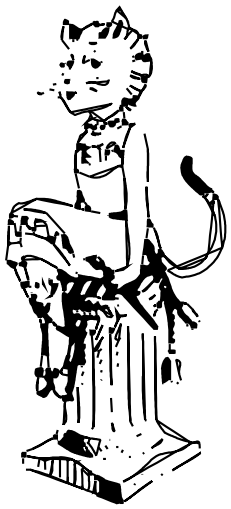
Voir en page x comment on peut imprimer ce manuel...

TOUT CE QUE VOUS AVEZ TOUJOURS VOULU SAVOIR SUR

LaTeX

SANS JAMAIS OSER LE DEMANDER



Ou comment utiliser LaTeX quand
on n'y connaît goutte

Vincent Lozano
lozano@enise.fr

Version du 5 mars 2008
Dernière mise à jour sur :
http://cours.enise.fr/info/latex

Enfin voilà deux jeunes corps enlacés qui jouissent de leur jeunesse en fleur ; déjà ils pressentent les joies de la volupté et Vénus va ensemencer le champ de la jeune femme. Les amants se pressent avidement, mêlent leur salive et confondent leur souffle en entrechoquant leurs dents. Vains efforts, puisque aucun des deux ne peut rien détacher du corps de l'autre, non plus qu'y pénétrer et s'y fondre tout entier. Car tel est quelquefois le but de leur lutte, on le voit à la passion qu'ils mettent à serrer étroitement les liens de Vénus, quand tout l'être se pâme de volupté. Enfin quand le désir concentré dans les veines a fait irruption, un court moment d'apaisement succède à l'ardeur violente ; puis c'est un nouvel accès de rage, une nouvelle frénésie. Car savent-ils ce qu'ils désirent, ces insensés ? Ils ne peuvent trouver le remède capable de vaincre leur mal, ils souffrent d'une blessure secrète et inconnaissable.

Lucrèce *De natural Rerum*, Livre IV

Introduction

Mieux vaut la malice d'un homme
que la bonté d'une femme.[1]
L'ecclésiastique Si **42** 14.

Il était une fois...

Tout a commencé lorsqu'au tout début des années 1990, j'utilisais sur un ordinateur PC 286 une version du logiciel Word Perfect pour m'initier à ce qu'on appelait alors le « traitement de texte ». Ce logiciel — qui existe toujours, édité par la société Corel — proposait à l'époque sous le désormais célèbre MS-DOS, une interface composée d'un vague aperçu du document, et surtout laissait à l'utilisateur la possibilité de « voir les codes » c'est-à-dire de visualiser le document avec une sorte de langage à balises en permettant un contrôle très souple.

Un peu plus tard, avec la prolifération de Windows 3.1 et l'engouement soudain pour les interfaces graphiques, je me laissais convaincre — faible que j'étais — d'utiliser le logiciel de traitement de texte devenu très célèbre aujourd'hui dans sa version d'alors : la version 2.0 (avec une petite lettre derrière qui avait toute son importance à l'époque)... Cette version, je ne l'appris qu'un peu plus tard, avait la particularité intéressante de comporter un bug très sérieux qui empêchait à partir d'un certain volume de données, la sauvegarde ! Il n'y a avait alors aucune solution pour sauvegarder ni récupérer son document ; les plus teigneux d'entre nous se hasardaient à supprimer quelques lignes et tentaient à nouveau une sauvegarde, mais en vain...

À cette époque où les logiciels édités par la société dont nous tairons le nom ici, faisaient l'objet de railleries non dissimulées,[2] la plupart des utilisateurs qui m'entouraient acceptaient malgré tout la situation : il était normal d'utiliser des logiciels qui se vautraient lamentablement et notoirement sans crier gare. Cette particularité a fait naitre en moi une certitude : *je n'accepterai pas d'utiliser de tels logiciels.* J'étais alors élève ingénieur et je pressentais qu'une partie de mon travail serait consacré à l'élaboration de documents et à l'utilisation de systèmes informatiques en général, il me fallait des outils robuste pour y parvenir.

C'est au cours de mon DEA (appelé aujourd'hui master recherche) à l'université Jean Monnet et à l'École des Mines de Saint-Étienne que j'ai découvert à la fois Unix (dans sa version Solaris) puis Linux. C'est alors que le mot « latèque » fut lâché pas loin de moi au début de ma thèse (1993-94). Il était apparemment question d'un logiciel indispensable pour produire des formules mathématiques, et surtout il semblait évident que LaTeX était *le* choix incontournable pour produire des documents scientifiques. À vrai dire la question n'avait même pas l'air de se poser !

[1] Les épigraphes de ce document sont tirées de l'Ancien et du Nouveau testament. Ces citations sont insérées par pure provocation de ma part, et ont — parfois — un lien avec le titre du chapitre.

[2] Parmi celles-ci, même si elles n'apparurent que quelques années après, on pourra noter la célèbre intervention du patron de General Motors en réponse à une provocation de Bill Gates et le non moins célèbre « Piège dans le cyberespace » de Roberto Di Cosmo.

J'entrepris donc d'installer cette « chose » qu'était LaTeX à la fois sur un système Mac avec une distribution nommée OzTeX et sur un système Solaris, avec la distribution fournie par l'association Gutenberg. Il avait fallu pour cela soudoyer l'administrateur système pour qu'il accepte de créer un utilisateur privilégié `texadm` dont le but était d'administrer la distribution...

Début 1994, je commençais ma thèse avec bien évidemment la ferme intention de la rédiger avec LaTeX. Courant 1995, enthousiasmé par ce que je découvrais, j'entrepris de rédiger un guide d'initiation à LaTeX pour mes collègues de laboratoire, guide qui est à l'origine du présent manuel. C'est au cours de l'année 1997, après environ deux ans de pratique et d'initiation au monde de la typographie, que je me confortais dans l'idée que LaTeX était effectivement le logiciel de choix pour la rédaction d'un document « sérieux » : contrôle global de la mise en page, gestion de la bibliographie, des indexes (nom communs et auteurs), légereté des fichiers manipulés et surtout : *la beauté du résultat*. Aujourd'hui encore, c'est pour moi l'argument le plus fort et le plus irréfutable pour utiliser LaTeX.

Aujourd'hui maître de conférence en informatique à l'école nationale d'ingénieurs de Saint-Étienne, j'utilise LaTeX pour la rédaction de documents scientifiques et de supports pédagogiques. Après maintenant plusieurs années de pratique, je continue à apprendre et à découvrir, tout en étant encore ébloui par l'ensemble des extensions proposées par les contributeurs du projet, ensemble d'extensions qui font de LaTeX un joyeux bazar, mais aussi un outil extraordinaire évoluant dans le sens de la véritable ergonomie,[3] un outil unique dont le souci permanent est « la belle ouvrage ».

Organisation du manuel

Ce manuel est une introduction au « traitement de texte » LaTeX ; il ne s'agit pas d'un manuel de référence, mais il a pour but de donner les bases pour utiliser LaTeX et si possible d'y prendre goût. Ainsi trouvera-t-on les informations nécessaires pour *commencer* en LaTeX et quelques conseils sur la rédaction des documents. Pour votre confort, nous avons eu l'idée lumineuse de diviser ce manuel en chapitres et annexes. La première partie présente les bases de LaTeX :

Principes de base expose les concepts fondamentaux de LaTeX à lire impérativement pour comprendre le reste.

Ce qu'il faut savoir présente les outils standard, ceux qu'il faut connaître pour rédiger un document simple.

Mathématiques ou comment produire des équations.

Un pas vers la sorcellerie pénètre un peu plus profondément dans les rouages de LaTeX ; à lire si vous voulez utiliser LaTeX de manière satisfaisante.

Graphisme permet de comprendre comment insérer des graphiques dans vos documents.

Documents scientifiques donne quelques conseils pour rédiger articles, bibliographies, index et transparents.

Documents en français fournit quelques notions de typographie élémentaires et présente les principaux aspects du package french.

[3] Pas celle qui consiste à ajouter une entrée dans un menu, ou un son à l'apparition d'une boîte de dialogue.

À vous de jouer ! une conclusion sous forme de conseils pour chercher des informations sur TEX et LATEX.

La deuxième partie a pour but d'aborder les aspects plus complexes de LATEX en prenant comme prétexte d'expliquer comment ce manuel a été produit. Ne la lisez pas avant d'avoir lu la première... Toute exposition — même non prolongée — à la deuxième partie peut provoquer des troubles du comportement et des traumatismes irréversibles.

Viennent ensuite les annexes :

Générer des documents en PDF comme son nom l'indique explique la méthode utilisée pour générer la version pdf de ce manuel ;

Mémento est un fourre-tout qui propose une liste non exhaustive d'extensions utiles, les raccourcis de AucTEX, et la configuration de aspell pour emacs ;

Symboles une liste des symboles mathématiques disponibles en standard et avec l'extension amssymb.

Il est conseillé de lire dans l'ordre les premiers chapitres jusqu'aux mathématiques. Les suivants peuvent se lire indépendamment les uns des autres. Encore une fois, la deuxième partie du manuel n'est à lire qu'après avoir maîtrisé les concepts de base. Un index en fin de document constitue un bon point d'entrée pour retrouver des informations. Enfin, à l'instar des auteurs de la FAQ française de LATEX, je n'ai pas fait d'effort particulier pour traduire systématiquement tous les termes du jargon de LATEX et de l'informatique en général.

Ce qu'il faudrait que vous sachiez

La lecture de ce manuel qui s'adresse aux débutants, ne demande aucun prérequis à propos de LATEX. Le lecteur devra cependant posséder une connaissance de base d'un système d'exploitation en tant qu'utilisateur, c'est-à-dire savoir manipuler des fichiers. Être capable de créer un fichier PostScript encapsulé sur son système, à partir d'un logiciel de dessin ou de manipulation d'image, est également souhaitable.

Ce que vous ne saurez pas

Ce fabuleux manuel que vous avez entre les mains souffre tout de même de quelques lacunes ; parmi celles-ci :

- il manque une explication claire de la manière dont TEX et LATEX gèrent les fontes. Vous ne trouverez d'ailleurs nulle part le mot METAFONT ;
- vous n'apprendrez pas comment installer et administrer une distribution LATEX sur un système UNIX ;
- vous ne trouverez pas de « catalogue » ou d'inventaire des extensions existantes, utiles ou inutiles, compatibles ou incompatibles, etc. ;
- la question de l'œuf ou la poule est également occultée, ainsi que celle des liens entre Dieu et la science ;
- ...

Il est important de noter que le titre de ce manuel est un mensonge éhonté. D'autre part, les images de la couverture mises à disposition par KNUTH[4] sont dessinées

[4] http://www-cs-faculty.stanford.edu/~uno/graphics.html

par Duane BIBBY et n'ont rien à voir avec le sujet du présent manuel puisqu'il s'agit de dessins censés représenter les programmes TEX et METAFONT.

TEX ?

Le mathématicien Donald Ervin KNUTH — à qui l'on doit de nombreux ouvrages de mathématiques et d'algorithmique (notamment *The Art of Computer Programing* [1]) — a conçu dans les années 70 un système de traitement de texte nommé TEX après avoir été déçu par la manière dont ses articles étaient imprimés par les systèmes du moment. TEX — accessible au public depuis le début des années 80 — est un environnement complexe de programmation composé d'un processeur de macro (*macro processsor*) et de quelques centaines de primitives. Un premier ensemble de macros pré-compilées est apparu assez rapidement sous le nom de *format plain*.

On pourra noter que TEX n'est ni un *traitement de texte* (KNUTH le nomme « typesetting system » que l'on pourrait traduire par système de composition) ni un langage de programmation compilé. Voici quelques citations de KNUTH à propos de TEX[5]

> *« Des mots anglais comme « technology » sont dérivés de racines grecques commençant par les lettres τεχ... ; ce même mot grec voulant dire à la fois* art *et technologie. D'où le nom TEX, qui est la forme en majuscules de τεχ. »*

Au sujet de la prononciation du « X » de TEX :

> *« [...] C'est le son « ch » en allemand comme dans* ach *; c'est le « j » espagnol [...]. Lorsque vous le dites correctement à votre ordinateur, l'écran doit devenir légèrement humide. »*

Votre humble serviteur se contente lui de le prononcer « TeK » pour contrecarrer l'aspect caoutchouteux et éviter d'avoir à nettoyer son écran régulièrement.

Enfin pour ce qui est du logo lui-même KNUTH fait remarquer que ce déplacement du E est là pour rappeler qu'il s'agit de typographie, et insiste sur le fait que dans une situation où l'on veut parler de TEX sans avoir les moyens d'abaisser le E, il faudra écrire « `TeX` ».

La version actuelle de TEX est 3.14159 (les versions comme vous l'avez compris tendent vers π) ; KNUTH estime que le dernier bug a été trouvé et corrigé dans TEX le 27 novembre 1985 (préface du livre TEX : le programme) et propose une récompense de $20.48 à qui en trouve un nouveau (!).

LATEX ?

En 1985, quelques années après la diffusion publique de TEX, Leslie LAMPORT crée un format composé de macros permettant d'avoir une vision de plus haut niveau d'un document, appelé LATEX et portant le numéro de version 2.09. Aujourd'hui,

[5] Tiré du chapitre introductif « The Name of the Game » du TEXBook.

LaTeX est un standard de fait, et seuls quelques sorciers produisent encore des documents uniquement avec TeX. Cependant, LaTeX étant une « surcouche » de TeX — contenant donc des appels à des macros de TeX — il est parfois utile de connaître quelques-uns des concepts de TeX pour se tirer d'un mauvais pas. Voici ce que dit Lamport à ce propos dans son livre [2] :

> *« Imaginez LaTeX comme une maison dont la charpente et les clous seraient fournis par TeX. Vous n'en avez pas besoin pour vivre dans la maison, mais ils sont pratiques pour y ajouter une nouvelle pièce. »*

Un peu plus loin :

> *« LaTeX a été conçu pour permettre à un auteur de faire abstraction des soucis de mise en page, et se concentrer sur l'écriture. Si vous passez beaucoup de temps sur la forme, vous faites un mauvais usage de LaTeX. »*

Aujourd'hui et depuis 1994, une équipe mi-européenne mi-américaine (autour de Frank Mittelbach) a pris en main le développement de LaTeX ; la version de LaTeX parue en 1994 se nomme LaTeX 2_ε. Le but à long terme est de concevoir un système nommé LaTeX3.

Licence

On peut souligner que TeX et LaTeX sont des logiciels faisant partie de la famille des logiciels libres et sont donc — entre autres — gratuits. Ce qui caractérise les logiciels libres (*free software*) est également l'aspect *ouvert* des logiciels. Il est donc possible d'avoir les sources Web[6] de TeX. Les macros de LaTeX sont quant à elles distribuées sous forme de code source TeX. Le fait de pouvoir obtenir les sources d'un logiciel peut sembler secondaire à la plupart des utilisateurs ; il faut comprendre que c'est parce que *rien n'est caché* que l'amélioration de l'existant et la création d'extensions sont possibles.

