

UNIVERSITÉ PIERRE ET MARIE CURIE
Module LM204 de la licence math-info

Apprentissage et pratique de \LaTeX

Manuel PÉGOURIÉ-GONNARD

1^{er} semestre 2008–2009

Licence

Ce document est placé sous la licence libre *GNU Free Documentation Licence* 1.3 : vous êtes libres de l'utiliser, le diffuser et le modifier, sous réserve de conserver une licence compatible. Pour plus de détails, voir le fichier `licence.txt` dans les sources du document, disponibles à l'adresse

<http://people.math.jussieu.fr/~mpg/lm204/files/sources.zip>.

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 en 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 des 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 approximative, qui semble « marcher » sur le moment, et qui deviendra vite une (mauvaise) habitude. On risque ainsi de rencontrer plus tard des problèmes inattendus, ou tout simplement de ne pas atteindre le maximum en termes d'efficacité ou de 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 GRIVAUD 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

Préambule	iii
1 Prise de contact	1
1.1 Présentation du module	1
1.1.1 Organisation	1
1.1.2 Contenu	1
1.2 Présentation de \LaTeX	2
1.2.1 Possibilités	2
1.2.2 Particularités	2
1.3 Installation	4
1.3.1 Windows	4
1.3.2 Mac OS X	4
1.3.3 Linux	5
1.4 Premiers documents	5
1.5 Bases théoriques	8
1.5.1 Commandes, arguments, environnements	8
1.5.2 Espaces et fins de ligne	9
1.6 Exercices	10
1.6.1 Contenu des exercices	10
2 Le mode texte	11
2.1 Caractères et symboles particuliers	11
2.1.1 Caractères réservés	11
2.1.2 Accents et symboles	11
2.1.3 Franchouillardises	13
2.1.4 Interlude : documentation	13
2.2 Changements de fonte	14
2.2.1 Modèle théorique	14
2.2.2 Tout sauf la taille	14
2.2.3 La taille	16
2.2.4 Le reste	17
2.3 Listes	18
2.4 Alignement du texte	19
2.5 Espaces	21
3 Structure du document	23
3.1 Classes de documents	23
3.2 Notes	23
3.2.1 Notes marginales	23

3.2.2	Notes de bas de page	25
3.3	Titre et résumé	25
3.3.1	Titre	25
3.3.2	Résumé	26
3.4	Structure globale	26
3.5	Références et liens hypertexte	28
3.5.1	Références croisées	28
3.5.2	Bibliographie	29
3.5.3	Liens hypertexte	30
3.6	Structure de la page	31
4	Les modes mathématiques	33
4.1	Découvertes des modes mathématiques	33
4.1.1	Les deux modes mathématiques	33
4.1.2	Outils de base	34
4.2	Points plus délicats	36
4.2.1	Distinguer texte et mathématiques	36
4.2.2	Styles mathématiques	37
4.2.3	Limites et grands opérateurs	37
4.2.4	Espaces en mode mathématique	38
4.2.5	Délimiteurs	38
4.3	Constructions mathématiques	39
4.3.1	Petites constructions	39
4.3.2	Alignements	39
4.4	Environnements numérotés	40
4.4.1	Formules numérotées	40
4.4.2	Environnements de type théorème	41
4.5	Documentation	42
5	Révisions et création de commandes	43
5.1	Rappels et compléments	43
5.1.1	Source minimal et encodages	43
5.1.2	Messages d'erreur courants	44
5.2	Définitions	45
5.2.1	Commandes simples	46
5.2.2	Commandes avec arguments	47
5.2.3	Commandes avec arguments optionnels	48
5.2.4	Environnements	48
5.2.5	Redéfinitions	49
5.2.6	Couleurs	50
6	Figures	51
6.1	Inclusion d'images	51
6.1.1	La question des formats	51
6.1.2	Inclusion simple	52
6.1.3	Options d'inclusion	53

6.2	Placement	55
6.2.1	Simple	55
6.2.2	Habillé par le texte	55
6.2.3	Flottant	57
6.2.4	Une astuce classique	58
6.3	Autres fioritures graphiques	59
6.3.1	Rotations et mises à l'échelle	59
6.3.2	Encadrement	59
6.3.3	Une police de symboles	60
7	Tableaux	61
7.1	Tableaux simples	61
7.1.1	Les bases	61
7.1.2	Colonnes de type paragraphe	62
7.1.3	Placement	63
7.1.4	En mode mathématique	63
7.2	Techniques plus avancées	64
7.2.1	Matériel automatique	64
7.2.2	Colonnes personnalisées	65
7.2.3	Fusion de cellules	65
7.2.4	Ajustement de la largeur	67
7.2.5	Couleurs	68
7.2.6	Aspects esthétiques	69
8	Compléments divers	71
8.1	Note technique	71
8.2	Code informatique	71
8.2.1	Outils de \LaTeX	71
8.2.2	Avec le module fancyrb	72
8.2.3	Aparté : le verbatim dans des arguments	74
8.2.4	Avec le module listings	75
8.3	Concentré d'orthotypographie	76
8.3.1	Références pour aller plus loin	77
9	Éléments de programmation	79
9.1	Rappels sur les définitions	79
9.2	Compteurs	79
9.3	Longueurs	82
9.4	Boîtes	85
9.4.1	Concepts	85
9.4.2	Boîtes horizontales	85
9.4.3	Boîtes verticales	87
9.4.4	Réglures	87
9.4.5	Aparté : espacement dans les tableaux	88
9.4.6	Remplissage de boîtes	89

10 Personnalisation	91
10.1 Polices	91
10.2 En-têtes et pieds de page	93
10.3 Titres de chapitres et sections	94
10.4 Listes	95
10.5 Flottants	97
11 Présentations vidéoprojetées	99

Listes des exemples, tables, sources et figures

Liste des exemples

1.1	Extrait de source	3
1.2	Exemple d'exemple	7
2.1	Quelques accents, cédilles et ligatures	12
2.2	Famille, graisse et forme de fonte	15
2.3	Changements de taille et interligne	17
2.4	Changements de couleurs	18
2.5	Quelques possibilités du module soul	18
2.6	Les trois types de listes	19
2.7	Environnements d'alignement du texte	20
3.1	Note de bas de page	25
3.2	Bibliographie et citations	30
3.3	Liens avec hyperref	30
4.1	Les deux modes mathématiques	33
4.2	Indices, exposants, fractions et racines	34
4.3	Sommes et intégrales	35
4.4	Déclaration et usage d'un nouvel opérateur	37
4.5	Styles mathématiques	37
4.6	Opérateur et placement des bornes	38
4.7	Taille automatique des délimiteurs	39
4.8	Alignements mathématiques moyens	40
4.9	Environnements de type théorème	42
5.1	Définitions simples	46
5.2	Utilisation de <code>\xspace</code>	47
5.3	Commandes avec arguments	47
5.4	Commande avec argument optionnel	48
5.5	Définitions d'environnements	49
6.1	Inclusion de graphiques avec rotation	54
6.2	Recadrage et découpage d'image	54
6.3	Habillage dans le coin en haut à droite	56
6.4	Image en fenêtre dans le texte	56
6.5	Rotation et dilatation de texte	59
6.6	Cadres plus sophistiqués	60
6.7	Boîtes colorées	60

6.8	Utilisation d'une police de symboles	60
7.1	Types de colonnes simples	61
7.2	Tableaux avec des filets	62
7.3	Matériel inter-colonnes	62
7.4	Colonnes de type paragraphe et alignement	63
7.5	Mauvais espaces horizontaux avec <code>array</code>	64
7.6	Matériel automatique pré- et post-colonne	64
7.7	Gestion des retours à la ligne	65
7.8	Fusion horizontale de cellules	66
7.9	Fusion verticale de cellules	66
7.10	Tableaux de largeur fixe	67
7.11	Trop de couleurs dans un tableau	68
7.12	Une utilisation raisonnable de la couleur	69
8.1	Commande et environnement de base pour le verbatim	72
8.2	Usage simple de <code>fancyvrb</code>	73
8.3	Verbatim : quand l'imitation vaut mieux que l'original	74
8.4	Reconnaissance de mots-clé du langage C	75
9.1	Modification de la représentation d'un compteur	81
9.2	Utilisation d'un compteur personnel	82
9.3	Paramétrage de <code>\fbox</code>	83
9.4	Mesure de mots pour un texte à trous	84
9.5	Utilisation de boîtes de largeur nulle	86
9.6	Utilisation de <code>\framebox</code> et <code>\raisebox</code>	86
9.7	Une boîte verticale dans une boîte horizontale	87
9.8	Réglures	87
10.1	En-têtes et pieds de page	94
10.2	Listes personnalisées	96

Liste des tables

2.1	Commandes pour les caractères réservés	11
2.2	Commandes fournies par <code>french</code> de <code>babel</code>	13
2.3	Famille, graisse et forme de fonte.	15
2.4	Tailles de fontes relatives	16
2.5	Environnements et commandes d'alignement.	20
4.1	Fontes disponibles en mode mathématiques	36
4.2	Commandes d'espace en mode mathématique	38
4.3	Petites constructions mathématiques	39
6.1	Modes de compilation et formats graphiques	51
8.1	Espacement autour des signes de ponctuation	77

9.1	Compteurs de base de \LaTeX	79
9.2	Unités de longueur	82
10.1	Exemples de polices avec leurs noms	92
10.2	Polices fantaisistes	93

Liste des code sources

1.1	Source minimum compilable	5
1.2	Tout petit document en français	6
1.3	Source développé de l'exemple 1.2	8
3.1	Structure d'un document en classe <code>article</code>	27
5.1	Fichier source de base	43
10.1	Création et gestion des types de flottants dans <code>ce poly</code>	97

Liste des figures

6.1	Centres de rotation possibles	54
9.1	Comment \LaTeX voit les caractères	85
9.2	Les trois dimensions des boîtes	85

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 consacrée 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 ; ces questions sont généralement signalées et il est conseillé de les garder pour la fin.

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, consiste 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**³ a 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 de 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 .

1. <http://people.math.jussieu.fr/~mpg/lm204/files/doc-eval-rapport.pdf>

2. <http://people.math.jussieu.fr/~mpg/lm204/prog.html>

3. <http://people.math.jussieu.fr/~mpg/lm204/files/doc-biblio.pdf>

- Vous donner des pistes pour être en mesure par la suite de continuer seul votre apprentissage.

Les cinq premiers chapitres du cours fournissent le minimum vital pour composer la plupart des documents scientifiques ne comportant pas d'autre élément complexe 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](#)⁴. 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'il est possible de programmer soi-même certains aspects de ses documents 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 forme⁵ 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

4. <http://people.math.jussieu.fr/~mpg/lm204/files/01-intro.pdf>

5. On dit que \LaTeX est un langage de balisage. Il est sur certains points comparables au couple HTML & CSS.

n'est plus modifiable comme l'est le fichier `.doc`. Cette séparation est heureuse car elle permet de bien distinguer les rôles : pour l'auteur, le `.tex`, pour les lecteurs, le `.pdf`.

Interrompons un moment cette présentation un peu théorique pour regarder un extrait d'une source \LaTeX (on verra bientôt à quoi ressemble un 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 vous faut donc installer une distribution \TeX et un éditeur adapté à \LaTeX (voir [section 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**⁶ : 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 en évoquant 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, un 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.

6. <http://www.april.org/fr/articles/intro>

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.

1.3 Installation

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

1. une distribution \TeX ⁷ ;
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 IDE⁸ 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 Live⁹ et MiK \TeX ¹⁰. Pour les éditeurs, le plus classique est `TeXnicCenter`¹¹, mais `TeXmaker`¹² 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**¹³ décrit en détails l'installation d'un environnement MiK \TeX & 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 le 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 `Mac \TeX` ¹⁴, dérivée de \TeX Live et munie d'un installateur spécifique pour Mac OS. En

7. \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.

8. Pour *integrated development environment*, soit environnement de développement intégré.

9. <http://tug.org/texlive/>

10. <http://www.miktex.org/>

11. <http://www.toolscenter.org/>

12. http://www.xmlmath.net/texmaker/index_fr.html

13. <http://people.math.jussieu.fr/~mpg/lm204/files/doc-install-miktex+txc.pdf>

14. <http://tug.org/mactex/>

outre, elle intègre un éditeur, TeXShop, et constitue donc à elle seule un environnement complet de travail.

D'autres bons éditeurs, ainsi qu'un peu plus de documentation, sont fournis dans le paquet supplémentaire `MacTeXtras`¹⁵, qui mérite donc d'être installé.

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 17 \(page 7\)](#), par rapport à certains exemples.

1.3.3 Linux

La distribution de référence est ici TeX Live (il faut mieux oublier TeX 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 17 \(page 7\)](#), par rapport à certains exemples. Sinon, vous pouvez sélectionner latin1 ou son synonyme iso-8859-1 comme encodage par défaut, généralement dans la partie « éditeur » du menu des préférences de votre IDE. (Pour Kile 2.1 : menu configuration, configurer Kile, puis dans la partie éditeur, ouvrir/enregistrer et liste encodage. Pour TeXmaker 1.8 : menu options, configurer TeXmaker, partie éditeur, liste encodage.)

1.4 Premiers documents

Voyons maintenant à quoi ressemble un source L^AT_EX 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 L^AT_EX. Les mots entourés d'accolades qui suivent une commande

15. <http://tug.org/mactex/mactextras.html>

sont les *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 destiné à 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 quelques éléments de la syntaxe de \LaTeX . Vous n'avez pas besoin de tout comprendre pour le moment, mais voici 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 utiliserons 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 modules¹⁶ 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 [section 1.3.2](#) ; pour TeXmaker ou Kile, voir la [section 1.3.3](#).¹⁷

16. Aussi appelés extensions, paquets ou en anglais *packages*. Dans tout ce cours, j'utiliserai « module » pour essayer d'écrire en français correct, mais l'anglicisme *package* reste le terme le plus couramment utilisé en général.

17. Cependant, afin de laisser le choix libre, je propose toutes les solutions des exercices en latin1 et en utf8. Si vous

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 .
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, reprenez absolument que certaines options et certains modules sont *obligatoires* pour chaque document. Un source de base est disponible *en ligne*¹⁸ (sinon, voir *source 5.1*) et je vous conseille *fortement* de vous en servir comme point 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          Le groupe d'utilisateurs de TeX.
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}` sera omis : 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 fait¹⁹ le *source complet 1.3*.

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

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 de 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,

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`.

18. <http://people.math.jussieu.fr/~mpg/lm204/files/doc-source-base.tex>

19. Pour être exact, il faut éventuellement remplacer `latin1` par `utf8`, selon le réglage de votre éditeur.

```

\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

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 instructions, comme le chargement de modules.

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

$$\ \{ \} \$ \& \# \wedge _ \sim \%$$

Vous connaissez déjà certains d'entre eux, les autres seront présentés petit à petit. Ce sont les dix caractères qui n'apparaîtront pas tels quels 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 ferai référence ainsi à un nom entre accolades, il s'agira du nom d'un environnement ²⁰.

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 au 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 ²¹ 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 « $\text{\LaTeX}c'est bien$ ». 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 ²². Plusieurs lignes vides successives sont équivalentes à une seule ligne vide.

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

21. 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.

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

Vu qu'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.

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 du [source 1.2](#) 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.

1.6.1 Contenu des exercices

Les exercices sont fournis dans des archives nommées `exos- $\langle nn \rangle$.zip` (ou `exos+cor- $\langle nn \rangle$.zip` si la correction est incluse), contenant tous les fichiers utiles, où $\langle nn \rangle$ est le numéro de la séance ou le nom (dm, exam). Ils consistent en général en un document à reproduire, qui contient parfois des indications sur l'exercice, ou des informations intéressantes, mais parfois seulement du texte de remplissage. Les types de fichiers potentiellement présents dans une archive sont :

- `exo- $\langle nn \rangle$.pdf` : le document à reproduire, en pdf ;
- `brut- $\langle nn \rangle$ - $\langle enco \rangle$.pdf` : le texte brut de l'exercice, prêt à être copie-collé, fourni dans différents encodages (choisir `latin1` pour `TeXnicCenter`) ;
- parfois des images utiles pour reproduire le document ;
- `sol- $\langle nn \rangle$ - $\langle enco \rangle$.tex` : le source complet du document à reproduire, prêt à compiler (et à modifier), mais sans commentaires ;
- `cor- $\langle nn \rangle$.pdf` : le source du document à reproduire, accompagné de commentaires.

Les solutions et corrigés des exercices sont une source importante d'information, car ils illustrent et expliquent les notions vues en cours sur des documents complets : n'hésitez pas à vous en inspirer lors de la rédaction de vos documents. Par ailleurs, le code des exemples est souvent mal présenté pour économiser la place : celui des solutions est un bien meilleur exemple sur ce point (à mon avis important). Enfin, les corrigés comportent parfois des informations qui ne sont pas reprises dans le cours.

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, table 2.1.

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 avez 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 :

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

TABLE 2.1 — Commandes pour les caractères réservés

1. Ils n'existent pas dans l'encodage d'entrée¹ utilisé.
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.

<code>\'A, \'A, ^A, \'A, \dots\par</code>	Á, À, Â, Ä, ...
<code>\'E, \'I, \'i, \'n, \dots\par</code>	É, Í, í, ñ, ...
<code>\c Ca me fend le c\oe ur !</code>	Ça me fend le cœur ! Tchüß!
<code>Tch\"u\ss!</code>	

EXEMPLE 2.1 — Quelques accents, cédilles et ligatures

Quelques commentaires sur cet exemple : observez l'utilisation de `\par` à la fin des deux premières 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` du³ module `marvosym`⁴.

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`⁵. Il est important en consultant ce document de faire attention aux modules à charger pour disposer des commandes indiquées. Nous verrons tout bientôt (section 2.1.4) comment accéder aux documents de référence.

Enfin, certains caractères particuliers s'obtiennent sans même faire appel à une commande : \LaTeX peut automatiquement « mélanger » deux ou plusieurs caractères⁶ pour en obtenir un autre.

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

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://ctan.org/pkg/marvosym>

5. <http://mirror.ctan.org/info/symbols/comprehensive/symbols-a4.pdf>

6. Un tel mélange s'appelle une *ligature*.

Exemple	Résultat
<code>\og guillemets \fg</code>	« guillemets »
<code>M\up{me}, D\up{r}</code>	M ^{me} , D ^r
<code>1\ier, 1\iere, 1\ieres</code>	1 ^{er} , 1 ^{re} , 1 ^{res}
<code>2\ieme 4\iemes</code>	2 ^e 4 ^{es}
<code>\No 1, \no 2</code>	N ^o 1, n ^o 2
<code>20~\degres C, 45\degres</code>	20 °C, 45°
<code>\bsc{M. Durand}</code>	M. DURAND
<code>\today</code>	8 septembre 2009

TABLE 2.2 — Commandes fournies par `french` de `babel`

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 fournit une bonne transition vers la prochaine section.

2.1.3 Franchouillardises

En plus de traduire certains textes automatiques, d’assurer une césure⁷ correcte des mots, et d’aider au respect de quelques règles orthotypographiques propres au Français, l’option `french`⁸ de `babel` fournit aussi quelques commandes utiles, résumées par la [table 2.2](#).

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 1^{er}jour ». Il faut penser à protéger l’espace en saisissant `le 1\ier{} jour` pour obtenir le résultat correct « le 1^{er} 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, décommentez⁹ la ligne 8 du `source minimal`¹⁰ 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 $\text{MiK}\TeX$, il s’appelle `mthelp`, sous $\text{T}\TeX$ Live c’est `texdoc`. Dans les deux cas, l’utilisation est la même : il s’agit de taper, en ligne de

7. Vous l’aurez remarqué, \LaTeX coupe automatiquement certains mots en fin de ligne. Les règles concernant la césure (l’action de couper les mots) dépendant fortement de la langue.

8. Synonyme des options `français` (sans cédille) et `frenchb` (pour `french babel`). J’utiliserai souvent `frenchb` par habitude, car les trois n’ont pas toujours été synonymes.

9. C’est-à-dire, retirez le signe `%` présent en début de ligne.

10. <http://people.math.jussieu.fr/~mpg/lm204/files/doc-source-minimal.tex>

commande¹¹ 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 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**¹². Il dispose d'une **page de recherches**¹³. De plus, 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 est 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 paramètres¹⁴ logiquement indépendants : la famille, la graisse, le forme, et la taille. Ici, « logiquement indépendant » signifie qu'on peut demander à \LaTeX de changer la 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 : une famille romaine (avec empattements) qui est en général 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 vus ci-dessus, la taille est un peu à part. La **table 2.3** montre comment changer les trois autres. Les polices par défaut de \LaTeX sont utilisées pour les illustrations.

11. 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.

12. <http://ctan.org/>

13. <http://ctan.org/search.html>

14. 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		
<code>\textrm{}</code>	<code>\rmfamily</code>	romain
<code>\textsf{}</code>	<code>\sffamily</code>	sans empattements
<code>\texttt{}</code>	<code>\ttfamily</code>	chasse fixe
Graisie		
<code>\textmd{}</code>	<code>\mdseries</code>	graisie normale
<code>\textbf{}</code>	<code>\bfseries</code>	gras
Forme		
<code>\textup{}</code>	<code>\upshape</code>	droit
<code>\textit{}</code>	<code>\itshape</code>	<i>italique</i>
<code>\textsl{}</code>	<code>\slshape</code>	<i>penché</i>
<code>\textsc{}</code>	<code>\scshape</code>	PETITES CAPITALES

TABLE 2.3 — Famille, graisie et forme de fonte.

Les valeurs par défaut de ces paramètres sont en général : famille romaine, forme droite et graisie 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 déjà à autre chose, comme à délimiter l’argument d’une commande. Il faut alors bien prendre 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 dans 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.

<code>Un {\bfseries mot gras \textit{et italique}}</code>	Un mot gras et italique aussi. Retour à
<code>aussi}. Retour à la normale. \texttt{Machine</code>	la normale. Machine à écrire .
<code>\textbf{à écrire}.} \textsc{Petites capitales</code>	PETITES CAPITALES GRASSES , à
<code>\textbf{grasses}, à \texttt{chasse fixe} ?}</code>	chasse fixe?

EXEMPLE 2.2 — Famille, graisie 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.

Taille	
<code>\tiny</code>	taille
<code>\scriptsize</code>	taille
<code>\footnotesize</code>	taille
<code>\small</code>	taille
<code>\normalsize</code>	taille
<code>\large</code>	taille
<code>\Large</code>	taille
<code>\LARGE</code>	taille
<code>\huge</code>	taille
<code>\Huge</code>	taille

TABLE 2.4 — Tailles de fontes relatives

```

! Paragraph ended before \text@command was complete.
<to be read again>
                                \par
1.18 bla}

```

L'intérêt d'un tel message d'erreur 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'une 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 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 toute une portion de texte. Les commandes disponibles sont listées dans la [table 2.4](#).

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

```
\documentclass[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

```
\documentclass[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 affiche).

```
\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 5.2.6](#) 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 un peu étrange.

<pre>Du \textcolor{red}{rouge} flashy.\par {\color{red!30!black} Moins} flashy.</pre>	<pre>Du rouge flashy. Moins flashy.</pre>
---	---

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 mises en forme au texte, comme le soulignement. Ce dernier est à *éviter* autant que possible. Il est très utilisé dans les documents 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 complétude, citons le module `soul` qui permet entre autres de souligner des portions de texte. Il fournit aussi des commandes pour d'autres 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](#).

<pre> \usepackage{soul} \ul{Souligné} \par \st{Barré} \par \so{Interlettré}</pre>	<pre>Souligné Barré Interlettré</pre>
---	---------------------------------------

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 numérotée, 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.

