaque module sont disponibles à l'adresse `http://ctan.org/pkg/`⟨*nom*⟩.

Par ailleurs, dans la version PDF du présent polycopié, le nom d'un module est en général un lien vers cette page de description, de même que le nom d'un fichier que je présente pour la première fois en souvent un lien vers sa version sur le CTAN.

À titre d'exercice, vous pouvez rechercher dans `symbols-a4` comment produire le symbole du Yen (¥). Faites-le vraiment, chercher des choses dans un document touffu n'est pas si facile au début et il faut vous y habituer.

2.2 Changements de fonte

2.2.1 Modèle théorique

Pour LATEX, une police est déterminée par 4 paramètres [12] logiquement indépendants : la famille, la graisse, le forme, et la taille. Ici, « logiquement indépendant » signifie qu'on peut demander à LATEX de changer le taille sans qu'il change la graisse, mais cela ne signifie pas que toutes les combinaisons existent nécessairement dans une fonte donnée.

Contrairement à certains logiciels qui encouragent à utiliser beaucoup de polices au sein d'un même document, LATEX ne propose par défaut que trois familles : un famille romaine (avec empattements) qui est en générale la police principale, une famille sans empattements (dite aussi sans sérif) et une de type `machine à écrire`, généralement à chasse fixe (c'est-à-dire que tous les caractères ont la même largeur).

2.2.2 Tout sauf la taille

Parmi les quatre paramètres vu ci-dessus, la taille est un peu à part. La table 2.1 montre comment changer les trois autres. Les polices par défaut de LATEX sont utilisées pour les illustrations. Les valeurs par défaut de ces paramètres sont en général : famille romaine, forme droite et graisse moyenne. On peut à tout moment revenir à ces valeurs par défaut avec la commande `\normalfont` qui n'affecte pas la taille.

Comme vous le constatez, chacune des commandes de changement de fonte présentée existe sous deux formes : une commençant par `\text`, qui prend un argument et n'agit que sur son argument, et une forme dite déclarative, qui agit sur toute la suite du texte jusqu'à ordre contraire.

Pour être plus précis, la forme déclarative agit jusqu'à la fin du *groupe* courant. Le moyen le plus simple de définir un groupe est d'utiliser une paire d'accolades, à condition qu'elles en servent pas

10. `http://ctan.org/`
11. `http://ctan.org/search.html`
12. C'est un peu faux car il faudrait aussi considérer l'encodage, mais c'est une notion technique heureusement inutile ici. On a déjà bien assez à faire avec l'encodage d'entrée.

Famille		
`\textrm{}`	`\rmfamily`	romain
`\textsf{}`	`\sffamily`	sans empattements
`\texttt{}`	`\ttfamily`	`chasse fixe`
Graisse		
`\textmd{}`	`\mdseries`	graisse normale
`\textbf{}`	`\bfseries`	**gras**
Forme		
`\textup{}`	`\upshape`	droit
`\textit{}`	`\itshape`	*italique*
`\textsl{}`	`\slshape`	*penché*
`\textsc{}`	`\scshape`	PETITES CAPITALES

TABLE 2.1 – Famille, graisse et forme de fonte.

déjà à autre chose, comme à délimiter l'argument d'une commande. Il faut alors bien prendre de garde de placer l'accolade ouvrante *avant* la commande déclarative dont on veut limiter l'action.

Les environnements définissent naturellement des groupes. On pourra donc aussi les utiliser comme délimiteurs de la portée d'une commande déclarative. On verra de tels exemples le corrigé de l'exercice 2. L'exemple 2.2 montre la technique utilisant des accolades, et illustre comment les paramètres peuvent se combiner. Comme on le constate, les certaines combinaisons « ne marchent pas », par exemple il n'y a pas de petites capitales en style machine à écrire. Dans ce cas, LaTeX ignore une des commandes.

```
Un {\bfseries mot gras \textit{et italique}
aussi}. Retour à la normale. \texttt{Machine
\textbf{à écrire}.} \textsc{Petites capitales
\textbf{grasses}, à \texttt{chasse fixe} ?}
```

Un **mot gras *et italique* aussi**. Retour à la normale. `Machine` **`à écrire.`** PETITES CAPITALES **GRASSES**, À `chasse fixe` ?

EXEMPLE 2.2 – Famille, graisse et forme de fonte.

Dans l'exemple 2.2, il aurait été plus naturel d'utiliser `\textbf` à la place de `\bfseries`. Par contre, il faut savoir que les commandes à argument ne peuvent pas servir pour des changements de fonte durant plus d'un paragraphe. C'est une sécurité censée vous aider à détecter les accolades fermantes oubliées : en cas d'oubli, vous obtenez le message d'erreur suivant.

```
! Paragraph ended before \text@command was complete.
```

L'avantage est que le numéro de ligne indiqué est proche du lieu où vous avez probablement oublié une accolade.

Enfin, il existe une commande particulière pour changer la forme de fonte courante : la commande `\emph` sert à mettre du texte en valeur, en passant en italique si l'on était en droit et réciproquement. Il est conseillé de l'utiliser plutôt qu'on commande de mise en forme directe, lorsque qu'on voudra *insister* sur un mot comme ici.

De façon générale, dès qu'on saura comment définir de nouvelles commandes et environnements (chapitre 5), il faudra éviter de faire apparaître les commandes de changement de fonte

Taille	
`\tiny`	taille
`\scriptsize`	taille
`\footnotesize`	taille
`\small`	taille
`\normalsize`	taille
`\large`	taille
`\Large`	taille
`\LARGE`	taille
`\huge`	taille
`\Huge`	taille

TABLE 2.2 – Tailles de fontes relatives

directement dans le document, et ne les utiliser que dans la définition d'autres commandes *sémantiques*. Nous y reviendrons en temps voulu.

2.2.3 La taille

Les commandes proposées par LaTeX pour changer de taille n'existent contrairement aux autres que sous forme déclarative. En effet, un changement de taille concerne rarement un mot isolé, mais plutôt tout une portion de texte. Les commandes disponibles sont listées dans la table 2.2.

La taille de texte normale est donnée par `\normalsize`, qui est indépendant de `\normalfont`. Cette taille est fixée par une option de classe de document. Par défaut, cette taille est de 10 points dans la classe `article`. Les options `11pt` et `12pt` sont disponibles. Par exemple, pour fixer une taille principale de 11 points pour l'ensemble du document, on écrira

```
\documenclass[a4paper, 11pt]{article}
```

comme première ligne du préambule. Toutes es autres commandes s'adapteront alors : `\large` avec une taille de base de 11 points est un peu plus grand que `\large` avec une taille de base de 10 points, etc.

Pour disposer de tailles de base plus variées (ce qui est rarement utile en fait), on devra remplacer la classe `article` par la classe `extarticle`. On dispose alors des nouvelles options `8pt`, `9pt`, `14pt`, `17pt` et `20pt` pour la taille de base. Ainsi, pour un livret écrit assez petit, on commencera son source par

```
\documenclass[a5paper, 9pt]{extarticle}
```

si le livret est imprimé sur du papier A5. (Écrire si petit sur du papier A4 n'est guère raisonnable.)

Ce modèle de tailles de fonte (une taille de base, et des commandes de taille relatives à cette dernière) est très pratique pour la plupart des documents, car il permet des changements faciles et cohérents. Cependant, on a parfois besoin de spécifier une taille absolue (par exemple pour obtenir un titre en très gros, ou réaliser une affichette).

`\fontsize{`⟨*corps*⟩`}{`⟨*interligne*⟩`} \selectfont`

La commande \fontsize permet d'indiquer un corps (un taille) de fonte en points ; il faut alors aussi préciser la taille de l'interligne (l'espace vertical entre la base de deux lettres de lignes consécutives). Une règle approximative est de choisir un interligne de 120% du corps : par exemple,

```
\fonsize{40}{48} \selectfont
```

pour écrire en taille 40. Il ne faut pas oublier de faire suivre \fontsize de \selectfont pour que les changements prennent effet.

Les commandes de changement de taille au sein du document qu'elles soient relatives comme \large ou absolue comme \fontsize, modifient aussi l'espace entre les lignes : pour \fontsize, c'est le rôle du deuxième argument, pour les autres c'est automatique. Un point auquel il faut prêter attention si l'on veut que l'interligne soit pris en compte est de bien terminer le paragraphe *avant* de revenir à la taille normale. Pour cela, on aura intérêt à terminer le paragraphe par un \par plutôt que par une ligne vide de façon à placer celui-ci avant l'accolade fermante qui délimite la fin du changement de taille, comme illustré par l'exemple 2.3.

Si par contre on utilise des environnements comme center, flushleft et flushright (vus plus loin, section 2.4) pour délimiter l'action du changement de taille, on n'a aucune précaution particulière à prendre, car ces environnements délimitent aussi un paragraphe. Voir le corrigé de l'exercice 2 pour un exemple correct d'utilisation de cette technique.

```
{\tiny Un petit paragraphe de texte écrit en
  tout petit avec un interligne incorrect.}\par
{\tiny Un petit paragraphe de texte écrit en
  tout petit avec un interligne correct.\par}
```

Un petit paragraphe de texte écrit en tout petit avec

un interligne incorrect.
Un petit paragraphe de texte écrit en tout petit avec
un interligne correct.

Exemple 2.3 – Changements de taille et interligne.

2.2.4 Le reste

Le dernier paramètre déterminant l'apparence d'un texte, en plus de la famille, la graisse, la forme et la taille de police, est sa couleur, au moins pour les documents électroniques. LaTeX ne permet pas tout seul de gérer la couleur, mais le module xcolor fournit les commandes nécessaires.

```
\textcolor{⟨couleur⟩}{⟨texte⟩}
{\color{⟨couleur⟩} ⟨texte long⟩}
```

Comme pour les commandes de changement de fonte normales, on dispose des deux formes : avec argument ou déclarative. Pour la forme déclarative, il faut penser à utiliser des accolades comme ci-dessus, ou à profiter d'un environnement pour restreindre l'action de la commande. À tout moment, on peut revenir à la couleur normale avec la commande déclarative \normalcolor.

Il y a plusieurs noms de couleurs prédéfinis : pour en avoir une liste complète, consulter la documentation d'xcolor (voir sec. 2.1.4 comment la trouver), puis aidez-vous des signets pdf ou des liens cliquables de la table des matières pour arriver rapidement à la section 4 « *colors by name* ». Par défaut, seuls les noms de la section 4.1 sont définis. Vous pouvez mélanger entre elles des couleurs existantes de la façon suivante : blue!30!white donne un mélange de 30%

de bleu et 70% de blanc. Nous verrons à la section 5.2.5 comment définir de nouveaux noms de couleur. Ces possibilités sont présentées sur l'exemple 2.4 : là aussi, il serait plus naturel d'utiliser `\textcolor` les deux fois, `\color` n'a été choisie la deuxième fois que pour illustrer sa syntaxe (très particulière).

```
Du \textcolor{red}{rouge} flashy.\par
{\color{red!30!black} Moins} flashy.
```

Du rouge flashy.
Moins flashy.