Le fait qu'un logiciel soit libre ne veut pas dire qu'on puisse en faire tout à fait ce que l'on veut. Il reste la propriété de son auteur et toute modification doit être documentée ; chacune de ces modifications doit également donner lieu à un nom de fichier différent de celui du fichier initial avant modification. Ceci pour assurer cohérence et portabilité au système (voir à ce sujet `ftp://ftp.lip6.fr/pub/TeX/CTAN/macros/latex/base/lppl.txt` pour la licence de LaTeX 2_ε).

Cinq bonnes raisons pour ne pas utiliser LaTeX

Il existe plusieurs raisons pour lesquelles il est « impératif » de ne pas utiliser LaTeX :

[6] Le langage Web conçu par Knuth est qualifié de langage de « programmation littéraire. » À partir d'un document source Web, on peut produire le code Pascal ou C du programme ainsi qu'une documentation en TeX de ce code.

1. vous utilisez un traitement de texte uniquement pour faire vos cartes de vœux, votre courrier, pour noter quelques idées, etc. ;
2. vous adorez les souris (1 ou 3 boutons indifféremment) et vous pensez que la seule manière d'écrire des équations est de les utiliser (les souris) de manière intensive ;
3. vous pensez qu'UNIX c'est « prise de tête » et « pas convivial » et/ou vous avez une aversion particulière pour tout langage de programmation ;
4. vous trouvez normal : 1° que votre logiciel préféré ne puisse pas lire le document que vous aviez produit avec la version précédente, et/ou 2° que la nouvelle version vous oblige à changer de système d'exploitation, et 3° que la nouvelle version dudit système d'exploitation vous oblige à changer d'ordinateur, et 4° que votre nouvel ordinateur...
5. vous ne savez pas où se trouve la touche \ sur votre clavier ;

si vous vous reconnaissez dans une de ces catégories, mieux vaut vous contenter de votre système actuel.

Quelques bonnes raisons d'utiliser LaTeX

Il n'est pas question ici de convaincre le lecteur de la supériorité de TeX et LaTeX par rapport à un autre système, de toutes manières, vous lisez ce manuel, donc vous êtes inconsciemment convaincu. Laissons donc la parole au concepteur de TeX :

> *« By preparing a manuscript in TeX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers. »*
> D. E. Knuth in the TeXbook [3]

Les documents générés par TeX ou LaTeX sont d'une qualité typographique exceptionnelle (avec possibilité de réglage très fin[7]), ceci grâce notamment à :

- un dessin de fontes très soigné ;
- des détails typographiques (tirets, ligatures,...) ;
  - avez-vous — bien — regardé ces tirets (page 19–23) ?
  - et le « fi » de fin, le « ffl » de souffle ou le « fl » de trèfle ?
- un algorithme de césure très performant ;
- des formules mathématiques particulièrement réussies ;
- ...

D'autre part, LaTeX est un des rares logiciels de traitement de texte orienté vers la production de documents *scientifiques*. Car outre les équations et autres formules, LaTeX possède un grand nombre de fonctionnalités axées autour de la rédaction d'*article* et la génération de *bibliographie* et d'*index*.

Enfin, LaTeX est particulièrement adapté à la production de gros documents. Pas seulement parce que la manipulation d'un document LaTeX exige par essence

[7] À titre indicatif l'unité interne de mesure de TeX est le *scaled point*, noté `sp` dans le TeXbook, qui vaut 1/65536 points ; 1 point valant environ 1/72e de pouce, 1 pouce valant 2.54 cm, l'unité de base est approximativement 50 Å, ce qui laisse de la marge vis à vis de la résolution des imprimantes actuelles.

peu de mémoire, mais parce que son mécanisme de *macros* et de *référence croisée* (*cross reference* en anglais) permettent de garder un contrôle global et très souple du document.

référence croisée : LaTeX permet de faire référence de manière *symbolique* à toute partie du document faisant l'objet d'une numérotation. Ainsi, le numéro des titres, figures, tableaux, équations, pages, références bibliographiques, items d'énumération, théorèmes,... peut être mentionné à plusieurs endroits dans un document de manière très simple, sans se soucier du numéro lui-même.

macros : sans doute l'aspect le plus puissant de LaTeX. Il faut savoir que **tout** processus qui mène à la génération d'un document est une séquence de commandes ou macros. Chaque utilisateur peut donc modifier l'allure d'un document, en modifiant l'une des ces macros. On peut bien évidemment définir ses propres macros pour mettre en page une partie spécifique d'un document. L'idée forte autour des macros est qu'on peut *a priori* séparer le fond de la forme lors de la rédaction d'un document.

Les limites du Wysiwyg

LaTeX est le contraire d'un Wysiwyg,[8] puisqu'un source LaTeX est un document texte composé du texte lui-même et des commandes de mise en page. Lamport présente ce type d'approche comme étant une mise en page *logique* par opposition à la mise en page *visuelle*.[9]

On pourrait cependant dire que LaTeX est un Wywsiewyg (what you will see is exactly what you get) puisqu'après compilation on peut visualiser à l'écran une image **exacte** du document futur sur papier.

Voici donc un exemple[10] parmi d'autres qui met en évidence les limites du Wysiwyg et les avantages de la mise en page logique : supposons que dans un document apparaisse un certain nombre de fois, une fonction quelconque ayant deux arguments. La notation étant un point délicat dans l'élaboration de documents scientifiques, on pourra définir une macro `\mafct` permettant de produire une telle fonction. Ainsi les séquences `\mafct{1}{2.5}` et `\mafct{x}{t}` produiront respectivement $\mathcal{F}_{\alpha,\beta}(1, 2.5)$ et $\mathcal{F}_{\alpha,\beta}(x, t)$. Mais si l'on a besoin de changer de notation, il suffira de redéfinir la commande `\mafct` pour produire aux endroits nécessaires : $\mathsf{F}^{\alpha\beta}[1, 2.5]$ et $\mathsf{F}^{\alpha\beta}[x, t]$. Et le tour est joué !

Un autre exemple : imaginons que votre document comporte beaucoup de mots techniques que vous voulez mettre en évidence d'une manière particulière. Vous écrirez alors dans votre document `\jargon{implémentation}` en ayant préalablement défini la macro `\jargon` de manière à ce qu'elle mette en italique le mot du vocabulaire technique. Les 235 mots de jargon auxquels vous faîtes référence dans votre document pourront être mis en évidence autrement qu'en italique si vous changez d'avis, et cela sans avoir à passer sur les 235 occurrences des mots du jargon, mais juste en changeant la définition de la macro `\jargon`. Avec un peu d'entraînement

[8] Pour « what you see is what you get » terme apparu au début des années 90 pour désigner les logiciels permettant à l'utilisateur de voir à l'écran sur qu'il obtiendrait sur le papier.

[9] Pour faire un peu de mauvais esprit, les logiciels du type Wysiwyg ont d'ailleurs été qualifiés par Kernighan (dixit Lamport dans son livre sur LaTeX) de "what you see is all what you've got" !

[10] Lamport propose un exemple analogue à celui-ci dans son manuel.

vous arriverez même à faire en sorte que cette macro insère automatiquement le mot du jargon dans l'index de votre document...

Voici un exemple un peu plus tordu : dans le titre du paragraphe intitulé « Cinq bonnes raisons de ne pas [...] » un peu plus haut dans ce chapitre, je n'ai pas écrit « Cinq »[11] en toutes lettres dans le document source. En réalité le titre du paragraphe est produit par : « `\ref{nbraisons} bonnes raisons...` » qui affiche en français le nombre correspondant aux nombres de bonnes raisons de ne pas utiliser LaTeX. Si jamais j'avais à rajouter d'autres entrées dans cette liste de bonnes raisons, il ne sera pas nécessaire de refaire la numérotation...

Vous trouverez le long de ce manuel, d'autres exemples mettant en évidence, les faiblesses du Wysiwyg. Ce « nota », vous avertissant d'un point important en est un autre exemple. Car au moment où l'auteur tape ces lignes, la présence du « panneau danger » est un détail — il s'agit simplement d'un nota. Et l'auteur a écrit :

```
\begin{nota}
  Vous trouvez le long de ce manuel...
\end{nota}
```

Pour en finir avec les macros, on peut dire qu'il s'agit d'une « généralisation » des styles du célèbre logiciel « Mot » de la société « Micrologiciel ». La lecture de ce document et en particulier la deuxième partie devrait vous convaincre que les macros permettent d'aller bien au-delà de ces fameux styles...

Pour les accros du Wysiwyg, une équipe de développeur a mis en œuvre une version *What you see is what you Mean* (*sic*) de LaTeX nommé LyX, dont je vous invite à prendre connaissance à `http://www.lyx.org`.

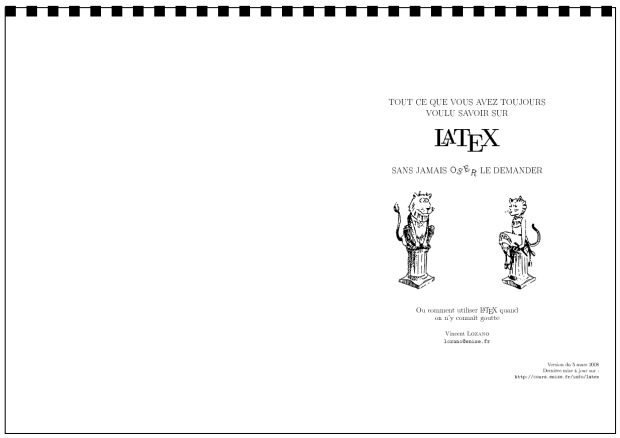
Comment imprimer ce manuel ?

Avec une imprimante,[12] en utilisant exclusivement les fichiers proposés sur `http://www.enise.fr/cours/info/latex`. Ce manuel est conçu pour être imprimé en deux pages logiques par page physique (*2up printing* comme disent nos amis anglo-saxons). En d'autres termes, chaque page de ce document doit apparaître sur une page au format A5. Quatre solutions — et quatre fichiers associés — sont possibles :

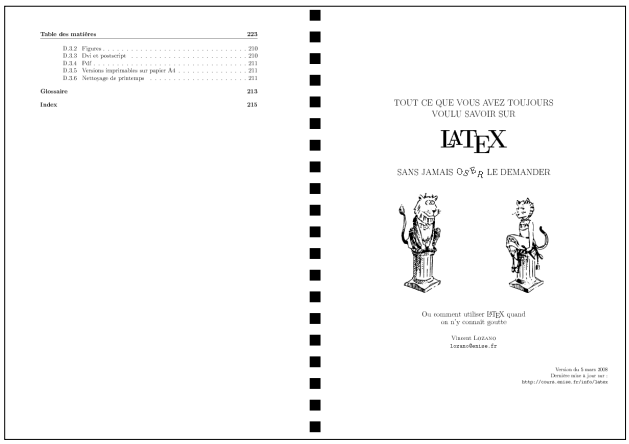
1. une version destinée à être reliée du long coté de la feuille A4, c'est la solution pour ceux qui ne disposent pas de massicot ; le fichier portant le nom `guide-latex-0.pdf` est en outre généré pour présenter la page impaire (dite belle page) à droite (cf. figure 1a page xi) ;
2. une version pour ceux qui disposent d'un massicot : dans ce cas, il faudra « plier » le document et le relier au niveau de cette pliure (voir figure 1b page xi). Le fichier correspondant se nomme `guide-latex-1.pdf` ;
3. une version contenant 2 exemplaires (fichier `guide-latex-2.pdf`). On reliera ces deux exemplaires comme indiqué à la figure 1c page xi ;
4. une version au format PostScript `guide-latex.ps.gz` pour ceux qui voudraient faire « à leur sauce »...

[11] Là non plus d'ailleurs.

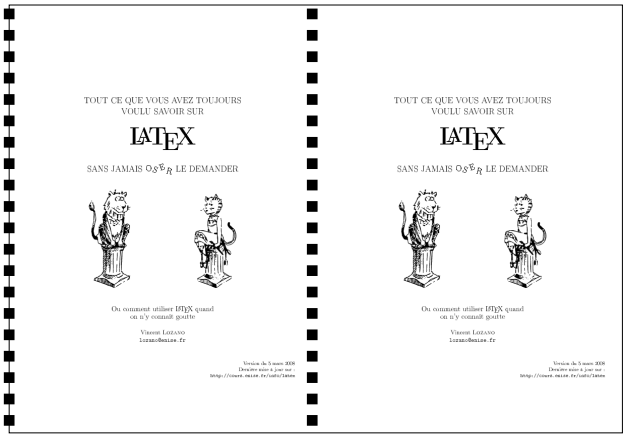
[12] Arf arf (comme disait Frank Zappa).



(a) Reliure longue



(b) Reliure courte « 1 exemplaire »



(c) Reliure courte « 2 exemplaires »

FIG. 1 – Les versions imprimables

⚠ Attention, les deux derniers fichiers au format Pdf, sont prévus pour être imprimés en recto verso en demandant à l'imprimante de retourner la feuille du côté court. Il faudra donc veiller à ce que le pilote d'imprimante utilise cette option.

Que pouvez-vous faire de ce manuel ?

Nom de l'auteur : Vincent LOZANO ;

Titre : Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais avoir osé le demander ;

Date : 5 mars 2008

Copyleft : ce manuel est libre selon les termes de la Licence Art Libre (http://www.artlibre.org) (LAL).

La LAL stipule en résumé que vous pouvez copier ce manuel. Vous pouvez également le diffuser à condition :

– d'indiquer qu'il est sous la LAL ;

- d'indiquer le nom de l'auteur de l'original : Vincent LOZANO et de ceux qui auraient apporté des modifications ;
- d'indiquer que les sources peuvent être téléchargés sur http://cours.enise.fr/latex ;

Enfin vous pouvez le modifier à condition :

- de respecter les conditions de diffusion énoncées ci-dessus ;
- d'indiquer qu'il s'agit d'une version modifiée et si possible la nature de la modification ;
- de diffuser vos modifications sour la même licence ou sous une licence compatible.

En avant !