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

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 2.5 : on prendra bien garde à l’inversion `left/right` : aligné à gauche est en effet équivalent à « déchiré » (*ragged*) à droite.

Environnement	Commande
<code>center</code>	<code>\centering</code>
<code>flushleft</code>	<code>\raggedright</code>
<code>flushright</code>	<code>\raggedleft</code>

TABLE 2.5 — Environnements et commandes d’alignement.

Si l’on choisit d’utiliser une commande et non un environnement, 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 environnements 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 de l'espace respectivement horizontalement et verticalement. On utilise ainsi `\hspace{3cm}` pour laisser un blanc 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 moment¹⁵ les unités métriques `mm` et `cm`, l'unité anglo-saxonne `in` (pouce), les unités (typo-)graphiques `pt` et `bp` (deux définitions différentes¹⁶ 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 et 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{<*>}`

15. La liste complète des unités que connaît \TeX , avec leur valeur, est donnée par la [table 9.2](#) si vous y tenez.

16. 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.

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 : \TeX 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 peu fréquent 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 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 subtile¹⁷ ; en général, utilisez `\clearpage`, ou encore n'utilisez aucune des trois, car elles ne sont pas souvent utiles.

`\stretch{<force>}`

Il existe en \TeX 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 le 2.

17. En fait, `\clearpage` force \TeX à placer immédiatement tous les flottants en attente (voir [section 6.2.3](#) pour la notion de flottant).

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 dernière 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` (source 1.1) que l'on n'utilise en pratique *jamais*, et la classe `article` (source 1.2) 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 grincer \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. (Voir aussi la section 9.4.6.)

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 la 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 utilisé 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 9.5](#)).

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 pas interférer avec la numérotation des notes du texte principal. Ceci se fait automatiquement avec la technique utilisée pour les exemples (environnement `minipage`, qui sera vu en [9.4.3](#)).

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 peut 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 *ETX 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 raisonnable⁸ 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 mot⁹ « 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.

<pre> \label{<étiquette>} \ref{<étiquette>} \pageref{<étiquette>} </pre>
--

Ce moyen est fourni 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’union¹¹. 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’on 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}{\langle exemple \rangle}
\bibitem{\langle étiquette \rangle}
\cite[\langle endroit \rangle]{\langle étiquette \rangle}
```

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 la 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 `\langle exemple \rangle`, pour donner la 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]{lcf}.
% plus loin dans le document
\begin{thebibliography}{LC}
  \bibitem{ttb} \emph{Tame the BeaST},
  \bsc{N. Markey}, CTAN.
  \bibitem{lcf} \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] *TEX 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
linkcolor=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 `linkcolor`, ou `urlcolor` selon le type de lien. Enfin, vous pouvez spécifier certaines 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 lien¹² 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'écire}.

```

Le CTAN : <http://ctan.org/>. Ou le CTAN. Pour m'écire.

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 en [section 2.5](#). Par exemple, on peut dire

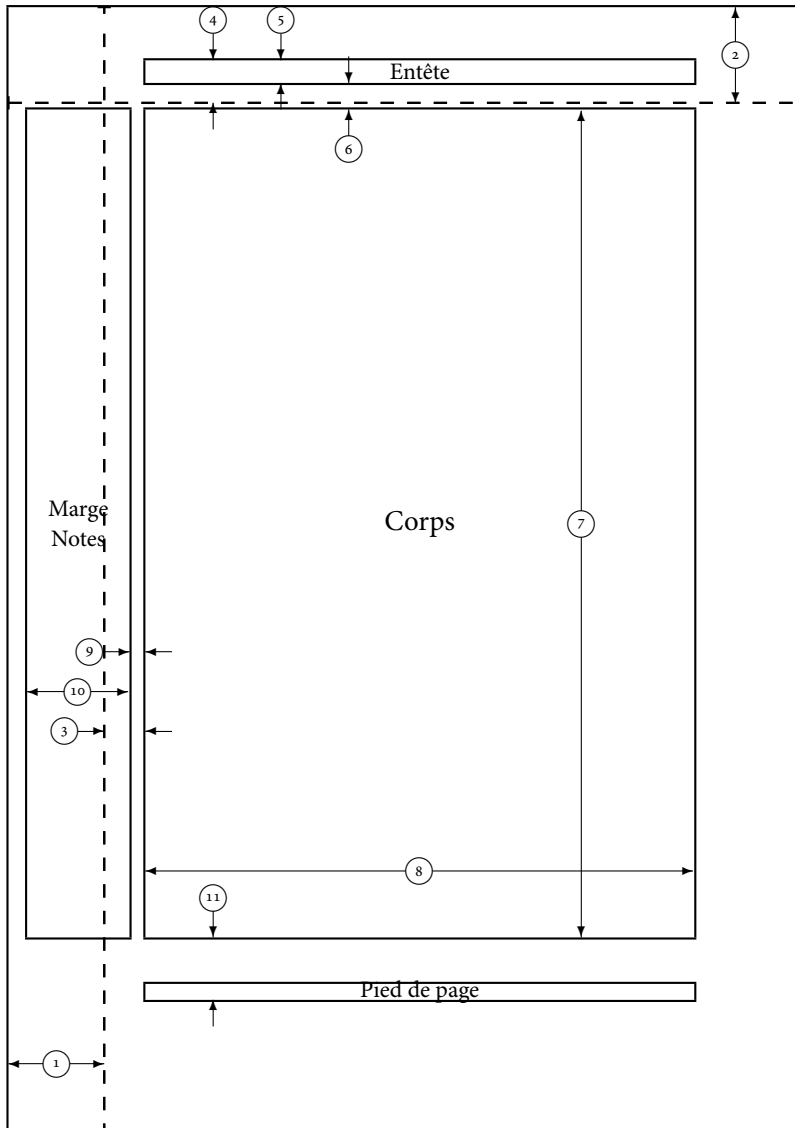
```
\usepackage[hmargin=3cm, landscape]{geometry}
```

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 présent document sont visibles sur la page suivante.

Toutes ces longueurs peuvent être spécifiées comme options de `geometry` (en retirant le `\` initial). Toutefois, il est conseillé de ne pas modifier `hoffset` ni `voffset`, réservés à des usages bien particuliers. Par ailleurs, on aura intérêt à ne pas régler directement `paperwidth` et `paperheight` si on utilise un format de papier standard et à préférer l'option correspondante (par exemple, `a4paper`).

On constate sur la figure 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 en [section 10.2](#) 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 défini par
`\[(\sqrt{2})^2 = 2 \]`
et `\(\sqrt{2} > 0\)`. C'est un nombre
réel irrationnel.

Le nombre $\sqrt{2}$ est défini 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 fortement¹ 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 autorisés ;
- 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 (comme l'alignement des exposants — un exemple est donné en exercices) et corrige de légers bugs.

4.1.2 Outils de base

```
^{\langle exposant \rangle} ^{\langle symbole unique \rangle}
_{\langle exposant \rangle} _{\langle symbole unique \rangle}
```

Les deux caractères réservés `^` et `_` deviennent utilisables en mode math, où 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 le même symbole, mais pas deux indices ou deux exposants. On peut par ailleurs utiliser toutes les constructions voulues (en particulier, des indices et exposants) dans un indice ou un exposant, comme le montre l'exemple 4.2.

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

EXEMPLE 4.2 — Indices, exposants, fractions et racines

```
\frac{\langle numérateur \rangle}{\langle dénominateur \rangle}
\sqrt[\langle indice \rangle]{\langle radicande \rangle}
```

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_{\langle borne basse \rangle}^{\langle borne haute \rangle} \prod_{\langle borne basse \rangle}^{\langle borne haute \rangle}
\int_{\langle borne basse \rangle}^{\langle borne haute \rangle} \iint \oint \dots
```

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 disponibles² 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.

<p>Posons $S = \sum_{i=1}^n u_n$. Ou bien</p> <p>Posons $S = \sum_{i=1}^n u_n$. Ou bien</p> <p><code>[S = \sum_{i=1}^n u_n \]</code> On a aussi</p> <p><code>(\iint f \neq \int\int f)</code>.</p>	$S = \sum_{i=1}^n u_n$
---	------------------------

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 quelques 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 de 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	<code>\$abc\$</code>
romaine	dx	<code>\$\$\mathrm{d}x\$</code>
grasse droite	$\mathbf{C} \supset \mathbf{R}$	<code>\$\$\mathbf{C} \ \supset \ \mathbf{R}\$</code>
grasse	\mathbf{k}	<code>\$\$\boldsymbol{k}\$</code>
fraktur	$\mathfrak{P} \mid \mathfrak{p}$	<code>\$\$\mathfrak{P} \ \mid \ \mathfrak{p}\$</code>
calligraphique	\mathcal{A}	<code>\$\$\mathcal{A}\$</code>
anglaise	\mathscr{C}	<code>\$\$\mathscr{C}\$</code>
ajourée	$\mathbb{N} \subset \mathbb{Z}$	<code>\$\$\mathbb{N} \ \subset \ \mathbb{Z}\$</code>

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 mathématique⁸ 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 moyen⁹ 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}`

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.

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, 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} \]
```

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

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*{\langle \rangle}{\langle \rangle}
```

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 se placer en-dessous et au-dessus. L'exemple 4.6 illustre l'usage de de cette version étoilée.

```

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

```

$$\operatorname{colim}_{x \rightarrow \infty} f(x) = \operatorname{colim}_{x \rightarrow \infty} f(x)$$

EXEMPLE 4.6 — Opérateur et placement des bornes

Commande	Nom	Effet
<code>\quad</code>	double cadratin	$x \equiv y \quad [\pi]$
<code>\quad</code>	cadratin	$x \equiv y \quad [\pi]$
<code>\</code>	inter-mot	$x \equiv y [\pi]$
<code>\;</code>	épaisse	$x \equiv y [\pi]$
<code>\:</code>	moyenne	$x \equiv y [\pi]$
<code>\,</code>	fine	$x \equiv y [\pi]$
<code>\quad</code>	<i>pas</i> d'espace	$x \equiv y[\pi]$
<code>\!</code>	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 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 ne sont en fait pas souvent 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` et `\rangle` (`\{`, `\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](#).

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 commandes `\big`, `\Big`, `\bigg`, `\Bigg`. Il n'y a alors plus aucun contrôle d'équilibrage contrairement à ce qui se passe avec `\left` et `\right`.

```

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

```

EXEMPLE 4.7 — Taille automatique des délimiteurs

Résultat	Code
$x \xrightarrow{f} y$	<code>\$x \stackrel{f}{\longmapsto} y\$</code>
$X_n \xrightarrow[n \rightarrow \infty]{L_2} X$	<code>\$X_n \xrightarrow[n \rightarrow \infty]{L_2} X\$</code>
$\prod_c^d a^b$	<code>\$\$\prod_c^d a^b\$</code>
$n < \overset{*}{n}$	<code>\$\$\underset{*}{n} < \overset{*}{n}\$</code>
$\binom{n}{p}$	<code>\$\$\binom{n}{p}\$</code>
$\sum_{\substack{i \in I \\ j \in J}}$	<code>\$\$\sum_{\substack{i \in I \\ j \in J}}\$</code>
tM	<code>\$\$\!{}^tM\$</code>
$x^n = \underbrace{x \cdots x}_n$	<code>\$\$x^n = \underbrace{x \cdots x}_n\$</code>

TABLE 4.3 — Petites constructions mathématiques

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} \quad \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{*}{\langle nom env \rangle}[\langle numéroté avec \rangle]{\langle titre \rangle}[\langle numéroté dans \rangle]
```

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 arguments obligatoires : `\langle nom env \rangle` est le nom de l'environnement tel qu'il apparaîtra dans la source. Il ne doit pas correspondre au nom d'un environnement ou d'une commande¹⁰ qui existe. `\langle titre \rangle` 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 le nom du premier comme argument optionnel `\langle numéroté avec \rangle`.

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 `\langle numéroté dans \rangle`. 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.

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

<pre> \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} </pre>	<p>Théorème 4.4.1 (de Fermat). <i>Cubum autem in dous cubos,...</i></p> <p><i>Démonstration.</i> La marge est trop étroite. □</p> <p>Exercice 4.4.2. <i>La changer avec \geometry.</i></p> <p>Question de cours. <i>Qu'est-ce qui est trop étroit ?</i></p>
--	---

EXEMPLE 4.9 — Environnements de type 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.

4.5 Documentation

Les possibilités mathématiques de \LaTeX sont immenses, les mathématiciens en étant les principaux utilisateurs. Pour ce chapitre comme les autres, il n'est pas possible de tout présenter. Je rappelle donc ici les principales sources d'information 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 documents 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

Rappelons l'allure d'un fichier source \LaTeX pour les documents courants.

```
\documentclass[a4paper]{article} % ou report ou book ou ...
\usepackage[latin1]{inputenc} % ou utf8 ou macroman
\usepackage[T1]{fontenc} % accents dans le pdf
% \usepackage{textcomp} % symboles complémentaires (euro)
% \usepackage{amsmath, amssymb} % minimum pour les maths
% \usepackage{xspace} % protection de certains espaces
<modules supplémentaires>
\usepackage[frenchb]{babel} % titres en français, typo française
% \usepackage{hyperref} % liens hypertexte

<commandes pour régler la mise en pages>
<définitions de commandes, environnements, etc.>

% \title{<titre>}
% \author{<nom de l'auteur>}
% \date{<date>} % \today pour la date du jour

\begin{document}
% \maketitle
<reste du corps du document>
\end{document}
```

SOURCE 5.1 — Fichier source de base

Ce source de base est disponible dans une **version prête à l'emploi**¹ que je vous recommande de télécharger et d'avoir toujours à portée de main, pour l'utiliser comme point de départ de tous vos documents.

Les lignes commentées dans ce source ne sont pas obligatoires, mais sont assez souvent utiles. Vous n'avez qu'à les dé-commenter (retirer le caractère % en début de ligne) et éventuellement les modifier pour les utiliser. Les autres lignes doivent figurer dans tous vos documents, éventuellement adaptées (vous pouvez changer les options de la classe ou de modules par exemple).

Une ligne que vous pouvez avoir besoin de modifier est l'appel à `inputenc` : en effet, la bonne option ici dépend du réglage de votre éditeur. Rappelons les principales options disponibles.

1. <http://people.math.jussieu.fr/~mpg/lm204/files/doc-source-base.tex>

- Avec TeXnicCenter (sous Windows, donc), aucun réglage n'est possible. L'encodage utilisé par cet éditeur est `cp1252`. Vous pouvez donc utiliser `cp1252` comme option d'`inputenc`. En fait, il se trouve que `cp1252` est une légère extension de `latin1`, qui est un encodage beaucoup plus standard. Je vous recommande donc d'utiliser plutôt `latin1` : certains caractères ne seront plus disponibles (par exemple, « € » et « œ ») mais en revanche votre source sera plus facilement utilisable sous d'autres plateformes que Windows.
- Avec la plupart des autres éditeurs, vous pouvez choisir l'encodage utilisé. Dans ce cas, tout ce qui compte, c'est que le réglage de votre éditeur corresponde à ce que vous indiquez en option à `inputenc`. Les deux encodages recommandés sont `latin1` (qui peut s'appeler ISO-8859-1 dans le menu de votre éditeur) ou éventuellement `utf8` (plus universel, mais pas encore accepté par 100% des modules \LaTeX).

Les questions d'encodages sont techniques et peu intéressantes pour les personnes n'utilisant que des langues d'Europe de l'Ouest, comme le français ou l'anglais. Je vous recommande d'évacuer ce problème une fois pour toutes en choisissant un encodage commun à tous vos documents (par exemple `latin1`), en réglant votre éditeur pour qu'il utilise tout le temps cet encodage, et en utilisant toujours `inputenc` avec l'option choisie.

Si vous travaillez en binôme avec un camarade sur un document, ou que vous travaillez sur un même document depuis plusieurs ordinateurs différents (par exemple, avec TeXnicCenter sur les sessions Windows de l'UTES, et chez vous avec TeXshop sous OS X), prenez soin d'utiliser le même réglage d'encodage partout. Dans le cas contraire, des problèmes peuvent survenir, allant jusqu'à rendre votre source totalement inutilisable. Donc, soyez prudents.

5.1.2 Messages d'erreur courants

Vous l'aurez constaté, la compilation d'un document ne se déroule pas toujours sans accroc. En \LaTeX comme avec tous les autres programmes informatiques, il est important de lire et savoir interpréter les messages d'erreur pour résoudre les problèmes. En ce qui concerne la lecture, je ne peux rien faire d'autre² que vous exhorter à les lire *vraiment*, voici en revanche quelques éléments pour les interpréter.

La première chose à savoir, c'est qu'en \LaTeX , une erreur peut en entraîner bien d'autres (et plus rarement, en cacher une autre). Quand votre document produit plusieurs erreurs, je vous recommande fortement de procéder ainsi : lisez seulement le *premier* message d'erreur, corrigez l'erreur, recompilez votre document et itérez. Une seule erreur peut parfois tellement désorienter \LaTeX qu'il produira des centaines d'erreurs dans la suite du document : si vous êtes dans ce cas, pas de panique, traitez les problèmes un par un dans l'ordre.

Le deuxième point important, surtout quand vos documents commencent à être longs, c'est de localiser l'erreur. Le message d'erreur indique toujours la ligne où \LaTeX a détecté l'erreur : comme \LaTeX ne voit pas dans le futur, l'erreur peut se trouver exactement à cette ligne, parfois un peu avant (le temps que \LaTeX s'en rende compte), mais jamais après. Par ailleurs, le message d'erreur comporte souvent une ligne de contexte, par exemple

```
1.16 C'est clair : \textcolour
                        {blue}{bleu} clair
```

2. Si les messages d'erreur n'apparaissent pas clairement avec votre éditeur (c'est souvent un problème avec TeX-maker), je peux bien sûr vous expliquer *comment* les lire.

On voit ici que l'erreur est survenue ligne 16, mais cette ligne pouvant être très longue, il est important de lire la suite. \LaTeX montre le contexte de l'erreur, et place un saut de ligne à l'endroit précis où il s'est arrêté : l'erreur est en général juste avant. Ici, c'est en effet une faute de frappe sur `\textcolour`, dont l'orthographe correcte est `\textcolor`.

Les deux points précédents sont valables pour tous les types d'erreur : traiter les erreurs une par une dans l'ordre, et utiliser les informations données par \LaTeX pour les localiser. Pour aller plus loin, il faut utiliser le message d'erreur lui-même. Ces messages sont en anglais³, et sont parfois très précis, d'autres fois plus difficiles à interpréter (typiquement, `Missing \endcsname`). Voici quelques messages courants qu'il faut absolument savoir reconnaître.

- `Undefined control sequence` : commande non définie. En général ceci signifie soit que vous avez fait une faute de frappe dans le nom de la commande (ex. : `\colour` pour `\color`), soit que le nom est correct mais que vous avez oublié de charger le module qui fournit cette commande (ex. : `\usepackage{xcolor}` pour la commande `\color`).
- `! LaTeX Error: Environment centre undefined` : environnement non défini. Pareil que le point précédent, mais pour un environnement.
- `! Missing $ inserted` : `$` manquant (rajouté par \LaTeX). Soit vous avez oublié de passer en mode mathématique pour une construction autorisée seulement en mode math, soit vous avez oublié de sortir du mode mathématique pour une construction autorisée seulement en mode texte. Variante courante : vous ne voulez pas faire des math, mais vous avez oublié que `^` et `_` sont des caractères réservés (voir [section 2.1.1](#)).
- `Runaway argument?` : argument sans fin ? Probablement, vous avez oublié une accolade fermante, et \LaTeX a donc l'impression que l'argument de la commande correspondante ne se termine jamais. Dans ce cas, il vous dit de quelle commande il s'agit ; par exemple :

`! File ended while scanning use of \textbf .`

Pour une liste complète de toutes les erreurs produites par \LaTeX ainsi que les modules les plus courants, expliquées en français, vous pouvez consulter l'[annexe B du \$\LaTeX\$ Companion](#)⁴, qui n'a pas été intégrée à l'édition papier faute de place, mais est disponible gratuitement en ligne. C'est un document qu'il est utile de garder à portée de main (de préférence en version PDF : il est ainsi plus facile de trouver un message précis grâce à la fonction « rechercher » de votre lecteur PDF).

Je rappelle que tous vos documents doivent compiler sans aucune erreur. \LaTeX produit parfois aussi des avertissements (*warning*) qui sont parfois importants, parfois pas. Pour simplifier, nous considérerons que tous les avertissements liés aux polices (*font warning*) peuvent être ignorés ; pour les autres, me consulter.

Enfin, le dernier type de problème que \LaTeX peut rapporter concerne les débordement (ou défauts de remplissage) de boîte : `overfull` ou `underfull \hbox` ou `\vbox`. Ces problèmes sont également à traiter ; nous les aborderons en [section 9.4.6](#).

5.2 Définitions

Vous avez peut-être remarqué une subtilité dans certains messages d'erreurs évoqués ci-dessus : \LaTeX dit qu'une commande « n'est pas définie », et pas qu'elle « n'existe pas ». La raison est simple :

3. Pour être précis, en américain, comme la plupart des commandes de \LaTeX . En anglais britannique, on aurait la commande `\colour`, la couleur `grey` et non `gray`, l'environnement `centre`, etc.

4. <http://www.latex-project.org/guides/lc2fr-apb.pdf>

la liste des commandes \LaTeX existante n'est pas figée, on peut l'étendre en chargeant des modules, mais surtout, on peut définir soi-même de nouvelles commandes.

Vous savez déjà le faire dans certains cas particulier : par exemple, vous savez déclarer des nouvelles commandes pour opérateurs mathématiques avec `\DeclareMathOperator`, ou des nouveaux environnements numérotés avec `\newtheorem`. Nous allons ici voir les techniques générales de définition de commandes et d'environnements.

Cette possibilité est une des plus intéressantes de \LaTeX . Elle permet d'automatiser certaines constructions fastidieuses, et d'homogénéiser le rendu du document en séparant son fond (la structure) de sa forme (l'aspect visuel) : idéalement, le corps du document ne doit contenir aucune commande relative à la forme ; celles-ci ne devraient apparaître que dans la définition de commandes personnelles. Ceci est surtout valable pour les documents de taille importante.

5.2.1 Commandes simples

```
\newcommand\<nom>\{<définition>\}
```

Les commandes les plus simples à définir sont celles qui ne prennent pas d'argument. Elles se comportent alors comme un raccourci, soit pour du texte plus long à saisir, soit pour du texte ou des commandes qui reviennent souvent et qu'on veut être sûr d'écrire tout le temps de la même manière.

<pre>\newcommand\sev{sous-espace vectoriel} \newcommand\defini{\textit} Une \defini{base} est une famille libre et génératrice. Une famille est \defini{libre} si\dots ; elle est \defini{génératrice} si\dots On appelle \defini{\sev} un sous-ensemble qui est aussi un espace vectoriel. Un sous-ensemble est un \sev{} si et seulement si\dots</pre>	<p>Une <i>base</i> est une famille libre et génératrice. Une famille est <i>libre</i> si... ; elle est <i>génératrice</i> si... On appelle <i>sous-espace vectoriel</i> un sous-ensemble qui est aussi un espace vectoriel. Un sous-ensemble est un sous-espace vectoriel si et seulement si...</p>
--	---

EXEMPLE 5.1 — Définitions simples

Par exemple, si à chaque fois que vous *définissez* un mot, vous voulez l'écrire en italique, vous pouvez définir une commande `\defini` que vous utiliserez dans le texte plutôt que `\textit`. Ceci présente (au moins) deux avantages⁵ :

homogénéité : vous êtes certains en procédant ainsi que toutes les définitions auront la même apparence, vous ne risquez pas d'utiliser une fois de l'italique, une fois du gras, par erreur. De façon générale,

abstraction : en séparant le *fond* (ce terme est défini ici) de la *forme* (ce mot doit être en italique), vous obtenez plus de souplesse pour changer cette dernière. Si par exemple vous décidez que l'italique n'est pas assez visible et que le gras est plus approprié, il vous suffit de changer la définition de `\defini`, pas besoin de remplacer la moitié de vos `\textit` par des `\textbf`.