Exemple 2.4 – Changements de couleurs

Sur certaines installations vieilles ou incomplètes, `xcolor` n'existe pas. On peut alors utiliser `color` à la place : les commandes citées ci-dessus restent les mêmes, ainsi que les noms des couleurs de base, mais certaines possibilités, comme le mélange de couleurs, ne sont pas disponibles.

Enfin, outre les changements de fontes et de couleurs, il est également possible d'appliquer quelques autres effets au texte, comme le soulignement. Ce dernier est *à éviter* autant que possible. Il est très utilisé dans les document manuscrits ou dactylographiés car c'est un des seuls moyen de mise en valeur qui existe dans ces cas. Avec LaTeX (ou un traitement de texte classique), on dispose de tellement d'autres moyens d'expression typographiques qu'il faut mieux se passer du soulignement, souvent peu élégant (problèmes de lettres à jambages, d'interligne, …)

Ceci dit, par souci de complétion, citons le module `soul` qui permet entre autres de souligner des portions de texte. Il fournit aussi des commandes pour quelques autre effets, comme le texte barré ou interlettré (c'est-à-dire avec un espace supplémentaire entre chaque lettre : cette mise en forme peut être utile pour des titres par exemple), illustrées par l'exemple 2.5.

```
\usepackage{soul}
\ul{Souligné} \par
\st{Barré} \par \so{Interlettré}
```

Souligné
~~Barré~~
I n t e r l e t t r é

Exemple 2.5 – Quelques possibilités du module `soul`

2.3 Listes

Nous connaissons désormais les commandes pour mettre en forme des fragments de texte, souvent de l'ordre de quelques mots, et traduire ainsi sa structure à petite échelle. Voyons maintenant quelques éléments de structure à moyenne échelle. Le plus important est la structure de liste, qui se décline en LaTeX (comme en HTML) en trois variantes : liste à tirets, liste numérotée, et liste descriptive.

Dans les trois cas, la liste elle-même est délimitée par un environnement : `itemize` pour une liste à tirets, `enumerate` pour une liste numérotée, et `description` pour une liste descriptive. Chaque élément, ou point, de la liste (y compris le premier) est précédé de la commande `\item`. Cette commande ne prend en général pas d'argument, mais on peut lui passer un argument optionnel entre crochets : dans le cas d'une liste à tirets ou numéroté, l'argument remplace le symbole de début d'item ou la numérotation ; dans le cas d'une liste descriptive, il est d'usage d'utiliser cet élément pour le nom du terme à décrire. Ces trois type de listes sont illustrés par l'exemple 2.6 ; on peut les emboîter entre elles comme on le verra en exercices.

```
Les trois types de listes sont :
\begin{itemize}
  \item les listes à tirets ;
  \item les listes numérotées ;
  \item les listes descriptives.
\end{itemize}
Les trois types de listes sont :
\begin{enumerate}
  \item les listes à tirets ;
  \item les listes numérotées ;
  \item les listes descriptives.
\end{enumerate}
Les trois environnement utiles sont :
\begin{description}
  \item[itemize] pour les listes à tirets ;
  \item[enumerate] pour les listes numérotées ;
  \item[description] pour les descriptives.
\end{description}
```

Les trois types de listes sont :
- les listes à tirets ;
- les listes numérotées ;
- les listes descriptives.

Les trois types de listes sont :

1. les listes à tirets ;
2. les listes numérotées ;
3. les listes descriptives.

Les trois environnement utiles sont :

itemize pour les listes à tirets ;

enumerate pour les listes numérotées ;

description pour les descriptives.

Exemple 2.6 – Les trois types de listes.

Il est essentiel de penser à utiliser ces environnements plutôt que de mettre en forme le texte « à la main », à grand renfort de sauts de ligne forcés et autres. C'est particulièrement vrai pour les listes numérotées : LaTeX s'occupe pour vous de maintenir les numéros dans l'ordre si vous ajoutez ou retirez des éléments de la liste.

2.4 Alignement du texte

Par défaut, LaTeX fait en sorte que les bords gauche et droits du texte soient bien rectilignes : on dit que le texte est *justifié*, ou pour être précis, justifié à droite et à gauche. On peut changer ce comportement pour certaines parties du texte, soit par choix (page de titre ou autre mise en pages personnalisée, soit pour des raisons techniques : quand on écrit du texte sur une faible largeur, essayer de l'aligner des deux côtés produit souvent un résultat assez moche, et il vaut alors mieux choisir un autre alignement.

On obtient du texte centré avec l'environnement `{center}`, aligné seulement sur la marge de gauche (on dit « au fer à gauche », ou encore « en drapeau ») avec `{flushleft}` en enfin aligné à droite avec `{flushright}`, comme le montre l'exemple 2.7. Par ailleurs, ces environnements laissent comme on le voit un petit espace avant et après, ce qui est en général nécessaire pour que le changement d'alignement ne soit pas choquant.

De même que les commandes de changement de fonte existent en deux formes (une commande à argument et une commande déclarative) on peut changer l'alignement avec un environnement comme on vient de le voir, mais aussi avec des commandes déclaratives. Tout ce qui a été dit précédemment sur les commandes déclaratives reste vrai, par exemple l'utilisation de groupes ou environnements pour limiter leur portée. La correspondance environnement-commande est donnée par la table 2.3 : on prendra bien garde à l'inversion `left`/`right` : aligné à gauche est en

```
Du texte d'exemple justifié à droite et à gauche.
\begin{center}
  Un peu de texte d'exemple centré sur la ligne.
\end{center}
\begin{flushleft}
  Cette fois du texte d'illustration en drapeau.
\end{flushleft}
\begin{flushright}
  Et enfin une partie de texte au fer à droite.
\end{flushright}
```

Du texte d'exemple justifié à droite et à gauche.

Un peu de texte d'exemple centré sur la ligne.

Cette fois du texte d'illustration en drapeau.

Et enfin une partie de texte au fer à droite.

Exemple 2.7 – Environnements d'alignement du texte.

Environnement	Commande
center	\centering
flushleft	\raggedright
flushright	\raggedleft

Table 2.3 – Environnements et commandes d'alignement.

effet équivalent à « déchiré » (*ragged*) à droite.

Si l'on choisit d'utiliser une commande, il faut prendre garde à plusieurs points : par exemple, terminer le paragraphe *avant* que l'effet de la commande ait cessé (comme dans le deuxième cas de l'exemple 2.3 précédent) car sinon elle n'aura tout simplement aucun effet. On peut aussi devoir ajouter un peu d'espace vertical avant et après. Pour ces raisons, il est recommandé de ne pas utiliser la forme commande et de préférer les environnements, sauf pour changer l'alignement à l'intérieur d'un autre environnement, ou bien dans la définition d'un environnement.

Enfin, signalons un point subtil : les environnements comme `center` se comportent différemment selon qu'ils sont au milieu d'un paragraphe ou pas (c'est-à-dire selon qu'il sont ou non entourés de lignes vides) : ceci affecte l'espace vertical précédent et suivant le texte centré, et l'alignement du texte qui suit (présence ou absence du retrait d'alinéa). Ceci est bien sûr valable pour les deux autres environnements ; des exemples seront vus en exercices.

Je présente pour finir dans cette section, faute d'un meilleur emplacement, deux environnements utiles : `{quote}` et `{quotation}` qui sont prévus pour faire des citations, et dont l'effet le plus visible est d'augmenter un peu la taille des marges de chaque côté, réduisant ainsi la largeur du texte. Les deux environnement sont quasiment identiques : le premier supprime le retrait d'alinéa au début de chaque paragraphe du texte cité, tandis que le deuxième le conserve.

2.5 Espaces

Comme on l'a vu à la section 1.5.2, LaTeX traite à sa manière les espaces et retours à la ligne dans le source. En particulier, pour obtenir un espace blanc important, que ce soit horizontalement (entre deux mots) ou verticalement (entre deux lignes), il est vain d'introduire plusieurs espaces ou retours à la ligne dans le source. Ceci est une bonne chose, car c'est une assez mauvaise habitude (même lorsqu'on utilise un traitement de texte classique) de procéder ainsi pour régler

l'espacement. Il est bien plus naturel de spécifier la longueur de l'espace à laisser blanc dans l'unité de son choix.

```
\hspace⟨*⟩{⟨longueur⟩}
\vspace⟨*⟩{⟨longueur⟩}
```

Pour cela, LaTeX propose les commandes `\hspace` et `\vspace` qui ont exactement la même syntaxe, et servent comme leur nom l'indique à insérer respectivement de l'espace respectivement horizontalement et verticalement. On utilise ainsi `\hspace{3cm}` pour laisser un espace horizontal de 3 centimètres, plutôt que d'essayer de le reproduire avec plusieurs espaces successifs. Les deux peuvent s'utiliser à n'importe quel endroit du texte, mais `\vspace` est surtout utile entre deux paragraphes : au sein d'un paragraphe, son effet est parfois étrange.

En LaTeX comme en physique, une longueur est un nombre suivi d'une unité. Ici, nombre signifie « nombre décimal », et le séparateur décimal (la « virgule ») est le point : la moitié de 5 s'écrit `2.5`. Les unités disponibles sont nombreuses, citons seulement pour le moment [13] les unités métriques `mm` et `cm`, l'unité anglo-saxonne `in` (pouce), les unités (typo-)graphiques `pt` et `bp` (deux définitions différentes [14] du point) et enfin deux unités relatives : `ex` et `em`. Ces dernières dépendent la fonte courante (et en particulier de sa taille) : un `ex` est approximativement la hauteur d'un « x » minuscule, et un `em` la largeur d'un « M » majuscule. Elles sont commodes pour obtenir des espaces proportionnelles à la taille d'un mot ou la hauteur d'une ligne.

Les distances négatives sont autorisées et servent à resserrer deux éléments, voire à créer des effets de surimpression si l'on resserre trop…

Normalement, les espaces introduits par `\hspace` et `\vspace` disparaissent s'ils se trouvent en début ou fin de page ou de ligne. C'est la plupart du temps une bonne chose ; quand on veut vraiment laisser un espace blanc à un de ces endroits, il faut utiliser `\hspace*` ou `\vspace*` : par exemple, `\vspace*{10cm}` juste après le début du document laisse un espace de 10 centimètres (en plus de la marge) en haut de la première page.

On découvre sur cet exemple un nouvel élément de la syntaxe de LaTeX : certaines commandes peuvent être suivies d'une étoile qui modifie leur comportement. Dans ce cours, `\commande⟨*⟩` désignera toujours une commande qui peut être suivie d'une étoile optionnelle. On dira que `\commande*` est la *version étoilée* de `\commande`.

On ne contrôle d'habitude pas les changements de ligne ni les changements de page : LaTeX s'occupe de calculer leurs emplacements optimaux. Si on souhaite forcer un retour à la ligne, on pourra la faire avec la commande `\newline` ou `\\`. Dans certains contextes (texte centré), seul `\\` fonctionnera. Ces deux commandes sont à utiliser avec une *grande* parcimonie : on a tendance à les utiliser à tort et à travers quand on débute, alors qu'elles sont en fait assez rarement utiles. La plupart du temps, soit le retour à la ligne est automatique (début de liste, de formule hors-texte, etc.) soit on veut en fait changer de paragraphe (ce qui se fait en laissant une ligne vide dans le source.) Un des cas où l'usage de `\\` se justifie toutefois est dans un court texte centré si l'on souhaite décider des coupures de ligne pour obtenir une forme de paragraphe équilibrée.