Comme beaucoup de logiciels puissants, LaTeX n'est pas toujours simple à utiliser. En fait lorsque l'on va dans son sens, LaTeX est toujours agréable et permet effectivement comme le souligne LAMPORT de ne pas se soucier de problèmes de mise en page. Lorsque l'on veut changer un comportement, et que la solution consiste à choisir une autre option d'une commande, tout va encore très bien. Cependant, même si les choix de LaTeX répondent à des conventions en vigueur chez tous les bons imprimeurs, il arrive un jour où l'on désire avoir une mise en page particulière qu'apparemment LaTeX est incapable de fournir. À ce stade, plusieurs solutions s'offrent à vous :

- inclure un *package* qui répond à votre problème (LaTeX étant un système ouvert, une multitude de packages plus ou moins standardisés sont disponibles pour réaliser des opérations variées voire farfelues);
- demander à un TeXnicien[13] de vous dépanner;
- si les deux premières solutions sont inefficaces, vous n'avez plus qu'à faire le détective et mettre le nez dans le code[14] pour trouver la commande qui vous fait du tort et la modifier. Vous aurez besoin à ce stade de connaissance de la première couche du système, à savoir TeX. On touche sans doute ici à un des défauts de LaTeX : si d'autres logiciels sont incapables de faire des choses compliqués, il est parfois difficile de faire faire à LaTeX des choses simples (vous en serez probablement convaincu après la lecture de la deuxième partie de ce manuel).

Conventions typographiques

Certaines conventions utilisées dans ce manuel nécessite d'être quelque peu éclaircies. Les d'extraits de code LaTeX qui parsèment le document peuvent apparaître comme ceci :

```
% attention les yeux
Ceci est \emph{déjà} du code \LaTeX.
```

[13] ou un TeXpert, mais c'est assez rare.

[14] C'est la solution la plus plaisante pour ceux qui ont certaines velléités pour passer du code...

Le choix s'est porté sur la fonte « `machine à écrire` » de LaTeX. Le code est également souvent présenté sous la forme suivante, avec un petit numéro sur la barre centrale auquel on se réfère parfois :

```
% attention les yeux
Ceci est \emph{déjà} du
code \LaTeX.
```

1 Ceci est *déjà* du code LaTeX.

Certaines parties sont présentées sous forme de « notes » pour éclaircir un point sans que la lecture soit indispensable au premier abord.

Si la lecture est indispensable, on aura recours au pictogramme ci-contre pour attirer l'attention du lecteur distrait...

Les logiciels et les packages de LaTeX sont typographiés comme indiqués ci-avant. Les mots en anglais sont produits *like this*. Pour mettre en évidence les parties génériques d'une commande on utilisera cette notation. Par exemple :

```
Ceci est \emph{texte à mettre emphase} du code \LaTeX.
```

Quelques rares fois sont insérées des commandes Unix, comme ceci :

```
grep -wi bidule /tmp/truc.dat | sort -n
```

On trouve même dans une des annexes, des commandes pour emacs :

```
M-x doctor
```

Et, comble de l'horreur, des extraits de `Makefile` :

```
bidule : bidule.o truc.o
⟼ gcc -o $@ $^
```

Dans la version papier apparaissent des renvois sur des chapitres ou des paragraphes, comme celui-ci dirigeant le lecteur vers la production de formules ▶mathématiques◀ avec LaTeX. Ch. 3 p. 33 ◀

Remerciements

La rédaction de cet ouvrage qui est initialement le guide local du laboratoire d'informatique graphique et d'ingénierie de la vision situé à Saint-Etienne, a débuté en 1995. Je tiens ici à remercier les membres de cette équipe de recherche qui m'ont fait part de leurs remarques et encouragements. Les personnes participant au forum `fr.comp.text.tex` m'ont indirectement apporté énormément d'informations qui ont enrichi ce document, qu'elles en soient ici remerciés.

Je voudrais également remercier Benjamin Bayart qui m'a aidé à créer certaines des extensions utilisées dans ce manuel, en particulier la version initiale de la boîte entourant les « mini » tables des matières en tête de chapitre ; ainsi que Guillaume Connan pour ses remarques sur l'annexe concernant le format Pdf et pour ses encouragements.

Un merci particulier à Denis Bitouzé pour sa lecture attentive ses conseils précieux et les nombreuses corrections qu'il a apportées à la première partie du manuel. Denis m'a mis sur la voie de la rédemption, j'ai fait une croix sur a4wide, `eqnarray`, et toutes ces horreurs qui faisaient de moi un pauvre pécheur...

La lecture des « grands classiques » de la littérature autour de TEX et LATEX m'a inconsciemment influencé dans la rédaction de ce document. La lecture de TEXbook de Knuth [3] m'a bien évidemment donné l'idée de créer le panneau danger ⚠, la lecture quasi compulsive du *LATEX Companion* de Goossens, Mittelbach et Samarin [4] a très certainement influencé beaucoup de passages de cet ouvrage aussi bien pour le fond que la forme. Enfin la lecture de plusieurs manuels en ligne a également dû orienter certains de mes choix (Le « Not So Short Introduction to LATEX » possède par exemple un chapitre que l'on peut traduire par « ce qu'il faut savoir »)...

Avant de commencer je tiens à signaler que même si ce document a mis plusieurs années à mûrir, il est dans un style tout à fait douteux. La preuve, sur ma machine, la commande :

```
grep -E -i 'on peut|permet' *.tex | wc -l
```

donne 343 (plus d'une occurrence par page) ce qui dénote un style assez pauvre.

Bonne lecture et bon courage ![15]

[15] Cette ligne est une illustration de la mise en page logique, elle est censée être au 2/3 du blanc restant sur la page, quelle que soit la taille de ce blanc bien sûr.

Sommaire

I

Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander

1 Principes de base

Sommaire

Lorsqu'un homme a un écoulement
sortant de son corps,
cet écoulement est impur.

Le Lévitique Lv **15** 2.

Ce chapitre expose les mécanismes de base de LaTeX. Vous y trouverez donc une courte introduction à l'installation de LaTeX, une présentation d'une « session » LaTeX classique, la structure d'un document type, des remarques sur les accents, quelques outils à connaître, et enfin, une présentation de l'attitude à avoir devant les messages d'erreurs de compilation.

1.1 Installation

Vous voulez utiliser LaTeX ? Il vous faudra installer une *distribution* correspondant à votre système d'exploitation.[1] Les distributions fournissent des programmes permettant d'automatiser la configuration et l'installation de LaTeX, TeX et tous les utilitaires connexes.

Sous Unix : on trouve encore la distribution teTeX bien que son développement ait été stoppé en 2006. Aujourd'hui on installe généralement la TeXLive (http://www.tug.org/texlive) sur un système Unix ;

Sous MacOS : la distribution de référence est MacTeX (http://www.tug.org/mactex) ;

Sous Windows : le plus simple est sans doute de choisir proTeXt (http://www.tug.org/protext) qui installe la distribution MiKTeX (http://www.miktex.org) et quelques outils de développement dont un programme de visualisation de fichiers au format PostScript (gsview).

Il faut parfois ajouter à ces distributions (si elles n'en contiennent pas déjà un) un éditeur de texte puisque vous le découvrirez bien assez tôt, utiliser LaTeX c'est taper du texte et des commandes dans des fichiers :

[1] Si vous ne savez pas ce qu'est un système d'exploitation, le vôtre est MacOS, si vous ne savez pas *quel est exactement* le système de votre machine vous avez Fenêtre, sinon vous avez un Unix...

- emacs ou vi sous Unix sont deux éditeurs de référence qui, bien que le premier soit nettement supérieur au second, continuent de faire l'objet d'une guerre stérile entre utilisateurs la plupart du temps de mauvaise foi ;
- kile et texmaker sont des environnements de développement intégré grâce auxquels les utilisateurs débutants pourront se sentir à l'aise pour commencer : ils ont en effet la particularité de centraliser dans une même interface : édition, compilation & visualisation. Ces environnement permettent également de découvrir les commandes de LaTeX par l'intermédiaire de menus, boîtes de dialogue et autres onglets (voir la figure 1.1a page suivante pour en avoir un aperçu) ;
- TeXnicCenter est l'équivalent (aperçu à la figure 1.1b) sous la marque « à la fenêtre » ;
- TeXshop et iTeXmax sont les équivalents sous la marque « à la pomme »

Vous apprendrez également bien assez vite que la production d'un document avec LaTeX consiste à traduire (on dit aussi *compiler*) un source — donc créé par un éditeur de texte — en un format destiné à l'affichage ou à l'impression. Il existe donc, plus ou moins intégré aux distributions, des outils célèbres pour la visualisation des différents fichiers résultants de la compilation :[2]

Format DVI : xdvi, kdvi sous Unix et yap sous Windows font partie des programmes permettant de visualiser le résultat de la compilation d'un fichier LaTeX ;

Format Postscript : la suite ghostscript disponible sous des noms qui peuvent varier selon la plateforme, permet de visualiser des fichiers au format PostScript ;

Format PDF : mis à part le célèbre acrobat reader, il existe sous Unix des utilitaires permettant de visualiser le format Pdf : xpdf, evince, ...

Il faudra veiller à ce que la distribution choisie comprenne le module « français » de LaTeX assurant (*hyphenation* en anglais) correcte des mots. On vérifiera les « logs » (voir § 1.6 page 12) au moment de la compilation d'un document pour contrôler le chargement des motifs pour le français :

```
LaTeX2e <2005/12/01>
Babel <v3.8h> and hyphenation patterns for english, usenglishmax,
dumylang, [french], loaded.
```

1.2 Cycle de production

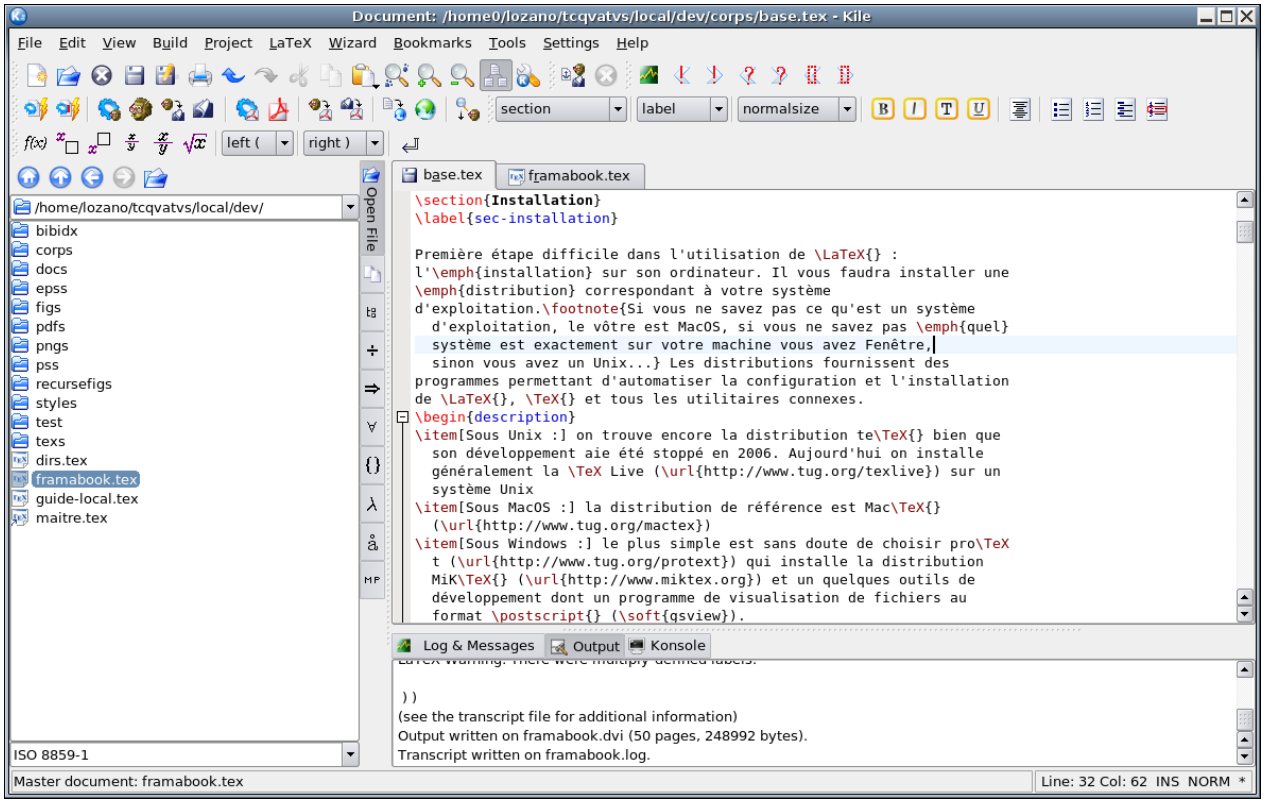
Même si LaTeX n'est pas à proprement parler un langage de programmation compilé, on peut malgré tout faire une analogie entre le cycle de production d'un document LaTeX et le cycle *édition-compilation-exécution* d'un développement de logiciel avec un langage de programmation classique.

1.2.1 Édition

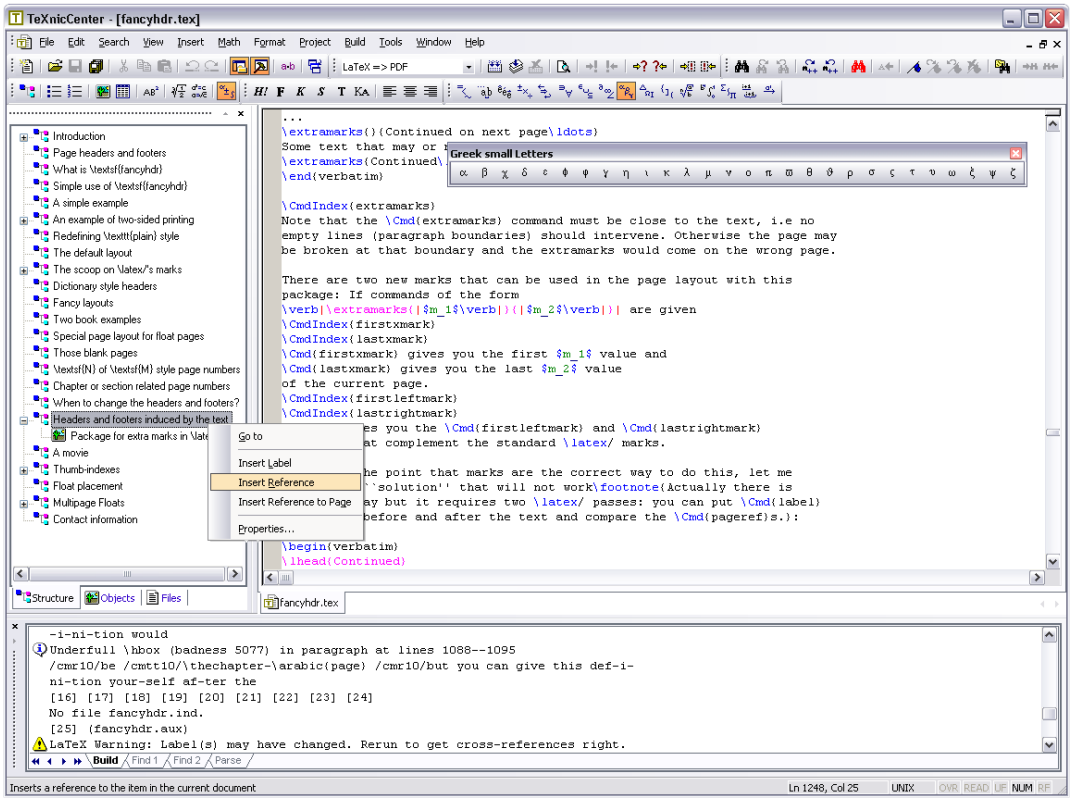
Un document *source* LaTeX est un fichier texte.[3] Ainsi la manipulation d'un fichier LaTeX ne demande pas de logiciel particulier, si ce n'est un éditeur de texte classique. Donc, pour manipuler un document LaTeX :

[2] Ces formats sont présentés peu plus loin dans ce chapitre.