5. Avantages qui devraient vous être familiers si vous avez déjà suivi des cours de programmation, où l'on poursuit les mêmes buts (entre autres).

Pour les commandes utilisées en mode texte qui remplacent une suite de mots, il faut faire attention au fait que les espaces qui suivent sont avalées. Observez par exemple les accolades après la deuxième utilisation de `\sev` dans l'exemple 5.1 qui a pour but de protéger l'espace suivant. On a beau être prévenu, il est trop facile d'oublier ceci, avec pour conséquence des mots attachés dans le document compilé.

Heureusement, une solution existe : il suffit, après avoir chargé le module `xspace`, d'ajouter la commande `\xspace` à la fin de la définition de sa commande. Ainsi, l'espace avalé sera automatiquement rajouté, sauf si la commande est suivie d'une ponctuation, comme illustré par l'exemple 5.2.

<code>\usepackage{xspace}</code>	Soit F un sous-espace vectoriel
<code>\newcommand\sev{sous-espace vectoriel\xspace}</code>	d'un espace vectoriel. Une partie
Soit F un <code>\sev</code> d'un espace vectoriel. Une	stable par les opérations est un
partie stable par les opérations est un <code>\sev</code> .	sous-espace vectoriel.

EXEMPLE 5.2 — Utilisation de `\xspace`

5.2.2 Commandes avec arguments

```
\newcommand*\nom[n]{définition avec #1 à #9}
\newcommand\nom[n]{définition avec #1 à #9}
```

TeX permet également de définir des commandes acceptant entre un et neuf arguments. La syntaxe est la même que pour des commandes simples, sauf qu'entre le nom et la définition, on indique le nombre d'arguments entre crochets. On peut ensuite utiliser les arguments zéro, une ou plusieurs fois dans l'ordre qu'on veut dans la définition, avec `#1` pour le premier, `#2` pour le deuxième, etc.

<code>\newcommand\guill[1]{\og #1 \fg}</code>	
<code>\newcommand\boldcolor[2]{\textbf{\textcolor{#2}{#1}}}</code>	Le « laser » pourrait
<code>\newcommand\vecma[2]{(x_{#1}, \dots, x_{#2})}</code>	tout brûler . Attention,
<code>\newcommand\vecbien[2]{(x_{#1}, \dots, x_{#2})}</code>	$(x_1, \dots, x_{42}) \neq$
Le <code>\guill{laser}</code> pourrait tout <code>\boldcolor{brûler}{red}</code> .	$(x_1, \dots, x_{42})!$
Attention, <code>\vecma{1}{42} \neq \vecbien{1}{42}</code> !	

EXEMPLE 5.3 — Commandes avec arguments

Parfois, il faut prendre garde dans la définition à entourer chaque argument d'une paire d'accolades, sinon, TeX le découpera impitoyablement en morceaux au moment du remplacement, comme le montre la comparaison entre `\vecma` et `\vecbien` dans l'exemple 5.3.

Vous aurez remarqué que pour définir des commandes à argument on dispose de deux variantes : `\newcommand*` et `\newcommand`. C'est la première qui est recommandée dans la plupart des cas : elle définira une commande à arguments courts, c'est-à-dire faisant moins d'un paragraphe (comme c'est le cas de la commande `\textbf` par exemple). C'est souvent utile pour détecter des erreurs : si vous oubliez l'accolade fermante d'un argument court, l'erreur sera détectée dès la fin du paragraphe plutôt qu'à la fin du document.

5.2.3 Commandes avec arguments optionnels

```
\newcommand{<*>\langle nom \rangle [ <n \rangle ] [ <valeur par défaut \rangle ] { <définition \rangle }
```

Dans certains cas, un des arguments de la commande est presque tout le temps le même. Il est alors intéressant de rendre cet argument optionnel : c'est-à-dire qu'on aura pas besoin de l'indiquer quand il a sa valeur « normale », mais qu'on peut si on veut le changer en l'indiquant entre crochets.

Vous connaissez déjà des exemples de commandes pré-définies admettant un argument optionnel, comme `\sqrt` : on a la racine carrée avec `\sqrt{x}` (cas le plus courant), mais on peut aussi écrire une racine cubique avec `\sqrt[3]{x}`.

```
\newcommand\fort[2][black]{%
  \textbf{\textcolor{#1}{#2}}}
Un \fort{mot} vraiment très
\fort[red]{important}.
```

Un **mot** vraiment très **important**.

EXEMPLE 5.4 — Commande avec argument optionnel

Vous pouvez obtenir le même résultat avec vos propres commandes. Pour rendre optionnel le premier argument, il suffit de lui indiquer une valeur par défaut entre crochets juste avant la définition de la commande. Attention, l'argument optionnel est toujours `#1`, et le nombre d'arguments à indiquer est le nombre *total*, en comptant l'argument optionnel. Ainsi, dans l'exemple 5.4, la commande `\fort` est déclarée comme ayant deux arguments, même si on peut l'utiliser avec un seul argument.

Avec `\newcommand`, on ne peut avoir qu'un argument optionnel, qui doit toujours être le premier. Si on veut définir des commandes ayant plusieurs arguments optionnels, on peut utiliser le module `xargs`⁶, dont la documentation est disponible en français (`texdoc xargs-fr`, `mthelp xargs-fr` ou [en ligne](#)⁷).

5.2.4 Environnements

```
\newenvironment{<nom \rangle}{<code de début \rangle}{<code de fin \rangle}
```

Vous pouvez aussi définir vos propres environnements. Rappelons que la partie entre le `\begin` et le `\end` s'appelle le *corps* de l'environnement. La principale différence d'usage entre un environnement et une commande est que le corps de l'environnement est généralement plus long que l'argument d'une commande : ainsi, on utilise une commande pour mettre quelques mots en italique, mais un environnement s'il s'agit de mettre en exergue tout un bloc de texte.

Remarquez qu'il n'y a pas de `#1` dans la définition : le corps de l'environnement est placé automatiquement entre le `<code de début \rangle` et le `<code de fin \rangle`. Une façon équivalente de voir les choses est de dire que le `\begin{<nom \rangle}` est remplacé par `<code de début \rangle` et `\end{<nom \rangle}` par `<code de fin \rangle`. C'est donc un peu comme si on avait défini deux commandes.

En fait, un environnement, c'est un peu plus que deux commandes : en bonus, toutes les modifications de police ou autre effectuées par le `<code de début \rangle` sont automatiquement oubliées à

6. <http://ctan.org/pkg/xargs>

7. <http://tug.ctan.org/get/macros/latex/contrib/xargs/xargs-fr.pdf>

<pre> \newenvironment{petit}{% \begin{quote}\small}\end{quote}} \newenvironment{remarque}{% \noindent\textbf{Remarque.}\itshape}{} \begin{remarque} Une remarque qui pourrait bien faire plusieurs paragraphes.\end{remarque} \begin{petit} Du texte petit sur une plus petite largeur. \end{petit} Du texte à nouveau normal ici pour voir. </pre>	<p>Remarque. <i>Une remarque qui pourrait bien faire plusieurs paragraphes.</i></p> <p>Du texte petit sur une plus petite largeur.</p> <p>Du texte à nouveau normal ici pour voir.</p>
---	---

EXEMPLE 5.5 — Définitions d'environnements

la fin de l'environnement. Il est conseillé d'utiliser cet effet : il n'est ainsi pas rare d'avoir un code de fin vide, comme pour l'environnement `remarque` de l'exemple 5.5. Il n'y a pas besoin d'utiliser `\upshape` ni `\normalfont` à la fin pour annuler le `\itshape` du début.

L'exemple appelle par ailleurs une remarque : il est souvent commode, pour des raisons de lisibilité, d'écrire les longues définitions sur plusieurs lignes. Il faut alors faire attention au fait que chaque saut de ligne équivaut à un espace : on risque d'introduire des espaces dans la définition sans le vouloir. Pour éviter cela, il suffit de placer un `%` en fin de ligne, sans espace devant, à chaque fois qu'on change de ligne uniquement pour des raisons de présentation⁸.

Par ailleurs, je rappelle que le code des exemples est souvent très mal présenté, pour respecter des contraintes de place dans le polycopié. Je vous encourage à consulter plutôt les corrigés des exercices pour des exemples de code source présenté correctement.

5.2.5 Redéfinitions

```

\renewcommand\langle nom \rangle { \langle définition \rangle }
\renewcommand\langle * \rangle \langle nom \rangle [ \langle n \rangle ] { \langle définition \rangle }
\renewcommand\langle * \rangle \langle nom \rangle [ \langle n \rangle ] [ \langle valeur par défaut \rangle ] { \langle définition \rangle }
\renewenvironment { \langle nom \rangle } { \langle code début \rangle } { \langle code fin \rangle }

```

En définissant une commande ou un environnement avec `\newcommand` ou `\newenvironment`, \TeX vérifie que le nom qu'on souhaite utiliser est bien disponible, et émet un message d'erreur sinon. La plupart du temps, c'est une très bonne chose, car cela vous empêche de re-définir par erreur une des commandes plus ou moins internes de \TeX , ce qui pourrait avoir des conséquences catastrophiques.

Il arrive parfois cependant qu'on veuille redéfinir une commande existante, par exemple parce que la définition normale ne nous convient pas. On peut alors précéder comme pour une définition normale, en utilisant `\renewcommand` ou `\renewenvironment` selon qu'il s'agit d'une commande ou d'un environnement. Ces commandes ont exactement la même syntaxe que leurs analogues sans `re`. Elles ont le même effet, sauf que cette fois elle émettent un message d'erreur si le nom n'était pas déjà défini.

Il faut être prudent en redéfinissant des commandes ; par exemple il faudrait toujours que la commande, une fois redéfinie, garde presque le même sens qu'avant. Par exemple,

8. C'est ce qu'on appelle souvent « commenter ses fins de lignes ».

```
\renewcommand\le{\leqslant}
```

est une re-définition acceptable car la nouvelle commande et l'ancienne désignent deux variantes d'un même symbole (inférieur ou égal). Elle évite ainsi de mélanger par mégarde \leq et \leqslant dans le texte, ce qui ne serait pas très joli.

5.2.6 Couleurs

```
\definecolor{<nom>}{<modèle>}{<valeur>}
\colorlet{<nom>}{<couleur existante>}
```

On peut définir des nouveaux noms de couleur (toujours avec `xcolor`) de deux façons différentes : soit en donnant la valeur numérique de la couleur dans un modèle de couleurs (HTML, RGB, etc.) soit par rapport à une couleur existante, qui peut être un mélange. Par exemple,

```
\definecolor{bleu1}{HTML}{000080}
\colorlet{bleu2}{blue!50!black}
```

sont deux façons de définir un bleu foncé. La première⁹ dit que `bleu1` sera défini par des niveaux rouge et de vert nuls (les 4 premiers 0) et un niveau de bleu de 128 (80 en hexadécimal) sur 256. La deuxième dit que `bleu2` sera un mélange de bleu et de noir à 50%.

Les noms ainsi définis sont utilisables de la même manière que les noms de couleurs pré-définis. Il est souvent de bon goût de n'utiliser qu'un nombre réduit de couleurs dans l'ensemble d'un document (par exemple, ce poly n'utilise que trois couleurs, en dehors des exemples). On a alors intérêt à les définir dans le préambule, plutôt que d'écrire la valeur de chaque couleur à chaque fois, avec les risques d'erreur et la difficulté de changement que cela implique.

9. Qui utilise le modèle de couleurs de l'HTML, le langage utilisé pour écrire les pages web.

6 Figures

6.1 Inclusion d'images

6.1.1 La question des formats

Il existe dans le monde informatique plusieurs façons de stocker des images dans des fichiers, c'est-à-dire plusieurs *formats graphiques* généralement repérés par l'extension du fichier : bmp, gif, tiff, jpg, png, ps, pdf, svg pour ne citer que les plus connus.

D'un point de vue technique, il se divisent en deux familles : les formats matriciels (*bitmap*) et les formats vectoriels. Dans le premier cas, l'image est représentée par une grille de points ayant chacun une valeur de couleur, c'est-à-dire une matrice de nombres. Dans le deuxième, chaque partie de l'image est représentée par les équations des courbes qui la délimitent.

Les images stockées sous forme vectorielle seront *in fine* mises sous forme matricielles (*rasterisées*) au moment de l'impression ou de l'affichage à l'écran, avec une résolution (la finesse de la grille) adaptée. Cette adaptation de la résolution au média fait la supériorité des images vectorielles sur les images matricielles : une image vectorielle qui rend bien à l'écran rendra ¹ bien à l'impression, ce n'est pas le cas des images matricielles. Il est donc recommandé d'utiliser des formats vectoriels aussi souvent que possible, et le reste du temps de veiller à choisir une résolution suffisante pour les documents destinés à être imprimés ².

Assez de généralités, passons à la pratique avec \LaTeX . Ce dernier ne reconnaît actuellement pas tous les formats graphiques existants, il faut donc choisir le format des images à inclure dans une liste limitée. De plus, \LaTeX offre plusieurs *modes de compilation* qui ne permettent pas d'inclure les mêmes types de fichiers. La [table 6.1](#) liste les formats disponibles dans les deux modes de compilation les plus courants.



	PDF direct	Passage par PostScript
Aperçu TeXnicCenter	LaTeX => PDF 	LaTeX => PS => PDF 
Formats matriciels	png, jpg	—
Formats vectoriels	pdf	ps, eps

TABLE 6.1 — Modes de compilation et formats graphiques

Un mot sur les différents modes de compilation : le mode que je recommande en général est PDF direct ; les différents exemples et exercices supposent que vous utilisez ce mode compilation. Il fonctionne bien et est souvent le plus simple. Il peut cependant arriver que l'usage de PostScript

1. Notez que la réciproque de cette implication est fautive : certaines images vectorielles qui rendent bien à l'impression rendront moins bien à l'écran : en effet, celui-ci n'est pas assez précis pour rendre correctement les détails qui sont alors déformés de façon plus ou moins élégante.

2. Les documents destinés à l'écran peuvent se contenter d'une résolution inférieure : un écran actuel offre environ 100ppp (points par pouces) de résolution, contre au moins 600ppp pour une imprimante.

comme format intermédiaire soit indispensable, par exemple pour si vous incluez beaucoup de graphiques provenant d'une application (Maple, Mathematica, etc.) qui n'exporte qu'en EPS et pas en PDF, ou pour utiliser des outils de dessin avancés en \LaTeX comme PStricks, que nous n'étudierons pas dans ce cours.

Par ailleurs, même si \LaTeX lui-même ne peut pas utiliser tous les formats existants, il est toujours possible de convertir les images entre différents formats en utilisant des logiciels spécialisés. Par exemple, les deux logiciels **Gimp**³ (formats matriciels) et **Inkscape**⁴ (formats vectoriels) permettent d'effectuer des conversions facilement en ouvrant une image dans un format et en la sauvegardant dans un autre. (Ils sont par ailleurs gratuits et fonctionnent sous Windows, Mac OS X et Linux.)

6.1.2 Inclusion simple

```
\usepackage{graphicx}
\includegraphics[\langle options \rangle]{\langle fichier \rangle}
```

Pour pouvoir inclure des images avec \LaTeX , il faut charger le module `graphicx` dans le préambule. Attention, un autre module du nom de `graphics` existe : il est plus ancien et fournit exactement les mêmes possibilités, mais avec une syntaxe moins agréable. Nous utiliserons donc toujours `graphicx`.

La principale commande fournie par ce module est `\includegraphics` : comme son nom l'indique, elle permet d'inclure des fichiers d'images, et elle ne fait que ça (en particulier, il faut utiliser d'autres commandes pour déterminer le placement de l'image, voir [section 6.2](#)) mais elle le fait bien : la prochaine section liste les options disponibles, mais parlons déjà du nom du fichier.

On peut au choix donner le nom complet du fichier, comme `image.png` ou bien omettre l'extension du moment qu'elle est reconnue par \LaTeX et donne juste le nom de base, comme `image`. Par ailleurs, les fichiers d'images doivent en principe se trouver dans le même répertoire que le fichier `.tex` que vous compilez pour que \LaTeX les trouve.

En fait, on peut également placer les images dans un autre répertoire, ce qui est pratique pour les gros documents utilisant de nombreuses images. On doit alors indiquer le chemin des images de la façon suivante : si les images sont dans un sous-répertoire nommé `img` de l'endroit où se trouve le fichier `.tex`, on indiquera `img/image.png` comme argument de `\includegraphics`. Attention : le séparateur de répertoires est `/` et non `\`, même sous Windows.

Par ailleurs, pour accéder à un répertoire parent, on utilise « `..` » : par exemple, si un répertoire `lm204` contient un sous-répertoire `poly` avec les sources d'un polycopié⁵ et un sous-répertoire `img` avec toutes les images utilisées dans les documents, on utilisera une commande comme

```
\includegraphics{../img/lion.jpg}
```

pour insérer l'image `lion.jpg` dans le polycopié.

Enfin, on peut aussi indiquer le chemin complet du fichier, commençant par exemple par `C:\` sous Windows et par `/` partout ailleurs. Ceci n'est souvent pas une bonne idée, car si jamais vous déplacez votre document, l'échangez avec un collègue⁶, il faudra alors changer tous les chemins :

3. <http://www.gimp.org/>

4. <http://www.inkscape.org/?lang=fr>

5. Exemple purement fictif, bien sûr. Toute ressemblance avec des personnes ou situations ayant réellement existé...

6. ou le montrez à votre enseignant...

préférez donc des chemins relatifs comme ci-dessus. Dans la plupart des cas, laisser les images dans le même répertoire que le source \LaTeX est la meilleure solution.

Trois mots d'avertissement pour finir : \LaTeX gère malheureusement assez mal les espaces dans les noms de fichiers, il faut donc mieux les éviter. De plus, utiliser des caractères accentués dans les noms de fichiers est souvent source de problèmes (pas seulement pour \LaTeX). Enfin, il faut prendre garde aux majuscules : sous Windows, on peut appeler `fiChIeR` un fichier qui s'appelle en réalité `Fichier` sans rencontrer de problèmes, mais le document ne compilera alors pas sur un autre système (Mac OS X, Linux). Ma recommandation personnelle est de n'utiliser que des lettres minuscules (non accentuées) et des traits d'union dans les noms de fichier (p. ex. `mode-compile.jpg`) pour éviter tout problème de ce côté.

6.1.3 Options d'inclusion

On peut inclure une image sans utiliser aucune option, comme indiqué ci-dessus, mais le plus souvent ou voudra contrôler au moins ses dimensions, et parfois tourner l'image ou la recadrer. Pour ceci, `\includegraphics` accepte plusieurs options sous la forme $\langle clé \rangle = \langle valeur \rangle$, que l'on peut regrouper en trois grandes familles.

Le premier groupe d'options concerne la taille : si l'on n'indique rien, l'image sera insérée à sa « taille naturelle », expression qui ne veut en général pas dire grand chose. On peut spécifier une échelle, qui est un nombre décimal, par rapport à cette taille naturelle, avec l'option `scale` : par exemple `scale=0.5` pour réduire l'image de moitié.

Il est souvent plus utile de donner les dimensions finales que l'on souhaite. Pour cela, on dispose des options `height` (hauteur) et `width` (largeur) ; on peut en fait n'en spécifier qu'une et \LaTeX calculera automatiquement l'autre de façon à ne pas déformer l'image. On peut utiliser des dimensions absolue, p. ex. en centimètres, ou des dimensions prédéfinies par \LaTeX , comme `\linewidth` qui représente la largeur d'une ligne de texte.

Par ailleurs, signalons que selon l'image et son format, elle peut comporter une *ligne de base* qui ne sera pas forcément en bas de l'image visuellement. Pensez par exemple à une image représentant la lettre « p » : la ligne de base⁷ se situerait en bas de la boucle du « p » et pas en bas de la lettre, pour assurer un bon alignement avec les autres lettres. Dans un tel cas, `height` ne représente que la hauteur de l'image au-dessus de la ligne de base (la « partie émergée de l'iceberg ») et il faut utiliser `totalheight` si l'on veut parler de la « vraie » hauteur de l'image. En fait, il vaut mieux toujours utiliser `totalheight` à la place de `height`. Voici quelques exemples d'inclusion d'une image avec mise à l'échelle. Notez que dans la troisième ligne, l'image sera déformée si elle n'était pas carrée.

```
\includegraphics[scale=2]{truc.pdf}
\includegraphics[width=2cm]{truc.pdf}
\includegraphics[width=2cm, totalheight=2cm]{truc.pdf}
\includegraphics[width=\linewidth]{truc.pdf}
```

Le deuxième groupe d'options permet de tourner l'image. Comme en math, une rotation est définie par son angle et son origine, correspondant aux options `angle` et `origin`. L'angle est exprimé en degré, et l'origine est choisie parmi un ensemble de 12 points prédéfinis sur l'image et

7. Voir la [figure 9.1](#) et plus généralement la [section 9.4.1](#).

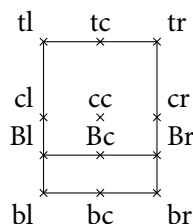
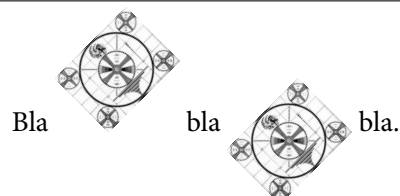


FIGURE 6.1 — Centres de rotation possibles

désignés par une suite de deux lettres. La première lettre désigne la hauteur de l'origine : **b** tout en bas, **B** sur la ligne de base (voir ci-dessus), **c** au centre et **t** en haut : la deuxième son emplacement horizontal : **l** à gauche, **c** au centre et **r** à droite, comme illustré sur la [figure 6.1](#).

On utilise toujours ensemble les options `angle` et `origin`, et on peut les combiner avec d'autres options comme sur l'[exemple 6.1](#).

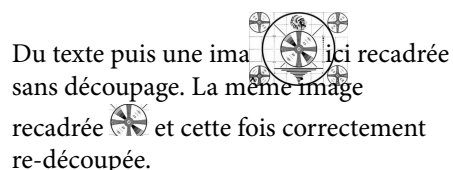
```
\usepackage{graphicx}
Bla \includegraphics[height=1cm,
angle=45, origin=bl] {mire-tele.jpg}
bla \includegraphics [height=1cm,
angle=45, origin=br]{mire-tele.jpg} bla.
```



EXEMPLE 6.1 — Inclusion de graphiques avec rotation

Enfin, il est possible de recadrer une image. On utilise pour cela `viewport=<x1 y1 x2 y2>`, qui a le sens suivant : recadrer l'image en utilisant de rectangle dont le coin inférieur droit a pour coordonnées $\langle x_1, y_1 \rangle$ et le coin supérieur gauche $\langle x_2, y_2 \rangle$. L'origine des coordonnées est le coin inférieur droit de l'image ; l'orientation positive est vers la droite et vers le haut ; on peut préciser une unité pour chaque coordonnée.

```
\usepackage{graphicx}
Du texte puis une image
\includegraphics[viewport=4cm 3cm 9cm 7cm,
width=0.5cm]{mire-tele} ici recadrée sans
découpage. La même image recadrée
\includegraphics[viewport=4cm 3cm 9cm 7cm,
clip, width=0.5cm]{mire-tele} et cette fois
correctement re-découpée.
```



EXEMPLE 6.2 — Recadrage et découpage d'image

Il faut de plus, dans la plupart des cas, utiliser l'option `clip` qui dit à \LaTeX de supprimer réellement la partie de l'image qui déborde du cadre ainsi défini : dans le cas contraire, il n'ajustera que les « dimensions logiques » de l'image sans couper le reste, et le résultat sera le plus souvent une superposition malencontreuse avec le texte autour, comme dans l'[exemple 6.2](#). Notez, dans ce même exemple, que les coordonnées du nouveau cadre sont plus grandes (9cm par exemple) que la taille finale (0.5cm) : elles sont calculées par rapport à la taille de l'image *avant* mise à l'échelle.