Il est également assez rare de devoir imposer les coupures de pages à la main. Pour les rares cas où cela sera utile, il existe les commandes `\newpage`, `\clearpage` et `\pagebreak`. Les deux

13. La liste complète des unités que connaît TeX, avec leur valeur, est donnée sec. 9.2 si vous y tenez.

14. Le premier vaut 1/72,27 pouces, le second 1/72 pouces. Le `pt` est l'unité utilisée par LaTeX pour mesurer par exemple les tailles de fontes ; le `bp` est la définition du point utilisée par la plupart des logiciels graphiques pour mesurer les images.

premières laissent un espace blanc à la fin de la page, alors que la troisième tente de préserver l'alignement en bas. La différence entre les deux premières est plus subtile et sera expliquée ultérieurement. Pour l'instant, utilisez `\clearpage`, ou encore n'utilisez aucune des trois, car elles ne sont pas souvent utiles.

`\stretch{`⟨*force*⟩`}`

Il existe en LaTeX des longueurs spéciales, qui peuvent s'étirer en fonction de l'espace disponible sur la page. La façon la plus simple d'y accéder est d'utiliser la commande `\stretch`, par exemple dans l'argument de `\vspace` : l'espace ainsi obtenu s'étirera alors autant que possible, poussant vers le bas de la page tout ce qui le suit. Si plusieurs espaces élastiques sont présents sur une même page, ils se partageront l'espace disponible de façon proportionnelle à leur ⟨*force*⟩. Ceci est pratique pour placer par exemple un texte au tiers de la page : on insère `\vspace*{\stretch{1}}` avant et `\vspace*{\stretch{2}}` après. Pour des exemples, voir les exercices et notamment l'exercice 2.

3 Structure du document

Même si, une fois imprimé sur du papier, un document a globalement une structure linéaire (dans le sens où on peut le lire dans l'ordre de la première à la dernières page), la structure logique du document est souvent bien plus complexe : on a en général une structure arborescente en chapitres, sections, sous-sections, etc., voire des morceaux de texte séparés comme des annexes ou des notes, auxquelles il est fait référence dans le corps du texte. Ce chapitre présente les outils proposés par LaTeX pour gérer ces structures : ils sont en général assez puissants dans le sens où on obtient beaucoup avec peu de commandes, et sans avoir à se soucier de détails comme maintenir à jour la numérotation.

3.1 Classes de documents

Une classe de document est quelque chose que l'on peut passer comme argument principal de la commande `\documentclass` présent au début de chaque source LaTeX. Nous connaissons pour l'instant la classe `minimal` (exemple 1.1) que l'on utilise en pratique *jamais*, et la classe `article` qui est la plus courante pour des documents pas trop longs. Les deux autres classes standard sont `report`, prévue pour des documents un peu plus longs (quelques dizaines de pages), et la classe `book`, qui convient pour réaliser des livres complets.

La classe de document détermine certains aspects de l'apparence du document, comme la présence ou non du numéro de page (absent avec `minimal`, présent avec toutes les autres) et sa place (centré en bas de page, sur un côté en en-tête, etc.), la présence ou non d'en-têtes de pages… Par ailleurs, elle détermine en partie les commandes de sectionnement disponibles.

Chaque classe de document accepte une ou plusieurs options, que l'on met toutes entre crochets, séparées par des virgules, entre le `\documentclass` et le nom de la classe entre accolades. Vous connaissez déjà les options `a4paper`, `10pt`, `11pt` et `12pt`.

Je n'en dis pas plus pour l'instant sur les particularités des classes et les options disponibles : elles seront présentées au fur et à mesure tout au long de ce chapitre.

3.2 Notes

3.2.1 Notes marginales

`\marginpar{`⟨*note*⟩`}`

La façon la plus simple de placer une note dans la marge est d'utiliser la commande `\marginpar`, qui prend en argument le texte à placer dans la marge. On peut l'utiliser dans le cours du texte ou entre deux paragraphes. La note sera placée *à peu près* à la même hauteur que la commande dans le texte : LaTeX a la possibilité de la déplacer un peu pour éviter des problèmes, par exemple si plusieurs notes se suivent, ou qu'on est trop près du bas de la page pour placer le texte en entier.

Ceci est un exemple de note marginale.

Ceci est un exemple de note marginale.

Signalons de suite un problème courant avec les notes marginales : l'alignement. En effet, par défaut le texte de la note est justifié à droite et à gauche (voir 2.4), ce qui pose souvent problème vu la faible largeur de la marge : certains espaces sont démesurément étirés comme dans l'exemple ci-contre, ce qui est assez moche, et fait d'ailleurs grincher LaTeX, qui émettra des messages comme

```
Underfull \hbox (badness 10000) in paragraph at lines 35--35
[]\EU1/MinionPro(0)/m/n/10.95 Ceci est un
```

où le numéro à la fin [1] de la première ligne du message indique la fin du `\marginpar` incriminé dans le source, et le texte à la fin [2] de la deuxième ligne (ici « `Ceci est un` ») est le texte présent sur la ligne mal remplie.

Ce problème admet une solution simple : il suffit de demander à LaTeX d'aligner à gauche le texte de la note, comme il a été fait dans le premier exemple de cette section. Pour cela, le plus simple est d'insérer le commande `\raggedright` au début du texte de la note ; il s'agit d'un cas où la commande est plus pratique que l'environnement. L'effet de la commande est automatiquement limité à la note et ne se propagera pas au reste du texte.

Par défaut, les notes marginales se trouvent dans la marge de droite. On peut inverser le côté des notes avec la commande `\reversemarginpar` ; cette commande agit comme un commutateur : elle inverse le coté de toutes les notes qui suivent, jusqu'au prochain `\reversemarginpar` ou la fin du document s'il n'y en a plus d'autre.

```
twoside
\marginpar[⟨texte si à gauche⟩]{⟨texte si à droite⟩}
```

J'ai légèrement menti ci-dessus en disant que les notes sont dans la marge de droite : ce n'est vrai que pour les documents que LaTeX considère comme étant en recto simple. C'est le cas par exemple de tous les documents en classe `report` ou `book`. En classe `article`, on peut forcer une mise en page recto/verso avec l'option de classe `twoside`. (À l'inverse, dans les classes qui sont par défaut en recto/verso, on peut imposer le recto simple avec l'option `oneside`.)

Attention, cette notion de recto/verso est indépendante de la façon dont le document sera réellement imprimé : vous pouvez parfaitement imprimer en recto/verso en classe `article` sans pour autant avoir l'obligation d'utiliser l'option `twoside`. En fait, ces options ont pour but de dire à LaTeX si la mise en page des pages paires et impaires doit être identique ou non : par exemple dans ce polycopié [3] les pages impaires portent en haut le nom de la section en cours et en bas le numéro de page à droite, tandis que les pages paires portent le nom du chapitre et le numéro de page à gauche. Dans un document de classe `article`, le numéro de page est centré sur toutes les pages.

Revenons donc à nos ~~moutons~~ notes marginales. La commande `\marginpar` accepte un argument optionnel qui, dans le cas où le document est en recto-verso et où la note tombe sur une page paire (donc dans la marge de gauche), sera utilisé à la place de l'argument principal. Ceci est principalement utilise pour régler les problèmes d'alignement : on pourra par exemple écrire

```
\marginpar[\raggedleft Texte de note.]{\raggedright Texte de note.}
```

pour s'assurer que le texte de la note soit toujours correctement aligné.

1. Le texte du début dit que le remplissage de la ligne est mauvais et même extrêmement mauvais : 10000 est la plus grande valeur de « mauvaiseté » pour LaTeX.
2. Le texte du début indique de façon un peu cryptique la fonte en cours.
3. Réalisé avec la classe `scrbook` qui est une variante plus moderne de `book`.

☞ Il faut bien être conscient que \marginpar n'est *pas* du tout adapté pour placer du matériel dans la marge comme la main en face du début de ce paragraphe, pour au moins deux raisons : on ne contrôle pas précisément l'emplacement vertical de la note, ni le côté où elle apparaît dans un document recto/verso. Il est important de réaliser que dans la plupart des cas ce manque de contrôle est une *bonne* chose : il signifie en fait que LaTeX s'occupe d'un nombre important de questions techniques dont il nous décharge. Une technique adaptée pour placer du matériel dans la marge de gauche à un emplacement fixé sera vue plus tard (exemple 9.1).

Enfin, signalons que dans les documents en recto/verso, un bug de LaTeX fait que les notes apparaissent parfois du mauvais côté près des changements de pages : il suffit pour corriger ce bug de charger le module mparhack.

3.2.2 Notes de bas de page

\footnote{⟨*texte de la note*⟩}

Les notes de bas de page sont en fait bien plus faciles à utiliser que les notes marginales : il suffit de placer la commande \footnote à l'endroit précis où on veut que l'appel de note [4] ait lieu, en lui passant en argument le texte de la note de bas de page. Ceci [5] est illustré par l'exemple 3.1.

```
Un paragraphe avec une
note\footnote{Oui, vraiment.}
de bas de page.
```

Un paragraphe avec une note [a] de bas de page.

a. Oui, vraiment.

Exemple 3.1 – Note de bas de page.

Il n'y a essentiellement aucun piège avec \footnote, sauf que c'est une commande qui ne peut pas être utilisée dans toutes les circonstances : elle ne peut par exemple pas être utilisée dans les titres de section. Nous verrons tout à l'heure (en 3.4) comment gérer cette limitation.

3.3 Titre et résumé

3.3.1 Titre

\title{⟨*titre*⟩} \author{⟨*auteur*⟩} \date{⟨*date*⟩}
\maketitle

On peut demander à LaTeX de gérer automatiquement la mise en forme du titre. Pour cela, il suffit de lui donner les informations nécessaires, à savoir le titre du document, le nom de l'auteur, et si l'on veut [6] la date. Ces trois commandes (\title, \author et \date) n'écrivent *rien* dans le document (on peut même les placer dans le préambule si on veut) : il faut ensuite utiliser \maketitle (dans le corps du document, et généralement au tout début) pour faire réellement apparaître ces informations, mises en forme, dans le document.

4. C'est-à-dire le petit numéro, par exemple le 4 après le mot « note » ici.
5. Dans l'exemple, la note est numérotée « a » pour ne par interférer avec la numérotation des notes du texte principal. Ceci se fait automatiquemenr avec la technique utilisée pour les exemples (environnement minipage, qui sera vu en 9.3.3).
6. Si l'on ne précise pas de date, c'est celle du jour de compilation qui sera utilisée.