[3] C'est-à-dire un fichier ne contenant que le code des caractères qui le composent.



(a) Kile



(b) TEXnicCenter

FIG. 1.1 – Deux exemples d'environement de développement intégré : Kile sous Linux et TEXnicCenter sous Windows. Ils permettent de centraliser dans une même interface : édition, compilation & visualisation.

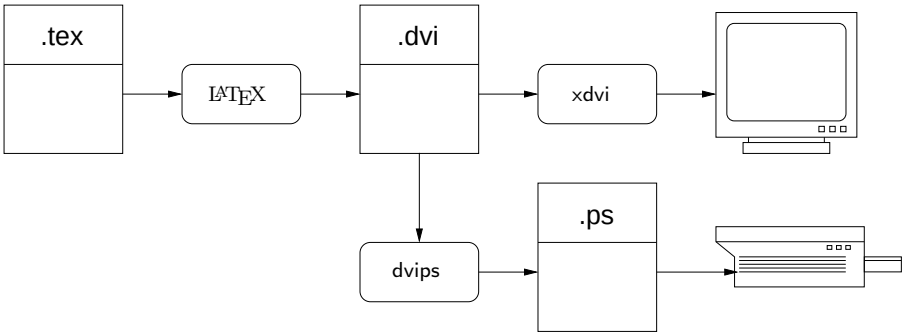


FIG. 1.2 – Cycle de production sur machines UNIX

```
emacs nom de fichier.tex &
```

ou :

```
vi nom de fichier.tex
```

devrait suffire pour entrer dans ce monde sauvage et inconnu qu'est la saisie d'un document LaTeX. Sous Windows, on s'équipera d'un éditeur de texte de son choix.[4] Notez qu'il est recommandé de donner l'extension `.tex` aux sources LaTeX.

1.2.2 Compilation

On lance la compilation grâce à la commande :

```
latex nom de fichier.tex
```

La compilation génère un jour ou l'autre des erreurs. Il en sera question à la section 1.6 page 12. En tout cas, après suppression des erreurs de compilation, on obtient un fichier portant l'extension `dvi` pour *device independant*. Ce qui signifie que le fichier contient des informations indépendantes du périphérique de sortie (écran, imprimantes, ...). Ce fichier de type binaire contenant une « image » du document portable sur tout système TeX quel que soit le système d'exploitation. Il existe ensuite des programmes permettant soit :

- de visualiser le document : `dvi` → bitmap écran ;
- de l'imprimer : `dvi` → langage imprimante ;
- de le convertir : `dvi` → fichier PostScript.

La figure 1.2 illustre les divers programmes entrant en jeu dans la production du document final sur une machine UNIX.

Il est également possible de générer un document au format PDF. On pourra consulter l'annexe A pour obtenir des informations à ce sujet.

1.2.3 Visualisation

La visualisation s'effectue simplement — après compilation sans erreur — grâce au programme xdvi en tapant la commande :

```
xdvi nom de fichier.dvi &
```

[4] Il existe une version d'Emacs pour Micrologiciel Fenêtre, avis aux amateurs.

il s'agit d'un logiciel tournant sous X Window, très intuitif, qui donne un aperçu très lisible du document. La distribution teTEX pour Windows propose un visualiseur nommé yap.[5]

On notera qu'il n'est pas nécessaire de relancer xdvi ou yap après chaque compilation. Ils mettent en effet à jour l'affichage automatiquement.

1.2.4 Impression

Pour imprimer un document, on utilise le programme dvips comme suit :

```
dvips nom de fichier.dvi
```

Il est aussi possible de générer un fichier PostScript en redirigeant la sortie de dvips :

```
dvips nom de fichier.dvi -o nom de fichier.ps
```

Le fichier nom de fichier.ps est un fichier ASCII pur contenant des commandes PostScript. Par exemple si le fichier source est nommé `truc.tex`, on écrira :

```
latex truc.tex
dvips truc.dvi -o bidule.ps
```

pour compiler d'abord `truc.tex`, et pour produire ensuite `bidule.ps` à partir de `truc.dvi`. Pour préparer un fichier destiné à d'autres imprimantes, il faut spécifier en ligne de commande :

```
dvips -Pconfig nom de fichier.dvi
```

où config désigne une imprimante particulière ou une option définie par l'administrateur du système LaTeX que vous utilisez.

Le format PostScript défini par Adobe est un langage d'impression très répandu sur les systèmes Unix. De nombreux utilitaires autour du Postscript sont disponibles, voir par exemple les très bons `psutils` disponibles par exemple à `ftp://ftp.lip6.fr/pub/TeX/CTAN/support/psutils` ou toute autre bonne crèmerie CTAN[6] près de chez vous ou sous la forme d'un paquet Debian.

1.3 Le document source : un document type

Nous allons présenter dans cette section un document type. Tous les documents LaTeX ont en effet une structure commune, de la forme suivante :

```
\documentclass[cOption1,cOption2,...)]{classe}
\usepackage[pOption1,pOption2]{package}
...
préambule
...
\begin{document}
...
le texte
...
\end{document}
```

[5] Pour « yet another previewer », soit « encore un visualiseur » (humour d'informaticien)...

[6] voir le chapitre 8 page 97 pour la signification de ce terme.

Ainsi tout document LaTeX peut se décomposer comme suit :
- spécification de la classe du document ;
- préambule :
  - utilisation de packages particuliers ;
  - initialisations et déclarations diverses ;
- corps du document (entre `\begin{document}` et `\end{document}`).

Voici quelques détails sur chacune de ces parties.

1.3.1 Classe du document

La classe est une indication donnée à LaTeX qui va déterminer la mise en page de certaines parties du document. Suivant la classe utilisée, certaines commandes seront disponibles ou non (`\chapter` disponible pour la classe `book` mais indisponible pour la classe `article`, par exemple). D'autre part, une commande donnée aura une signification spécifique selon la classe choisie (titres, tables des matières,...). En première approche,[7] tout document LaTeX commence donc nécessairement par l'instruction `\documentclass` avec entre accolades une classe de document qui peut-être :
- `article` pour un article ;
- `proc` pour un article dans le style IEEE proceedings ;
- `report` pour un rapport de plusieurs dizaines de pages ;
- `book` pour un livre ou une thèse ;
- `letter` pour une lettre ;
- `slides` pour produire des transparents

On peut évidemment définir sa propre classe de document. Les options de la classe sont précisées entre crochets et peuvent être les suivantes (non-exhaustif) :
- `11pt,12pt` pour changer la taille des caractères de manière globale ;
- `twoside` pour générer un document en recto verso ;
- `draft` pour générer le document en mode brouillon ;
- ...

On pourra donc par exemple, entrer :

```
\documentclass{article}
```

pour avoir toutes les options par défaut (10pt, une colonne, mode recto,...).

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

pour un article en 12pt (par défaut la taille est de 10pt), ou encore :

```
\documentclass[twoside,draft]{report}
```

pour un rapport en recto verso et mode brouillon.

1.3.2 Le préambule

Le préambule est la zone située entre la clause `\documentclass` et la clause `\begin{document}`. Cette zone est la zone où l'on peut spécifier les extensions que l'on veut inclure (voir le paragraphe suivant), l'initialisation de variables globales (marges,...), la définition de styles (titres, numérotation,...), ou de macros particulières.

[7] En réalité on peut insérer d'autres incantations magiques avant le `\documentclass`...

1.3.3 Ajout d'extension

La commande `\usepackage` de LaTeX pourrait être comparée à une directive `#include` du langage C. Elle permet de rajouter des fonctionnalités à LaTeX sous forme de macros et/ou d'environnement.[8] À ce stade, il faut juste noter que l'on peut inclure plusieurs packages en une seule ligne :

`\usepackage{`module$_1$`,` module$_2$`,` module$_3$`,...}`

Si module$_1$, module$_2$ et module$_3$ ont en commun une option `opt1`, on peut entrer :

`\usepackage[opt1]{`module$_1$`,`module$_2$`,` module$_3$`}`

Par contre si l'option opt1 ne concerne que l'extension module2, il sera nécessaire d'entrer les deux lignes suivantes :

```
\usepackage{module1, module3}
\usepackage[opt1]{module2}
```

Voici deux exemples :

```
% package graphicx avec option draft et xdvi
\usepackage[xdvi,draft]{graphicx}
% packages array et subfig
\usepackage{array,subfig}
```

Toutes les options (de classe, de packages, ou de commandes) sont par définition des arguments *optionnels*. On peut donc déjà retenir le fait que tout argument LaTeX donné entre crochets `[...]` est un argument facultatif.

1.4 C'est parti !

Nous allons tenter dans cette section, de présenter, à partir d'un document ne contenant que quelques commandes de mise en page, les principes de base de LaTeX.

```
\documentclass{article}
\begin{document}
Un outil qui vous tombe des mains    tombe
toujours   dans       l'endroit le plus
inaccessible, ou sur le composant le plus
fragile.

Cette loi est l'une des lois de \emph{Murphy}.
\end{document}
```

Un outil qui vous tombe des mains tombe toujours dans l'endroit le plus inaccessible, ou sur le composant le plus fragile.

Cette loi est l'une des lois de *Murphy*.

Cet exemple illustre un certain nombre de principes parmi les plus importants de LaTeX, à savoir :

Ligne vierge ≡ saut de paragraphe : une ligne vierge indique à LaTeX la fin d'un paragraphe. Ainsi dans l'exemple précédent, le premier paragraphe commence à « `Un outil` » et finit avec « `fragile.` ». La commande `\par` est équivalente à la ligne vierge et peut donc également être utilisée pour commencer ou finir un paragraphe.

[8] Ce terme est expliqué au chapitre suivant.

LaTeX ignore les sauts de lignes : ce ne sont pas les sauts de lignes dans le document source qui définissent les sauts de lignes dans le document final. LaTeX *coupe*, *indente* et *justifie* automatiquement chaque paragraphe, sauf contre-ordre de votre part.

LaTeX ignore les espaces multiples : taper un espace ou dix huit mille sept cent quatre vingt quatre espaces est équivalent, comme le montrent les espaces insérés avant « `tombe` » et avant « `l'endroit` ». Ceci est aussi vrai pour les sauts de paragraphes : entrer une ligne vierge ou plusieurs revient au même.

\ est le caractère d'échappement : il indique à LaTeX que le « mot » qui suit est une séquence de contrôle, c'est-à-dire une commande (ou macro) dans le sens le plus général du terme. Ici, il s'agit de mettre en évidence le mot `Murphy`. Ceci est effectué grâce à la commande `\emph`.

{ et } sont des délimiteurs de *groupe*, notions expliquées un peu plus bas.

1.4.1 Quelques caractères sont spéciaux

Comme le suggère l'intervention du caractère \, il existe d'autres symboles ayant une signification spéciale pour LaTeX. Il s'agit des 10 caractères suivants :

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

Voici un petit exemple, utilisant une partie de ces symboles :

```
% paragraphe sans intérêt
\textbf{To be} a subscript : $x_{i+1}$,
or a superscript : $e^{i\pi}$ ?
that's  question~1 ! % or question 2 ?
```

To be a subscript : x_{i+1}, or a superscript : $e^{i\pi}$? that's question 1 !

Pour l'instant, il faut donc savoir que :

- `%` indique à LaTeX d'ignorer le restant de la ligne. C'est donc le symbole du commentaire (équivalent au `//` du C++) ;
- `$` est le symbole de début et fin de formule. Lorsque LaTeX rencontre un `$` il commute en mode mathématique jusqu'au symbole `$` suivant ;
- `~` est l'espace insécable[9], il empêche LaTeX de faire une césure à cet endroit particulier. Il existe un nombre important de situations où il est nécessaire d'insérer un caractère insécable (tout ce qui est du style : `figure~4`). Cependant, il n'existe pas de règles systématisant l'usage d'un tel caractère ;
- `_` et `^` permettent respectivement de passer en indice et en exposant. **Attention,** ces symboles ne sont autorisés qu'en mode mathématique ;
- `{` et `}` sont respectivement les caractères de début et fin de groupe. Deux types de groupement sont donnés à titre d'exemple : l'un, en mode mathématique, pour grouper la « sous-formule » à mettre en indice ou en exposant ; l'autre pour grouper les mots à mettre en gras.

On peut produire une partie des caractères spéciaux dans le texte grâce aux commandes suivantes :

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

qui donnent respectivement : $ & % # { } _. La section 2.2.5 page 21 explique comment produire les autres caractères spéciaux (\ ~ ^).

[9] Voir aussi, le paragraphe 2.10 page 29 sur le contrôle de la césure.

1.4.2 Appel des commandes

Vous avez compris que pour appeler une commande ou macro, il est nécessaire d'insérer le caractère d'échappement — *escape char* en anglais — et de le faire suivre par le nom de la macro que vous voulez utiliser. Mais comment fait LaTeX pour repérer la fin du nom de la macro ? Prenons comme exemple la macro `\TeX` qui produit le logo TeX.

```
The \TeX book is for \TeX       hackers.

\TeX\  has some powerful macros.

\LaTeX{} is a document preparation system
```

1.3

The TeXbook is for TeXhackers.
TeX has some powerful macros.
LaTeX is a document preparation system

On peut résumer le mécanisme en deux règles. Il y a deux types de caractères qui indiquent à LaTeX la fin du nom de la macro :

- les espaces ; ils sont cependant ignorés dans la production du document ;
- tout caractère autre que les caractères de la catégorie « lettre » (alphabet majuscule et minuscule).

Le caractère \␣ (où ␣ est le caractère espace) est appelé un espace de contrôle ; cet espace n'est jamais ignoré par LaTeX. C'est pourquoi : `et\␣\␣\␣hop␣!` produira ; « et hop ! ». En fait, il est bon de prendre l'habitude d'appeler les macros sous la forme \fonction{arguments} et donc d'utiliser la troisième forme de l'exemple précédent plutôt que la deuxième.Cela évite de se poser le problème de l'espace ignoré[10]. On écrira donc « the TeXbook » avec « `the \TeX{}book` » et « LaTeX is a ... » avec « `\LaTeX{} is a ...` ».

1.4.3 Accents

Les français ont souvent une appréhension à utiliser LaTeX à cause des accents. Pas d'affolement ! vous n'aurez pas à saisir les caractères accentués comme indiqué dans le tableau 1.1. Il est quand même bon de noter qu'il est possible d'accentuer (et « cédiller ») n'importe quel type de caractère, y compris les majuscules.