6.2 Placement

6.2.1 Simple

Par défaut, \LaTeX n'applique aucune règle particulière pour placer les images : elles apparaissent à l'endroit où la commande `\includegraphics` est placée ; par exemple au milieu de deux mots comme dans l'exemple 6.1, ce qui n'est souvent pas l'effet désiré.

Le plus simple pour inclure des images en dehors d'un paragraphe, c'est d'utiliser un environnement `center` qui contiendra uniquement l'image : ceci la centrera horizontalement, mais aussi placera un peu d'espace vertical avant et après. En fait, même pour une image occupant tout la largeur disponible (option `width=\linewidth`), il est souvent intéressant d'utiliser un environnement `center`, juste pour cet espacement vertical (et aussi pour supprimer le retrait de début de paragraphe).

6.2.2 Habillé par le texte

On peut au contraire ne pas vouloir séparer l'image du texte qui l'entoure, mais l'inclure au milieu d'un paragraphe avec le texte qui « coule » autour. C'est ce qu'on appelle une image *habillée* par du texte. Il existe plusieurs modules en \LaTeX qui fournissent ce genre de possibilité. Malheureusement, il n'y a pas un module unique proposant toutes les options possibles, et les syntaxes utilisées sont assez variées (et pas toujours très intuitives). De plus, ils ne fonctionnent pas en toutes circonstances : par exemple, près d'un titre de section, d'une liste, ou en mode deux colonnes, des problèmes⁸ peuvent survenir.

Voyons cependant ce qu'on peut faire dans des cas simples avec les deux⁹ principaux modules existants : `wrapfig`¹⁰ et `picinpar`¹¹.

```
\begin{wrapfigure}[<nb ligne>]{<lrío>}{<largeur>}[<débordement>]
  <figure>
\end{wrapfigure}
<texte>
```

Il est facile avec `wrapfig` de placer un figure dans l'angle en haut à gauche ou en haut à droite d'un paragraphe, comme illustré par l'exemple 6.3. Il faut pour cela mettre l'image dans un environnement `wrapfigure` et la texte qui doit couler autour *après* cet environnement. Dans sa forme la plus simple, l'environnement `wrapfigure` prend deux arguments (en plus de son contenu) : le premier est une lettre indiquant de quel côté l'image doit être placée : `l` (*left*) à gauche, `r` (*right*) à droite, `i` (*inside* à l'intérieur (à droite sur les pages de gauche et réciproquement), `o` (*outside*) à l'extérieur. Ces deux dernières options ne sont bien sûr utiles que si le document est en recto/verso (voir page 24).

Le deuxième argument de `wrapfigure` est l'espace horizontal à réserver à l'image. C'est à vous de prendre soin d'indiquer une dimension légèrement plus grande que la taille de l'image, comme

8. Par exemple, en rédigeant ce polycopié, j'ai du recourir à une astuce pour l'exemple 6.3 car `wrapfigure` ne fonctionne pas dans mon environnement d'exemple habituel, et j'ai mis longtemps à trouver un réglage satisfaisant pour l'exemple 6.4 car `window` ne fonctionne plus dès que la largeur de colonne est trop petite.

9. En fait, il y en a un troisième, `picins`, mais il n'est pas inclus dans \TeX Live pour des raisons de licence ; je n'en dirai donc pas plus à son sujet.

10. <http://ctan.org/pkg/wrapfig>

11. <http://ctan.org/pkg/picinpar>

```

|\usepackage{graphicx, wrapfig}
\begin{wrapfigure}{l}{2.4cm}
  \includegraphics[width=2cm]{lion.png}
\end{wrapfigure}
Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Ut purus elit,
vestibulum ut, placerat ac, adipiscing
vitae, felis. Curabitur dictum gravida
mauris. Nam arcu libero, nonummy eget,
consectetur id, vulputate a, \dots

```



Lorem ipsum dolor
 sit amet, consectetur
 adipiscing elit.
 Ut purus elit, vestibulum
 ut, placerat ac, adipiscing
 vitae, felis. Curabitur
 dictum gravida mauris.

Nam arcu libero, nonummy eget, consectetur
 id, vulputate a, ...

EXEMPLE 6.3 — Habillage dans le coin en haut à droite

dans l'exemple 6.3 : 2.4cm pour une image de 2cm, de façon à éviter que le texte soit « collé » sur l'image. La dimension verticale, en revanche, est calculée automatiquement. Vous pouvez cependant l'ajuster au besoin en indiquant le nombre de lignes qui doivent être raccourcies comme premier argument optionnel. Le dernier argument optionnel sert, lui, à faire déborder l'image dans la marge d'une longueur donnée.

Il est seulement possible, avec `wrapfig`, de placer l'image sur un des deux côtés, en haut. Pour la placer au milieu du texte, il faut recourir au module `picinpar` et à son environnement `window`.

```

\begin{window}[<lignes avant>,<lcr>,<{figure}>,<titre>]
  <texte>
\end{window}

```

Observez tout de suite la syntaxe très différente de celle de `wrapfigure` (et assez déroutante, je vous l'accorde) : la figure est le troisième élément d'une liste séparée par des virgules, et le texte qui l'entoure est le contenu de l'environnement. Deux pièges subtils vous guettent : il ne faut pas mettre d'espace après ni avant les virgules, et penser à entourer d'accolades la commande de la figure (L^AT_EX gère mal les imbrications de crochets [] carrés) comme dans l'exemple 6.4.

```

\begin{window}[1,c,%
  {\includegraphics[width=2em]{lion}},%
  \centering Lui]
  Un paragraphe au milieu duquel
  apparaît ici la chère mascotte de
  \LaTeX{} : un lion à lunettes. Rappelons
  que celle de \TeX{} (le programme sur
  lequel est basé \LaTeX{}) est un lion
  \emph{sans} lunettes mais en toge.
\end{window}

```

Un paragraphe au milieu duquel apparaît ici la chère mascotte de L^AT_EX : un lion à lunettes. Rappelons que celle de T_EX (le programme sur lequel est basé L^AT_EX) est un lion *sans* lunettes mais en toge.

EXEMPLE 6.4 — Image en fenêtre dans le texte

Les « arguments » de `window` sont d'abord le nombre de lignes complètes dans le paragraphe avant le début de la figure, puis l'emplacement de la fenêtre : `l` à gauche, `c` au centre et `r` à droite. Viennent ensuite la figure elle-même et un titre qui sera placé au-dessous.

6.2.3 Flottant

L'habillage par du texte convient bien aux figures de dimensions réduites. Les figures plus larges gagneront à être placées hors de tout paragraphe, dans un environnement `center`, comme expliqué précédemment. Dès que la hauteur de la figure devient importante, un nouveau problème se pose : les coupures de pages.

Imaginons en effet une page remplie environ à 75% au moment où survient une figure qui fait environ une demie-page de hauteur : il n'y a donc pas la place d'insérer la figure ici, et \LaTeX est obligé de changer de page, mais le résultat sera moche¹² car pour aligner le bas de page, il va falloir étirer démesurément les espaces verticaux présents dans la page. Il suffirait pourtant de déplacer un peu de texte avant la figure pour que tout rentre dans l'ordre.

Le problème, c'est que c'est \LaTeX et pas vous qui décidez des coupures de pages. Il faut donc que ce soit aussi lui qui décide où placer la figure (sur cette page ou la suivante, et éventuellement après du texte qui la suit) en fonction des sauts de page. Ceci s'appelle faire *flotter* la figure, qu'on appelle alors un *flottant*. Mais, maintenant que la figure peut se promener, il faut pouvoir s'y référer : pour cela on pensera à la munir d'un titre et à la numéroter.

```
\begin{figure}[\langle htbp \rangle]
  \langle figure \rangle
  \caption[\langle lof \rangle]{\langle titre \rangle} \label{\langle clé \rangle}
\end{figure}
\listoffigures
```

Pour rendre une figure flottante, il suffit de la placer dans un environnement `figure`. Attention, cet environnement est très mal nommé : il *ne sert pas* à construire des figures¹³ ou à les insérer, uniquement à les faire flotter ! Il faut être conscient, en l'utilisant, que votre figure va très probablement être placée ailleurs que là où elle est dans le source ; surtout, tant que la page contenant la figure, ainsi que la suivante, le placement peut être incohérent : c'est normal — continuez et tout rentrera dans l'ordre.

À l'intérieur de l'environnement `figure`, une nouvelle commande, `\caption`, est disponible, qui sert à donner un titre et un numéro la figure. Cette commande est similaire aux commandes de sectionnement en ce qu'elle admet un argument optionnel `\langle lof \rangle` qui permet de donner un titre alternatif (par exemple, plus court) qui sera utilisé pour la liste des figures. Celle-ci s'obtient de façon automatique avec la commande `\listoffigures`, similaire à `\tableofcontents`.

Le titre sera placé en-dessous ou au-dessus de la figure, selon que dans le source la commande `\caption` est placée avant ou après la figure elle-même. Ceci ne sera plus forcément le cas avec certains modules que nous verrons à la [section 10.5](#). Par ailleurs, on peut utiliser les mécanismes standard de références de \LaTeX (vu en [section 3.5](#)) pour se référer au numéro de la figure avec `\ref{\langle clé-figure \rangle}`. Pour cela, il suffit comme d'habitude de dire `\label{\langle clé-figure \rangle}`, mais il faut prendre garde de placer cette commande *après* la commande `\caption`, et non pas au début de l'environnement `figure` : en effet c'est `\caption` qui produit le numéro, et `\label` ne peut pas fonctionner correctement avant que le numéro ne soit produit.

12. Et provoquera une complainte du type `Underfull \vbox` de la part de \LaTeX .

13. En fait, il est strictement équivalent à l'environnement `table` que nous verrons en [section 7.1.3](#), sauf que `\caption` écrira « figure » ou « table », et que l'objet sera ajouté à la liste des figures ou des tables.

Même si c'est \LaTeX qui décide où place la figure, on peut influencer sa décision au moyen de l'argument optionnel de `figure`, qui contient une suite de lettre parmi `h` (*here*, ici), `t` (*top*, en haut d'une page, par exemple la suivante), `b` (*bottom*, en bas d'une page) et `p` (*page*, sur une page à part, en comportant que des flottants). Les options par défaut sont `tbp`. Il est souvent raisonnable de rajouter l'option `h`, ce qui donne `htbp`, mais c'est en général une mauvaise idée d'enlever des options pour essayer de forcer le flottant à être placé où vous voulez : rappelez-vous que l'intérêt même des flottants est que \LaTeX s'occupe à votre place de la lourde tâche de leur trouver un endroit convenable.

Parfois, on veut absolument placer une figure à un endroit précis, mais on aimerait bien qu'elle soit numérotée et ait un titre comme les autres. Dans ce cas, il faut charger le module `caption`¹⁴, qui permet d'utiliser la commande `\caption` en dehors de tout environnement `figure`, et placer la figure à la main (par exemple dans un environnement `center`).

Enfin, il faut savoir que l'environnement `\figure` ne règle pas lui-même l'alignement horizontal de l'image : pour cela, on utilise les mécanismes standard de \LaTeX , vus à la [section 2.4](#). Pour une fois, on aura intérêt à utiliser une commande comme `\centering` plutôt que l'environnement `center`. En effet, ce dernier ajoute de l'espace vertical, ce qui est inutile voire dérangent car l'environnement `figure` s'en est déjà chargé. Nous verrons en [section 10.5](#) d'autres moyens de gérer l'alignement des flottants.

La nature d'un flottant étant de flotter (sans blague), il n'y aurait pas beaucoup d'intérêt à vouloir en faire rentrer dans une petite case comme tous les exemples de ce polycopié. Je renvoie donc aux exercices¹⁵ pour des exemples de figures flottantes.

6.2.4 Une astuce classique

Les flottants sont parfois un moyen un peu extrême de résoudre le problème du placement des figures. Une solution intermédiaire, qui n'est pas spécifique aux figures, est fournie par la commande `\afterpage` du module `afterpage`¹⁶.

```
\afterpage{\matériel}
```

Sa syntaxe est simple : elle admet un seul argument obligatoire, qui est une liste de commandes arbitraires (par exemple, `\includegraphics`, `\caption`, etc. mais pas `figure` qui n'a pas sa place ici) qui seront exécutées seulement au tout début de la page suivante. Ceci permet d'attendre tranquillement la prochaine coupure de page pour placer une figure dont on est pas sûr qu'elle tienne sur la page en cours, quand on ne veut pas laisser \LaTeX choisir sa place lui-même.

C'est par exemple ce mécanisme que j'ai utilisé pour placer le schéma des dimensions du document, [page 32](#). J'ai ainsi¹⁷ inséré la commande `\afterpage{\layout}` juste après les mots « sur la page suivante » [page 31](#) et le schéma s'est automatiquement déplacé à la page suivante comme annoncé.

14. <http://ctan.org/pkg/caption>

15. Remarque technique : les exercices comme les transparents mentionnent `floatrow`, qui n'est finalement traité qu'à la [section 10.5](#) dans ce polycopié.

16. <http://ctan.org/pkg/afterpage>

17. Je mens un peu : d'une part, j'ai aussi inséré des `\label` pour pouvoir y faire référence ici : d'autre part, `\layout` produit par défaut deux pages de schéma en mode recto/verso : j'ai donc utilisé une commande modifiée pour obtenir le résultat voulu.

6.3 Autres fioritures graphiques

6.3.1 Rotations et mises à l'échelle

Dans les sections précédentes, on a étudié relativement en détails la commande principale fournie par le module `graphicx` : `\includegraphics`. Cette commande permet en particulier d'appliquer des rotations ou des affinités orthogonales¹⁸ à l'image insérée. On peut en fait faire subir le même traitement à n'importe quel texte ou plus généralement « truc » produit par \LaTeX .

```
\rotatebox[origin=lcr]{tcBb}{angle}{matériel}
\scalebox{échelle}{matériel}
\scalebox{échelle h}[échelle v]{matériel}
```

Le sens de l'option `origin` de `\rotatebox` est celui expliqué par la [figure 6.1](#), sauf qu'il ne se rapporte plus à l'image, mais à la « boîte » formée par \LaTeX avec le `<matériel>` donné (voir [section 9.4](#) pour la notion générale de boîte). La syntaxe de `\scalebox` est un peu particulière (le sens de l'argument obligatoire change selon que l'argument optionnel est utilisé ou non, et l'argument optionnel est le deuxième) mais elle devrait être claire après avoir étudié l'[exemple 6.5](#). Le lecteur féru de mathématiques remarquera immédiatement qu'une échelle horizontale négative permet d'effectuer une symétrie orthogonale d'axe vertical.

```
\usepackage{graphicx}
\rotatebox[origin=rb]{-10}{ça s'en va}
et \rotatebox[origin=lb]{10}{ça
   revient}, c'est \scalebox{2}{fait}
de \scalebox{1.5}[.8]{tous petits}
\scalebox{-1}[2]{riens}, \dots
```

ça s'en va et ça revient, c'est fait de
tous petits riens, ...

EXEMPLE 6.5 — Rotation et dilatation de texte

J'espère qu'il est inutile d'attirer votre attention sur le fait qu'il ne faut user de ces possibilités qu'avec parcimonie et à bon escient ☺

6.3.2 Encadrement

```
\fbox{matériel}
```

Pour mettre une image (ou autre chose) en valeur, il peut être intéressant de l'encadrer. Ceci se fait tout simplement avec la commande `\fbox`, fournie par \LaTeX , qui dessine un cadre simple autour de son argument. Pour des cadres un peu plus sophistiqués, on pourra essayer les commandes fournies par `fancybox`¹⁹, illustrées par l'[exemple 6.6](#). Comme on le constatera, les cadres arrondis ne rendent pas toujours bien (à l'écran en tout cas).

On peut aussi faire des cadres en couleur, éventuellement avec un fond coloré, ou même mettre du texte sur fond coloré sans cadre. Les commandes nécessaires sont fournies par `xcolor` et illustrées par l'[exemple 6.7](#).

18. En clair, de changer l'échelle suivant un axe.

19. <http://ctan.org/pkg/fancybox>

```
|\usepackage{fancybox}
\shadowbox{Ombré} ou \doublebox{doublé}
ou \ovalbox{arrondi} mais c'est limite.
```

Ombré ou doublé ou arrondi mais c'est limite.

EXEMPLE 6.6 — Cadres plus sophistiqués

```
|\usepackage{xcolor}
Plus \colorbox{blue!30!white}{bleu} que le
\fcolorbox{blue!70!white}{white}{bleu} de
tes yeux, je ne vois rien de mieux, même le
\fcolorbox{blue}{blue!30!white}{bleu} des cieux.
```

Plus bleu que le bleu de tes yeux, je ne vois rien de mieux, même le bleu des cieux.

EXEMPLE 6.7 — Boîtes colorées

6.3.3 Une police de symboles

Enfin, pour insérer de petits dessins, de la taille d'un caractère, on fait souvent appel à une *police de symboles* plutôt qu'à des images. Au sein d'une telle police, les caractères usuels sont remplacés par des symboles, auxquels on se réfère par leur numéro. Plusieurs polices de symboles existent pour \LaTeX ; une des plus standard est fournie par le paquet `pifont`. Pour des raisons historiques, sa documentation est dans `psnfss2e.pdf`²⁰. Vous y trouverez notamment la liste des symboles avec leurs numéros.

On peut obtenir un symbole isolé au moyen de la commande `\ding{<numéro>}`. Le module fournit quelques autres commandes utiles, comme l'environnement `dingautolist`, qui fournit une liste numérotée par des symboles consécutifs : il suffit d'indiquer le premier, comme dans l'exemple 6.8.

```
|\usepackage{pifont}
Choses à faire :
\begin{dingautolist}{192}
  \item finir le photocopie ;
  \item le mettre en ligne ;
  \item téléphoner \ding{37}
  aux amis et faire la fête.
\end{dingautolist}
```

Choses à faire :

- ① finir le photocopie ;
- ② le mettre en ligne ;
- ③ téléphoner ☎ aux amis et faire la fête.

EXEMPLE 6.8 — Utilisation d'une police de symboles

20. <http://mirror.ctan.org/macros/latex/required/psnfss/psnfss2e.pdf>

7 Tableaux

7.1 Tableaux simples

7.1.1 Les bases

Nous avons déjà abordé la structure de base des tableaux en [section 4.3.2](#) : des lignes séparées par `\\`, formées de cellules séparées par le caractère réservé `&`, le tout placé dans un environnement adapté. L'environnement de base pour les tableaux hors du mode mathématique est `tabular`.

```
\begin{tabular}{<motif>
  <cellule> & <cellule> & <...> \\ <...>
\end{tabular}
```

La nouveauté par rapport aux alignements mathématiques est qu'il faut spécifier un *<motif>*, décrivant le nombre de colonnes du tableau et leur type. Un type de colonne est généralement décrit par une lettre ; les trois types les plus simples sont `l`, `c` et `r` qui fournissent des cellules simple respectivement alignées à gauche, centrées, et alignées à droite, comme le montre l'[exemple 7.1](#).

```
\begin{tabular}{lcr}
  cellule & cellule & cellule \\
  texte & texte & texte \\
  x & x & x \\
\end{tabular}
```

EXEMPLE 7.1 — Types de colonnes simples

Comme on le constate sur cet exemple, un tableau sert juste à aligner les cellules entre elles, la séparation entre les cases n'est pas matérialisée par un quadrillage visible. Pour obtenir des filets (ou réglures) entre les cellules, on procède différemment selon qu'il s'agit d'un filet vertical ou horizontal. Dans le premier cas, il suffit d'insérer un ou plusieurs signes `|` entre les lettres représentant les colonnes dans le motif. Dans le second, c'est dans le corps même du tableau qu'on insère des filets, par la commande `\hline` placée entre deux lignes. L'[exemple 7.2](#) montre un tableau avec quelques réglures.

Les filets verticaux indiqués dans le motif (on dit aussi le *préambule* du tableau) s'étendent sur toute la hauteur. On peut obtenir un filet au milieu d'une cellule avec la commande `\vline`. Nous verrons plus tard comment obtenir un filet vertical seulement sur certaines cases, ou bien sur toutes les cases sauf quelques-unes.

Les filets horizontaux sont plus faciles à gérer : pour obtenir un filet partiel, il suffit d'utiliser `\cline{<déb>-<fin>}` qui donne un filet couvrant les colonnes *<déb>* à *<fin>* incluses. Comme on le constate sur l'[exemple 7.2](#), on peut utiliser plusieurs fois cette commande pour créer un filet horizontal avec un « trou ».

```

\begin{tabular}{r|r|cl|} \hline
A 1 & Machin & Patin & Couffin \\ \hline
B 2 & Bidule & Chose & Chouette \\ \cline{1-1} \cline{3-4}
C & \vline{} 3 & Fou & Barre & Base \\ \hline
\end{tabular}

```

A 1	Machin	Patin	Couffin	
B 2	Bidule	Chose	Chouette	
C	3	Fou	Barre	Base

EXEMPLE 7.2 — Tableaux avec des filets

En fait, le préambule d'un tableau peut contenir un peu plus que des spécifications de colonnes. D'une part, on dispose de raccourcis pour les morceaux de motif se répétant : écrire `*{3}{< sous-motif >}` revient à écrire trois fois de suite `< sous-motif >`. Ça simplifie la saisie et les modifications, et ça évite les erreurs de frappe. Par contre, il faut faire attention à ne pas doubler les filets verticaux par inadvertance.

Par ailleurs, on peut modifier ce qu'il y a entre les colonnes elles-même (par défaut, juste un blanc d'une certaine dimension), en insérant `@{< matériel >}` entre les colonnes dans le motif : dans ce cas, `< matériel >` remplace le blanc. En fait, ceci fonctionne aussi avant la première et après la dernière colonne, ou entre une colonne et un filet vertical, comme le montre l'exemple 7.3. On verra bientôt comment *ajouter* du matériel au début ou à la fin de chaque cellule d'une colonne, ce qui est différent.

```

\begin{tabular}{r@{ et }l|r@{ ans\ }|}
Roméo & Juliette & 420 \\
Jeanne & Serge & 25 \\
Tic & Tac & 66 \\
\end{tabular}

```

Roméo	et Juliette	420 ans
Jeanne	et Serge	25 ans
Tic	et Tac	66 ans

EXEMPLE 7.3 — Matériel inter-colonnes

7.1.2 Colonnes de type paragraphe

Les colonnes simples vues jusqu'à présent correspondent à ces cellules ne pouvant comporter qu'une seule ligne de texte. Il existe des colonnes de type paragraphe, dont les cellules peuvent contenir plusieurs lignes de texte, coupées automatiquement par \LaTeX . Il faut par contre indiquer la largeur de ces colonnes, pour que \LaTeX sache à quelle longueur couper les lignes.