Par défaut en classe `article`, le titre apparaît centré, et le texte suit sur la même page (voir par exemple la mise en page des corrigés des exercices). On peut décider que le titre occupe une page à lui tout seul (comme pour ce polycopié par exemple), avec l'option de classe `titlepage`. C'est ce qui se passe automatiquement avec les classes `report` et `book` ; pour ces classes, on peut forcer le titre à être sur la même page que la suite du texte avec l'option `notitlepage`.

Les paragraphes ci-dessus décrivent la façon la plus simple de faire un titre. Cependant, pour des documents plus sophistiqués, la mise en page automatique de LaTeX peut sembler un peu trop sobre. Dans ces cas-là, on peut changer totalement de stratégie et faire la page de titre soi-même à la main. On n'utilisera alors pas du tout les commandes comme `\title` et `\maketitle`, et on devra spécifier soi-même le texte, son ordre d'apparition sur la page et sa mise en forme à l'aide des outils standard vus au chapitre précédent. On doit par contre réserver une page non numérotée pour faire la page de titre : ceci se fait au moyen de l'environnement `{titlepage}` ; voir l'exercice 2 pour un exemple. Dans ce cas, on reprend totalement le contrôle, et les options de classe comme `titlepage` ou `notitlepage` n'ont donc plus aucun effet.

Il est d'ailleurs intéressant de noter cette dualité des méthodes : la première consistant à laisser LaTeX s'occuper de tout, la deuxième à tout faire soi-même. L'avantage de la première méthode est qu'on gagne beaucoup de temps lors de la préparation du document, en laissant de côté beaucoup de détails, et en obtenant quand même au final un résultat visuellement correct. Dans un premier temps, on s'intéressera principalement à ce type de méthode (laisser faire LaTeX), car c'est un de ses points forts de pouvoir procéder ainsi pour rédiger plus vite. Il est néanmoins important de savoir qu'à tout moment on eut regagner le contrôle des détails, si on est prêt à y passer le temps nécessaire.

Une partie des techniques nécessaires pour contrôler totalement les divers éléments de mise en page et réaliser des constructions complexes seront présentées aux chapitres 9 et 10. En attendant, laissez-vous guider par les mises en pages automatiques de LaTeX.

3.3.2 Résumé

On obtient une présentation automatique du résumé (marges réduites, texte plus petit, titre « résumé » en gras au-dessus) en mettant le contenu du résumé dans un environnement [7] `{abstract}`. Je renvoie aux exercices pour un exemple.

3.4 Structure globale

Les outils pour diviser son document en sections, sous-sections, etc. dépendent de la classe utilisée : il est clair que plus le document est important, plus ces outils seront nombreux. Les différentes commandes de sectionnement proposées par la plus petite des trois classes standard, `article`, sont présentées sur le source 3.1.

Quelques remarques sur les commandes de sectionnement présentées ici. Le nom est en général assez explicite pour qu'il n'y ait pas besoin de préciser le sens. Elles partagent toutes (à l'exception de `\appendix`) la même syntaxe ; celle présentée dans le source 3.1 est la version la plus simple. Voici la syntaxe complète, présentée uniquement sur la commande `\section` pour simplifier, mais valable pour toutes les autres commandes de sectionnement.

7. Attention, l'ouvrage « LaTeX pour l'impatient » présente `abstract` comme une commande : c'est une erreur.

```
\part{⟨titre de partie⟩}
\section{⟨titre de section⟩}
\subsection{⟨titre de sous-section⟩}
\subsubsection{⟨titre de sous-sous-section⟩}
\paragraph{⟨titre de paragraphe⟩}
\subparagraph{⟨titre de sous-paragraphe⟩}
\appendix
\section{⟨titre d'appendice⟩}
\section{⟨titre d'appendice⟩}
```

Source 3.1 – Structure d'un document en classe `article`

`\section⟨*⟩[⟨titre table des matières⟩]{⟨titre document⟩}`

Premièrement, l'étoile optionnelle permet d'obtenir une section non numérotée. Par défaut, toutes les commandes de sectionnement produisent une numérotation, sauf celles de niveau trop faible (par exemple `\paragraph`). On peut néanmoins souhaiter supprimer la numérotation d'une section, par exemple pour une introduction : c'est ce qui est souvent fait en exercices.

Il est important de distinguer une section non numérotée d'un texte « juste écrit en gros » : d'une part, une section gère aussi beaucoup de détails comme l'espacement avant et après, qu'il est fastidieux d'essayer de reproduire à la main. D'autre part, si plus tard vous changez la mise en forme des titres, il sera plus facile de le faire uniformément en ayant utilisé `\section*` à bon escient qu'en ayant essayé de reproduire la mise en forme à la main. À l'inverse, il ne faut pas utiliser systématiquement `\section*` pour écrire en gros, mais seulement quand il s'agit d'un titre. Je renvoie aux corrigés des exercices pour des exemples d'usage de l'une ou l'autre solution.

On a ensuite un argument optionnel permettant de spécifier un titre alternatif qui n'apparaîtra pas dans le corps du document, mais uniquement dans la table des matières. Ceci peut être utile pour les sections ayant un titre très long, qui apparaîtrait sur plusieurs lignes dans la table des matières : on peut donner une version courte du titre qui perturbera moins la mise en pages de la table des matières.

Comme on l'a signalé un peu plus haut, il n'est pas possible d'utiliser `\footnote` dans le titre d'une section : ceci est essentiellement dû aux problèmes qui surviennent en « déplaçant » la note jusqu'à la table des matières. La façon la plus raisonnable [8] de contourner ce problème est d'écrire par exemple

```
\section[Section importante]{Section importante\footnote{Vraiment.}}
```

de façon à ce que la note de bas de page apparaisse uniquement dans le document, mais pas dans la table des matières : on évite ainsi le problème.

La commande spéciale `\appendix` agit comme un commutateur, à usage unique : il y a un avant et un après. Avant, tout se passe normalement. Après, la commande `\part` devient illégale, et la numérotation des sections change : elle passe en lettres, pour signifier que les sections sont en fait des appendices.

8. Une autre façon est de placer `\protect` juste devant `\footnote` : de cette façon, la note apparaîtra correctement dans le titre *et* dans la table des matières.

Évoquons maintenant rapidement les caractéristiques des autres classes. En classe `report`, une nouvelle commande `\chapter` apparaît : elle s'insère entre `\part` et `\section`. Elle provoque un saut de page et le mot [9] « chapitre » est écrit à côté du numéro. Après `\appendix`, c'est `\chapter` qu'il faut continuer à utiliser, et le nom devient alors « annexe ».

La classe `book` propose d'autres commutateurs en plus de `\appendix` : il s'agit de `\frontmatter` à utiliser au début du document pour, par exemple la table des matières, la préface, etc. On utilise ensuite `\mainmatter` pour passer aux choses sérieuses (par exemple, dans le présent polycopié, on a utilisé `\mainmatter` juste avant de commencer le chapitre 1), puis éventuellement `\backmatter` une postface (non utilisé dans ce poly). Observez l'effet sur la numérotation des pages.

3.5 Références et liens hypertexte

3.5.1 Références croisées

Les différents niveaux de sectionnement sont numérotés automatiquement par LaTeX. Une des fonctions principales de ces numéros est de pouvoir faire référence à une partie du document (par exemple en disant que vous avez compilé votre premier document à la section 1.4). Vu que les numéros changent automatiquement à chaque fois que vous insérez ou supprimez des sections, ou changez leur ordre, il faut un moyen de mettre à jour les références automatiquement.

```
\label{⟨étiquette⟩}
\ref{⟨étiquette⟩}
\pageref{⟨étiquette⟩}
```

Ce moyen est fournit par les commandes `\label` et `\ref`, qui s'utilisent de la façon suivante : on place la commande `\label` à l'endroit auquel on voudra faire référence plus tard, par exemple juste *après* [10] la commande `\section` et son argument. L'argument de `\label` est une chaîne de caractères libre, mais il est prudent de n'y utiliser que des lettres non accentuées et éventuellement des traits d'union [11]. Plus tard, quand on voudra faire référence au numéro de cette section, on utilisera `\ref` avec la même étiquette comme argument. Si on veut faire référence à la page (par exemple, la section 1.4 page 5), on utilisera `\pageref` toujours avec la même étiquette.

L'étiquette est un nom privé qui ne se verra pas dans le document. Par exemple, l'étiquette qui me permet de faire référence au source 3.1 est `src-struct-article`. Observez que le nom fait référence au contenu du source et non pas à sa position, qui pourrait changer. Par ailleurs, le préfixe `src-` indique sa nature : vous n'êtes pas obligés d'utiliser de tels préfixes, mais c'est souvent une habitude utile.

Ainsi, la commande `\label` permet de marquer certains endroits du document, et les commandes `\ref` et `\pageref` de se référer au numéro de cet endroit, ou à sa page. Pour l'instant, les seuls endroit numérotés que l'ont connaît sont les sections, sous-sections, etc. ainsi que les notes de bas de page (dans ce cas, il faut placer le label dans le texte de la note), mais l'on verra plus tard d'autres objets numérotés, auxquels on fera toujours référence par le même mécanisme.

9. Je suppose bien sûr que vous utilisez l'option `french` de `babel`, sinon ce sera *chapter* ou autre chose.
10. Si on place le `\label` avant, il fera référence à la section précédente.
11. On rencontre souvent des « : » dans les noms d'étiquettes, mais cela peut (exceptionnellement) causer des problèmes en français, alors que le trait d'union est à ma connaissance toujours inoffensif.

On peut aussi faire référence à des objets situées plus loin dans le document : c'est de là que vient l'expression références *croisées*. Pour permettre ceci, le mécanisme de référence est asynchrone et nécessite deux compilations successives pour fonctionner correctement. Lors de la première compilation, vous obtiendrez des avertissements comme

```
LaTeX Warning: Reference 's-first-doc' on page 28 undefined on input line 330.
LaTeX Warning: There were undefined references.
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
```

qui disparaissent automatiquement aux compilations suivantes, et ré-apparaissent à chaque fois que vous changez quelque chose aux références croisées. Si l'avertissement de la deuxième ligne ci-dessus persiste après deux compilations, il est probable que vous ayez effectivement oublié de définir un `\label`, ou fait une faute de frappe dans l'étiquette. Dans le corps du document, une référence à une étiquette non définie fait apparaître un double point d'interrogation qui signale l'erreur.

`\tableofcontents`

À tout endroit du document, la commande `\tableofcontents` fait apparaître la table des matières. Comme les commandes de références, elle nécessite deux compilations pour fonctionner correctement. La table des matières est surmontée du titre « table des matières » qui est écrit comme si on avait utilisé la commande `\section*` : il n'y a rien à faire pour cela. (Bien sûr le titre n'est en français que si on utilise `babel` avec l'option `french`, je ne le répéterai plus.)

3.5.2 Bibliographie

Parallèlement au système de références croisées, internes au document, LaTeX propose un système de citations pour faire référence à des documents externes.