TAB. 1.1 – Saisie des accents avec des caractères 7 bits

accent aigu	`\'z`	ź
accent grave	`\`z`	ż
accent circonflexe	`\^z`	ẑ
cédille	`\c{z}`	ȥ
tréma	`\"{z}`	ż

Attention ! S'il est possible de saisir des documents avec des caractères accentués il ne faut pas perdre de vue qu'il faut alors faire appel à un encodage qui est pour l'instant local à une région du globe. On utilise en France le codage Iso8859 avec l'extension latin1 qui permet de manipuler nos jolis accents. Avant de lire précisément le chapitre dédié aux documents rédigés en français, nous vous suggérons de rajouter dans votre préambule :

[10] Mais pourquoi il nous en parle, alors !?

```
\usepackage[latin1]{inputenc} % codage du fichier source
\usepackage[T1]{fontenc}      % codage des fontes TeX
\usepackage[francais]{babel}  % document en français
```

pour « attaquer » un document en français.

1.5 Premiers outils

Voici quelques macros et ligatures à connaître car souvent utilisées dans un document. Tout d'abord, LaTeX distingue trois types de tirets :

- `-` pour « Saint-Étienne » ;
- `--` pour « page 12–24 » ;
- `---` pour ouvrir une parenthèse — comme cela.

Les guillemets doivent être entrés comme ceci :

- ` `` ` et `''` pour les textes en "anglais" ;
- `«` et `»` si votre clavier le permet,[11] pour les textes en « français ». La partie française du package babel (cf. chapitre 7 page 87) permet la saisie de caractères à l'aide des commandes `\og` et `\fg`, ainsi : `\og français\fg{}`.

Voici pour finir quelques commandes utiles :

- `\today` produit la date du jour (de la compilation) : 5 mars 2008 ;
- `\S` donne le signe paragraphe : § ;
- `\ldots` permet de saisir les points de suspension dans un document anglais... Ils doivent être saisis avec trois points : `...` dans un document français (voir le chapitre 7 pour quelques notions de typographie française)...

Enfin, souvenez-vous qu'en anglais, on ne saisit pas d'espace avant les ponctuations doubles (: ; ! ?) — contrairement au français. Rappelez-vous aussi que dans ce doux pays qu'est la France, on roule surtout à droite.

1.6 Premières erreurs

Dans ce qui suit nous vous proposons d'examiner les états d'âme de LaTeX pendant qu'il compile votre document. Lorsqu'on lance la compilation en ligne de commande, on voit directement cette sortie dans le terminal. De manière à utiliser LaTeX de manière la plus enrichissante nous vous incitons à trouver dans votre propre environnement la manière d'examiner les « logs » de LaTeX, qui vous indiqueront les messages d'erreurs et autres avertissements survenant lors de la compilation.

1.6.1 Symptômes

Si vous utilisez LaTeX en interactif vous serez amenés un jour ou l'autre à voir apparaître à l'écran, un message barbare de ce type :

```
1 This is TeX, Version 3.1415 (C version 6.1)
2 (erreur.tex
3 LaTeX2e <1995/12/01>
4 (/usr/local/lib/texmf/tex/cls/article.cls
5 Document Class: article 1995/11/30 v1.3p Standard LaTeX document class
```

[11] Alt Gr z Alt Gr x sur un clavier sous Linux, par exemple.

```
6  (/usr/local/lib/texmf/tex/clo/size10.clo)) (erreur.aux)
7  ! Undefined control sequence.
8  l.5  paragraphe de ce \empha
9                               {document}
10 ?
```

Ce message qui vous semble sûrement incompréhensible, est le résultat produit sur le terminal après avoir exécuté LaTeX sur le document `erreur.tex` que voici :

```
\documentclass{article}

\begin{document}
Il me semble bien qu'il y ait une erreur dans le premier
paragraphe de ce \empha{document} somme toute assez court.
\end{document}
```


1.6.2 Diagnostic

On peut donc expliquer de manière simple le message d'erreur :

ligne 1 vous utilisez TeX version π à 10^{-4} près ;

ligne 2 vous compilez le fichier `erreur.tex` ;

ligne 3 vous utilisez LaTeX 2_ε version de décembre 95 ;

ligne 4–5 vous utilisez la classe de document standard `article` ;

ligne 6 par défaut, la taille 10pt est utilisée ;

ligne 7 le message d'erreur proprement dit ;

ligne 8–9 la ligne où s'est produite l'erreur ainsi que son numéro dans le document source `erreur.tex` ;

ligne 10 le prompt `?` particulièrement angoissant de TeX

La « coupure » formée par les lignes 8 et 9, indique précisément l'endroit où LaTeX a perdu les pédales. Le message :

```
! Undefined control sequence.
```

vous indique que la commande que vous avez entrée n'est pas connue par LaTeX. Et effectivement, la commande `\empha` n'existe pas.

1.6.3 Soins

Mais que répondre à LaTeX, lorsqu'il nous affiche son fameux prompt « `?` » ? Voici, trois solutions, les plus couramment utilisées pour communiquer un peu avec LaTeX :

- appuyer sur `<Entrée>` pour ignorer l'erreur ;
- taper `x` permet de quitter la compilation ;
- taper `r` pour demander à LaTeX de continuer en ignorant tous les autres messages d'erreur ;
- taper `i` pour insérer une correction et continuer la compilation. Sachant que cette correction ne sera pas insérée dans le document source ;
- taper `h` pour demander un plus d'information quant à l'erreur ; voici ce que vous dit TeX pour le `Undefined control sequence` :

```
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
```

1.6.4 Une collection de message

TEX et LATEX disposent d'un nombre important de messages d'erreur qui correspondent à diverses situations. Ces messages ne sont pas toujours compréhensibles au premier abord. Cependant on peut dire que la plupart des erreurs viennent le plus souvent :

- d'une erreur de syntaxe sur les mots réservés de LATEX ;
- de paires d'accolades mal construites ;
- d'une commande mathématique utilisée en mode texte ;
- d'un mode mathématique non refermé ;
- d'un package que vous avez oublié d'inclure ;
- d'une fin de journée difficile ;
- ...

Y a plus qu'à !

Vous avez maintenant compris comment on pouvait créer un document imprimable à partir d'un source LATEX. Ce chapitre vous a également permis de comprendre le principe de l'appel des commandes. Il ne vous reste qu'à entamer le chapitre suivant pour connaître les différentes fonctionnalités que vous propose le langage LATEX.

2 Ce qu'il faut savoir

Sommaire

Quand on châtie le railleur, le simple s'assagit ;
quand on instruit le sage, celui-ci gagne en savoir.

Les proverbes Pr **21** 11.

Il sera question dans ce chapitre, des commandes de mise en page de base à connaître pour générer un document avec LaTeX. Nous traiterons en vrac des mises en évidences, des environnements standard LaTeX, des titres, des notes de bas de page, des entête et pied de page et des environnements flottants. Nous terminerons le chapitre par un exposé du système de références suivi d'une présentation des fichiers auxiliaires générés par LaTeX. Enfin, ceux qui auront tenu jusque là, auront la chance de pouvoir lire quelques considérations sur la césure.

Toutes ces commandes seront à utiliser avec leur comportement par défaut, c'est-à-dire que nous ne présenterons pas ici la manière de les redéfinir. Vous serez par contre en mesure de produire un document classique avec les mises en page traditionnelles. Pour taper un article plus évolué, vous aurez besoin d'informations sur la manière de produire des formules mathématiques (chapitre 3), quelques infos sur les documents scientifiques (chapitre 6), et éventuellement sur l'inclusion de graphiques (chapitre 5).

2.1 Mise en évidence

Pour comprendre un tant soit peu le mécanisme de sélection de fontes de LaTeX, il faut savoir qu'on distingue au moins quatre paramètres dans une fonte :

famille : c'est la forme globale de la fonte. LaTeX utilise par défaut 3 types de familles : roman, sans sérif et `machine à écrire`. Le mot anglais utilisé par LaTeX est *family*

style : c'est l'allure (en anglais *shape*) de la fonte : *italique penché* et Petites Capitales.

TAB. 2.1 – Déclarations de changement de fontes

Commande	Déclarations	Output
`\textrm{...}`	`{\rmfamily ...}`	roman
`\textsf{...}`	`{\sffamily ...}`	sans sérif
`\texttt{...}`	`{\ttfamily ...}`	`machine à écrire`
`\textup{...}`	`{\upshape ...}`	droit
`\textit{...}`	`{\itshape ...}`	*italique*
`\textsl{...}`	`{\slshape ...}`	*penché*
`\textsc{...}`	`{\scshape ...}`	PETITES CAPITALES
`\textmd{...}`	`{\mdseries ...}`	medium
`\textbf{...}`	`{\bfseries ...}`	**gras**

graisse : c'est l'épaisseur (*serie* pour LaTeX) des traits. Par défaut 2 épaisseurs : médium et **gras** ;

taille : taille des caractères.

2.1.1 Family-shape-series

On distingue deux types de macros pour changer les trois premiers paramètres (cf. tableau 2.1) : les *commandes* et les *déclarations*. Les commandes agissent sur leur argument donné entre accolades. Les déclarations agissent comme des interrupteurs en changeant la valeur d'un de ces paramètres jusqu'à nouvel ordre. En règle générale, on utilisera les commandes pour mettre en évidence un mot ou un groupe de mots :

```
une \emph{variable} de type \texttt{char} est
\textsc{Toujours} codée sur \textbf{8 bits}.
```

2.1 une *variable* de type `char` est TOUJOURS codée sur **8 bits**.

Notez l'utilisation dans l'exemple précédent, de la commande `\emph` (dont la déclaration équivalente est `\em` qui permet de mettre en évidence de manière élégante un groupe de mots. Il est fortement conseillé d'utiliser les *commandes* plutôt que les déclarations. Cependant lorsqu'une longue portion de texte est à changer, il sera parfois plus judicieux d'utiliser les commandes :[1]

```
{\em The music of \bfseries Magma \mdseries is
like a mirror where  everyone can see
a reflection  of who he is.}
```

2.2 *The music of **Magma** is like a mirror where everyone can see a reflection of who he is.*

L'exemple suivant illustre l'utilisation de groupes. La déclaration `\slshape` se situe dans un groupe, elle est donc *locale* à ce groupe. D'autre part, un groupe hérite les paramètres du groupe qui l'englobe. Ainsi, « silence » est écrit en fonte sans sérif (groupe englobant) et *penché* (déclaration locale).

[1] Ainsi que lors de la définition de commandes.

```
\sffamily Le jazz est une musique où le
{\slshape silence\/} a toujours raison ;
c'est pour cela qu'il n'a pas d'autre
issue que l'impossible.
```

2.3

Le jazz est une musique où le *silence* a toujours raison ; c'est pour cela qu'il n'a pas d'autre issue que l'impossible.

2.1.2 Correction italique

Une autre raison pour laquelle il est recommandé d'utiliser les commandes plutôt que les déclarations, est que les commandes effectuent la *correction italique* contrairement aux déclarations. La correction italique est un espace qu'il est nécessaire de rajouter à la fin d'un groupe de mots en italique, pour éviter que celui-ci ne « touche » le mot suivant. Cet espacement est fonction du caractère mis en jeu :

```
le {\em chef} a toujours raison.\par
le {\em chef\/} a toujours raison.\par
le \emph{chef} a toujours raison.\par
```

2.4

le *chef* a toujours raison.
le *chef* a toujours raison.
le *chef* a toujours raison.

On voit donc clairement que la commande `\emph` effectue la correction, alors qu'il est nécessaire de la faire explicitement à l'aide de la macro `\/`, quand on utilise la forme déclaration.

2.1.3 Tailles

On dispose des macros données au tableau 2.2 pour changer la taille de la fonte en cours. Ces macros sont des *déclarations* et il existe pour chacune d'entre elles un environnement portant le même nom.

TAB. 2.2 – Changement de taille

`\Huge`	immense	`\normalsize`	normal
`\huge`	énorme	`\small`	petit
`\LARGE`	très très gros	`\footnotesize`	plus petit
`\Large`	très gros	`\scriptsize`	rikiki
`\large`	gros	`\tiny`	minuscule

2.1.4 Quelques recommandations

L'usage veut que dans la mesure du possible, on utilise avec parcimonie les changements de fontes. Il est en effet de mauvais goût d'effectuer des mises en évidence intempestives et inutiles ; le plus généralement elles surchargent le document au lieu de le rendre plus lisible. Voici trois suggestions (toujours d'usage !) sur l'utilisation des changements de fontes :

- préférer la commande `\emph` (par défaut *italique*) que tout autre commande pour mettre en évidence ;
- réserver le **gras** pour une remarque particulièrement importante ;

– utiliser les petites capitales ne sont à utiliser quasiment exclusivement que pour les noms dans un document en français : Donald KNUTH ;
– la famille `machine à écrire` est souvent utilisée pour produire du texte en langage de programmation ou équivalent.

À bon entendeur...

D'autre part nous vous donnons ci-dessous deux considérations quant à l'utilisation du changement de taille et du souligné (commande `\underline`) :

> *« Perhaps poets who wish to speak in* a still small voice *will cause future books to make use of frequent font variations, but nowadays it's only an occasional font freak* (like the author of this manual) *who likes such experiments. »*
> Donald KNUTH in the *TEXbook* [3]

> *« Note that underlining for emphasis is considered bad practice in the publishing world. Underlining is only used when the output device can't do highlighting in another way — for example, when using a typewriter. »*
> Michel GOOSSENS *et al.* in the *LATEX Companion* [4]

2.2 Environnements

LaTeX propose une série d'outils sous la forme d'*environnements*. Il s'agit d'une structure de bloc dont la syntaxe est la suivante :

```
\begin{nom env}
  ...
\end{nom env}
```

où nom env est le nom d'un environnement. Le premier environnement rencontré jusqu'ici est l'environnement `document`. Entre le `\begin` et le `\end` on insère une portion de texte qui va subir une mise en page particulière.

Notons tout de suite que toute déclaration est locale à un environnement ; et qu'il est bien sûr possible de définir ses propres environnements éventuellement à partir d'autres existants.

Le reste de cette section sera consacré à la description des environnements normalisés de LaTeX.

2.2.1 Centrage et alignement

Pour centrer quelques lignes, on utilise l'environnement `center` :

```
... fin de phrase.
\begin{center}
  quelques lignes \\
  parfaitement centrées \\
  entre les marges
\end{center}
et le paragraphe continue...
```

... fin de phrase.

quelques lignes
parfaitement centrées
entre les marges

2.5

et le paragraphe continue...