La seule colonne de type paragraphe qui soit pré-définie en \LaTeX est `p{< largeur >}`. Le module `array`¹ fournit deux autres types : `m{< largeur >}` et `b{< largeur >}`. La différence entre ces trois types est la façon dont le paragraphe sera aligné verticalement par rapport aux autres cellules. Avec `p`, c'est par le haut, `m` par le milieu et `b` par le bas.

Comme on le constate sur l'exemple 7.4, quand une cellule est alignée par le haut, cela veut dire que son contenu déborde en bas, et réciproquement. Il faut prendre garde de ne pas s'emmêler.

Par ailleurs, les colonnes de type paragraphe sont souvent assez étroites et posent donc de gros problèmes de coupure de ligne lorsque \LaTeX essaie (comme il le fait d'habitude) d'avoir des marges

1. <http://ctan.org/pkg/array>

```

\begin{tabular}{rpb{1cm}l}
  Base & Par le haut & Par le centre & Par le bas & Base \\
& & & & \\
& & & & \\
\end{tabular}

```

EXEMPLE 7.4 — Colonnes de type paragraphe et alignement

gauches et droites bien rectilignes (texte justifié). Nous verrons plus tard comment gérer ce problème en changeant l'alignement, dès que nous saurons introduire automatiquement du matériel en début de cellule (section 7.2.1).

7.1.3 Placement

De façon similaire à `\includegraphics`, l'environnement `tabular` ne gère absolument pas le placement du tableau : par défaut, il tentera de l'intégrer au paragraphe courant. Comme pour les figures, la solution la plus simple et la plus courante consistera à placer notre tableau dans un environnement `center` (ou `flushleft`) pour ajuster l'espace vertical autour du tableau et le center (ou l'aligner à gauche).

```

\begin{table}[htbp]
  <tableau>
  \caption[lot]{<titre> \label{<clé>}
\end{table}
\listoftables

```

Toutes les autres solutions de placement évoqués pour les figures sont aussi disponibles pour les tableaux, même si certaines sont peu adaptés esthétiquement (habillage). Celle qui sert le plus souvent est de faire flotter les tableaux de taille importante. Pour cela, on procède exactement comme pour les figures, en remplaçant l'environnement `figure` par `table`. Je le répète, `table` ne sert pas à construire un tableau, mais à le faire flotter. Relisez la section 6.2.3 si vous n'êtes pas sûr de savoir ce que cela signifie. Par ailleurs, `\listoftables` permet d'obtenir la liste des tableaux, de façon similaire à `\tableofcontents` pour la table des matières.

7.1.4 En mode mathématique

Nous avons vu en section 4.3.2 plusieurs façons de construire des alignements en mode mathématique, adaptées à tel ou tel cas d'usage. Il existe un environnement de tableau mathématique : `array`, à ne pas confondre avec le module du même nom, vu précédemment. Cet environnement fonctionne exactement comme `tabular` sauf sur les deux points suivants :

- il faut être en mode mathématique pour l'utiliser ;
- chaque cellule est automatiquement en mode mathématique.

Il prend aussi un argument décrivant les colonnes ; en général on n'utilise que des colonnes de type `l`, `c` ou `r`.

De façon générale, il faut éviter le plus possible d'utiliser `array` lorsqu'une solution alternative est disponible : en effet, il est très difficile d'obtenir des espacements cohérents entre les colonnes

avec `array`, comme le montre l'exemple 7.5 : dans (7.1) et (7.2), les espaces autour du signe d'égalité sont identiques et ils sont corrects. Dans (7.3) et (7.4), ils sont trop grands d'un côté ou des deux, et dans (7.5) trop petits. Dans ce cas, `aligned` est la bonne solution.

```

|\usepackage{amsmath}
\begin{gather}
x = 2 + 2 \label{good1} \\
\begin{aligned} x &= 2 + 2 \\ &= 4 \end{aligned} \label{good2} \\
\begin{array}{rl} x &= 2 + 2 \\ &= 4 \end{array} \label{bad1} \\
\begin{array}{rcl} x &=& 2 + 2 \\ &=& 4 \end{array} \label{bad2} \\
\begin{array}{r@{}c@{}l} x &=& 2 + 2 \\ &=& 4 \end{array} \label{bad3}
\end{gather}

```

$x = 2 + 2$

(7.1)

$x = 2 + 2$
 $= 4$

(7.2)

$x = 2 + 2$
 $= 4$

(7.3)

$x = 2 + 2$
 $= 4$

(7.4)

$x=2 + 2$
 $=4$

(7.5)

EXEMPLE 7.5 — Mauvais espaces horizontaux avec `array`

Ceci dit, `array` est utile pour des constructions spécifiques, comme des tableaux de variation. Il faut seulement l'éviter pour aligner des équations sur le signe = (ou un autre signe de relation), ce qui se fait avec `align` ou une de ses variantes vues en section 4.3.2.

7.2 Techniques plus avancées

7.2.1 Matériel automatique

$\>\{matériel\ avant\}\{colonne\}\<\{matériel\ après\}$

Le module `array` est pratiquement indispensable dès que l'on veut faire des choses un peu sérieuses avec les tableaux. En plus de fournir les types de colonnes `m` et `b`, il fournit surtout la possibilité d'indiquer dans le préambule du tableau du matériel qui sera inséré automatiquement au début ou à la fin de chaque cellule de la colonne. L'exemple 7.6 illustre l'utilisation de ces fonctionnalités.

```

|\usepackage{array}
\begin{tabular}{|>\bfseries|l|}
>\{(\}c\{\}\}|r\{~\texteuro}\}
du texte & 2^2 & 1 \\
en gras & \sqrt{2} & 3 \\
\end{tabular}

```

du texte

2^2

$1 \in$

en gras

$\sqrt{2}$

$3 \in$

EXEMPLE 7.6 — Matériel automatique pré- et post-colonne

Remarquez que pour mettre une texte en gras, on utilise la forme déclarative `\bfseries` : en effet on ne peut pas mettre `\textbf{` au début et `}` à la fin, car le début et la fin doivent être équilibrés en accolades. Heureusement, une commande déclarative n'agit qu'à l'intérieur de la cellule où elle a été utilisée.

Revenons maintenant aux colonnes de type paragraphe : c'est souvent une mauvaise idée de les laisser justifiées, il faut donc les centrer ou les aligner à droite ou à gauche. Ceci se fait avec les commandes standard, comme `\centering`, que l'on peut insérer comme matériel automatique en début de colonne. Le problème qui survient alors est que ces commandes re-définissent `\\`, qui est déjà utilisé pour séparer les lignes du tableau. Deux solutions sont alors disponibles :

- utiliser `\\` pour aller à la ligne dans une cellule, et le remplacer par `\tabularnewline` entre deux lignes du tableau ;
- faire suivre `\centering` de la commande spéciale `\arraybackslash` : la commande `\\` retrouve alors son sens habituel de séparateur des lignes du tableau, mais on ne peut plus l'utiliser pour aller à la ligne au sein d'une cellule.

L'exemple 7.7 montre deux tableaux utilisant chacun une des deux approches.

<code>\usepackage{array}</code>	a	blabla
<code>\begin{tabular}{r>{\centering}p{3cm}}</code>		bla
<code> a & blabla \\ bla \\ lala \tabularnewline</code>		lala
<code> b & blibli \\ bli \\ lili \tabularnewline</code>	b	blibli
<code>\end{tabular}\par</code>		bli
<code>\begin{tabular}{r</code>		lili
<code> >{\centering\arraybackslash}p{3cm}}</code>	a	blabla bla lala baba
<code> a & blabla bla lala baba bla blabla \\</code>		bla blabla
<code> b & blibli bli lili bibi bli blibli \\</code>	b	blibli bli lili bibi bli
<code>\end{tabular}</code>		blibli

EXEMPLE 7.7 — Gestion des retours à la ligne

7.2.2 Colonnes personnalisées

```
\newcolumntype{<lettre>}{<motif>}
```

Si l'on utilise souvent le même type de colonnes, avec le même matériel automatique de début et de fin, il peut être pratique de définir un raccourci. C'est ce que fait `\newcolumntype` : associer tout un morceau de préambule à une seule lettre. Attention, une lettre, ça représente assez peu de choix possibles : il faut donc être prudent pour éviter de prendre une lettre déjà utilisée par \LaTeX ou un module existant.

7.2.3 Fusion de cellules

```
\multicolumn{<nombre>}{<motif>}
```

Il est possible de fusionner horizontalement plusieurs cellules entre elles, par exemple pour avoir un titre s'étalant sur toute la largeur du tableau. Pour cela, on utilise la commande standard `\multicolmun` fournie par \LaTeX , dans la cellule la plus à gauche de celles à fusionner en indiquant le nombre total de cellules participant à la fusion. On doit ensuite indiquer un motif de colonnes qui remplacera le motif habituel des cellules utilisées. Il faut bien faire attention aux filets verticaux en changeant ce motif, comme dans l'exemple 7.8.

```

\begin{tabular}{|*3{1|}} \hline
\multicolumn{3}{|c|}{\textbf{Titre
général assez long}} \\\hline
\multicolumn{1}{|c|}{\textit{T. 1}} &
\multicolumn{1}{|c|}{\textit{T. 2}} &
\multicolumn{1}{|c|}{\textit{T. 3}} \\\hline
machin 1 & machin 2 & machin 3 \\\hline
c1 & c2 & c3 \\\hline
\end{tabular}

```

Titre général assez long		
<i>T. 1</i>	<i>T. 2</i>	<i>T. 3</i>
machin 1	machin 2	machin 3
c1	c2	c3

EXEMPLE 7.8 — Fusion horizontale de cellules

De plus, on voit sur cet exemple un usage particulier, mais assez courant, de `\multicolumn` : fusionner une cellule avec elle-même, juste pour pouvoir changer son motif (ici, il s'agit de centrer les titres alors que le texte est normalement aligné à gauche comme sur la dernière ligne).

```

\multirow{<nb lignes>}{<largeur>}{<paragraphe>}
\multirow{<nb lignes>}*{<texte court>}

```

Pour fusionner des cellules verticalement, on doit utiliser le module `multirow` pour disposer de la commande éponyme. Celle-ci se place dans la cellule en haut de celles à fusionner ; ces dernières devront être laissées vides dans les lignes suivantes. Le premier argument indique le nombre total de cellules fusionnées. Par défaut, la cellule ainsi créée est de type paragraphe : son contenu peut s'étaler sur plusieurs lignes, il faut alors préciser sa largeur comme deuxième argument. Si au lieu d'une largeur, on met une étoile, alors la cellule ne peut plus contenir qu'une ligne de texte. Dans tous les cas, son contenu est centré verticalement dans l'espace disponible. Souvent, on peut vouloir tourner le texte dans la cellule fusionnée, comme dans l'exemple 7.9.

```

\usepackage{multirow, graphicx}
\begin{tabular}{rcl}
Premier & \multirow{3}{*}{\rotatebox
[origin=cc]{90}{Podium}} & Premier & Or
Deuxième & & Deuxième & Argent
& Or & Troisième & Bronze
Troisième & & &
\end{tabular}

```

EXEMPLE 7.9 — Fusion verticale de cellules

Par ailleurs, dans l'autre sens, on peut aussi diviser des cellules. Ceci est rarement utile, sauf pour couper une cellule en deux en diagonale. Le module `slasbox`² fournit la commande éponyme, qui permet de faire cela. Je renvoie aux exercices pour un exemple. Attention, le résultat est souvent douteux d'un point de vue esthétique : il vaut souvent mieux trouver une autre façon de disposer son tableau.

2. <http://ctan.org/pkg/slasbox>

7.2.4 Ajustement de la largeur

```

\begin{tabular*}{\langle largeur \rangle}{@{\extracolsep\fill}\langle motif \rangle}
\begin{tabularx}{\langle largeur \rangle}{\langle motif avec X \rangle}
\begin{tabulary}{\langle largeur \rangle}{\langle motif avec L, C, R, J \rangle}

```

Pour l’instant, tous nos tableaux s’ajustaient en largeur à leur contenu. Pour faire un tableau dont on choisit la largeur totale, il y a deux types de solutions : ajuster la largeur en ajoutant de l’espace entre les colonnes, on en changeant la taille des colonnes. La première solution est fournie par l’environnement `tabular*`, à condition de penser à rajouter `@{\extracolsep\fill}` en début de motif (ce n’est pas très intuitif, mais c’est comme ça).

Cette approche n’est en général pas très intéressante, car il est dommage de laisser du vide entre les colonnes alors qu’on pourrait souvent utiliser cet espace. On peut bien sûr chercher à ajuster la largeur des colonnes à la main, mais c’est fastidieux : il faut mieux automatiser ceci avec les modules `tabularx`³ ou `tabulary`⁴ et leurs environnements éponymes. Le premier fournit un nouveau type de colonne : `X`, qui est l’équivalent de `p` sauf qu’on a pas à indiquer la largeur : elle est déterminée par `tabularx` en fonction de la largeur des autres colonnes pour atteindre la largeur totale voulue.

<pre> \usepackage{array, tabularx, tabulary} \newcommand\tcourt{Un texte plutôt court.} \newcommand\tlong{Un texte nettement plus long pour illustrer la différence entre \texttt{tabularx} et \texttt{tabulary}.} \newcommand\rr{\raggedright} \begin{tabular*}{\linewidth}{ @{\extracolsep\fill} >{\rr}p{1.5cm} >{\rr}p{3cm} } \tcourt & \tlong \end{tabular*} \par \begin{tabularx}{\linewidth} { >{\rr}X >{\rr}X } \tcourt & \tlong \end{tabularx} \par \begin{tabulary}{\linewidth}{ L L } \tcourt & \tlong \end{tabulary} </pre>	<p>Un texte plutôt court.</p>	<p>Un texte nettement plus long pour illustrer la différence entre <code>tabularx</code> et <code>tabulary</code>.</p>
<pre> \begin{tabular*}{\linewidth}{ @{\extracolsep\fill} >{\rr}p{1.5cm} >{\rr}p{3cm} } \tcourt & \tlong \end{tabular*} \par \begin{tabularx}{\linewidth} { >{\rr}X >{\rr}X } \tcourt & \tlong \end{tabularx} \par \begin{tabulary}{\linewidth}{ L L } \tcourt & \tlong \end{tabulary} </pre>	<p>Un texte plutôt court.</p>	<p>Un texte nettement plus long pour illustrer la différence entre <code>tabularx</code> et <code>tabulary</code>.</p>
<pre> \begin{tabular*}{\linewidth}{ @{\extracolsep\fill} >{\rr}p{1.5cm} >{\rr}p{3cm} } \tcourt & \tlong \end{tabular*} \par \begin{tabularx}{\linewidth} { >{\rr}X >{\rr}X } \tcourt & \tlong \end{tabularx} \par \begin{tabulary}{\linewidth}{ L L } \tcourt & \tlong \end{tabulary} </pre>	<p>Un texte plutôt court.</p>	<p>Un texte nettement plus long pour illustrer la différence entre <code>tabularx</code> et <code>tabulary</code>.</p>

EXEMPLE 7.10 — Tableaux de largeur fixe

Dans le cas où plusieurs colonnes de type `X` sont utilisées, elles se partagent à égalité la largeur disponible. Ceci n’est pas toujours l’idéal, par exemple si le contenu d’une colonne est nettement plus long que celui de l’autre, comme dans l’exemple 7.10. Dans ce cas, il faut mieux utiliser `tabulary`, qui répartit équitablement⁵ la largeur entre les colonnes en fonction de leurs besoins.

3. <http://ctan.org/pkg/tabularx>

4. <http://ctan.org/pkg/tabulary>

5. Notez la différence entre *égalité* et *équité*.

De plus, `tabulary` possède quatre types de colonne de largeur automatique, qui diffèrent par l'alignement horizontal du texte : **L** aligné à gauche, **C** centré, **R** aligné à droite et **J** justifié (comme `p`). Ceci offre rend l'écriture du préambule plus concise, puisqu'on a pas besoin de jouer avec `>\raggedright`.

7.2.5 Couleurs

Il est possible d'utiliser des couleurs dans les tableaux. Pour mettre le contenu d'une cellule en couleur, on utilise `\color` ou `\textcolor` comme d'habitude. Pour tout le reste, on dispose de commandes spéciales obtenues en chargeant `xcolor` avec l'option `table`.

```
\cellcolor{<couleur>}
>{\columncolor{<couleur>}}
\rowcolor{<couleur>}
\arrayrulecolor{<couleur>}
```

Ces commandes ont des noms assez clairs (si l'on sait que *cell* signifie cellule, *column* colonne, *row* ligne et *rule* filet) et servent à déterminer la couleur de différents éléments. La seule difficulté est de les placer au bon endroit : `\cellcolor` se place au tout début d'une cellule, `\columncolor` dans le préambule du tableau, comme matériel automatique de début de cellule, `\rowcolor` se place au tout début de n'importe quelle cellule d'une ligne et agit jusqu'à la fin de la ligne. Enfin, `\arrayrulecolor` s'utilise entre deux lignes d'un tableau. L'usage de toutes ces commandes est illustré par l'exemple 7.11, qui est de très mauvais goût.

```
\usepackage[table]{xcolor}
\arrayrulecolor{red}
\begin{tabular}{l| >{\color{green}}c |
 >{\color{blue}\columncolor{yellow}}r}
\rowcolor{pink} Titre 1 & Titre 2 & Titre 3 \\
\hline \cellcolor{cyan} Cellule 1a & Cell. 1b & Cell. 1c \\
& Cell. 2a & Cell. 2b & Cell. 2c \\
& Cell. 3a & Cell. 3b & Cell. 3c \\
\arrayrulecolor{black} \color{cyan} Cell.
3a & Cell. 3b & Cell. 3c \\
\hline
\end{tabular}
```

Titre 1	Titre 2	Titre 3
Cellule 1a	Cell. 1b	Cell. 1c
Cell. 2a	Cell. 2b	Cell. 2c
Cell. 3a	Cell. 3b	Cell. 3c

EXEMPLE 7.11 — Trop de couleurs dans un tableau

```
\rowcolors{<délai>}{<couleur>}{<couleur>}
```

Une utilisation bien plus raisonnable de la couleur consiste à colorer une ligne sur deux. Dans les tableaux longs, ceci aide le lecteur à suivre les lignes, sans pour autant trop alourdir le tableau. L'option `table` d'`xcolor` fournit une commande automatisant ceci, à placer avant le tableau. Les deux derniers arguments de `\rowcolors` sont les couleurs à appliquer successivement aux lignes, tandis que le premier argument permet de commencer l'alternance avec un peu de retard, par exemple pour laisser la place aux lignes de titre. C'est que qu'on fait dans l'exemple 7.12, où l'on voit aussi que laisser vide un des deux derniers arguments revient à laisser la ligne en blanc.

EXEMPLE 7.12 — Une utilisation raisonnable de la couleur

7.2.6 Aspects esthétiques

Vous savez maintenant, en combinant les éléments déjà vus, faire des tableaux très sophistiqués. Les exemples donnés sont souvent assez laids et servent juste à illustrer rapidement l'ensemble des techniques disponibles. Pour finir ce chapitre, je vous encourage, pour les tableaux encore plus que pour le reste de votre mise en pages, à rester *simple* et *sobre*. Un excès d'information ou une structure trop complexe désorientent le lecteur plus qu'ils ne l'aident.

En principe, un tableau n'a pas besoin de beaucoup de filets. L'alignement des éléments entre eux est suffisant pour guider l'œil du lecteur. Il est conseillé d'utiliser une structure commune pour la plupart de ses tableaux, à la fois par souci esthétique et pour ne pas détourner l'attention du lecteur.

Le module `booktabs`⁶ fournit trois commandes : `\toprule`, `\midrule` et `\bottomrule`, qui remplacent avantageusement `\hline` pour des filets horizontaux en haut, milieu et bas de tableau. Elles ajustent légèrement l'espace vertical de façon à obtenir une disposition plus équilibrée. Ces commandes ont été utilisées pour tous⁷ les tableaux du présent polycopié, et aucun filet vertical n'a été inséré. Ceci fournit, je l'espère, des tableaux aérés et lisibles.

Enfin, il est parfois nécessaire d'ajuster manuellement certains espacements dans les tableaux, notamment autour des filets horizontaux. Nous verrons quelques techniques pour cela en [section 9.4.5](#).

6. <http://ctan.org/pkg/booktabs>

7. Sauf bien sûr ceux des exemples de ce chapitre...

8 Compléments divers

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 poly. Il s'agit des sections 3.5.2 (bibliographie), 6.2.4 (astuce pour les figures), 7.1.4 (tableaux en mode mathématique) et 7.2.5 (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 huit.

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 exigibles¹ à l'examen. Elle ne sera donc pas reproduite ici.

8.2 Code informatique

8.2.1 Outils de \LaTeX

Le problème qui se pose lorsque l'on veut saisir du code informatique avec \LaTeX est que de nombreux symboles réservés sont utilisés fréquemment dans la plupart de langages courants. Pour contourner ce problème, \LaTeX dispose d'un mode spécial, appelé *verbatim*, où plus aucun caractère n'est réservé ou spécial.

```
\verb{*}<symbole><texte verbatim><symbole>  
\begin{verbatim}*} <contenu verbatim>
```

Un peu comme pour les math, il y a deux moyens d'écrire en verbatim : soit dans le cours du texte, avec la commande `\verb`, soit hors-texte, éventuellement sur plusieurs lignes, avec l'environnement `verbatim`. Dans les deux cas, le texte sera aussi composé en style machine à écrire, comme avec `\texttt`, et si l'on utilise l'étoile optionnelle de la commande ou de l'environnement, les espaces seront matérialisés par le caractère `␣`, comme le montre l'exemple 8.1.

On constate aussi que plus aucun espace n'est ignoré : ni en début de ligne, ni après un autre espace. La syntaxe de la commande `\verb` est particulière : son « argument » n'est pas délimité par des accolades. Il se termine à la première apparition du symbole (autre qu'une étoile) qui suit immédiatement la commande `\verb`. Ainsi, dans l'argument, on peut utiliser tous les symboles sauf un. En pratique ce n'est pas un problème.

L'environnement `verbatim` a aussi une particularité syntaxique, qui est moins visible : la chaîne `\end{verbatim}` doit impérativement être seul sur une ligne. Par ailleurs, ni la commande ni l'environnement ne fonctionnent dans l'argument d'une autre commande. Par exemple, le code

1. Ce qui ne vous empêche pas de m'éblouir par votre maîtrise de `bibtex` dans votre document libre si vous maîtrisez déjà bien le reste du cours et désirez passer d'une très bonne note à une très, très bonne note. Cette remarque est valable pour la plupart des points hors-programme, bien sûr.

Du texte et des caractères réservés :

`\verb+{ $ }+`, éventuellement avec des espace visibles : `\verb*+{ $ }+`.

```
\begin{verbatim}
#include <stdlib.h>
int main(void) {
    exit 0;
}
\end{verbatim}
```

Du texte et des caractères réservés : `{ $ }`, éventuellement avec des espace visibles : `{_ $ _}`.

```
#include <stdlib.h>
int main(void) {
    exit 0;
}
```

EXEMPLE 8.1 — Commande et environnement de base pour le verbatim

`\fbox{\verb+\lipsum+}`

ne donnera pas `\lipsum`, mais provoquera en fait une erreur à la compilation. Il faut être prudent car ce genre d'erreur est parfois difficile à comprendre, elles sont parfois mal détectées par \LaTeX et le message d'erreur n'est alors pas clair du tout. Ceci est dû au fonctionnement très particulier du verbatim, et les commandes et environnements présentés dans les deux prochaines section présentent la même limitation.