`{thebibliography}{`⟨*exemple*⟩`}`
`\bibitem{`⟨*étiquette*⟩`}`
`\cite[`⟨*endroit*⟩`]{`⟨*étiquette*⟩`}`

Pour composer la liste des références bibliographiques, on utilise `{thebibliography}` : c'est un environnement très similaire à `{itemize}`, sauf qu'on remplace `\item` par `\bibitem` pour introduire chaque entrée. Il prend un argument obligatoire, qui est une texte libre dont le largeur doit être environ celle de la plus large clé de citation. Par exemple, quand des numéros sont utilisés comme dans l'exemple 3.2, si on a entre 10 et 99 références, on pourra indiquer `00` comme ⟨*exemple*⟩, pour donner le largeur de deux chiffres.

La commande `\bibitem` prend en argument obligatoire une étiquette privée, n'apparaissant pas dans le document, mais seulement utilisée comme argument obligatoire de `\cite` pour citer le document en question. Le couple `\bibitem` et `\cite` est donc analogue au couple `\label` et `\ref` de ce point de vue, et le rôle de l'étiquette identique. La commande `\cite` accepte de plus un argument optionnel permettant de préciser un emplacement précis dans la référence citée. Tout ceci est illustré par l'exemple 3.2, dont les deux première lignes peuvent se situer n'importe où dans le document, et les autre sont par exemple à la fin.

```
Pour en savoir plus, lire
\cite{ttb} ou \cite[chap.~12]{lcfr}.
% plus loin dans le document
\begin{thebibliography}{LC}
  \bibitem{ttb} \emph{Tame the BeaST},
    \bsc{N. Markey}, CTAN.
  \bibitem{lcfr} \emph{\LaTeX{}
      Companion}, \bsc{Mittlebach}
    \& \bsc{Goossens}, Pearson.
\end{thebibliography}
```

Pour en savoir plus, lire [1] ou [2, chap. 12].

Bibliographie

[1] *Tame the BeaST*, N. Markey, CTAN.

[2] *LaTeX Companion*, Mittlebach & Goossens, Pearson.

Exemple 3.2 – Bibliographie et citations.

3.5.3 Liens hypertexte

Comme vous l'aurez constaté, dans la version PDF de ce polycopié, toutes les références, ainsi que les titres dans la table des matières, sont des liens vers la destination. De plus, vous disposez de signets PDF pour naviguer dans les sections. Ceci est obtenu automatiquement grâce au module hyperref.

```
\usepackage[⟨option1, option2, ...⟩]{hyperref}
```

Il suffit de charger le module sans aucune option pour que tout ce qui peut raisonnablement être un lien en devienne un. Vous pouvez cependant personnaliser certains aspects à l'aide des nombreuses options. En voici quelques-unes ; pour les autres, je renvoie à la documentation du module.

```
colorlinks=true
linkscolor=⟨nom de couleur⟩, urlcolor=⟨nom de couleur⟩
pdfauthor=⟨votre nom⟩, pdftitle=⟨titre⟩
```

L'option colorlinks permet aux liens d'être en couleur, au lieu d'être entourés d'un cadre coloré ; la couleur peut être choisie avec linkscolor, ou urlcolor selon le type de lien. Enfin, vous pouvez spécifier certains méta-informations qui n'apparaîtront pas dans le PDF lui-même mais seront visibles via le menu « propriétés » du lecteur de PDF, ou de la commande pdfinfo.

En fait, hyperref permet aussi de créer des liens vers adresses externes. On peut au choix montrer le texte de l'adresse avec la commande \url, ou transformer en lien [12] un texte arbitraire, comme le montre l'exemple 3.3.

```
Le CTAN : \url{http://ctan.org/}.
Ou \href{http://ctan.org/}{le CTAN}. Pour
\href{mailto:mpg@math.jussieu.fr}{m'écrire}.
```

Le CTAN : http://ctan.org/. Ou le CTAN. Pour m'écrire.

Exemple 3.3 – Liens avec hyperref.

12. La commande utilisée pour cela a un nom familier pour ceux d'entre vous qui connaissent un peu d'HTML.

3.6 Structure de la page

La classe de document prévoit un style de page, avec des dimensions prédéfinies en fonction de certaines options : par exemple, en recto/verso, les marges de gauche et de droite ne font pas la même taille. Vous pouvez régler à votre guise ces dimensions avec le module geometry.

```
\usepackage[⟨options⟩]{geometry}
\geometry{⟨options⟩}
```

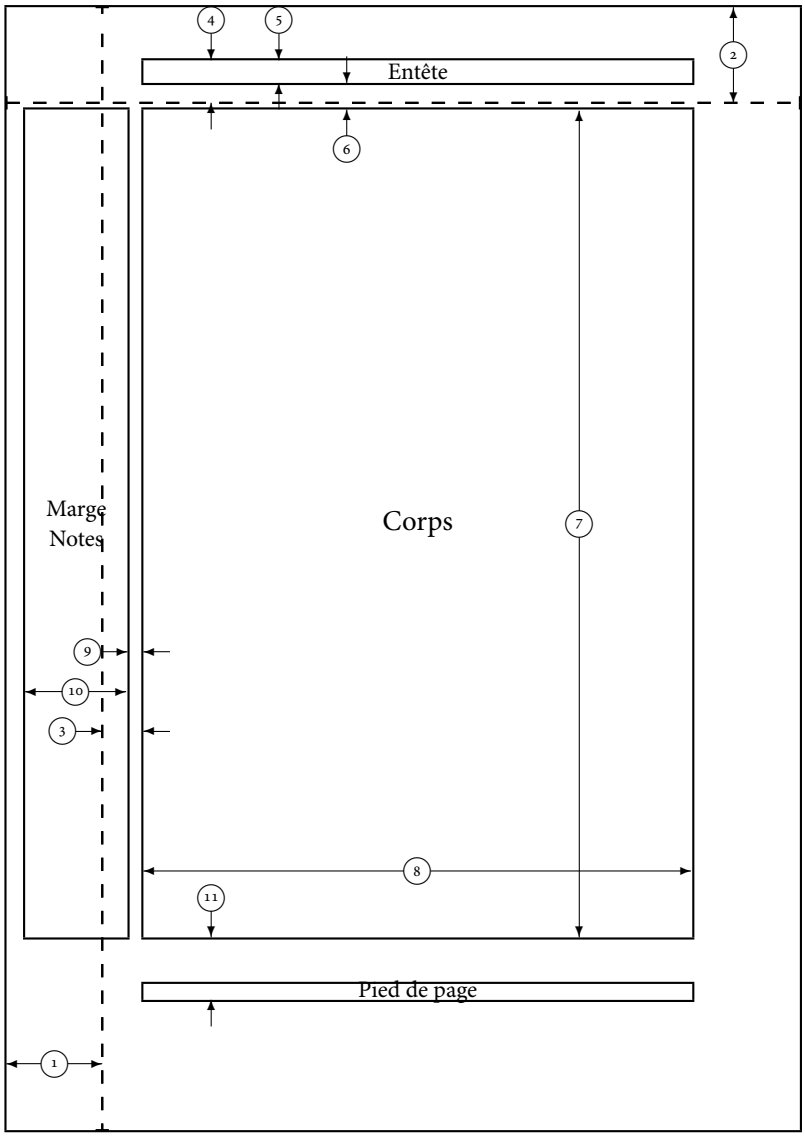
Les options peuvent être au choix indiqués au chargement du module, ou plus tard dans le préambule avec la commande \geometry : les deux sont rigoureusement équivalents. Les options disponibles sont nombreuses et je renvoie à la documentation pour une liste complète. Les plus importantes à connaître sont vmargin et hmargin qui règlent les marges du haut et du bas (resp. de gauche et de droite) simultanément, rmargin, lmargin, tmargin et bmargin pour les spécifier séparément, et landscape pour changer l'orientation du papier. Pour toutes les options acceptant une longueur, la définition de longueur est celle vue sec. 2.5. Par exemple, on peut dire

```
\usepackage[hmargin=3cm, landscape]{geometry}
```

Si on charge geometry sans options, il modifiera quand même les marges par rapport à celles prévues par la classe (il fait en général des marges un peu plus petites).

Un moyen commode de se familiariser avec les différentes dimensions d'une page est d'utiliser la commande \layout du module layout : elle dessine la maquette de la page en cours, avec une légende portant les noms et les valeurs des différentes dimensions. Pour avoir cette légende en français, il suffit de passer l'option french au chargement du module layout. Par exemple, les dimensions utilisées par le document présent sont visibles sur la page suivante.

On y voit notamment que des espaces particuliers sont réservés pour un éventuel en-tête ou pied de page. Pour l'instant, on a en général juste un pied de page consistant en le numéro de page, centré ; en classe book on aurait aussi un en-tête de page. On verra sec. 10.1 comment personnaliser le contenu de ces zones.



1	un pouce + \hoffset	2	un pouce + \voffset
3	\evensidemargin = 31pt	4	\topmargin = -32pt
5	\headheight = 17pt	6	\headsep = 20pt
7	\textheight = 623pt	8	\textwidth = 413pt
9	\marginparsep = 12pt	10	\marginparwidth = 77pt
11	\footskip = 47pt		\marginparpush = 6pt (non affiché)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 597pt		\paperheight = 845pt

4 Les modes mathématiques

4.1 Découvertes des modes mathématiques

4.1.1 Les deux modes mathématiques

Pour écrire des maths en LaTeX, il faut entrer dans un mode spécial appelé mode mathématique. Celui-ci existe en deux variantes : en ligne et hors texte, selon que les éléments mathématiques doivent être intégrés au paragraphe en cours ou bien apparaître sur une ligne à part. Sur l'exemple 4.1, le premier $\sqrt{2}$ est en mode en ligne, et sa définition en mode hors-texte.

```
Le nombre \(\sqrt{2}\) est définit par
\[ (\sqrt{2})^2 = 2 \]
et \(\sqrt{2} > 0\). C'est un nombre
réel irrationnel.
```

Le nombre $\sqrt{2}$ est définit par

$$(\sqrt{2})^2 = 2$$

et $\sqrt{2} > 0$. C'est un nombre réel irrationnel.

Exemple 4.1 – Les deux modes mathématiques

Ici, on est allé à la ligne dans le source pour la formule hors-texte, mais c'est juste pour rendre le source plus clair : ça ne change rien au résultat ; le retour à la ligne et le centrage de l'équation sont automatiques en passant en mode hors-texte. Les modes mathématiques sont délimités de la façon suivante :

en ligne avec `\(...\)` ou `$...$`, ou l'environnement `{math}` ;

hors-texte avec `\[...\]`, ou l'environnement `{displaymath}`.

Avec le module `amsmath`, on passe aussi en mode hors-texte avec l'environnement `{equation*}`. Une mauvaise façon de passer en mode hors-texte est d'utiliser `$$...$$` ; on la trouve mentionnée dans de nombreuses références. Je recommande fortement [1] de ne pas l'utiliser, car elle présente différents défauts (espaces verticaux incohérents, non-respect de l'option de classe `fleqn`, incompatibilités avec `amsmath`, …) et que `\[...\]` n'est pas beaucoup plus long à saisir.

En mode mathématique, plusieurs choses diffèrent du mode texte :
- une police différente est utilisée, les lettres sont par défaut en italique ;
- les espaces sont totalement ignorés : `$a b$` est équivalent à `$ab$` et donnera ab ;
- les deux caractères spéciaux `^` et `_` deviennent autorisé ;
- un certain nombre de commandes deviennent disponibles.