De même on peut aisément aligner un paragraphe à droite grâce à l'environnement `flushright` :

```
... fin de phrase.
\begin{flushright}
  deux lignes\\
  alignées à droite
\end{flushright}
et le paragraphe continue...
```

2.6

... fin de phrase.

deux lignes
alignées à droite

et le paragraphe continue...

Noter l'emploi dans les deux précédents exemples de la commande \\ pour passer à la ligne. En dehors de cas particuliers (tableaux, titre et auteur d'un document, centrage et alignements notamment), cette commande est à proscrire : pour passer à la ligne, il faut laisser une ligne vierge ou utiliser la commande `\par`.

En général, on emploie l'environnement `flushleft` avec des commandes \\. Mais on peut l'utiliser pour produire un paragraphe comme celui-ci, non justifié à droite, en laissant à LaTeX le soin d'insérer les sauts de lignes.

La grande majorité des environnements passent à la ligne pour insérer leur contenu. Cependant, il est important de comprendre qu'un environnement *interrompt* le paragraphe dans lequel il est inséré, mais ne le termine pas — vous pourrez d'ailleurs remarquer que la phrase « et le paragraphe continue » n'est pas indentée. En outre, LaTeX insère gracieusement autour de chaque environnement un espace vertical.

On peut noter qu'aux trois environnements précédents correspondent respectivement les trois déclarations : `\centering`, `\raggedleft` et `\raggedright`. On peut par exemple écrire :

```
Emacs stands for :

{\centering Emacs\\Makes\\
  A\\Computer\\Slow\\}
```

2.7

Emacs stands for :

Emacs
Makes
A
Computer
Slow

2.2.2 Listes

LaTeX offre la possibilité d'utiliser trois principaux types de *listes* sous forme d'environnement : `itemize`, `enumerate` et `description`. Il est possible de définir ses propres listes, si celles de LaTeX de vous conviennent pas. Mais voici les listes standard :

Tout d'abord `itemize` qui est une liste d'« items » non numérotés dont le premier niveau est marqué par un tiret (–) en version française et par un point (•) par défaut :

```
... toujours la fin d'une phrase.
\begin{itemize}
\item dans un calcul complexe, un facteur
  du numérateur passe toujours au
  dénominateur
\item une virgule est toujours mal placée
\end{itemize}
et le truc continue, imperturbablement...
```

2.8

... toujours la fin d'une phrase.

- dans un calcul complexe, un facteur du numérateur passe toujours au dénominateur
- une virgule est toujours mal placée

et le truc continue, imperturbablement...

Vient ensuite l'environnement `enumerate`, sur le même principe que le précédent mais où les items sont numérotés. Étant donné que ces environnements peuvent être inclus les uns dans les autres, nous vous présenterons `enumerate` et `description` dans un même exemple :

```
... encore la fin d'une phrase.
\begin{description}
\item[\TeX] The \TeX{}book
\item[\LaTeX] deux livres importants :
  \begin{enumerate}
  \item \LaTeX{} : A Document preparation
    System
  \item The \LaTeX{} Companion
  \end{enumerate}
\end{description}
et le paragraphe continue, encore et encore...
```

2.9

... encore la fin d'une phrase.

TEX The TEXbook

LATEX deux livres importants :

1. LATEX : A Document preparation System
2. The LATEX Companion

et le paragraphe continue, encore et encore...

Les listes de description, qui n'ont pas d'équivalent dans les traitements de texte habituels, sont malheureusement au mieux mal employées, au pire ignorées des débutants sous LATEX.

2.2.3 Tabulations

L'environnement `tabbing` permet d'utiliser les bonnes vieilles tabulations de la machine à écrire. On pose les taquets de tabulations grâce à la commande `\=` et on se déplace de taquet en taquet avec la commande `\>`. En outre, la commande `\\` permet de passer à la ligne.

```
\begin{tabbing}
  à gauche \= au centre \= à droite \\
  \> modéré \\
  \> \> conservateur \\
  xxxxxxxxxxx \= \kill
  \> sans opinion
\end{tabbing}
```

2.10

à gauche au centre à droite
modéré
conservateur
sans opinion

Cet exemple illustre deux autres principes :

- on peut positionner une tabulation avec un « modèle » et ne pas afficher la ligne correspondante avec la commande `\kill` ;
- une nouvelle commande `\=` enlève le taquet qui suit logiquement, s'il existe.

2.2.4 Tableaux

L'environnement pour produire les *tableaux* en LATEX se nomme `tabular`. Le système de bordures n'est pas très sophistiqué, mais, pour des tableaux à bordures simples les résultats sont acceptables :[2]

[2] L'annexe B donne quelques pistes pour trouver des packages permettant de créer des tableaux plus complexes.

```
Voici un tableau :
\begin{tabular}{|r|c|}
  \hline
  deux & trois \\
  cinq & six  \\ \hline
\end{tabular}
```

2.11

Voici un tableau :

deux	trois
cinq	six

on peut donc comprendre grâce à cet exemple, les chose suivantes :

- l'environnement `tabular` attend un paramètre qui est en quelque sorte une « chaîne de format ». À chaque colonne doit correspondre un caractère de positionnement :
  - `r` : alignement à droite ;
  - `c` : centrage ;
  - `l` : alignement à gauche ;
- le caractère `&` est le séparateur des colonnes ;
- la commande `\\` permet de passer à la ligne ;
- les bordures verticales s'insèrent dans la chaîne de mise en page grâce au caractère `|` ;
- les bordures horizontales à l'aide de la commande `\hline`.

On peut donc jouer sur le nombre de `\hline` et de `|` pour changer l'allure des bordures. Le package array permet quelques fantaisies avec les tableaux.

Si la plupart des environnements commencent une nouvelle ligne, ce n'est pas le cas de l'environnement `tabular`. Il crée juste une boîte dans le texte courant.

On peut en outre préciser le positionnement vertical du tableau grâce à un argument optionnel :

```
un :
\begin{tabular}[b]{|c|} a\\ b\end{tabular}
et deux :
\begin{tabular}[t]{|r|} c\\ d\end{tabular}
```

2.12

un : | a / b | et deux : | c / d |

Vous avez donc compris que l'argument `b` (resp. `t`) « pose » (resp. « accroche ») le tableau sur (resp. à) la ligne. Sans cet argument le tableau est centré verticalement, comme dans le premier exemple de la section.

Les tableaux peuvent évidemment ne pas être insérés dans des « phrases » et constituer des « paragraphes » à eux seuls, par exemple en figurant centrés au moyen d'un environnement `center`.

2.2.5 Simulation de terminal

L'environnement `verbatim` insère son contenu mot pour mot. Il offre donc la possibilité de rentrer n'importe quel caractère même spécial, et donc, par exemple d'écrire une portion de code C++:[3]

```
\begin{verbatim}
  class pixel{
     int x,y;
  public:
     pixel(int i=0, int j=0);  ...};
\end{verbatim}
```

```
class pixel{
   int x,y;
public:
  pixel(int i=0, int j=0);  ...};
```

[3] Notez que le package listings est bien mieux adapté à ce genre de problème.

On peut tout écrire dans un environnement `verbatim` *sauf* la séquence de caractères : `\end{verbatim}` !

Il existe deux commandes permettant de produire une portion de texte comme le fait l'environnement `verbatim` : il s'agit de `\verb` et `\verb*`. La forme «étoilée» remplace le caractère « » par «␣».

L'argument de ces commandes n'est pas donné entre accolades ({ }) mais par tout autre caractère : 1° autre que les caractères spéciaux et 2° n'étant pas contenu dans l'argument.

```
La déclaration : \verb+#include<stdlib.h>+
permet d'inclure les prototypes de la
librairie standard du~C.
```

2.14

La déclaration : `#include<stdlib.h>` permet d'inclure les prototypes de la librairie standard du C.

La commande `\verb` ne peut en aucun cas se trouver dans l'argument d'une commande, quelle qu'elle soit.

2.2.6 Citations

Les environnements `quote` et `quotation` permettent d'insérer une citation dans le texte. Voici d'abord `quote` :

```
... encore la fin d'une phrase.
\begin{quote}
  Tout est relatif.\hfill\textbf{Einstein}.

  Il n'est pas certain que tout soit certain.
  \hfill\textbf{Pascal}.
\end{quote}
et le paragraphe interrompu, continue...
```

2.15

... encore la fin d'une phrase.

> Tout est relatif. **Einstein**.
>
> Il n'est pas certain que tout soit certain. **Pascal**.

et le paragraphe interrompu, continue...

La commande `\hfill` insère un espace qui s'étend horizontalement de manière infinie. L'environnement `quotation` diffère quelque peu de `quote` :

```
... encore la fin d'une phrase.
\begin{quotation}
  L'homme est plein d'imperfections mais on
  ne peut que se montrer indulgent si l'on
  songe à  l'époque où il fut créé.\par
  \raggedleft Alphonse \textsc{Allais}.
\end{quotation}
et ce  brave paragraphe qui continue...
```

2.16

... encore la fin d'une phrase.

> L'homme est plein d'imperfections mais on ne peut que se montrer indulgent si l'on songe à l'époque où il fut créé.
>
> Alphonse Allais.

et ce brave paragraphe qui continue...

En fait ces deux environnements sont présentés par Leslie Lamport, l'un (`quote`) pour une ou plusieurs citations courtes, et l'autre (`quotation`) pour une citation longue.

2.3 Notes de marge

La commande `\marginpar` crée un mini-paragraphe dans la marge, la syntaxe est la suivante :

TAB. 2.3 – *Sectioning commands*

`\part{...}`	`\chapter{...}`	
`\section{...}`	`\subsection{...}`	`\subsubsection{...}`
`\paragraph{...}`	`\subparagraph{...}`	

`\marginpar{`texte`}`

Pour distinguer la page droite de la page gauche en mode recto-verso, on pourra utiliser :

`\marginpar[`textegauche`][`textedroite`]`

où textgauche et textdroite seront respectivement les textes qui apparaîtront en marge selon la parité du numéro de la page. Ainsi :

```
\marginpar[Youhou !][Coucou !]
```

Coucou

donne ce que vous pouvez constatez dans la marge.

2.4 Titres

Le tableau 2.3 montre les commandes de *section* disponibles dans LaTeX. La commande `\chapter` n'est pas disponible pour la classe de document `article` ; et aucune commande de titres ne peut être utilisée dans la classe `letter`. Pour l'instant, il faut savoir les deux choses suivantes :

- chaque titre résultant d'une commande de section est automatiquement *numéroté* et mis dans la table des matières le cas échéant ;
- la commande `\tableofcontents` produit une *table des matières* à l'endroit où est insérée cette commande.

D'autre part toutes les commandes de titres ont un style associé que l'on peut éventuellement redéfinir. Enfin, ces commandes effectuent automatiquement les espacements verticaux avant et après le titre ; ainsi toute ligne vierge insérée avant ou après la commande est ignorée.

```
... tiens c'est la fin d'une phrase.
\section{La quantification}
Le processus de quantification...
```

... tiens c'est la fin d'une phrase.

2.17

3.2 La quantification

Le processus de quantification...

Il existe une forme « étoilée » (par exemple : `\section*`) de chaque commande de titres permettant d'insérer un titre non numéroté. Mais attention, ce titre n'apparaîtra pas dans la table des matières. Les commandes de section prennent également un argument optionnel permettant de préciser une entrée de table matières différente du titre de la section. Par exemple :

```
\section[Paulette]{C'était bien, c'était chouetteuuuu}
```

insère « Paulette » dans la table des matières en lieu et place du titre inséré dans le document.

2.5 Notes de bas de page

L'insertion d'une note de bas de page s'effectue de manière simple par la commande `\footnote{`texte`}`. La numérotation est automatique, et par défaut, les notes sont numérotées à l'intérieur d'un chapitre. Voici ce que donne LaTeX :

```
Contre toute attente, c'est la commande
\verb+footnote+\footnote{Comme son
nom l'indique...} qui insère une note
de bas de page.
```

Contre toute attente, c'est la commande `footnote`[a] qui insère une note de bas de page.

[a] Comme son nom l'indique...

Il arrive lorsqu'on travaille en milieu particulièrement hostile que la commande `\footnote` ne produise pas l'effet désiré. Il est alors nécessaire de procéder en deux temps :

1. poser une *marque de note*, commande `\footnotemark` ;
2. entrer le *texte* de la note de base de page — commande `\footnotetext` — lorsque les conditions sont plus favorables.

Par exemple, il semble délicat de mettre une note de bas de page dans un tableau, on écrira donc :

```
\begin{tabular}{cc}
  un & deux\footnotemark \\
  trois & quatre
\end{tabular}\footnotetext{Une note.}
```

un	deux[a]
trois	quatre

[a] Une note.

2.6 Entête et pied de page

Les commandes standard de LaTeX permettant de personnaliser les entêtes et pied de page sont assez rudimentaires mais méritent d'être mentionnées ici, puisqu'elles peuvent suffire dans certains cas.

Nous ne nous attarderons pas plus sur ces commandes, car il nous semble que le package fancyhdr — documenté dans `fancyhdr.dvi` — est beaucoup plus confortable à utiliser et offre des fonctionnalités bien plus intéressantes que les options standard de LaTeX. L'utilisation de ce package pour produire les entête et pied de page du manuel que vous avez sous les yeux est expliquée à la section 10.4.

Sans faire appel à un package particulier, on peut spécifier le style d'entête et pied de page à l'aide de la commande `\pagestyle` :

`\pagestyle{`style`}`

dans le préambule du document ; le paramètre *style* pouvant prendre les valeurs suivantes :

- `empty` ni entête, ni pied de page ;
- `plain` c'est le style par défaut, le pied de page contient les numéros de pages centrés ;
- `headings` suivant le style de document, un certain nombre d'informations est inséré dans l'entête et le pied de page (par exemple dans le style `report` en recto-verso, est inséré en entête : soit le titre du chapitre en cours, soit le titre de la section en cours) ;

– `myheadings` un style qui permet de personnaliser les informations à insérer.

Il existe d'autre part une commande `\thispagestyle`, qui permet de changer ou de spécifier le style de la page courante.

2.7 Flottants

LaTeX offre à ses valeureux utilisateurs la possibilité d'utiliser des environnements *flottants*. Ces environnements ont la particularité de rendre « flottants » leur contenu. C'est-à-dire que LaTeX choisit à partir d'un algorithme qui tient compte d'un certain nombre de paramètres, la position de l'environnement dans le document.

Contrairement à ce que leur nom laisse croire, les environnements de LaTeX `figure` et `table` ne sont pas spécialement conçus pour insérer des figures et des tables ! En fait ils sont conçus uniquement pour faire flotter leur contenu et laisser la possibilité d'insérer une légende. Le contenu à proprement parler peut être constitué de ce que bon vous semble, pas nécessairement du graphique.