Les outils fournis par \LaTeX répondent plutôt bien au problème de saisir des morceaux de texte comportant beaucoup de caractères spéciaux, mais n'offrent aucune possibilité de mettre en forme (police, couleur, encadrement, etc.) ces fragments. (Les commandes standard ne sont pas utilisables pour la raison évoquée ci-dessus.) Comme d'habitude, de nombreux modules offrent des possibilités supplémentaire. Nous allons en étudier deux, qui sont sans doute les plus aboutis : `fancyvrb`² et `listings`³.

8.2.2 Avec le module fancyvrb

Tout d'abord, `fancyvrb` fournit une commande et un environnement (`\Verb` et `Verbatim`) qui ont pour but de remplacer ceux fournis par \LaTeX et peuvent s'utiliser avec la même syntaxe. Leur avantage est que l'apparence du résultat est réglable.

```
\fvset{<option>=<valeur>, <...>}
formatcom=<commande>
```

Pour ceci, on dispose de la commande `\fvset` qui a une action globale. C'est la seul moyen de régler l'apparence du résultat de `\Verb`. L'environnement `Verbatim`, par contre, accepte un argument optionnel qui permet de spécifier ses options valables uniquement pour l'environnement en cours.

Parmi les options, certaines ont une influence à la fois sur `\Verb` et `\Verbatim`. C'est le cas par exemple de `formatcom` qui prend comme argument une commande ou une suite de commande déterminant la police et la couleur du résultat. Attention, il faut utiliser ici la forme déclarative (ex. `\bfseries`) et non la forme avec argument (ex. `\textbf`), comme le montre l'exemple 8.2.

2. <http://ctan.org/pkg/fancyvrb>

3. <http://ctan.org/pkg/listings>

Par ailleurs, on y voit que chaque utilisation de l'option `formatcom` annule la précédente, les commandes ne s'ajoutent pas entre elles.

```
numbers=(none, left, right)
stepnumber=(nombre)
```

D'autres options n'ont de sens que pour l'environnement, comme `number` qui détermine si les lignes sont numérotées, et de quel côté sont placés les numéros. On dispose d'ailleurs de `stepnumber` pour éviter de numéroté toutes les lignes, également illustré par l'exemple 8.2.

<pre>\usepackage{fancyvrb, xcolor} \fvset{formatcom=\color{pink}\bfseries, numbers=left, stepnumber=2} D'abord en ligne : \Verb+{#}+. Puis \begin{Verbatim}[formatcom=\slshape] #!/usr/bin/perl use warnings; print "Hello, world!"; exit 0; \end{Verbatim}</pre>	<p>D'abord en ligne : <code>{#}</code>. Puis</p> <pre>#!/usr/bin/perl 2 use warnings; print "Hello, world!"; 4 exit 0;</pre>
---	---

EXEMPLE 8.2 — Usage simple de `fancyvrb`

Il existe bien d'autres possibilités pour mettre en forme le contenu avec `fancyvrb`. Voyons par exemple comment encadrer le code et lui donner un titre.

```
frame=(none, leftline, topline, bottomline, lines, single),
framerule=(dimension), framesep=(dimension), framecolor=(commande),
label=(texte), labelposition=(none, topline, bottomline, both)
```

Ces options ont des noms assez explicites, du moins si l'on connaît un peu d'anglais, et il est facile d'expérimenter avec jusqu'à obtenir l'effet désiré. J'attire votre attention sur deux points délicats : le premier est que la valeur de `framecolor` doit être une *commande* comme `\color{blue}` et pas un simple *nom* de couleur comme `blue`. Par ailleurs, l'option `label` n'a rien à voir avec la commande `\label`. En fait elle ressemble plus à `\caption` : il s'agit de donner un titre au morceau de code ; par contre, aucune numérotation n'est produite.

```
\begin{BVerbatim}[boxwidth=(auto, dimension), (autres options)]
```

Par défaut, une ligne de texte verbatim prend toute la largeur, ce qui devient visible quand on l'encadre. Il existe en fait une variante de `Verbatim` nommée `BVerbatim` (pour *boxed verbatim*) qui peut créer des lignes plus petites. En plus des options habituelles, elle reconnaît l'option `boxwidth` qui permet de spécifier la dimension des lignes créées, ou de s'adapter automatiquement à la largeur nécessaire avec le mot-clé `auto`.

Enfin, `fancyvrb` propose quelques moyens de simplifier la saisie. Par exemple, si comme moi vous aimez indenter le contenu des environnements avec deux espaces, vous serez vite gênés par le fait que ces deux espaces apparaissent dans la sortie (comme c'est le cas dans l'exemple 8.2). Pour les éliminer, il suffit d'utiliser l'option `gobble=<n>`, où `<n>` est le nombre de caractères à supprimer au début de chaque ligne. Attention, si une ligne de commence pas pas au moins `<n>` espaces, vous⁴ perdrez des caractères !

Par ailleurs, si vous avez beaucoup de verbatim à saisir au cours du texte, vous pouvez créer un raccourci pour `\Verb` : placez `\DefineShortVerb|` dans votre préambule, et partout dans le document, `|` sera équivalent à `\Verb|` : c'est-à-dire que vous pouvez saisir seulement `|\lipsum|` au lieu de `\Verb|\lipsum|`. Par contre, il devient bien sûr impossible d'utiliser le caractère `|` normalement. Vous pouvez au besoin désactiver ce raccourci avec `\UndefineShortVerb|`. (Il est bien sûr possible de choisir n'importe quel autre caractère à la place de `|`, mais les caractères non-ASCII⁵ ne fonctionneront pas en utf8.)

8.2.3 Aparté : le verbatim dans des arguments

Comme je l'ai dit, le mode verbatim ne peut pas fonctionner dans l'argument d'un autre commande. Comment faire pour contourner cette limitation ? Cette section présente trois approches possibles.

La première consiste tout simplement à ne pas utiliser de verbatim, mais à le simuler. En effet, chacun des caractères réservés peut s'obtenir avec une commande : il suffit de l'utiliser. Pour simplifier la saisie, on a intérêt à se définir des commandes adaptées au contexte, comme dans l'exemple 8.3. C'est la méthode que j'utilise la plupart du temps : à chaque fois que vous croyez voir du verbatim dans le titre d'un exemple ou d'un tableau (c'est-à-dire dans l'argument de la commande `\caption`) par exemple, il s'agit de faux verbatim.

<pre>\usepackage{fancyvrb, xcolor} \newcommand\code{\ttfamily\color{magenta}} \fvset{formatcom=\code} \newcommand\macro[1]{\code\textbackslash#1} \newcommand\argu[1]{\code\textbraceleft #1\textbraceright}}</pre>	<p>Du vrai verbatim : <code>\textbf{gras}</code>. Mais pour encadrer, il faut tricher : <code>\textbf{gras}</code>. Une autre méthode : <code>\textbf{gras}</code>.</p>
<p>Du vrai verbatim : <code>\Verb+\textbf{gras}+</code>. Mais pour encadrer, il faut tricher : <code>\fbox{% \macro{textbf}\argu{gras}}</code>. Une autre méthode : <code>\SaveVerb{bold}+\textbf{gras}+\fbox{\UseVerb{bold}}</code>.</p>	

EXEMPLE 8.3 — Verbatim : quand l'imitation vaut mieux que l'original

La deuxième méthode, illustrée dans ce même exemple, consiste à utiliser les commandes, conçues à cet effet, `\SaveVerb` et `\UseVerb`, fournies par `fancyvrb`, qui permettent de sauvegarder un fragment verbatim sous un nom libre, dans un contexte sûr (en dehors de l'argument

4. Et pour citer GARY déformant HUGO : « Quand je dis *vous*, c'est aussi de *moi* que je parle ». C'est en effet une erreur que vous m'avez sûrement vu commettre dans une version de certains slides...

5. Les caractères visibles ASCII sont, en plus des lettres des chiffres et de l'espace, les 33 caractères suivants : `!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~.` Ces 96 caractères sont les seuls avec lesquels vous n'aurez jamais de problème d'encodage.

d'une commande), puis de l'utiliser ensuite dans n'importe quel contexte. C'est une méthode que j'utilise aussi dans ce polycopié, mais jamais directement, car je la trouve fastidieuse : je l'enrobe donc dans des macros ou environnements personnels qui rendent son usage plus facile.

Enfin la dernière méthode ⁶ est spécifique à la commande `\footnote`. Il est en fait possible de modifier cette commande particulière de façon à ce qu'on puisse utiliser du verbatim dans son argument ⁷ sans précaution particulière. Pour cela, il suffit de charger `fancyvrb` et de placer la commande `\VerbatimFootnotes` juste après le `\begin{document}`. Attention, il est possible que cette commande ne fonctionne pas si vous utilisez d'autres paquets qui modifient la commande `\footnote`.

8.2.4 Avec le module listings

Le module `listings` fournit beaucoup d'options similaires à celles de `fancyvrb`, sur lesquelles je ne m'étendrai pas. Parlons plutôt de ce qu'il offre de plus : il permet de reconnaître des mots-clé selon le langage informatique utilisé, et de leur appliquer automatiquement des mises en formes particulières, comme le montre l'exemple 8.4, qui utilise d'abord les réglages par défaut, puis personnalise un peu l'apparence.

```

|\usepackage{listings}
\begin{lstlisting}[language=C]
#include <stdlib.h>
int main(int argc, char** argv) {
    exit 0; /* ouf, fini ! */
}
\end{lstlisting}
\begin{lstlisting}[language=C,
basicstyle=\ttfamily,
keywordstyle=\color{blue},
commentstyle=\color{black!30},
numbers=left]
#include <stdlib.h>
int main(int argc, char** argv) {
    exit 0; /* ouf, fini ! */
}
\end{lstlisting}

```

EXEMPLE 8.4 — Reconnaissance de mots-clé du langage C

Attention, les caractères non-ASCII (voir note 5, page 74) sont susceptibles de poser problème avec `listings`. Si on veut en utiliser dans le code présenté, il faut penser à spécifier l'option `extendedchars=true`. Malheureusement, ceci ne fonctionnera pas si l'encodage utilisé est `utf8`, pour des raisons techniques et sérieuses. (Cette limitation pourra être levée à long terme.)

Le module `listings` offre bien d'autres possibilités, comme d'ajouter des mots-clés, de définir des nouveaux styles de commentaire, de sortir temporairement du mode verbatim, de couper

6. Que j'utilise aussi dans ce poly, décidément...

7. En fait, ce n'est alors plus un argument, au sens \TeX nique du terme, mais ce genre de considération dépasse largement le cadre de ce cours.

les lignes à une longueur fixée, etc. Pour plus de détails, je vous renvoie à sa documentation, assez complète, même s'il n'est pas toujours facile de s'y orienter. D'ailleurs, je n'ai pas non plus présenté toutes les fonctionnalités de `fancyvrb`, et vous invite à consulter la documentation, riche de nombreux exemples, pour apprendre le reste.

8.3 Concentré d'orthotypographie

Pour clore ce chapitre, une section assez différente des autres. D'habitude, nous parlons beaucoup des *moyens* d'obtenir tel ou tel résultat, mais peu du *but*, qui est de produire un document beau à regarder et agréable à lire, ce pour quoi \LaTeX n'est qu'un outil. Après tout, le *T_EXbook* présente \TeX (sur lequel est basé \LaTeX) en ces termes :

Gentil lecteur, ceci est un manuel décrivant \TeX , un nouveau système de composition conçu pour créer de beaux livres — spécialement des livres traitants de mathématiques.

Je n'ai ni la compétence ni le temps de vous apprendre comment faire de *beaux* documents, mais parlons déjà du premier pas, qui est de produire des documents orthotypographiquement corrects, c'est-à-dire respectant les règles de base de la typographie de la langue dans laquelle ils sont écrits.

La première règle est la sobriété. Un texte bien composé est assez régulier ; il n'y a donc pas besoin de beaucoup d'effets pour mettre des mots en valeur. Pour insister légèrement sur un mot, l'italique est le moyen préféré (avec la commande `\emph` en \LaTeX). Pour insister lourdement, on peut utiliser le gras, qui est déjà un moyen agressif : même en regardant la page de loin, on repère immédiatement la tache produite par ce mot au sein du bloc de texte (on dit qu'il rompt le *gris typographique*).

Il convient d'éviter absolument les changements de taille au sein d'un paragraphe, car ils ruinent totalement son homogénéité. De même, le souligné est généralement à proscrire : au sein d'un paragraphe, il trouble l'interligne ou entre en collision avec les caractères ; en dehors d'un paragraphe, il est parfaitement inutile car les mots sont suffisamment mis en valeur par leur position sur la page, et éventuellement leur taille et leur graisse.

Par ailleurs, en français, on n'utilise les majuscules⁸ qu'avec parcimonie. Si aucune règle de grammaire ou d'orthographe ne dit qu'une lettre particulière doit être en majuscule, alors elle est en minuscule. En particulier, on ne met pas de majuscule à un mot juste pour souligner son importance ; dans les titres, seul le premier mot prend une majuscule (c'est en anglais que tous les mots d'un titre prennent une majuscule).

Par ailleurs, les majuscules conservent les accents et les cédilles⁹ : ceux-ci font partie de l'orthographe du mot, qui n'a pas de raison de changer selon sa place dans la phrase ou dans un titre. La fausse croyance, très répandue, en une règle disant le contraire, provient en fait des limitations techniques des machines à écrire. \LaTeX ne souffre heureusement pas de ce genre de limitation.

Un autre point soumis à des règles précises est l'espacement autour des signes de ponctuation, qui ne doit pas être utilisé de façon fantaisiste : plus encore que l'esthétique, c'est la lisibilité du texte qui est en jeu. Ces règles sont résumées par la [table 8.1](#).

8. Note pour les connaisseurs : j'évite volontairement de distinguer majuscules et capitales, pour ne pas compliquer l'exposé.

9. Plus généralement, ce qu'on appelle les *diacritiques*, qui comprennent les accents, les cédilles, les trémas, etc.

Avant	Signe	Après
rien	,	espace normale
rien	.	espace normale
fine	;	espace normale
fine	!	espace normale
fine	?	espace normale
insécable	:	espace normale
espace normale	—	espace normale
espace normale	«	insécable
insécable	»	espace normale
espace normale	(rien
rien)	espace normale

TABLE 8.1 — Espacement autour des signes de ponctuation

Ces règles peuvent vous faire peur à première vue, surtout celles impliquant des espaces fines¹⁰ ou insécables. La bonne nouvelle, c'est que lorsque vous utilisez `babel` avec l'option `french`, ces espaces sont gérées automatiquement pour vous (avant les ponctuations hautes `;!:` et à l'intérieur des guillemets avec `\og` et `\fg`). Par contre, c'est à vous de gérer le reste des espaces : en mettre là où il doit y en avoir, et pas ailleurs.

Ce sont des habitudes de saisie au clavier que l'on prend très rapidement (que vous avez peut-être même déjà). Il faut y faire attention, au même titre qu'on veille à ne pas faire (trop) de fautes d'orthographe : en fait c'est quand même bien plus facile de l'orthographe !

8.3.1 Références pour aller plus loin

Pour une introduction un peu plus poussée aux règles essentielles de la typographie française, notamment pour les documents scientifiques, je recommande *le Petit typographe rationnel*¹¹ d'Édie SAUDRAIS, un fascicule de 15 pages résumant l'essentiel de ce qu'il faut savoir. En fait, même si vous décidez de ne plus utiliser \LaTeX après ce cours, je vous encourage quand même à lire (et relire) ce fascicule, car il me semble important de savoir rédiger des documents relativement corrects d'un point de vue orthotypographique, quel que soit le moyen technique utilisé.

Il existe de très nombreuses références concernant la typographie en général et ses règles en particulier. Pour un résumé des règles, l'Imprimerie nationale publie un *Lexique des règles typographiques* très pratique. Pour une référence plus exhaustive, le *Dictionnaire d'orthotypographie*¹² de LACROUX est disponible en ligne ou en librairies. Enfin, pour s'ouvrir aux aspects esthétiques de la typographie, *The elements of typographic style* de Robert BRINGHURST est un classique incontournable (et lui-même un très beau livre).

10. Dans son sens typographique, *espace* est féminin. Il fait partie de ces mots, qui comme *amour*, changent de genre selon le contexte ou l'usage.

11. <http://tex.loria.fr/typographie/saudrais-typo.pdf>

12. <http://www.orthotypographie.fr/>

9 Éléments de programmation

9.1 Rappels sur les définitions

Vous savez déjà créer et manipuler plusieurs sortes d'objets en \LaTeX : commandes et environnements avec `\newcommand` et `\newenvironment`, environnements numérotés avec `\newtheorem`, couleurs avec `\definecolor` et `\colorlet`, etc. Comme je l'ai déjà dit, la possibilité de pouvoir créer des objets pour adapter \LaTeX à ses besoins est un des éléments qui fait sa force.

Quand on connaît suffisamment bien les mécanismes internes de \LaTeX , on peut aussi modifier son comportement pour l'adapter à ses besoins. Les (re-)définitions de commandes ne suffisent pas toujours pour parvenir à ses fins. Dans ce chapitre, je présente tous les types d'objets manipulés par \LaTeX , qui vous permettront de créer et faire fonctionner des commandes complexes, ainsi que de mieux comprendre de contrôler \LaTeX .

Hormis les commandes (et environnements), il y a trois types d'objets qui intéressent le bricoleur \LaTeX ien : les compteurs, les longueurs et les boîtes.

9.2 Compteurs

Il y a toutes sortes d'éléments numérotés en \LaTeX : sections, sous-sections, théorèmes, équations, éléments d'un `enumerate`, pages, notes, etc. Derrière chacun d'entre eux se cache un compteur, c'est-à-dire une variable stockant la valeur d'un nombre entier.

En \LaTeX , le nom d'un compteur ne comporte que des lettres (sans `\` au début, contrairement à un nom de commande). Souvent, un compteur porte le même nom (sans la contre-oblique) que la commande servant à créer l'élément correspondant. La [table 9.1](#) donne les noms des compteurs pré-définis en \LaTeX .

<code>part</code>	<code>subsubsection</code>	<code>figure</code>	<code>enumi</code>
<code>chapter</code>	<code>paragraph</code>	<code>table</code>	<code>enumii</code>
<code>section</code>	<code>page</code>	<code>footnote</code>	<code>enumiii</code>
<code>subsection</code>	<code>subparagraph</code>	<code>equation</code>	<code>enumiv</code>

TABLE 9.1 — Compteurs de base de \LaTeX

En principe, vous n'avez pas besoin de toucher à ces compteurs, qui augmentent et se remettent éventuellement à zéro aux bons moments. Par contre, certains compteurs sont spécifiquement prévus pour régler certains aspects du document.

```
\setcounter{<nom>}{<valeur>}
\addtocounter{<nom>}{<valeur>}
secnumdepth tocdepth
```

Vous avez peut-être remarqué que les commandes de sectionnement ne produisent pas toutes de numéro ni d'entrée dans la table des matières : par exemple, en classe `article`, les sous-sections (`\subsubsection`) sont numérotées, mais pas les `\paragraph`. En fait, vous pouvez choisir vous-même à quel niveau s'arrête la numérotation, grâce à un code numérique : 1 pour `\section`, 2 pour `\subsection`, etc. C'est la valeur du compteur `secnumdepth` qui contrôle le dernier niveau numéroté. Par exemple, après `\setcounter{secnumdepth}{1}`, seules les sections seront numérotées, pas les sous-sections.

Par contre, les sous-sections apparaîtront toujours dans la table des matières. Si vous voulez que la table des matières se limite¹ aux sections, vous pouvez faire `\setcounter{tocdepth}{1}`. Ceci est indépendant du réglage de `secnumdepth`. Par ailleurs, si vous n'avez pas envie de retenir les codes numériques, vous pouvez juste dire `\addtocounter{tocdepth}{-2}` pour enlever deux niveaux de la table des matières par rapport aux réglages par défaut.

Plus généralement, vous l'aurez compris, `\setcounter` sert à régler la valeur d'un compteur, et `\addtocounter` permet de lui ajouter une valeur. Par exemple, si vous voulez numéroté vos sections en partant de 0, il vous suffit de dire `\setcounter{section}{-1}` avant la première section. En effet, le compteur est augmenté de 1 par `\section` juste avant d'être affiché.

En principe, vous devez donner un nombre entier comme *(valeur)*, pas une expression. Si vous chargez le module `calc`² par contre, vous avez le droit à des expressions arithmétiques simples : additions, soustractions, multiplications, parenthèses. Pour utiliser la valeur d'un compteur dans une expression, on utilise `\value{<nom>}`. Attention, `\value` ne sert que dans une expression, pas à afficher la valeur du compteur.

```
\the<nom>
\arabic{<nom>} \roman{<nom>} \Roman{<nom>}
\alph{<nom>} \Alph{<nom>} \fnsymbol{<nom>}
```

À chaque compteur est associé une commande spéciale pour afficher sa valeur, formée en ajoutant `\the` devant le nom du compteur. Par exemple, on affiche la valeur du compteur `page` avec la commande `\thepage`. Pour la plupart des compteurs, ceci affiche la valeur du compteur en chiffres arabes. En fait, la commande `\thesection` peut être redéfinie, comme n'importe quelle commande, pour modifier l'affichage du compteur.

Dans sa définition, nous allons avoir besoin de commande élémentaires pour afficher les valeurs de compteur. Celles-ci sont au nombre de six : `\arabic` pour les nombres arabes, `\roman` pour des chiffres romains minuscules, `\Roman` pour des chiffres romains majuscules, `\alph` pour des lettres minuscules, `\Alph` pour des lettres majuscules, et `\fnsymbol` pour des symboles, traditionnellement utilisés pour les notes de bas de page en typographie anglo-saxonne. L'exemple 9.1 montre comment les utiliser pour changer la représentation des compteurs de section et sous-section.

Comme on le constate, modifier `\thesubsection` change la représentation de `subsection` partout, y compris dans les références. Si on veut juste modifier la façon dont le numéro s'affiche dans le titre, il ne faut pas modifier `\thesubsection` (voir section 10.3 pour une solution).

Par ailleurs, remarquez que dans la définition de `\thesubsection` on a utilisé `\thesection` pour reprendre automatiquement la bonne représentation de `section`. Attention par contre à ne

1. Une table des matières courte au début du document peut être par exemple complétée par des tables détaillées au début de chaque section ; le module `minitoc` permet d'obtenir de telles tables des matières partielles. Si vous en avez besoin un jour, sachez que la documentation est disponible en français (`mini toc - fr . pdf`).

2. <http://ctan.org/pkg/calc>

<code>\renewcommand\thesection{\Roman{section}}</code>	I Une section
<code>\renewcommand\thesubsection{% \thesection-\roman{subsection}}</code>	I-i Une sous-section
<code>\section{Une section}</code>	Un peu de texte ici.
<code>\subsection{Une sous-section}</code>	
<code>\label{texte} Un peu de texte ici.</code>	I-ii Une autre sous-section
<code>\subsection{Une autre sous-section}</code>	
Il y avait du texte en~\ref{texte}.	Il y avait du texte en I-i.