J'insiste sur le fait que les espaces sont tous ignorés : c'est en général une bonne chose car l'espace en mode mathématique est très délicat à régler, il faut souvent utiliser des demi-espaces, des tiers d'espace, etc. Il vaut donc mieux que ce soit LaTeX qui s'en occupe plutôt que de nous obliger à saisir des commandes compliquées pour obtenir l'espacement correct. Mais c'est potentiellement déroutant au début.

1. Lire : ne pas respecter cette recommandation comptera comme une faute.

Enfin, même si LaTeX seul propose de nombreuses commandes pour les math, certains modules étendent là aussi des possibilités. Deux d'entre eux sont incontournables : il s'agit de amsmath et amssymb. Il est recommandé de les charger dans le préambule de tout document utilisant des formules mathématiques. Dans toute la suite de ce chapitre, tous les exemples avec des formules supposeront que ces deux modules sont chargés, sans le préciser à chaque fois. Si le paquet mathtools est disponible, il peut avantageusement remplacer amsmath : il fournit les même fonctionnalités, ainsi que quelques autres commandes pour simplifier certaines constructions ou régler des points de détail.

4.1.2 Outils de base

^{⟨*exposant*⟩} ^⟨*symbole unique*⟩
_{⟨*exposant*⟩} _⟨*symbole unique*⟩

Les deux caractères réservés ^ et _ deviennent utilisable en mode math, et il servent à mettre respectivement en exposant ou en indice le symbole qui les suit. Attention, si on veut mettre plusieurs symboles en indice ou en exposant, il faut les entourer par des accolades. On peut mettre un indice et un exposant dans l'ordre qu'on veut sur la même symbole, mais pas deux indices ou deux exposants. On peut par ailleurs utiliser toutes les constructions voulues (y compris des indices et exposants) en particulier dans un indice ou un exposant, comme le montre l'exemple 4.2.

```
$2^{2^2} = 2^4$ mais $u_n+1 \neq u_{n+1}$
et $\sqrt[3]{2} = 2^{\frac{1}{3}}$.
```

$2^{2^2} = 2^4$ mais $u_n + 1 \neq u_{n+1}$ et $\sqrt[3]{2} = 2^{\frac{1}{3}}$.

Exemple 4.2 – Indices, exposants, fractions et racines.

\frac{⟨*numérateur*⟩}{⟨*dénominateur*⟩}
\sqrt[⟨*indice*⟩]{⟨*radicande*⟩}

Comme on le voit aussi sur l'exemple 4.2, on utilise \frac pour obtenir une fraction, et \sqrt pour une racine. Par défaut c'est une racine carrée, mais l'argument optionnel permet de préciser. L'argument optionnel n'est disponible qu'avec le module amsmath : je rappelle que tous les exemples mathématiques supposent que ce module est chargée, ainsi que amssymb, je ne le répéterai plus.

Comme de nombreux autres éléments, les fractions ont une apparence différente selon qu'elles sont en mode hors-texte simple, on mode en-ligne, ou imbriquées dans d'autres constructions : LaTeX règle automatiquement le style (taille et espacement) sans que vous ayez à vous soucier de rien. Nous verrons en 4.2.2 comment reprendre le contrôle des styles pour les rares cas où c'est souhaitable.

\sum_{⟨*borne basse*⟩}^{⟨*borne haute*⟩} \prod_{⟨*borne basse*⟩}^{⟨*borne haute*⟩}
\int_{⟨*borne basse*⟩}^{⟨*borne haute*⟩} \iint \oint ...

On obtient respectivement des sommes, produits et intégrales avec \sum, \prod et \int. La syntaxe pour les bornes est commune à ces trois commandes : c'est la même que s'il s'agissait

d'indices et exposants normaux. Cependant, ils obéissent à des règles de placement spéciales, qui dépendent du mode : notamment les bornes seront bien placées dessus et dessous en mode hors-texte. Pour les intégrales, de nombreuses variantes sont disponibles [2] et il convient de les utiliser : `\iint` pour une intégrale double fournit par exemple un plus joli résultat que `\int\int`. La liste de ces commandes fait partie de la liste de symboles mathématiques [3] distribuée.

```
Posons $S = \sum_{i=1}^n u_n$. Ou bien
\[ S = \sum_{i=1}^n u_n \] On a aussi
\( \iint f \neq \int\int f \).
```

Posons $S = \sum_{i=1}^{n} u_n$. Ou bien

$$S = \sum_{i=1}^{n} u_n$$

On a aussi $\iint f \neq \int \int f$.

Exemple 4.3 – Sommes et intégrales

De nombreux symboles spéciaux sont disponibles en mode mathématique. Citons par exemple toutes les lettres grecques, qui s'obtiennent par leur nom : `\alpha`, `\beta`, …, mais aussi en majuscule quand elles existent : `\Gamma`, `\Delta`, ainsi que quelque lettres issues de l'alphabet hébraïque : `\aleph`, `\beth`. De nombreuses flèches existent, avec des noms souvent assez descriptifs : `\leftarrow` pour une flèche vers la gauche, `\to` et `\mapsto` pour les fonctions. Enfin, une grande quantité de symboles ensemblistes (`\in`, `\subset`, etc.) de relation (`\leq` pour *Less or EQual*, `\neq` pour *Not EQual*, etc.), `\cdot` pour un point centré désignant la multiplication, `\cdots` pour une suite de points centrés.

Il serait vain de vouloir lister tous les symboles disponibles : je renvoie donc à la liste de symboles courants [4] distribuée, qui est extraite de `flshort-3.20.pdf` [5]. Encore plus de symboles mathématiques sont listés dans le document `symbols-a4.pdf` [6] déjà évoqué à propos des symboles en mode texte.

Voyons, pour finir ce premier tour d'horizon, les différentes fontes disponibles en mode mathématique. Elles sont listées et illustrées par la table 4.1. Attention, pour que la commande `\mathscr` soit disponible, il faut charger le module `mathrsfs`. On peut par ailleurs obtenir une variante de `\mathcal` [7] en chargeant le module `eucal`. Enfin, les deux fontes calligraphiques (`\mathcal` et `\mathscr`) ne sont disponibles que pour les majuscules.

Par ailleurs, des commandes de changement de fontes similaires à celle du mode texte sont disponibles, mais seules `\mathrm` et `\mathbf` sont réellement utiles en pratique. On prendra garde en tout cas à ne pas utiliser les commandes ce changement de fontes du mode texte en mode mathématique, notamment les commandes de changement de taille. On verra plus loin comment contrôler la taille *via* le style (sec. 4.2.2) et comment passer en mode texte au milieu du mode mathématique (sec. 4.2.1).

2. Enfin, avec `amssymb`, mais j'avais déjà dit que je ne le répéterai plus.
3. `http://people.math.jussieu.fr/~mpg/lm204/files/doc-symboles-math.pdf`
4. `http://people.math.jussieu.fr/~mpg/lm204/files/doc-symboles-math.pdf`
5. `http://mirror.ctan.org/info/lshort/french//flshort-3.20.pdf`
6. `http://mirror.ctan.org/info/symbols/comprehensive/symbols-a4.pdf`
7. Ou bien obtenir une variante de `\mathscr` en chargeant `eucal` avec l'option `mathscr` au lieu de charger `mathrsfs`. Il n'y a pas de moyen simple d'avoir ces trois polices calligraphiques simultanément disponibles.

Police	Exemple	Code
par défaut	abc	`$abc$`
romaine	$\mathrm{d}x$	`$\mathrm{d}x$`
grasse droite	$\mathbf{C} \supset \mathbf{R}$	`$\mathbf{C} \supset \mathbf{R}$`
grasse	$\boldsymbol{k}$	`$\boldsymbol{k}$`
fraktur	$\mathfrak{P} \mid \mathfrak{p}$	`$\mathfrak{P} \mid \mathfrak{p}$`
calligraphique	$\mathcal{A}$	`$\mathcal{A}$`
anglaise	$\mathscr{C}$	`$\mathscr{C}$`
ajourée	$\mathbb{N} \subset \mathbb{Z}$	`$\mathbb{N} \subset \mathbb{Z}$`

TABLE 4.1 – Fontes disponibles en mode mathématiques

4.2 Points plus délicats

4.2.1 Distinguer texte et mathématiques

Pour obtenir des documents corrects, il faut prendre garde de bien distinguer le mode texte du mode mathématique : il est dangereux de passer en mode math lorsque ce n'est pas justifié (par exemple, utiliser `^` pour mettre du texte en exposant alors que `\up` est fait pour ça en mode texte), et réciproquement il est incorrect de rester en mode texte pour des éléments mathématiques. Par exemple, pour écrire « soit f une fonction », on passera en mode mathématique [8] pour le f, qui représente un objet mathématique.

En mode mathématique, les lettres sont par défaut considérées comme des variables et composées en italique. Il arrive de devoir passer en police droite, pour différentes raisons.

- Pour les lettres représentant des constantes, comme dans e^x (`$\mathrm{e}^x$`) où le « e » est une constante, la base des logarithmes népériens, alors que x est une variable et reste donc en italique. On utilisera dans ce cas `\mathrm`.
- Pour les noms d'opérateurs, comme lim, sin ou cos. De nombreux noms sont prédéfinis (`\lim`, `\sin`, `\cos`), mais on peut en définir soi-même au besoin (voir ci-dessous comment faire). Il faut toujours utiliser une commande prédéfinie, ou définie comme ci-dessous, pour les opérateurs.
- Pour écrire réellement du texte en mode maths. Ceci peut être l'expression « tel que » au milieu d'une formule, ou encore une abréviation comme dans P_{effectif}, que l'on saisit comme `$P_{\text{effectif}}$`. On utilisera `\text` à l'exclusion de tout autre moyen [9] pour ce cas.

Il est important d'apprendre à bien distinguer ces trois cas. En particulier dans le dernier cas, on passe vraiment en mode texte : dans l'argument de `\text`, les espaces sont de nouveaux respectés et les commandes spécifiques au mode mathématique interdites.

`\DeclareMathOperator{`⟨*commande*⟩`}{`⟨*nom*⟩`}`

Pour déclarer un nouvel opérateur, on utilisera `\DeclareMathOperator`, qui est disponible uniquement dans le préambule. Le premier argument doit être une commande qui n'existe pas encore,

8. À l'exclusion de tout autre procédé, comme `\textit` : même si l'apparence est parfois la même, ce n'est pas toujours le cas et de toutes façon la signification est différente.

9. On trouve dans la littérature de nombreuses astuces comme `\textrm` ou `\mbox` pour passer localement en mode texte : elles sont toutes moins robustes que la commande `\text` pour des raisons diverses même si elles donnent un résultat correct dans certaines circonstances. Elles seront systématiquement considérées comme incorrectes.

et le deuxième est le nom tel qu'il doit apparaître dans le document. On n'est pas obligé de prendre le même nom pour la commande, comme le montre l'exemple 4.4.

```
\DeclareMathOperator{\acos}{arccos}
On a toujours $\cos(\acos(x)) = x$.
```

On a toujours $\cos(\arccos(x)) = x$.