2.7.1 Figure et table

L'environnement `figure` est en général utilisé pour les graphiques, et l'environnement `table`, pour les tableaux. Chacun de ces environnements possède une légende. La syntaxe d'utilisation est la suivante :

```
Ce paragraphe contient un environnement
flottant de type « figure ». Le
contenu est donc susceptible de se
déplacer dans la page.
\begin{figure}
  \begin{center}
    0 + 0 \\
    =
  \end{center}
  \caption{La tête à toto}
\end{figure}
```

Ce paragraphe contient un environnement flottant de type « figure ». Le contenu

$$0 + 0$$
$$=$$

Figure 1: La tête à toto

est donc susceptible de se déplacer dans la page.

Vous noterez que c'est la commande `\caption` qui produit la légende. Le texte « Figure 1: » est inséré automatiquement avec le numéro correspondant à la figure. Le « style » de la légende est bien entendu personnalisable.

2.7.2 Placement

LaTeX tente de placer le contenu flottant en fonction des paramètres qu'on indique entre crochets après le `\begin` du flottant :

– `h` : là où il apparaît dans le source ;
– `t` : en haut de la page ;
– `b` : en bas de la page ;
– `p` : seul sur une page

Notons qu'il arrive parfois que l'on s'arrache les cheveux, pour placer les environnements flottants. Pour ne pas s'énerver, il faut comprendre — et accepter — que LaTeX utilise plusieurs paramètres pour placer les `figure` et `table`. Notons parmi ces paramètres :

– le nombre maximum d'environnements flottants en haut et en bas de page ;

– le pourcentage maximum de la surface de la page qu'occupe un flottant en haut et en bas de la page ;
– les espacements avant et après le flottant.

Si vous avez des problèmes[4] pour placer vos figures, nous vous conseillons de suivre ces quelques recommandations :

– si vous tenez à écrire « *comme le montre la figure :* » en attendant la figure à la suite, *n'utilisez pas l'environnement* `figure` !
– utilisez plutôt le système de *référence* et écrivez « comme le montre la figure 3 » ;
– on a toujours tendance à faire des figures énormes : rétrécissez-les !
– si vous avez des tableaux à rallonge, mettez-les en annexe, puisque de toutes façons ils gêneront le lecteur ;
– les paramètres de LaTeX sont étudiés pour équilibrer le texte et les figures dans le document. Donc, si votre document est une bande dessinée, attendez vous au pire...
– ne vous souciez du placement des figures qu'au moment d'**imprimer votre document final**.

2.7.3 Liste des figures

La commande `\listoffigures` (resp. `\listoftables`) insère une liste des figures (resp. des tableaux) de votre document. La liste est imprimée là où apparaît la commande. Ces commandes produisent un fichier portant l'extension `.lof` (resp. `.lot`). En outre, de manière analogue aux commandes de sections qui alimentent la table des matières, la commande `\caption` prend un argument optionnel permettant de définir l'entrée dans la table des figures. Par défaut cette commande utilise la légende comme entrée :

```
\caption[Hop]{Ici on peut raconter sa vie puisque ça mettra
  pas le « foin » dans la liste des figures avec un titre à
  rallonge vu qu'on a mis « Hop » à la place de cette légende qui
n'en finit pas...}
```

2.8 Références

Le système de référence de LaTeX permet de manipuler le numéro de toute partie d'un document faisant l'objet d'une numérotation, de manière symbolique. Donc sans se soucier de savoir s'il s'agit par exemple, de la figure 4 ou de la figure 5. C'est un des aspects de LaTeX qui vous évitera beaucoup de travail. Et qui s'explique en quelques lignes.

2.8.1 Principe

Pour utiliser une référence, on a deux tâches à effectuer : 1° poser une étiquette symbolique dans le texte, 2° appeler cette étiquette pour faire référence, soit au numéro de l'objet référencé, soit au numéro de la page où se trouve l'objet référencé. C'est d'une simplicité enfantine :

[4] Et vous en aurez sûrement...

1. On pose une étiquette avec la commande `\label` :

   `\label{`étiquette`}`

   où étiquette est une chaîne de caractères ne comprenant pas de caractères spéciaux.
2. On fait référence au numéro de l'objet référencé avec la commande `\ref` :

   `\ref{`étiquette`}`

On fait ensuite référence à la page avec `\pageref` :

`\pageref{`étiquette`}`

2.8.2 Que référencer ?

Les objets que l'on peut référencer sont les suivants :
- les titres ;
- les flottants (`figure`, `table`, ...) ;
- les équations (cf. chapitre 3) ;
- les items de liste énumérée (`enumerate` par exemple) ;
- etc.

Voici un exemple synthétisant les trois commandes de référencement :

```
\section{Second degré}\label{sec-2degre}
Ce sont les équations du type :
\begin{equation}
  ax^2 + bx + c = 0 \label{equ}
\end{equation}
L'équation \ref{equ} de la section
\ref{sec-2degre} page \pageref{sec-2degre}
patati patala...
```

3.5 Second degré

Ce sont les équations du type :

$$ax^2 + bx + c = 0 \tag{2.12}$$

2.21

L'équation 2.12 de la section 3.5 page 13 patati patala...

Dans cet exemple on fait référence à une `\section` et une `\equation` (cf. chapitre 3). En outre, on fait référence à la page où apparaît la `section` en question.

Lorsque vous placez un `\label` dans un environnement flottant, placez le toujours **après** la commande `\caption`. Sinon, la référence « pointera » sur la section et non sur la figure.

2.9 Fichiers auxiliaires

Pour bien comprendre le mécanisme de référencement, il nous reste à examiner ce que LaTeX écrit sur votre disque lorsqu'il compile un fichier source. Pour l'instant, voici les fichiers que vous pourrez rencontrer :

`dvi` l'image de votre document ;

`log` c'est le bavardage de LaTeX lors de la dernière compilation. En général, il correspond peu ou prou à ce que vous avez sur votre terminal au moment de la compilation ;

`aux` le fichier auxiliaire, il stocke les informations concernant les références, les numéros de pages, les titres, ... ;

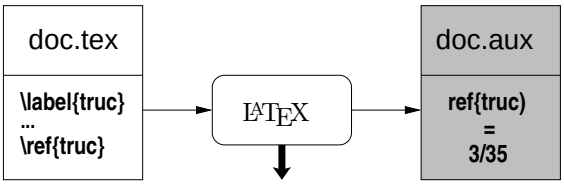
`toc` le fichier contenant la table des matières ;

`lof` le fichier contenant la liste des figures.

2.9.1 Interaction avec les références

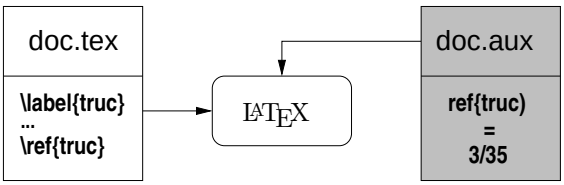
LaTeX gère les références de la manière suivante : lors d'une première compilation, il stocke les références dans le fichier nom-doc`.aux` où nom-doc est le nom de votre document. À l'aide d'un exemple où l'on aurait placé une étiquette `truc` pour la section 3 à la page 35 d'un document, voyons le principe du mécanisme de résolution des références.

1. la première compilation avec LaTeX stocke dans le fichier auxiliaire `.aux` le numéro de l'étiquette (le numéro de la section dans notre exemple) et le numéro de la page où cette étiquette apparaît :



   LaTeX envoie donc lors de cette compilation un avertissement précisant que l'étiquette `truc` est indéfinie ;

2. on effectue donc une deuxième compilation qui va cette fois exploiter le contenu du fichier auxiliaire :



Les références peuvent être incorrectement définies dans les situations suivantes :

1. vous avez inséré une nouvelle étiquette, et c'est la première compilation que vous effectuez (les références sont *indéfinies*) ; et vous aurez pour cette nouvelle étiquette un message :

```
Reference ‘vlunch’ on page 2 undefined on input line 17.
```

2. les changements que vous avez apportés à votre document ont sans doute changé la numérotation des pages ou le placement des objets (figures, équations,...), les références sont alors *mal définies*, et vous serez averti par un message en fin de compilation :

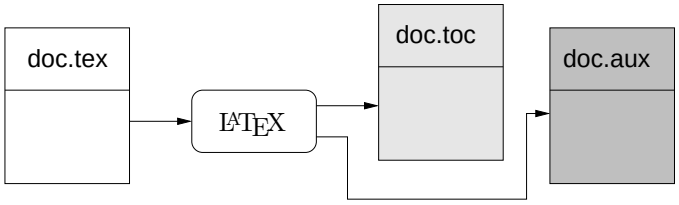
```
Label(s) may have changed.
Rerun to get cross-references right.
```

3. vous faites référence à une étiquette qui n'existe pas. Dans ce cas, 18 compilations ne changeront rien à votre problème.

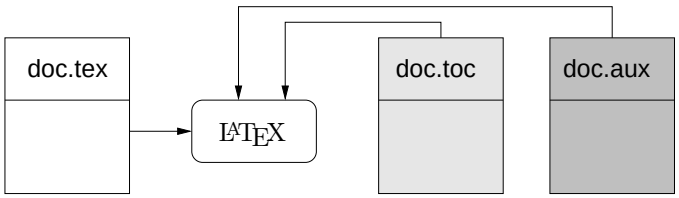
2.9.2 Interaction avec la table des matières

On retrouve un peu le même principe avec la table des matières. Lorsque vous insérez la commande `\tableofcontents` dans votre document, la table des matières va être créée en deux étapes, comme suit :

1. un premier parcours pour récupérer les informations liées aux *titres* de tout le document et stockage dans le fichier nom-du-document`.toc` :



2. un deuxième passage pour inclure nom-du-document`.toc`— donc la table des matières — dans le document final :



Vous serez alors confrontés au phénomène suivant : lorsqu'au cours de la rédaction d'un document contenant déjà l'ordre `\tableofcontents`, vous insérez une commande de section, elle n'apparaîtra dans la table des matières qu'après **deux** compilations.

2.9.3 Petits conseils

Prenez l'habitude de créer un répertoire pour chaque document que vous rédigez. LaTeX crée en effet plusieurs fichiers autour de votre `.tex`[5]. D'autre part, ne vous souciez pas trop, lors de la rédaction de votre document, de savoir si les références ou la table des matières sont à jour : elles le seront bien un jour ou l'autre ! En fait, il faut s'assurer que les références sont correctes avant d'**imprimer**.

Enfin, de même qu'on effectue de temps en temps un `make clean` lorsqu'on n'est plus sûr de ses fichiers objets, il est bon quand il vous semble que tout va mal, d'effacer les fichiers auxiliaires et de reprendre la compilation.

2.10 Où il est question de césure

LaTeX s'appuie sur TeX pour effectuer la césure des mots en fonction d'une langue déterminée. Cet algorithme décrit à l'annexe H du TeXbook constitue un des aspects les plus réussis de TeX. Une manière de reconnaître un document généré par LaTeX est d'examiner la manière dont sont coupés les paragraphes ; beaucoup d'autres logiciels se contentent d'insérer des blancs entre les mots. Il existe cependant des situations où LaTeX ne peut effectuer une césure correcte. Dans ce cas, LaTeX vous avertira par l'un des deux messages terrifiants :

[5] Et encore il n'a pas encore été question de bibliographie, ni d'index et de glossaires...

```
Underfull \hbox (badness 1810) detected at line 33
```

ou bien :

```
Overfull \hbox (14.24376pt too wide) detected at line 41
```

À un très bas niveau, TEX produit votre document en assemblant des *boîtes*. Chaque caractère est contenu dans une boîte qui lui est propre ; les mots sont formés par assemblage de ces boîtes. Et ainsi de suite, pour les lignes qui forment des paragraphes puis des pages.

Pour résumer et présenter les choses de manière simple, disons que TEX est en mode horizontal pour assembler les « mots » et manipule alors des `\hbox` ; il est en mode vertical et manipule des `\vbox` lorsqu'il crée les pages. Aussi, lors de l'assemblage de ces boîtes, si TEX juge que le résultat ne sera pas esthétique, il vous avertira par les deux types de messages présentés plus haut. Ces messages ont la signification suivante :

- `Underfull \hbox` les boîtes sont assemblées de manière un peu lâche ; TEX vous donne la « laideur » de la ligne (badness) sachant qu'une ligne parfaite a une laideur de 0, et que la pire des lignes, une laideur de 10000 ;
- `Overfull \hbox` les boîtes sont un peu trop serrées ; TEX vous indique en `pt` le dépassement dans la marge.

Si une page est trop lâche, LATEX parlera de `\vbox` dans ses messages. Le tableau 2.4 illustre le phénomène sur une phrase.

Ô rage ! ô désespoir ! ô vieillesse ennemie !	underfull
Ô rage ! ô désespoir ! ô vieillesse ennemie !	underfull
Ô rage ! ô désespoir ! ô vieillesse ennemie !	underfull
Ô rage ! ô désespoir ! ô vieillesse ennemie !	ok
Ô rage ! ô désespoir ! ô vieillesse ennemie !	overfull
Ô rage ! ô désespoir ! ô vieillesse ennemie !	overfull
Ô rage ! ô désespoir ! ô vieillesse ennemie !	overfull

TAB. 2.4 – Under et overfull hbox

Il est possible en utilisant l'option de document `draft` de faire apparaître dans la marge une barre noire comme en marge de ce paragraphe, indiquant les `Overfull \hbox`. Cette option permet de localiser rapidement la ligne en cause.

2.10.1 Contrôler la césure

LATEX peut avoir des difficultés à couper une phrase pour les raisons suivantes :

- il ne reconnaît pas le mot à couper : ce cas est exceptionnel ;
- l'endroit où la césure devrait avoir lieu est un objet qui ne peut être coupé, par exemple un objet du type `\verb|...|`, une équation,...

Nous vous donnons ci-dessous quelques méthodes pour contrôler la césure.

Lorsqu'aucune de ces méthodes ne vous donne satisfaction — ceci peut se produire si votre phrase contient trop d'objets que TEX ne peut couper — il n'y a pas d'autre solution que de tourner sa phrase différemment pour contourner le problème.

Guider la césure

On peut aider LaTeX à couper un mot en lui indiquant les endroits où la césure peut être effectuée, en insérant aux endroits nécessaires, la commande `\-`. Par exemple, si LaTeX a du mal à couper le mot « nonmaiçavapamieu », vous pouvez entrer :

```
non\-mai\-ça\-va\-pa\-mieu
```

Si vous utilisez ce mot fréquemment, vous pouvez, pour vous épargner d'indiquer les césures comme ci-dessus, entrer dans le préambule la commande `\hyphenation` :

```
\hyphenation{non-mai-ça-va-pa-mieu}
```

qui indique à LaTeX comment couper ce mot étrange.

Forcer la césure

Vous pouvez forcer la césure, en insérant la commande `\linebreak[`nombre`]`, mais cela peut avoir des résultats catastrophiques. Si vous voyez ce que je veux dire. Le paramètre nombre permet de moduler la commande `\linebreak`. Vous avez la possibilité de formuler un souhait timide : `\linebreak[0]` ou un ordre à ne pas discuter : `\linebreak[4]`.