EXEMPLE 9.1 — Modification de la représentation d'un compteur

pas utiliser une commande dans sa propre (re-)définition. Par exemple si vous voulez ajouter des parenthèses à la représentation du compteur `page`,

```
\renewcommand\thepage{(\thepage)}
```

ne fonctionnera pas, car \LaTeX appellera `\thepage` de façon récursif, et finira par échouer avec une erreur de dépassement de capacité. La bonne solution est bien sûr

```
\renewcommand\thepage{(\arabic{page})}
```

```
\newcounter{<nom>}
\refstepcounter{<nom>}
```

Vous pouvez créer vos propres compteurs, pour des objets numérotés personnels. Pour déclarer un nouveau nom de compteur, on utilise `\newcounter{<nom>}` : ceci permet ensuite d'utiliser `<nom>` comme un compteur ; par ailleurs il est initialisé avec la valeur 0. L'opération la plus courante à faire avec un compteur est d'augmenter sa valeur d'un cran : plutôt qu'`\addtocounter`, il faut utiliser `\refstepcounter`, qui s'occupe de quelques détails supplémentaires, comme de faire fonctionner les références. L'exemple 9.2 montre une utilisation basique d'un nouveau compteur pour des exercices numérotés.

Bien sûr, on aurait pu utiliser `\newtheorem` pour créer cet environnement numéroté. L'intérêt de savoir le faire « à la main » à partir des commandes de base de \LaTeX , est qu'on est beaucoup plus libre de modifier la mise en forme de l'environnement comme on le souhaite.

```
\newcounter{<nom>}[<parent>]
\numberwithin{<nom>}{<parent>}
```

En fait, en déclarant un nouveau compteur, on peut aussi lui attribuer un « parent », ce qui a l'effet suivant : à chaque fois que le parent est incrémenté (c'est-à-dire quand sa valeur augmente), notre compteur est remis à zéro. Par exemple, si dans l'exemple 9.2, on avait ajouté `[section]` à la fin de la première ligne, la numérotation des exercices reprendrait à 1 au début de chaque section. Bien sûr, pour rester cohérent, il faudrait redéfinir `\theex` pour y faire apparaître le numéro de section.

Le module `amsmath` fournit une commande bien utile, `\numberwithin`, qui permet d'attribuer un parent à un compteur existant, et de modifier en même temps sa représentation pour y ajouter celle du parent. Par exemple,

<pre> \newcounter{ex} \newenvironment{ex}{\refstepcounter{ex}% \par\noindent\textbf{Exercice \theex.}\quad \itshape}{\par} Du texte de remplissage pour commencer. \begin{ex} \label{facile} Lire l'énoncé de cet exercice en entier. \end{ex} \begin{ex} \label{dur} Démontrer le théorème de \bsc{Fermat}. \end{ex} L'exercice~\ref{facile} était facile, mais le~\ref{dur} est trop difficile. </pre>	<p>Du texte de remplissage pour commencer.</p> <p>Exercice 1. Lire l'énoncé de cet exercice en entier.</p> <p>Exercice 2. Démontrer le théorème de FERMAT.</p> <p>L'exercice 1 était facile, mais le 2 est trop difficile.</p>
--	--

EXEMPLE 9.2 — Utilisation d'un compteur personnel

Symbole	En millimètres	En points
<code>sp</code>	$5,363 \cdot 10^{-5}$ mm	$1,526 \cdot 10^{-5}$ pt
<code>pt</code>	0,3515 mm	1 pt
<code>bp</code>	0,3528 mm	1,00375 pt
<code>dd</code>	0,3528 mm	0,9346 pt
<code>cc</code>	3,942 mm	11,21 pt
<code>pc</code>	4,218 mm	12 pt
<code>in</code>	25,4 mm	72,27 pt
<code>cm</code>	10 mm	28,45 pt
<code>mm</code>	1 mm	2,845 pt
<code>ex</code>	<i>hauteur d'x de la fonte</i>	
<code>em</code>	<i>largeur d'M de la fonte</i>	

TABLE 9.2 — Unités de longueur

`\numberwithin{equation}{subsection}`

permet d'avoir des équations numérotées par sous-section.

9.3 Longueurs

Nous avons déjà eu à fournir des longueurs comme arguments de certaines commandes, et vu qu'elles étaient constituées d'un nombre décimal et d'une unité; nous connaissons d'ailleurs quelques-une des unités disponibles. Pour être complet, la [table 9.2](#) liste toutes les unités reconnues par \LaTeX , avec leur valeur en millimètres et en points.

Rappelons que les unités spéciales `ex` et `em` dépendent de la police actuellement utilisée (en particulier de sa taille), et représentent à peu près la hauteur d'un « x » et la largeur d'un « M » dans cette police. Elles sont utiles pour exprimer des longueurs qui doivent s'adapter à la taille du texte environnant.

De même qu'on peut définir et manipuler des compteurs (variables de type « nombre entier »), on peut définir et manipuler des longueurs (variables de type « longueur »). Les noms utilisés par les longueurs sont les mêmes que ceux des commandes, et commencent par une contre-oblique contrairement aux compteurs.

Plusieurs variables de longueur sont prédéfinies par \LaTeX : vous connaissez déjà `\linewidth` qui représente la largeur d'une ligne de texte à l'endroit où on se trouve. C'est une longueur dont on va en général utiliser la valeur sans la modifier³. En revanche, certaines longueurs sont prévues pour pouvoir être réglées, et \LaTeX fournit des commandes pour cela, analogues à celles prévues pour les compteurs.

```
\setlength{<nom>}{<longueur>}
\addtolength{<nom>}{<longueur>}
\fboxsep \fboxrule
```

On règle une longueur au moyen de la commande `\setlength`, et on peut lui ajouter une autre longueur au moyen de `\addtolength`. Dans les deux cas, le deuxième argument `<longueur>` peut être :

- une longueur constante : nombre et unité ;
- le nom d'une autre longueur, par exemple `\linewidth` ;
- le nom d'une autre longueur, précédé d'un facteur multiplicatif (sans signe de multiplication), par exemple `0.5\linewidth` pour la moitié de la largeur d'une ligne.

Comme pour les compteurs, avec le module `calc`, on a de plus droit à des expressions utilisant les opérations et des parenthèses. Ce n'est pas très utile pour les compteurs, c'est par contre extrêmement pratique pour les longueurs, comme on le voit à l'[exemple 9.7](#).

Nous avons déjà vu que la commande `\fbox` permet d'encadrer des éléments. Son comportement est paramétré par les deux longueurs `\fboxsep` qui représentent l'espacement entre le contenu et le cadre, et `\fboxrule`, l'épaisseur du trait. On peut donc facilement modifier l'apparence du cadre en réglant ces longueurs, comme le montre l'[exemple 9.3](#).

<pre>Par défaut : \fbox{blabla} \begin{flushright} \setlength\fboxsep{2\fboxsep} Deux fois moins serré : \fbox{blabla}\ \setlength\fboxsep{0pt} Totalement collé : \fbox{blabla}\ \setlength\fboxrule{2pt} Avec un trait épais : \fbox{blabla}\ \end{flushright} Retour à la normale : \fbox{blabla}</pre>	<p>Par défaut : blabla</p> <p>Deux fois moins serré : blabla</p> <p>Totalement collé : blabla</p> <p>Avec un trait épais : blabla</p> <p>Retour à la normale : blabla</p>
--	---

EXEMPLE 9.3 — Paramétrage de `\fbox`

Par ailleurs, cet exemple illustre le fait que les réglages de longueurs sont soumis aux règles de \LaTeX sur la portée : toutes les modifications faites à l'intérieur d'un groupe (un environnement

3. Modifier `\linewidth` n'aurait pas d'autre effet que créer des problèmes : en fait, c'est les dimensions des marges qu'il faut modifier pour changer la longueur de la ligne. `\linewidth` est maintenue à jour par \LaTeX pour notre confort.

par exemple) sont oubliées à la fin de groupe. En fait, les compteurs sont les seuls objets dont les modifications survivent à la fin d'un groupe, ce qui est indispensable : imaginez que le compteur `equation` soit remis à zéro à la fin de chaque environnement mathématique... Pensez à profiter de ce phénomène de portée pour les longueurs : si vous modifiez une longueur dans la définition de début d'un environnement, vous n'avez rien à faire à la fin (comme pour les changements de police).

```
\settoheight\langle nom \rangle{\langle matériel \rangle}
\settodepth\langle nom \rangle{\langle matériel \rangle}
```

Il existe trois façons supplémentaires de régler une longueur, qui permettent en fait de mesurer les dimensions d'un élément. Le sens de ces trois dimensions est expliqué en [section 9.4.1](#). Ces commandes sont utiles pour adapter une longueur à la taille de certains éléments de façon automatique. Par exemple, on peut l'utiliser pour fabriquer un texte à trous dont les trous mesurent exactement deux fois la taille des mots qu'ils remplacent, comme dans [l'exemple 9.4](#).

```
\newlength\lgtrou
\newcommand*\trou[1]{%
  \settoheight\lgtrou{#1}%
  \hspace*{2\lgtrou}}
\setlength\baselineskip{1.5\baselineskip}
On laisse un trou \trou{plus} grand que
le mot parce qu'une \trou{écriture}
manuscrite prend plus de \trou{place}. On
augmente \trou{aussi} l'interligne.
```

On laisse un trou grand que le mot
parce qu'une manuscrite
prend plus de . On augmente
l'interligne.

EXEMPLE 9.4 — Mesure de mots pour un texte à trous

On découvre aussi sur cet exemple la longueur `\baselineskip` qui représente l'écart entre les bases de deux lignes au sein d'un paragraphe. Il est acceptable de le modifier à la main pour un texte court, comme dans l'exemple, mais en général il faut mieux passer par le module `setspace`⁴, car sinon on perturbe aussi la mise en page des listes sans le vouloir.

Il existe d'autres longueurs susceptibles d'être modifiées. Par exemple, `\parindent` est la largeur du retrait en début de paragraphe : on peut la régler à `Opt` pour toujours supprimer ce retrait. (Pour le supprimer ponctuellement au début d'un paragraphe donné, on utilise `\noindent`, qui a déjà été vue dans plusieurs exemples).

Par contre, il faut éviter de modifier manuellement (avec `\setlength` ou `\addtolength` les longueurs qui représentent les dimensions de la page (voir la figure [page 32](#)). En effet, ceci est souvent source d'erreur car ces longueurs doivent satisfaire un certain nombre de contraintes. Il faut mieux, pour modifier la disposition des pages, utiliser les options du module `geometry`.

Nous verrons à la [section 9.4.5](#) d'autres longueurs qui contrôlent la mise en page des tableaux et qu'il peut être utile de modifier avec `\setlength`.

4. <http://ctan.org/pkg/setspace>

9.4 Boîtes

9.4.1 Concepts

Pour \LaTeX , tout est boîtes : il ne connaît pas les formes des caractères, qu'il considère comme des boîtes rectangulaires. Sur la [figure 9.1](#), \LaTeX ne voit que la première ligne, nous ne voyons que la dernière. De façon générale, une boîte est un rectangle, qui peut contenir du matériel plus ou moins compliqué, comme du vide, du noir, un caractère, ou d'autres boîtes.

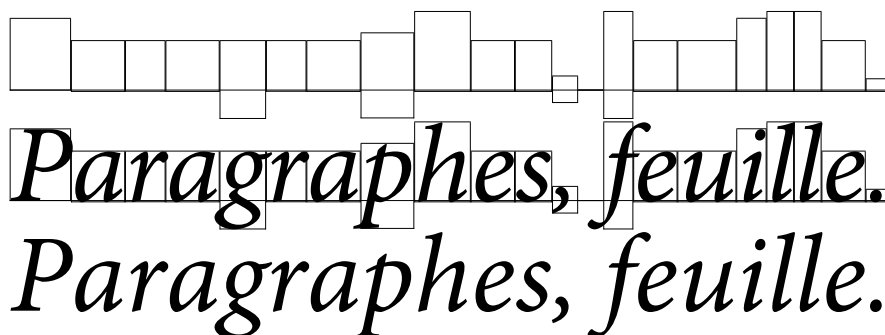


FIGURE 9.1 — Comment \LaTeX voit les caractères

En fait, une boîte est un tout petit peu plus qu'un rectangle, car elle possède en fait trois dimensions : une largeur, une profondeur et une hauteur, comme le montre la [figure 9.2](#). Sur une même ligne, les boîtes (par exemple des caractères) ont leurs points de base alignés horizontalement, et non par la bas, de façon à placer correctement les caractères qui débordent par le bas. C'est qui explique le besoin de trois dimensions pour mesurer une boîte. Parfois, on appellera hauteur totale la somme de la hauteur et de la profondeur.

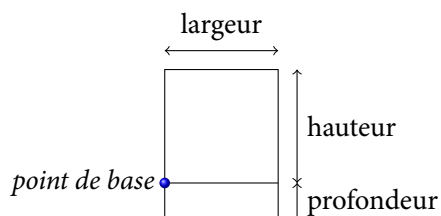


FIGURE 9.2 — Les trois dimensions des boîtes

Les trois prochaines sections examinent les boîtes et les commandes permettant de les manipuler en fonction de leur contenu : une boîte horizontale contient d'autres boîtes dont les points de base sont alignés horizontalement, une boîte verticale contient des boîtes alignées verticalement (par exemple des paragraphes, qui sont des empilements verticaux de boîtes horizontales (les lignes) et les règles sont des boîtes remplies en noir.

9.4.2 Boîtes horizontales

```
\mbox{<contenu>}
\makebox[<largeur>][<lcrs>]{<contenu>}
```

La commande la plus simple pour produire une boîte horizontale est `\mbox`. Elle n'a aucun effet visible (pas de cadre autour de la boîte, par exemple), mais rend par contre impossible de couper en deux le contenu. Elle est donc seulement utile pour empêcher un mot (un nom propre⁵ ou un sigle) d'être coupé.

Une commande un peu plus sophistiquée est `\makebox` qui permet de construire une boîte de largeur donnée. On peut alors préciser comment le contenu doit être réparti dans cette largeur : `l` pour l'aligner à gauche, `r` aligné à droite, `c` centré, et `s` (*stretch*) étiré. L'option `s` posera des problèmes si la différence entre la dimension naturelle du contenu et la largeur demandée est trop grande, et que le contenu ne possède pas assez d'éléments susceptibles d'être étirés ou comprimés.


La figure 9.1 montre un *f* qui déborde de sa boîte. Plus généralement, en \LaTeX , rien n'impose au contenu d'une boîte d'être physiquement contenu dans le rectangle associé, même si c'est généralement le cas. Un cas extrême est celui des boîtes de largeur nulle, qui permettent de placer du matériel à un endroit mais en faisant ensuite comme s'il n'était pas là : on peut ainsi obtenir des effets particuliers, comme ceux de l'exemple 9.5.

<pre>Du texte pour voir la marge.\ \makebox[0pt][r]{Bouh }ça fait \texttt{\makebox[0pt][l]{xxxx}peur} plaisir.\ On s'amuse avec les boîtes.</pre>	<p>Du texte pour voir la marge. Bouh ça fait XXXX plaisir. On s'amuse avec les boîtes.</p>
---	---

EXEMPLE 9.5 — Utilisation de boîtes de largeur nulle

```
\makebox[⟨largeur⟩][⟨lcrs⟩]{⟨contenu⟩}
\raisebox{⟨décalage⟩}[⟨hauteur⟩][⟨profondeur⟩]{⟨contenu⟩}
```

Pour obtenir une boîte encadrée, on peut utiliser `\fbox` qui est analogue à `\mbox`, ou bien `\framebox` qui correspond à `\makebox` dont il partage la syntaxe. Ceci permet d'obtenir des cadres de largeur donnée, comme dans l'exemple 9.6.

<pre> \setlength\fboxrule{2pt} Du texte d'abord pour visualiser la largeur des lignes de texte.\ \framebox[\linewidth] [c]{% \bfseries Coucou !}\ Puis une image \raisebox{-1cm}{% \includegraphics[height=2cm]{lion}} centrée sur la ligne.</pre>	<p>Du texte d'abord pour visualiser la largeur des lignes de texte.</p> <div style="border: 2px solid black; text-align: center; padding: 5px; width: fit-content; margin: 0 auto;"> Coucou ! </div>  <p>Puis une image centrée sur la ligne.</p>
--	---

EXEMPLE 9.6 — Utilisation de `\framebox` et `\raisebox`

La dernière commande de gestion des boîtes horizontales que nous verrons est `\raisebox`, qui permet de décaler une boîte par rapport à la ligne de base. Elle est utile pour régler d'alignement avec des objets volumineux comme des images (exemple 9.6), ou pour obtenir des effets spéciaux. On peut tricher sur la hauteur et la profondeur de la boîte obtenue (typiquement, les régler à `0pt` pour ne pas en tenir compte) grâce aux arguments optionnels.

5. Si on utilise `\bsc` pour les noms de famille, il n'y a pas besoin de rajouter `\mbox`, car `\bsc` empêche déjà les coupures.

9.4.3 Boîtes verticales

```
\begin{minipage}[<tc>][<hauteur>][<tcbs>]{<largeur>}
  <contenu>
\end{minipage}
```

Dès qu'on veut pouvoir aller à la ligne dans une boîte, il faut que ce soit une boîte verticale. La façon la plus simple de les construire est d'utiliser l'environnement `minipage`. L'argument obligatoire est la largeur de la boîte : en effet, \TeX va couper automatiquement les lignes, il faut donc savoir à quelle largeur couper. Notons qu'on peut parfaitement mettre une boîte verticale dans une boîte horizontale, comme le montre l'exemple 9.7 (observez au passage le calcul de la largeur de la minipage, pour laisser la place au cadre, en utilisant `calc`).

<pre>\usepackage{calc} Un peu de texte d'abord pour bien voir la largeur disponible.\par\noincent \fbbox{\begin{minipage}{% \linewidth-2\fbboxsep-2\fbboxrule} Plusieurs lignes de texte encadrées, le cadre prenant toute la largeur d'une ligne.\end{minipage}} Encore un peu de texte pour mieux voir la largeur disponible.</pre>	<p>Un peu de texte d'abord pour bien voir la largeur disponible.</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p>Plusieurs lignes de texte encadrées, le cadre prenant toute la largeur d'une ligne.</p> </div> <p>Encore un peu de texte pour mieux voir la largeur disponible.</p>
---	---

EXEMPLE 9.7 — Une boîte verticale dans une boîte horizontale

Le premier argument optionnel de l'environnement `minipage` sert à déterminer comment elle sera alignée avec les autres éléments de la ligne : `t` par le haut (*top*), `c` par le centre (*center*) et `b` par le bas (*bottom*). Le deuxième permet d'imposer une hauteur, et le troisième sert alors à indiquer comment le contenu sera réparti dans cette hauteur : `t`, `c` ou `b` pour le placer en haut, au centre ou en bas, et `s` pour tenter de l'étirer (comme pour `\makebox`, ceci peut causer des problèmes).

9.4.4 Réglures

```
\rule[<décalage>]{<largeur>}{<hauteur>}
\hrulefill
```

Les réglures sont les boîtes les plus simples : ce sont des rectangles noirs pleins. On les construit avec la commande `\rule` en spécifiant leur hauteur et leur largeur. On peut, de plus, décaler le rectangle ainsi produit vers le haut ou vers le bas (décalage négatif) grâce à l'argument optionnel, sans avoir besoin d'utiliser `\raisebox`.

<pre>Un centimètre : \rule[.5ex]{1cm}{.4pt}\ Du \hrulefill{} remplissage.</pre>	<p>Un centimètre : ———</p> <p>Du _____ remplissage.</p>
---	---

EXEMPLE 9.8 — Réglures

Placer des gros rectangles noirs sur la page n'est pas très intéressant. Aussi, la plupart du temps, on utilisera que des réglures dont une dimension est très petite, et l'autre plus grande : on obtient ainsi un « trait ». Dans l'exemple 9.8, on utilise une hauteur de `0.4pt`, la valeur par défaut de `\fboxrule`, qui convient en général bien comme épaisseur de trait. De telles lignes sont par exemple utiles pour structurer une page de titre ou mettre en valeur des éléments particuliers.

Pour obtenir un trait horizontal occupant toute la place restant disponible sur la ligne, on dispose de la commande `\hrulefill` qui construit une réglure étirable, de façon similaire aux espaces étirables obtenus avec `\stretch`.

Une autre utilisation courante des réglures consiste à placer des réglures de largeur nulle (ou d'épaisseur nulle), donc invisibles, à ces endroits bien choisis, pour réserver de la place. Nous en parlerons dans la prochaine section.

9.4.5 Aparté : espacement dans les tableaux

Dans le chapitre sur les tableaux, nous avons laissé de côté le réglage de différents paramètres, et le contrôle de l'espacement. Nous avons maintenant tous les outils pour y revenir.

```
\tabcolsep \arraycolsep
\doublerulesep
\arrayrulewidth
```

Les quatre longueurs ci-dessous, réglables avec `\setlength`, contrôlent divers aspects de l'apparence des tableaux. La première, `\tabcolsep`, représente la moitié de l'espace séparant deux colonnes. Autrement dit, c'est l'espace séparant le contenu d'une case et le bord de la case : il est tout à fait semblable à `\fboxsep`. Le longueur `\arraycolsep` joue le même rôle, mais pour les tableaux mathématiques construits avec l'environnement `array`.

Quand deux filets (verticaux ou horizontaux) se succèdent dans un tableau, leur espacement est donnée par `\doublerulesep`. Enfin, l'épaisseur des filets (simple ou doubles) eux-même est contrôlée par `\arrayrulewidth`. Attention, les noms ne sont pas cohérents : `\arrayrulewidth` est utilisé par `tabular` et par `array`, alors que pour la séparation des colonnes, on distingue `\tabcolsep` de `\arraycolsep`.

```
\extrarowheight
\arraystretch
```

Il arrive que dans un tableau, les lignes soient trop proches les unes de autres. En effet, \TeX possède des règles assez sophistiquées pour garantir un bon espacement entre les lignes d'un paragraphe, mais elles ne s'appliquent pas aux tableaux. Deux⁶ paramètres sont fournis pour augmenter globalement l'espace entre toutes les lignes d'un tableau.

Le premier, `\extrarowheight`, est une longueur (et se règle donc avec `\setlength`). Si elle est positive, un espace de cette dimension est ajouté en haut de chaque ligne. S'il n'y a pas de filet entre les lignes, on peut donc considérer qu'elle sont écartées de `\extrarowheight`. S'il y a un filet, l'espace est ajouté entre le haut du contenu de la ligne et le filet supérieur : ceci règle par exemple

6. En fait, `\extrarowheight` n'existe pas en standard : il est fourni par le module `array`. Mais celui-ci est tellement utile que je suppose que vous le chargez toujours quand votre document comporte des tableaux...

le problème des majuscules qui touchent le filet supérieur, mais pas les problèmes d'espacement avec le filet inférieur.

Le deuxième paramètre, `\arraystretch` n'est pas une longueur, ni un compteur, mais une commande, qui sert à stocker un nombre décimal (par défaut 1). Pour le régler, on fait par exemple

```
\renewcommand\arraystretch{1.5}
```

Il contrôle l'espace vertical minimum réservé pour une ligne. D'habitude, chaque ligne d'un tableau dispose d'au moins `0.7\baselineskip` en hauteur et `0.3\baselineskip` en profondeur (rappelons que `\baselineskip` est le déplacement entre deux lignes de texte au sein d'un paragraphe). `\arraystretch` est un facteur multiplicatif qui s'applique à cet espace minimum réservé : par exemple, s'il vaut 2, la hauteur minimum passera à `1.4\baselineskip` et la profondeur minimum à `0.6\baselineskip`.

Les deux paramètres précédents sont à régler avant le tableau, et s'appliquent à toutes les lignes. (En fait, ils s'appliqueront même aux tableaux suivants, sauf si les modifications sont faites au sein d'un environnement qui restreindra leur portée.) Pour régler un problème d'espacement sur une ligne précise, le plus simple est souvent d'insérer une réglure invisible. En jouant sur la hauteur totale et le décalage, on réserve de la place en haut et en bas de la ligne. Par exemple, insérer `\rule[-20pt]{0pt}{50pt}` dans n'importe quelle cellule d'une ligne réserve une place de 20pt en bas et 30pt en haut.