Exemple 4.4 – Déclaration et usage d'un nouvel opérateur.

4.2.2 Styles mathématiques

`\displaystyle \textstyle \scriptstyle \scriptscriptstyle`

Il existe en LaTeX quatre styles mathématiques auxquels on peut accéder par les commandes ci-dessus. Il faut bien distinguer les notions de *mode* et de *style* mathématique : par défaut, quand on entre en mode mathématique hors-texte, on est en style `\displaystyle`, alors que quand on entre en mode mathématique en-ligne, on est en style `\textstyle`. Ensuite, LaTeX va faire évoluer le style tout au long de la formule en fonction de l'emplacement des éléments : par exemple, l'indice d'un élément en `\textstyle` sera en `\scriptstyle`, etc.

On peut quand on le souhaite forcer le passage dans un certain style avec l'une des quatre commandes de style : celles-ci agissent jusqu'à la fin de la formule (ou sous-formule) en cours, ou jusqu'à ordre contraire. Le style contrôle la taille des symboles, mais aussi l'espace entre eux : observer par exemple l'espace autour des différents signes « + » dans l'exemple 4.5.

```
\[ \frac{1}{1-\frac{1}{2}} \neq
\frac{1}{\displaystyle 1-\frac{1}{2}}
\text{ et } 1+2 \neq 1^{1+2} \]
```

$$\frac{1}{1-\frac{1}{2}} \neq \frac{1}{\displaystyle 1-\frac{1}{2}} \text{ et } 1+2 \neq 1^{1+2}$$

Exemple 4.5 – Styles mathématiques

4.2.3 Limites et grands opérateurs

`\limits \nolimits`

Comme on l'a vu, le placement des bornes autour d'opérateurs comme `\sum` et `\prod` est réglé par LaTeX ; la règle de base est la suivante. En mode `\displaystyle`, les bornes sont placées au-dessus et en dessous ; dans tous les autres modes elles sont placées comme des exposants normaux. On peut forcer les bornes à se placer dessus et dessous en utilisant la commande `\limits` immédiatement après l'opérateur, comme dans l'exemple 4.6. Dans le sens inverse, on peut utiliser `\nolimits`.

`\DeclareMathOperator*{`⟨*commande*⟩`}{`⟨*nom*⟩`}`

La syntaxe de `\DeclareMathOperator` présentée précédemment n'était pas tout à fait complète : il existe en fait une variante étoilée qui permet de déclarer des opérateurs dont les « bornes » pourront de placer en-dessous et au-dessus. L'exemple 4.6 illustre l'usage de de cette version étoilée.

```
\DeclareMathOperator*{\colim}{colim}
\[ \colim_{x\to\infty} f(x) =
\colim\nolimits_{x_\to\infty} f(x) \]
```

$$\operatorname*{colim}_{x\to\infty} f(x) = \operatorname{colim}_{x_\to\infty} f(x)$$

Exemple 4.6 – Opérateur et placement des bornes.

Commande	Nom	Effet
\qquad	double cadratin	$x \equiv y \quad [\pi]$
\quad	cadratin	$x \equiv y \quad [\pi]$
\	inter-mot	$x \equiv y\ [\pi]$
\;	épaisse	$x \equiv y\ [\pi]$
\:	moyenne	$x \equiv y\ [\pi]$
\,	fine	$x \equiv y\ [\pi]$
	pas d'espace	$x \equiv y[\pi]$
\!	fine négative	$x \equiv y[\pi]$

Table 4.2 – Commandes d'espace en mode mathématique.

4.2.4 Espaces en mode mathématique

On peut parfaitement utiliser \hspace pour retoucher l'espacement en mode mathématique, mais c'est assez rarement une bonne idée. On dispose en fait de commande prédéfinies, présentées dans la table 4.2, qui sont bien plus commodes. Certaines de ces commandes sont même disponibles en mode texte, comme \quad qui est équivalent à \hspace{1em}.

Ces commandes sont au fond assez rarement utiles, car LaTeX se débrouille en général bien pour placer les choses comme il faut en mode mathématique. Un cas cependant où il faut toujours rajouter l'espace à la main est celui d'un quantificateur à séparer du reste d'une formule hors-texte : on écrira donc souvent \quad\forall x. L'usage des commandes d'espacement prédéfinies (par opposition à \hspace permet de rester cohérent entre les différentes formules. D'autres exemples sont donnés en exercices.

4.2.5 Délimiteurs

Certains délimiteurs peuvent se saisir directement au clavier comme (et [, d'autres sont accessibles *via* des commandes comme \{ ou \langle/\rangle ({, ⟨, ⟩). Tous les délimiteurs peuvent s'adapter en taille à leur contenu : pour que LaTeX calcule automatiquement la taille nécessaire, il suffit de faire précéder le délimiteur ouvrant par \left et le délimiteur fermant par \right. Si un symbole intermédiaire doit aussi être adapté en taille, on le fera précéder de \middle ; si un seul symbole est à adapter, on utilisera en face le délimiteur spécial « . » qui est invisible, comme l'illustre l'exemple 4.7.

```
\[\left( \frac{1}{2}\right)^2 \quad
\left. \frac{\partial f}{\partial x}
\right|_{x=0} \quad \left\{
\frac{a}{b} \ \middle|\ b=10^n \right\}\]
```

$$\left(\frac{1}{2}\right)^2 \quad \left.\frac{\partial f}{\partial x}\right|_{x=0} \quad \left\{\frac{a}{b} \,\middle|\, b = 10^n\right\}$$

Exemple 4.7 – Taille automatique des délimiteurs

Résultat	Code
$x \overset{f}{\longmapsto} y$	`$x \stackrel{f}{\longmapsto} y$`
$X_n \xrightarrow[n\to\infty]{L_2} X$	`$X_n \xrightarrow[n\to\infty]{L_2} X$`
${}_a^b\prod_c^d$	`$\sideset{_a^b}{_c^d}\prod$`
$\underset{*}{n} < \overset{*}{n}$	`$\underset{*}{n} < \overset{*}{n}$`
$\binom{n}{p}$	`$\binom{n}{p}$`
$\sum_{\substack{i\in I \\ j\in J}}$	`$\sum_{\substack{i\in I \\j\in J}}$`
${}^t\mathrm{M}$	`${}^t\!M$`
$x^n = \underbrace{x \cdots x}_{n}$	`$x^n = \underbrace{x \cdots x}_n$`

Table 4.3 – Petites constructions mathématiques

Par ailleurs, si la taille calculée par LaTeX ne convient pas, on est toujours libre d'ajuster soi-même la taille des délimiteurs en les faisant précéder d'une des commande `\big`, `\Big`, `\bigg`, `\Bigg`. Il n'y a alors plus aucun contrôle d'équilibrage contrairement à ce qui se passe avec `\left` et `\right`.

4.3 Constructions mathématiques

4.3.1 Petites constructions

Diverses petites constructions sont disponibles avec LaTeX et amsmath. Les plus courantes sont présentées dans la table 4.3.

4.3.2 Alignements

Voyons maintenant comment réaliser divers alignements en mode mathématique : matrices, systèmes d'équations, longs calculs… La syntaxe générale est la même que celle des tableaux, qui seront étudiés en détails au chapitre 7 : les lignes sont séparées entre elles par `\\` et au sein de chaque ligne, les cellules sont séparées par le caractère réservé `&` (qui est illégal en dehors d'un tableau). Tout ceci est entouré d'un environnement qui détermine l'apparence générale de l'alignement en fonction de sa nature.

```
{matrix} {pmatrix} {vmatrix} {Vmatrix} {bmatrix} {Bmatrix}
```

Par exemple, on obtient une matrice simple avec l'environnement `{matrix}`. Les autres variantes de cet environnement ajoutent des délimiteurs autour de la matrice : dans l'ordre ci-dessus, on a des parenthèses (exemple 4.8), des barres verticales, des doubles barres verticales, des crochets, des accolades.

`{aligned} {cases}`

Pour aligner entre elles les équations d'un système, on utilise `{aligned}` : il suffit de placer le repère d'alignement `&` juste devant le signe d'égalité (ou d'inégalité), et de terminer chaque ligne par `\\`. Pour faire une accolade devant le système, on utilisera `\left\{` d'un côté et `\right.` de l'autre, comme illustré à l'exemple 4.7. Une construction particulière revient souvent, pour les disjonctions de cas, comme dans l'exemple 4.8, pour laquelle on dispose de l'environnement spécialisé `{cases}`, qui nous dispense même d'avoir à prendre soin de l'accolade.

```
\[ M = \begin{pmatrix}
    a & b \\ c & d
  \end{pmatrix}  \qquad
  \delta_i^j = \begin{cases}
    0 &\text{si } i \neq j \\
    1 &\text{si } i=j
\end{cases} \]
```

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \delta_i^j = \begin{cases} 0 & \text{si } i \neq j \\ 1 & \text{si } i = j \end{cases}$$

Exemple 4.8 – Alignements mathématiques moyens.

`{align⟨*⟩} {multline⟨*⟩}`

On peut enfin prévoir des alignements au niveau de l'ensemble de la formule. Les environnements `align` et `multline` se distinguent des environnements précédents dans le sens où ils ne s'utilisent pas en mode mathématique, mais directement à la place de `\[...\]` pour *passer* en mode mathématique. En fait, les deux constructions suivantes sont équivalentes :

```
\begin{align*}...\end{align*}
\[ \begin{aligned}...\end{aligned} \]
```

L'environnement `multline*` est particulier dans le sens où vous n'avez pas de repère d'alignement à indiquer, mais seulement les coupures de lignes : il sert pour les formules trop longues pour tenir sur une ligne, mais où aucun alignement sur un symbole particulier n'est requis.

Ces deux derniers environnements existent en variante étoilée ou non étoilée : ceci sert à contrôler la présence ou l'absence de numérotation comme on va le voir à la section suivante.

4.4 Environnements numérotés

4.4.1 Formules numérotées

Pour obtenir une formule numérotée, il suffit de remplacer `\[...\]` par un environnement `{equation}`. En fait, tous les environnements qui permettent de passer en mode math hors-texte (pour l'instant on connaît `{equation}`, `{align}` et `{multline}`), sauf `{displaymath}`, provoquent une numérotation automatique des équations. Ils admettent une version étoilée qui a pour effet de supprimer cette numérotation.

À l'intérieur d'une formule numérotée, on peut utiliser `\label` pour définir une nouvelle étiquette à laquelle on pourra faire référence avec `\ref` et `\pageref` comme on l'a vu à la section 3.5. On peut même remplacer `\ref` par `\eqref`, ce qui a pour effet d'insérer automatiquement les parenthèses autour du numéro d'équation.