9.4.6 Remplissage de boîtes

Terminons ce tour d'horizon des boîtes par des explications sur quelques problèmes que vous avez peut-être déjà rencontrés : le mauvais remplissage de certaines boîtes, signalé par des messages comme

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

Les éléments intéressants dans ce message sont :

- le type de problème, indiqué au début. Ici, c'est une boîte horizontale (`\hbox`) dans un paragraphe (une ligne de texte, donc), qui n'est pas assez remplie (`Underfull`) ;
- l'emplacement dans le code source où le problème a été repéré. En général, c'est une indication assez exacte ;
- un extrait du texte que \TeX n'arrive pas à disposer de façon à bien remplir la ligne, avec les points de coupure possibles matérialisés par des traits d'union.

Comme une boîte peut être horizontale ou verticale, et pas assez ou au contraire trop remplie, il y a quatre cas que je vais présenter avec leur cause probable et des approches possibles pour trouver une solution.

1. `Underfull \hbox` : la plupart du temps, causé par du texte justifié sur une largeur trop étroite, par exemple dans une note marginale, une colonne d'un tableau, une minipage, etc. La solution est simple : ne pas justifier le texte à cet endroit, l'aligner seulement à gauche ou à droite, ou le centrer (voir [section 2.4](#)).
2. `Overfull \hbox` : le problème le plus courant et le plus embêtant. Souvent causé par un élément long et insécable (morceau de verbatim, formule, texte rendu insécable) dans la ligne de texte, mais survient parfois dans des paragraphes sans rien de bizarre. L'algorithme

de \LaTeX est très bon pour résoudre ces problèmes, et s'il n'y est pas arrivé tout seul, c'est souvent le signe qu'il n'y a pas de solution simple. Si vous êtes l'auteur du texte, le plus simple est de changer un ou deux mots ou la tournure de phrase en espérant que ça passe mieux. Sinon, selon le passage concerné, vous pouvez parfois jouer sur les marges ou ne pas justifier le texte à cet endroit. Une astuce pour repérer le problème dans le document : régler la longueur `\overfullrule` à, par exemple `5pt` pour que \LaTeX mette le problème en évidence par un rectangle noir à côté.

3. **Underfull `\vbox`** : un autre problème courant et parfois délicat. Cause : à un moment, \LaTeX doit changer de page, pour placer un élément important et qui ne peut pas être coupé, mais il n'y a pas assez de texte sur la page en cours pour la remplir. Si c'est possible, une solution est de faire flotter l'élément perturbateur. Sinon, on peut essayer de modifier les coupures des pages précédentes avec `\pagebreak` pour ramener un peu plus de matériel sur la page en cours. Il est aussi parfois possible d'allonger un paragraphe d'une ligne en le faisant précéder de `\looseness=1`, si sa dernière ligne était déjà bien remplie. Le reste du temps, on est bien embêté. Si on est l'auteur du texte, on peut le modifier légèrement.
4. **Overfull `\vbox`** : un problème bien moins courant. Il survient par exemple en classe `beamer`, où aucune coupure de page n'est faite automatiquement, quand on essaie de placer trop de choses sur une seule page. La solution est claire : mettre moins de choses ! Éventuellement, on peut tasser un peu le matériel présent avec des `\vspace` de longueur négative entre les différents éléments, mais ce n'est pas très propre.

Depuis le début du cours, je vous répète que tout document doit compiler sans erreurs. Les avertissements (*warning*) par contre sont parfois anodins (c'est en général le cas pour ceux liés aux fontes), parfois pas : il faut les examiner un par un. Voici la troisième règle : il ne doit rester aucun problème de remplissage de boîte dans la version finale de vos documents. À la rigueur, on peut accepter des problèmes de petite ampleur, s'il est certain qu'on ne les voit pas à l'œil nu, mais c'est rare.

Souvent, il vaut mieux ne pas se préoccuper des problèmes de boîtes pendant la rédaction et la mise au point du document : parfois, ils disparaîtront d'eux-même avant la fin, ou bien une solution apportée sur une première version du document ne fonctionnera plus après. Gardez donc ces problèmes, souvent épineux, pour la fin. (Les erreurs de compilations restent à régler immédiatement, bien sûr.)

10 Personnalisation

10.1 Polices

Nous avons vu en [section 2.2](#) le modèle de gestion des fontes de \LaTeX , et comment changer de police au sein d'un document en suivant ce modèle. En particulier, trois familles sont disponibles : avec empattements, sans empattements, et façon machine à écrire. Or, il existe de très nombreuses polices avec empattements, comme *Times*, *Computer modern roman* (la police par défaut de \LaTeX), *Garamond* (utilisée dans la Pléiade), *Minion* (utilisée dans ce poly), etc. Comment choisir laquelle sera utilisée en tant que `\rmfamily` ?

En général, cela se fait en chargeant un module, qui s'occupe de tous les détails nécessaires (parfois nombreux). Certains modules modifient les trois familles de police, d'autres n'en modifient qu'une. Par ailleurs, certains possèdent des options permettant d'accéder à des variantes. Comme \LaTeX est souvent utilisé pour des documents contenant des mathématiques, il est aussi important de prévoir une police contenant suffisamment de symboles mathématiques.

Voici une liste de quelques modules classiques permettant de changer les polices de base.

- **lmodern** change les trois familles, pour des variantes un peu retravaillées (familles *Latin Modern*) des polices par défaut de \LaTeX (les familles *Computer modern*). Si les polices par défaut vous conviennent, il est conseillé de toujours utiliser ce module : en effet il vaut mieux sélectionner explicitement cette variante de police, plutôt que de se reposer sur le choix de \LaTeX , qui pourra varier d'une installation à l'autre (surtout sur des installations vieilles ou incomplètes).
- **kpfonts** est un autre triplet de polices spécialement développées pour \LaTeX . Il s'agit de familles extrêmement complètes, comprenant beaucoup de symboles mathématiques. Le module propose des options pour accéder à des variantes, trop nombreuses pour être présentées ici : je renvoie donc à sa documentation.
- **fourier** modifie la police romaine (`\rmfamily`) et les polices mathématiques. C'est une version de la police souvent appelée *Utopia*. Le module, développé par un Français, propose une option (`upright`) pour adapter la composition des math aux traditions françaises, avec les lettres grecques et les majuscules droites¹ (et non en italique). (Des options similaires existent par les nombreux réglages de **kpfonts**, en fait.)
- **mathpazo** change la police romaine pour utiliser *Palatino* et inclus des polices mathématiques adaptées. Il propose des variantes, comme les chiffres minuscules avec l'option `osf`.
- **mathptmx** change la police romaine et les polices mathématiques pour le (trop ?) célèbre *Times*.
- **helvet** change la police sans empattements `\sffamily` pour *Helvetica* (dont une variante est connue sous le nom d'*Arial*). L'option `scaled` est souvent utile pour ajuster la taille de la police, qui a souvent l'air plus grosse que la famille romaine.

1. Vous pouvez d'ailleurs remarquer que c'est le réglage qui a été retenu dans ce polycopié, avec l'option `frenchmath` du module **MinionPro**.

Nom privé	Nom courant	Version italique
<code>lmr</code>	Latin modern roman	<i>Latin modern roman</i>
<code>jkp</code>	Kepler roman	<i>Kepler roman</i>
<code>futs</code>	Fourier (Utopia)	<i>Fourier (Utopia)</i>
<code>ppl</code>	Palatino	<i>Palatino</i>
<code>ptm</code>	Times	<i>Times</i>
<code>lmss</code>	Latin modern sans	<i>Latin modern sans</i>
<code>jkpss</code>	Kepler sans	<i>Kepler sans</i>
<code>phv</code>	Helvetica	<i>Helvetica</i>
<code>lmtt</code>	Latin modern typewriter	<i>Latin modern typewriter</i>
<code>jkptt</code>	Kepler typewriter	<i>Kepler typewriter</i>
<code>ul9</code>	Luximono	<i>Luximono</i>
<code>pzc</code>	<i>Zapf chancery</i>	<i>Zapf chancery</i>

TABLE 10.1 — Exemples de polices avec leurs noms

Vous pouvez bien sûr combiner ces paquets entre eux, par exemple charger `mathptmx` et `helvet` pour utiliser le duo *Times* et *Arial*². Tous les paquets cités proposent des polices de bonne qualité, raisonnablement riches, et couramment disponibles dans les distributions T_EX. Je vous encourage à en essayer quelques-uns pour varier un peu l'apparence de vos documents.

Malheureusement, la gestion des polices sous L^AT_EX est très compliqué pour des raisons techniques et historiques. À moins d'être un expert, il est en pratique impossible d'utiliser une police choisie au hasard avec L^AT_EX : il faut que les nombreux fichiers de support nécessaires aient été créés auparavant.

Heureusement, un nombre relativement grand de polices avec le support nécessaire pour L^AT_EX existe. Elles sont toutes listées, avec des exemples et la façon de les utiliser en L^AT_EX, dans un [catalogue des polices pour L^AT_EX](#)³ disponible en ligne.

```
\fontfamily{nom cryptique}\selectfont
```

Il est possible d'utiliser une autre famille que les trois polices principales du document, avec `\fontfamily`. Cette commande présente deux particularités : la première est qu'elle doit impérativement être suivie par `\selectfont` pour avoir un effet. La deuxième est qu'on ne peut pas directement utiliser le nom « humain » de la police, mais qu'on doit utiliser son nom privé « L^AT_EXien ». Celui-ci est parfois donné dans le catalogue, ou se trouve en chargeant le module pour cette fonte, et en regardant⁴ le contenu de `\rmdefault` pour une police romaine (`\sfdefault` ou `\ttdefault` sinon).

2. Ajoutez la commande `\sloppy` pour que L^AT_EX remplisse les lignes n'importe comment, et votre document commencera à ressembler à quelque chose fait avec Word — ce n'est évidemment pas mon choix préféré.

3. <http://www.tug.dk/FontCatalogue/allfonts.html>

4. Par exemple, créez un document de test qui charge ce module, tapez `\rmdefault` dans le source et regardez le résultat dans le pdf. En fait, la façon officielle de trouver les noms est de regarder dans le document `fontname.pdf`. En pratique, ce document est difficile à exploiter, et je trouve la méthode précédente plus commode.

Module	Commande	Nom
calligra	\calligra	<i>Calligra</i>
humanist	\hminfamily	Humanift
yfonts	\gothfamily	Gotic
foekfont	\foekfamily	FOEK

TABLE 10.2 — Polices fantaisistes

La [table 10.1](#) montre quelques exemples de fontes, dont celles citées ci-dessus ; la [table 10.2](#) montre des fontes plus fantaisistes, choisies dans le catalogue, pour vous convaincre que les possibilités de \LaTeX ne sont pas limitées aux documents sérieux (même si c’est principalement pour ça qu’il est fait).

Après cette débauche de fontes diverse et variées, j’aimerais lancer un appel à l’ordre : dans vos documents sérieux, choisissez vos fontes une fois pour toutes dans le préambule, n’en changez pas à tire-larigot, et n’utilisez pas trop de polices différentes. Comme d’habitude, un document sobre et cohérent est plus efficace qu’un fouillis baroque.

10.2 En-têtes et pieds de page

En classe [article](#), le numéro de page est automatiquement inséré en bas et au milieu de chaque page. En classe [book](#), le numéro de page est en haut et à l’extérieur, tandis qu’à l’intérieur figure le nom du chapitre en page de gauche, et le nom de la section en page de droite (sauf pour les première page de chapitre qui n’ont rien en en-tête et le numéro de page en pied au milieu). Ces réglages par défaut sont raisonnables, mais il est possible de les modifier, indépendamment de la classe, grâce au module [fancyhdr](#)⁵.

```
\pagestyle{fancy}
\lhead{<contenu>} \chead{<contenu>} \rhead{<contenu>}
\lfoot{<contenu>} \cfoot{<contenu>} \rfoot{<contenu>}
\renewcommand\headrulewidth{<longueur>}
\renewcommand\footrulewidth{<longueur>}
```

Pour profiter des fonctionnalités de [fancyhdr](#), il faut penser à utiliser `\pagestyle{fancy}` : le mieux est de le faire dans le préambule, juste après le chargement du module, pour être sûr de ne pas oublier. On peut ensuite attribuer un contenu à six zones : en-tête ou pied, gauche, centre ou milieu, à l’aide des commandes `\{l,c,r\}{head,foot}`. Certaines ont un contenu prédéfini : on peut alors les supprimer en les remplaçant par le contenu vide, comme le montre l’[exemple 10.1](#).

Il est de plus possible de séparer l’en-tête et le pied du corps de la page, en définissant l’épaisseur du trait utilisé (`\Opt` pour le supprimer). Attention, il ne s’agit pas d’une longueur, mais d’une commande utilisée pour stocker une longueur : on n’utilise donc pas `\setlength`. (Oui, ça a l’air bizarre, en fait il y a de bonnes raisons techniques à cette bizarrerie.)

5. <http://ctan.org/pkg/fancyhdr>

```

\usepackage{fancyhdr} \pagestyle{fancy}
\usepackage{lipsum, xcolor, lastpage}
\rhead{} \chead{\itshape Mon mémoire}
\lfoot{compilé le \today}
\rfoot{\thepage} sur \pageref{LastPage}
\renewcommand\headrulewidth{0pt}
\renewcommand\footrulewidth{.4pt}
\lipsum

```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum

compilé le 26 juillet 2009

1

1 sur 4

EXEMPLE 10.1 — En-têtes et pieds de page

Pour les documents recto-verso, on peut avoir intérêt à utiliser plutôt les commandes alternatives `\fancyfoot` et `\fancyhead` dont l'argument optionnel permet de spécifier des listes de position dépendant de la parité de la page. Ainsi,

```
\fancyfoot[LE,RO]{\thepage}
```

affiche le numéro de page dans le pied de page à l'extérieur : **LE** signifiant à gauche les pages paires (*left even*) et **RO** à droite les pages impaires (*right odd*).

Le module `lastpage` permet d'insérer le nombre total de pages, comme dans l'exemple 10.1. Techniquement, il n'a pas de rapport avec les en-têtes et pied de pages, mais c'est souvent dans ce contexte qu'on l'utilise, et c'est pourquoi je le signale ici.

Par ailleurs, on peut, dans les en-têtes et pieds, accéder au numéro et titre de la section en cours avec `\rightmark` et au chapitre avec `\leftmark`. Les noms reflètent le comportement par défaut de la classe `book` qui les place en effet dans ces positions.

10.3 Titres de chapitres et sections

Pour modifier l'apparence des titres de chapitre et de sections, on dispose de plusieurs moyens. Pour les chapitres uniquement, le module `fncychap`⁶ propose quelques thèmes prédéfinis, et assez sophistiqués. La documentation du module contient des exemples de chacun de ces style : je ne vois pas l'intérêt de reproduire ces exemples ici, et vous invite donc à aller les consulter directement à la source⁷. Pour sélectionner un thème, il suffit de donner son nom comme option au chargement du module.

```
\usepackage[options globales]{titlesec}
```

6. <http://ctan.org/pkg/fncychap>

7. D'autant plus que pour une fois il n'y a même pas besoin de comprendre l'anglais : il s'agit juste de regarder à quoi ressemblent les thèmes, et leurs noms.

Pour modifier l'apparence des titres à tous les niveaux (chapitre, section, sous-section, etc.), le module `titlesec`⁸ est le plus adapté. La façon la plus simple de l'utiliser est de donner des options globales qui s'appliquent à tous les niveaux de titre. Voici quelques exemples d'options, regroupés par famille :

- `bf`, `it`, `sc`, `sf`, etc. Ces options permettent de sélectionner la fonte utilisée pour les titre. Ces codes de deux lettres devraient vous être familiers ; dans le cas contraire, révisez la [table 2.3](#) ;
- `big`, `medium`, `small` et `tiny` permettent de régler la taille des titres (taille de fonte et espaces verticaux). La taille normale est `medium` ; `big` c'est plus grand, `small` plus petit et `tiny` beaucoup plus petit ;
- `raggedright`, `center` et `raggedleft` permettent de choisir la position des titres : `center` au centre, `raggedleft` à droite, etc.

```
\titleformat*{\commande de section}{\commandes de fonte}
\titlelabel{\matériel}
```

Pour plus de finesse, on peut modifier la police (ou la couleur utilisée) pour chaque niveau de sectionnement. Par exemple,

```
\titleformat*{\section}{\color{red}\sshape}
\titleformat*{\subsection}{\itshape}
```

donnera des titres de section en petites capitales rouges, et des titres de sous-sections en italiques.

Dans la [section 9.2](#) j'avais dit que pour modifier l'affichage du numéro de section dans le titre, il ne fallait pas modifier `\thesection`, car cela modifier aussi son affichage dans les références. Voici une bonne solution pour les cas simples : utiliser `\titlelabel`. Cette commande permet de redéfinir comment sont affichés les numéros de sections, sous-sections, etc. dans les titres. On lui donne en argument la façon d'afficher le numéro, en utilisant `\thetitle` à la place de `\thesection` ou `\thesubsection` ou ... Par exemple,

```
\titlelabel{\fbox{\thetitle}\quad}
```

encadrera le numéro et laissera un espace d'un `em` après, dans les titres de tous les niveaux.

Il est possible d'obtenir des réglages plus fins, par niveau de titre, et de modifier de fond en comble leur présentation, mais nous en resterons là. Comme d'habitude, si vous voulez connaître tous les détails, lisez⁹ la documentation de `titlesec`.

10.4 Listes

TeX fournit trois environnements de liste, chacun bénéficiant d'une mise en page par défaut convenable. On peut toutefois vouloir modifier ces mises en pages, ou bien créer ses propres environnements de liste comme variantes de types standard. Pour tout cela, le module le plus puissant est `enumitem`¹⁰. Pour modifier seulement les listes numérotées, le module `enumerate`¹¹ est parfois plus facile à utiliser, même s'il est en général moins puissant ; je décris seulement `enumitem` ci-dessous.

8. <http://ctan.org/pkg/titlesec>

9. Si vous ne lisez pas encore très bien l'anglais, c'est l'occasion de pratiquer !

10. <http://ctan.org/pkg/enumitem>

11. <http://ctan.org/pkg/enumerate>

```

\setitemize[niveau]{option=valeur, <...>}
\begin{itemize}{option=valeur, <...>}
\setenumerate[niveau]{option=valeur, <...>}
\begin{enumerate}{option=valeur, <...>}
\setdescription[niveau]{option=valeur, <...>}
\begin{description}{option=valeur, <...>}

```

Il y a deux façons de régler les listes avec `enumitem` : la première consiste à gérer globalement l'apparence de tous les `itemize` avec `\setitemize` (et ainsi de suite pour les autres environnements). Dans ce cas, on peut effectuer des réglages différents selon le niveau de la liste (par exemple, un `itemize` qui apparaît dans une autre liste utilisera les réglages du niveau 2). Sinon, on peut modifier un environnement donné, en lui passant directement la liste des options voulues en argument optionnel, comme le montre l'exemple 10.2.

<pre> \usepackage{enumitem} \setenumerate[1]{label=\Roman*} \setenumerate[2]{label*=-\roman*} \frenchbsetup{StandardLists=true} \begin{enumerate} \item D'abord, pour commencer : \begin{enumerate} \item Le chef a toujours raison. \item J'ai dit toujours. \end{enumerate} \item Ensuite, relire la première règle. \end{enumerate} Un peu de texte, pour se reposer de toutes ces énumérations, puis on reprend. \begin{enumerate}[resume] \item Pour finir, vive le chef ! \end{enumerate} </pre>	<p style="margin-left: 2em;">I D'abord, pour commencer :</p> <p style="margin-left: 4em;">I-i Le chef a toujours raison.</p> <p style="margin-left: 4em;">I-ii J'ai dit toujours.</p> <p style="margin-left: 2em;">II Ensuite, relire la première règle.</p> <p style="margin-left: 2em;">Un peu de texte, pour se reposer de toutes ces énumérations, puis on reprend.</p> <p style="margin-left: 2em;">III Pour finir, vive le chef !</p>
--	---

EXEMPLE 10.2 — Listes personnalisées

Cet exemple illustre quelques options, comme `label` pour choisir la façon dont l'étiquette est fabriquée à partir du numéro. Dans ce cas, une étoile après `\Roman` ou tout autre commande d'affichage de compteur, est automatiquement remplacée par le nom de compteur approprié (`enumi`, `enumii`, etc. suivant le niveau). Par ailleurs, `label*` insère le le numéro du niveau de liste précédent devant celui du niveau courant. Enfin, `resume` permet de reprendre la liste précédente, en conservant une numérotation continue.

Le module `enumitem` propose bien d'autres possibilités, pour lesquelles je renvoie à sa documentation.

Attention, ce module est incompatible avec l'option `french` de `babel`, car les deux essaient de modifier les environnements de liste de \LaTeX (`enumitem` pour les rendre configurables, et `babel` pour les adapter aux usages du français). Pour pouvoir utiliser `enumitem` dans un document fran-

çais, il faut malheureusement renoncer aux réglages des listes par `babel` en incluant la ligne ¹²

```
\frenchsetup{StandardLists=true}
```

dans son préambule après le chargement de `babel` avec l'option `french`.

10.5 Flottants

Par défaut, \LaTeX définit deux types de flottants : `figure` et `table`. Les environnements associés ne prennent en charge que le placement du flottant et la production du titre. Il faut faire la mise en page à la main, par exemple en plaçant un `\centering` dans chaque figure ou table pour les centrer.

```
\floatsetup[⟨type⟩]{⟨option⟩=⟨valeur⟩, ⟨...⟩}
\DeclareNewFloatType{⟨nom⟩}{⟨option⟩=⟨valeur⟩, ⟨...⟩}
```

Il est très fastidieux, et source d'erreurs, de recopier à la main les commandes de mise en page à chaque figure. Heureusement, il est possible d'automatiser ceci grâce au module `floatrow` ¹³ et sa commande `\floatsetup`. Cette dernière prend un certain nombre d'options, que je ne détaillerai pas ici : je renvoie à la documentation, comme d'habitude dans ce chapitre.

Par ailleurs, il est possible de créer de nouveaux types de flottants, pour des exemples, des codes source, etc. Pour la création du présent polycopié, `floatrow` a été utilisé, ainsi que les commandes listées au [source 10.1](#).

```
\DeclareNewFloat{fltexa}{name=\bsc{Exemple}}
\DeclareNewFloat{fltsrc}{name=\bsc{Source}}
\numberwithin{fltexa}{chapter}
\numberwithin{fltsrc}{chapter}
\floatsetup[fltexa]{precode=rule, midcode=rule}
\floatsetup[fltsrc]{precode=rule, midcode=rule}
\floatsetup[figure]{justification=centering}
\floatsetup[table]{justification=centering}
```

SOURCE 10.1 — Création et gestion des types de flottants dans ce poly

12. Qui ne fonctionne qu'avec une version suffisamment récente de `babel`.

13. <http://ctan.org/pkg/floatrow>

11 Présentations vidéoprojetées

Le contenu de ce chapitre, ayant été traité seulement une semaine avant l'examen, ne figure pas au programme, qui est déjà bien assez chargé comme ça. En conséquence, il ne sera pas développé dans le présent polycopié. Le lecteur intéressé est invité à se reporter aux diapos utilisées en cours pour une introduction, et à la documentation officielle (fichier `beameruserguide.pdf`) de la classe `beamer`¹ pour tous les détails.

1. <http://ctan.org/pkg/beamer>