4.4.2 Environnements de type théorème

En plus des équations, il est souvent utile de numéroter des théorèmes, définitions, etc. pour pouvoir y faire référence plus tard. Ces environnements munis d'un titre (p. ex. « théorème ») et d'un numéro sont traditionnellement appelés environnements *de type théorème*, même s'ils peuvent en fait être utilisés dans d'autres contextes.

```
\usepackage{amsthm}
\newtheorem⟨*⟩{⟨nom env⟩}[⟨numéroté avec⟩]{⟨titre⟩}[⟨numéroté dans⟩]
```

Aucun n'est défini par défaut : il faut donc les définir soi-même dans le préambule. Pour cela, le plus simple est d'utiliser la commande `\newtheorem`, avec le module `amsthm` qui étend ses possibilités. La syntaxe complète de cette commande est un peu compliquée, mais dans le cas le plus simple elle se résume aux deux argument obligatoires : ⟨*nom env*⟩ est le nom de l'environnement tel qu'il apparaîtra dans le source. Il ne doit pas être égal au nom d'un environnement ou d'un commande [10] qui existe. ⟨*titre*⟩ est le nom de l'environnement tel qu'il apparaîtra dans le document : « Théorème », « Définition », etc.

Par défaut, ceci produit un environnement numéroté. Les numéros des différents environnements sont indépendants, ce qui peut être pénible pour le lecteur, s'il y a dans le même document un lemme 1, une proposition 1, un théorème 1, une définition 1, etc. et que le scholie 1 arrive après le corollaire 3 se rapportant au théorème 4. Pour éviter ceci, on peut demander que plusieurs environnements partagent la même numérotation : pour le premier, on procède normalement, et pour les suivants on utilise comme argument optionnel ⟨*numéroté avec*⟩ le nom du premier.

Par ailleurs, pour un long document, on veut peut-être éviter d'arriver au théorème 42 au bout d'un moment. On peut alors demander que les théorèmes soient numérotés par exemple par section, en passant `section` comme deuxième argument optionnel ⟨*numéroté dans*⟩. Si l'on veut une numérotation par chapitre, on utilisera `chapter`, etc. Par exemple, dans ce polycopié, les exemples sont numérotés par chapitre. On ne peut pas utiliser simultanément le premier et le deuxième argument optionnel de `\newtheorem` : on utilisera le second pour définir un premier environnement, puis le premier pour les environnements partageant la même numérotation, comme le montre l'exemple 4.9.

Enfin, on peut définir des environnements nommés mais non numérotés, comme l'environnement `qc` de l'exemple 4.9, avec la variante étoilée de `\newtheorem`. Il n'est alors bien sûr pas possible d'utiliser les deux arguments optionnels qui n'ont plus de sens. Enfin, un environnement `proof` est disponible automatiquement : il s'agit d'un environnement avec titre et sans numérotation, muni d'un symbole spécial de fin d'environnement « □ ».

Les environnements définis par `\newtheorem`, ainsi que l'environnement `proof`, admettent toujours un argument optionnel, qui permet de préciser le titre du théorème, comme on le voit à l'exemple 4.9 avec la mention « de Fermat » pour le premier théorème.

```
\theoremstyle{⟨nom de style⟩}
```

On peut par ailleurs obtenir différents styles d'environnements en utilisant `\theoremstyle` pour changer le style des environnements de type théorème qui seront définis ultérieurement. Les styles disponibles sont `plain`, `definition` et `remark`. Je renvoie aux exercices et à la documentation d'`amsthm` pour plus de détails.

10. En particulier, ça ne peut pas être `def` pour les définitions.

```
\usepackage{amsthm}
\newtheorem{thm}{Théorème}[section]
\newtheorem{exo}[thm]{Exercice}
\newtheorem*{qc}{Question de cours}
\begin{thm}[de Fermat]
  Cubum autem in dous cubos,\dots \end{thm}
\begin{proof}
  La marge est trop étroite. \end{proof}
\begin{exo}
  La changer avec \verb+\geometry+. \end{exo}
\begin{qc}
  Qu'est-ce qui est trop étroit ? \end{qc}
```

Théorème 4.4.1 (de Fermat)**.** *Cubum autem in dous cubos,…*

Démonstration. La marge est trop étroite. □

Exercice 4.4.2. *La changer avec* `\geometry`.

Question de cours. *Qu'est-ce qui est trop étroit ?*

Exemple 4.9 – Environnements de type théorème.

4.5 Documentation

Les possibilités mathématiques de LaTeX sont immenses, les mathématiciens étant les principaux utilisateurs. Pour ce chapitre comme les autres, il n'est pas possible de tout présenter. J'aimerai rappeler ici les principales sources de documentation concernant les math en LaTeX.

- La liste de symboles courants [11] distribuée, extraite de `flshort-3.20.pdf` [12]. Pour les symboles plus rares, les sections 2 et 3 de `symbols-a4.pdf` [13].
- Les documentations d'`amsmath` [14] et de `mathtools` [15] pour plus de détails sur les alignements et certains points délicats.
- Le document présentant (presque) tout ce qui est possible avec des math en LaTeX : `Mathmode.pdf` [16]. De préférence, ne pas trop s'attarder sur la partie un, qui fait souvent de façon compliquée ce qui est présenté de façon plus simple dans les parties ultérieures en utilisant des modules.

Certains de ces document vous aideront à résoudre des problèmes difficiles que vous rencontrez sans doute plus tard, mais vous êtes d'ores et déjà encouragés à garder toujours sous la main la liste des symboles courants. En particulier, pour l'examen et les exercices, tous les symboles présents sur cette liste sont supposés connus même s'ils n'ont pas été présentés indépendamment en cours.

11. http://people.math.jussieu.fr/~mpg/lm204/files/doc-symboles-math.pdf
12. http://mirror.ctan.org/info/lshort/french//flshort-3.20.pdf
13. http://mirror.ctan.org/info/symbols/comprehensive/symbols-a4.pdf
14. http://ctan.org/pkg/amsmath
15. http://ctan.org/pkg/mathtools
16. http://mirror.ctan.org/info/math/voss/mathmode/Mathmode.pdf

5 Révisions et création de commandes

5.1 Rappels et compléments

5.1.1 Source minimal et encodages

5.1.2 Messages d'erreur courants

5.1.3 Trouver la documentation et l'aide

5.2 Définitions

(rappels sur ce qui est su : `\DeclareMathOperator` et `\newtheorem`, intro et intérêt)

5.2.1 Commandes simples

5.2.2 Commandes avec arguments

5.2.3 Commandes avec arguments optionnels

5.2.4 Environnements

5.2.5 Couleurs

5.2.6 Redéfinitions

6 Figures

Le corps de ce chapitre arrivera prochainement. Merci de votre compréhension et de votre patience.

6.1 Inclusion d'images

6.1.1 La question des formats

6.1.2 Options d'inclusion

6.2 Placement

6.2.1 Habillé par le texte

6.2.2 Flottant

6.2.3 Astuces

6.3 Autres fioritures graphiques

6.3.1 Rotations et mises à l'échelle

6.3.2 Encadrement

6.3.3 Une police de symboles

7 Tableaux

Le corps de ce chapitre arrivera prochainement. Merci de votre compréhension et de votre patience.

7.1 Tableaux simples

7.1.1 Les bases

7.1.2 Types de colonnes particuliers

7.1.3 Placement

7.1.4 En mode mathématique

7.2 Techniques plus avancées

7.2.1 Colonnes personnalisées

7.2.2 Fusion de cellules

7.2.3 Ajustement de la largeur

7.2.4 Couleurs

8 Compléments divers

Le corps de ce chapitre arrivera prochainement. Merci de votre compréhension et de votre patience.

8.1 Note technique

Par rapport au cours donné en salle (et donc aux supports de présentation utilisés, notamment `03-texte.pdf`, `06-graph.pdf`, `07-tab.pdf` et bien sûr `08-comp-divers.pdf`, des éléments ont été placés plus tôt dans ce cours. Il s'agit des sections 3.5.2 (bibliographie), 6.2.3 (astuces pour les figures), 7.1.4 (tableaux en mode mathématique) et 7.2.4 (couleurs dans les tableaux). Les exercices suivront la progression du cours réel : les notions vues dans les sections citées ne seront donc pas présentées avant l'exercice `exo-08.pdf`.

Par ailleurs, la démonstration d'utilisation de `bibtex` avec `biblatex` faite en cours était proposée à titre purement culturel et ne correspond en aucun cas à des connaissances exigibles à l'examen. Elle ne sera donc pas reproduite ici.

8.2 Code informatique

8.2.1 Outils de LaTeX

8.2.2 Avec le module fancyvrb

8.2.3 Avec le module listings

8.3 Concentré d'orthotypographie

9 Élements de programmation

Le corps de ce chapitre arrivera prochainement. Merci de votre compréhension et de votre patience.
La table des matières de ce chapitre est incomplète et provisoire.

9.1 Compteurs

9.2 Longueurs

9.3 Boîtes

9.3.1 Concepts

9.3.2 Boîtes horizontales

`(Plus tard.)`	(Plus tard.)

Exemple 9.1 – Boîtes de largeur nulles.

9.3.3 Boîtes verticales

9.3.4 Réglures

10 Personnalisation

Le corps de ce chapitre arrivera prochainement. Merci de votre compréhension et de votre patience.
La table des matières de ce chapitre est incomplète et provisoire.

10.1 En-têtes et pieds de page

11 Présentations vidéoprojetées

Le corps de ce chapitre arrivera prochainement. Merci de votre compréhension et de votre patience.

A Encodages

B Documentation

C Aide-mémoire

```
\documentclass[a4paper]{article}   % ou report ou book ou ...
\usepackage[latin1]{inputenc}      % ou utf8 ou macroman
\usepackage[T1]{fontenc}
% \usepackage{textcomp}            % symboles complémentaires (euro, etc.)
% \usepackage{amsmath, amssymb}    % minimum pour les maths
% \usepackage{xspace}              % protection de certains espaces
% (autres modules)
\usepackage[frenchb]{babel}
% \usepackage{hyperref}            % liens hypertexte
% (définitions et commandes diverses)
\begin{document}
% (corps du document)
\end{document}
```

Source C.1 – Source de base pour document en français.

Caractère	Usage	LaTeX	textcomp	Math
\	commandes		`\textbackslash`	`\backslash`
{	argument ou	`\{`	`\textbraceleft`	`\{`
}	groupe	`\}`	`\textbraceright`	`\}`
$	mode math	`\$`		`\$`
&	alignements	`\&`		
#	définitions	`\#`		
^	exposant	`\^{}`		
_	indice	`\_`		
~	espace insécable	`\~{}`	`\textasciitilde`	
%	commentaire	`\%`		

Table C.1 – Caractères réservés et comment les obtenir.

Ex.	Code	Ex.	Code	Ex.	Code	Ex.	Code
Ä ä	`\"A \"a`	Á á	`\'A \'a`	Ă ă	`\u A \u a`	Ā ā	`\=A \=a`
Â â	`\^A \^a`	À à	`\`A \` a`	Ã ã	`\~A \~ a`	Å å	`\r A \r a`
Ċ ċ	`\.C \.a`	Ř ř	`\v R \v r`	Ç ç	`\c A \c a`	Ą ą	`\k A \k a`

Table C.2 – Commandes d'accent en mode texte