La commande `\pagebreak[`nombre`]` est la commande correspondant aux coupures de pages. D'autre part, deux commandes sont disponibles pour effectuer un saut de page :

- `\clearpage` finit la page actuelle ;
- `\cleardoublepage` finit la page actuelle, et assure de commencer sur une page impaire, en mode recto verso.

Ces deux commandes forcent LaTeX à insérer toutes les figures flottantes en cours de placement.

Une autre intervention manuelle pratique dans certaines situations, consiste à agrandir la hauteur de la page actuelle avec la commande `\enlargethispage` suivi d'une dimension puis d'insérer un saut de ligne :

```
\enlargethispage{10cm}          ← au niveau de la page trop courte
[...le texte un peu trop long...]
\clearpage                      ← fin explicite de la page allongée de 10cm
```

Empêcher la césure

Il existe trois moyens de forcer LaTeX à ne pas couper le texte :

1. insérer l'espace insécable `~` ;
2. mettre un mot dans une boîte[6] avec la commande `\mbox{`mot`}` ;
3. utiliser l'ordre `\nolinebreak` :

   `\nolinebreak[`nombre`]`

   pour empêcher les sauts de lignes, et la commande `\nopagebreak` :

   `\nopagebreak[`nombre`]`

   où nombre a la même signification que pour les commandes `\linebreak` et `\pagebreak`.

[6] Car TeX ne coupe **jamais** une boîte.

Conclusion

Ce chapitre a présenté les fonctionnalités standard de LaTeX. Si vous avez lu attentivement jusqu'ici, vous devriez pouvoir produire n'importe quel document simple (sans formule ni graphique, pour l'instant). Si vous n'êtes pas encore en mesure de personnaliser vos documents, ils seront tout de même d'une très bonne qualité typographique en vous évitant de vous poser des questions métaphysiques sur la « bonne » largeur d'une marge ou le « bon » écart entre un titre et le texte,... En effet, les comportements par défaut de LaTeX répondent, pour la plupart, à des règles en usage dans le monde de l'imprimerie.

3 Mathématiques

Sommaire

Voici les noms des douze apôtres :
en tête Simon que l'on appelle Pierre [...]
L'Évangile selon Saint Matthieu Mt **10** 2.

Un des aspects pratique et rigolo[1] de LaTeX est bien sûr la génération de formules mathématiques ; elles seront naturellement belles, sans que vous n'ayez à faire quoique ce soit.[2] De plus, si vous avez un mauvais souvenir d'un certain éditeur d'équations, réjouissez vous : vous n'avez pas besoin de souris pour écrire des équations ! La génération d'équations avec LaTeX est un domaine particulièrement vaste. Nous présenterons ici les bases requises pour produire les formules « usuelles ». Ce chapitre ne constitue donc qu'une petite introduction à la manipulation des formules avec LaTeX.

Les commandes standard de LaTeX permettent de produire la plupart des équations mathématiques usuelles. Il est cependant conseillé d'utiliser les extensions de l'*American Mathematical Society* nommées amsmath et amssymb simplifiant la mise en forme dans beaucoup de situations.

3.1 Les deux façons d'écrire des maths

LaTeX distingue deux manières d'écrire des mathématiques. L'une consiste à insérer une formule dans le texte, comme ceci : $ax + b = c$, l'autre à écrire une ou plusieurs formules dans un environnement, par exemple :

$$\mathrm{d}U = \delta\mathcal{W} + \delta\mathcal{Q}$$

sachant que chacun de ces deux modes respectent un certain nombre de principes quant à la taille et la position des différents symboles. Voici un exemple avec les deux modes :

[1] Si, si ! Il y a même des gens qui font des formules juste pour le plaisir !
[2] Ou alors juste deux ou trois petites choses...

```
Déterminer la fonction dérivée
de $f(x)$ :
\begin{displaymath}
  f(x)=\sqrt{\frac{x-1}{x+1}}
\end{displaymath}
si elle existe.
```

3.1

Déterminer la fonction dérivée de $f(x)$:

$$f(x) = \sqrt{\frac{x-1}{x+1}}$$

si elle existe.

Cet exemple nous montre donc que l'on entre en mode mathématique « interne » grâce au symbole `$`, et que le même symbole `$` permet d'en sortir. D'autre part, on utilise ici l'environnement `displaymath` qui est le plus simple pour produire des équations. Ce dernier peut être saisi grâce aux commandes `\[` et `\]` (cf. § 3.7.1 page 42).

Nous vous présenterons au § 3.7 les différents environnements de LaTeX.

3.2 Commandes usuelles

3.2.1 Indice et exposant

Comme mentionné au § 1.4.1 page 10, `_` et `^` sont les commandes permettant de produire respectivement *indice* et *exposant*. Il est nécessaire de « grouper » les arguments entre accolades pour que ces commandes agissent sur plusieurs symboles.

`x_2`	x_2	`x_{2y}`	x_{2y}	`x_{t_0}`	x_{t_0}
`x^2`	x^2	`x^{2y}`	x^{2y}	`x_{t^0}`	x_{t^0}
		`x^{2y}_{t_0}`	$x^{2y}_{t_0}$	`x_{t^1}^{2y}`	$x^{2y}_{t^1}$

3.2.2 Fraction et racine

Voici comment produire *racines* et *fractions* :

- la commande `\frac{num}{denom}` produit une fraction formée par le numérateur num et le dénominateur denom ;
- la commande `\sqrt[n]{expr}` affiche la racine n$^{\text{e}}$ de son argument arg.

Notons que ces deux commandes ne produisent pas le même affichage selon le mode mathématique : interne ou équation. Ainsi voici une fraction : $\frac{1}{\sin x+1}$ et une racine : $\sqrt{3x^2-1}$ et leur équivalent en mode équation :

$$\frac{1}{\sin x + 1} \quad \sqrt{3x^2 - 1}$$

Pour en finir avec ces deux commandes, voyons comment elles peuvent être imbriquées et l'effet que cela produit :

```
\begin{displaymath}
  \sqrt{\frac{1+\sqrt[3]{3x+1}}
             {3x+\frac{1-x}{1+x}}}
\end{displaymath}
```

3.2

$$\sqrt{\frac{1+\sqrt[3]{3x+1}}{3x+\frac{1-x}{1+x}}}$$

3.2.3 Symboles

Symboles usuels

Le tableau 3.1 donne les macros produisant une partie des symboles dont vous pourriez avoir besoin.

TAB. 3.1 – Symboles mathématiques usuels

\pm	$\pm$	\otimes	$\otimes$	\cong	$\cong$	\imath	$\imath$
\mp	$\mp$	\oslash	$\oslash$	\subset	$\subset$	\jmath	$\jmath$
\div	$\div$	\odot	$\odot$	\supset	$\supset$	\ell	ℓ
\ast	$\ast$	\leq	$\leq$	\subseteq	$\subseteq$	\aleph	$\aleph$
\times	$\times$	\geq	$\geq$	\supseteq	$\supseteq$	\nabla	∇
\bullets	$\bullet$	\equiv	$\equiv$	\in	$\in$	\\|	$\|$
\circ	$\circ$	\ll	$\ll$	\ni	$\ni$	\partial	∂
\star	$\star$	\gg	$\gg$	\emptyset	$\emptyset$	\wedge	$\wedge$
\setminus	$\setminus$	\sim	$\sim$	\forall	$\forall$	\vee	$\vee$
\oplus	$\oplus$	\simeq	$\simeq$	\infty	∞	\cup	$\cup$
\ominus	$\ominus$	\approx	$\approx$	\exists	$\exists$	\cap	$\cap$

Nous avons recensé près de 450 symboles disponibles avec les packages latexsym et amssymb. Notre but n'est donc pas de les présenter ici ! Le tableau 3.1 est une sélection parmi les symboles standard. Nous avons jugé qu'ils faisaient partie des symboles les plus utiles — ce qui, malgré la présence tout à fait fortuite de l'aleph dans ce tableau, démontre que le niveau en mathématiques de l'auteur de ce document avoisine le ras des pâquerettes.

Points de suspension

On utilise couramment pour économiser de l'encre des points de suspension dans des formules. Il en existe de trois types. La commande \dots produit des points « posés » sur la ligne :

```
$C=\{\vec{c}_0,\vec{c}_1,\dots,\vec{c}_N\}$
est l'ensemble des $N$ couleurs.
```

3.3 $C = \{\vec{c}_0, \vec{c}_1, \dots, \vec{c}_N\}$ est l'ensemble des N couleurs.

La commande \cdots produit des points centrés verticalement sur le signe égal :

```
$\vec{\mu}=\frac{1}{N}
(\vec{c}_0+\vec{c}_1+\cdots+\vec{c}_N)$
est la moyenne des $N$ couleurs.
```

3.4 $\vec{\mu} = \frac{1}{N}(\vec{c}_0 + \vec{c}_1 + \cdots + \vec{c}_N)$ est la moyenne des N couleurs.

Enfin les commandes \vdots et \ddots sont à utiliser essentiellement dans les matrices (cf. § 3.6 et l'exemple 3.15). Ces deux commandes produisent respectivement : $\vdots$ et $\ddots$.

Flèches

Voici un moyen simple pour mémoriser les commandes permettant de générer des flèches :

- toutes les commandes finissent par `arrow` ;
- le préfixe obligatoire `left` ou `right` indique la direction ;
- le préfixe facultatif `long` donne une version longue ;
- la première lettre de la commande mise en majuscule rend la flèche double ;
- on peut mettre des flèches aux deux extrémités en collant les deux mots `left` et `right`.

ainsi :

`\rightarrow`	donne	$\rightarrow$
`\Longleftarrow`	donne	$\Longleftarrow$
`\Leftarrow`	donne	$\Leftarrow$
`\Longleftrightarrow`	donne	$\Longleftrightarrow$

Lettres grecques

Les lettres grecques s'utilisent de la manière la plus simple qui soit : en les appelant par leur nom. Ainsi : `\alpha` donne « α » et `\pi`, « π ». Mettre une majuscule à la première lettre de la commande, donne la majuscule correspondante : `\Gamma` donne « Γ ». Attention, toutes les majuscules ne sont pas disponibles dans l'alphabet grec, on mettra par exemple α en majuscule, avec la lettre A (la commande `\Alpha` n'existe pas).

L'ensemble des réels

Une question « cruciale » que se posent les rédacteurs potentiels de documents scientifiques est : « Comment peut/doit-on écrire le 'R' de l'ensemble des réels ? ». Les avis sont partagés à ce sujet. Historiquement il semble qu'initialement, dans les ouvrages de mathématiques, le symbole des réels était typographié en gras (« Soit $x \in \mathbf{R}$ ») et que les professeurs pour reprendre ces notations sur un tableau avec une craie avaient recours à l'artifice de repasser plusieurs fois sur la lettre « R » ; cette pratique pénible aurait évoluée vers l'écriture « bien connue » : « Soit $x \in \mathbb{R}$ ». Il y a donc les adeptes du $\mathbf{R}$, du $\mathbb{R}$, etc. Pour choisir par soi-même, voir les packages :

- bbm qui propose la commande `\mathbbm{R}` produisant $\mathbb{R}$, la commande `\mathbbmss{R}` produisant $\mathbb{R}$, etc.
- bbold qui propose la commande `\mathbbm{R}` produisant $\mathbb{R}$, etc.
- amssymb qui propose les commandes `\mathbb{R}` produisant $\mathbb{R}$ et `\mathbf{R}` produisant $\mathbf{R}$

3.3 Fonctions

3.3.1 Fonctions standards

Lorsqu'on veut produire des fonctions mathématiques classiques (logarithmes, trigonométrie,...), il faut utiliser les fonctions de LaTeX prévues à cet effet. Voici un exemple pour vous en convaincre.

```
$\sin^2x + \cos^2 x=1$
```

3.5

$\sin^2 x + \cos^2 x = 1$

Et sans les fonctions LaTeX :

```
$sin^2x + cos^2x=1$
```

3.6

$sin^2x + cos^2x = 1$

La différence réside dans le fait que LaTeX traite la chaîne `cos` comme une suite de variable (donc produites en italiques) alors que la fonction `\cos` produit « cos » en roman. Une autre différence importante est le placement d'éventuels indices (cf. l'exemple de la fonction `\max` ci-dessous). Parmi les fonctions mathématiques standard de LaTeX, on trouvera :

- toutes les fonctions trigonométriques : `\sin`, `\cos` et `\tan`. En rajoutant `arc` devant, vous aurez les réciproques, et `h` derrière vous obtiendrez les versions hyperboliques.
- les logarithmes népérien et décimal définis respectivement par les fonctions `\ln` et `\log`.
- les fonctions `\sup`, `\inf`, `\max`, `\min`, et `\arg` qui vous permettront de générer des formules de ce genre :

```
\begin{displaymath}
  T=\arg \max_{t<0} f(t)
\end{displaymath}
```

3.7

$$T = \arg\max_{t<0} f(t)$$

Notez l'utilisation de l'opérateur indice `_` et le placement résultant avec la commande `\max`.

3.3.2 Intégrales, sommes et autres limites

LaTeX utilise une syntaxe simple pour produire *intégrales*, *sommes*, etc. La syntaxe est la suivante :

\op_{inf}^{sup}

où op est l'un des opérateurs `sum`, `prod`, `int` ou `lim` et inf et sup sont les bornes inférieure et supérieure de la somme ou de l'intégrale. Ainsi on peut donc écrire :

```
Somme des termes d'une suite
géométrique :
\begin{displaymath}
  \sum_{i=0}^{n}q^i=\frac{1-q^{n+1}}{1-q}
\end{displaymath}
```

3.8

Somme des termes d'une suite géométrique :

$$\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q}$$

Le produit $\prod$ s'utilise de manière analogue avec la commande `\prod`. Un exemple avec une intégrale, en veux-tu en voilà :

```
On définit le logarithme
népérien de $x>0$ comme suit :
\begin{displaymath}
  \ln(x)=\int_{1}^{x}\frac{1}{t}\,\mathrm{dt}
\end{displaymath}
```

3.9

On définit le logarithme népérien de $x > 0$ comme suit :

$$\ln(x) = \int_1^x \frac{1}{t}\,\mathrm{dt}$$

La commande `\,` permet d'insérer un léger blanc avant le « dt » (cf. § 3.5.1). Si vous êtes plutôt *curviligne*, vous pouvez utiliser `\oint` qui donne : $\oint$. Bon, je vous donne juste un exemple avec une limite mais c'est bien parce que c'est vous :

```
$f(x)$ admet une limite $\ell$ en $x_0$ :
\begin{displaymath}
  \lim_{x\rightarrow x_0}f(x)=\ell
\end{displaymath}
```

3.10

$f(x)$ admet une limite ℓ en x_0 :

$$\lim_{x\rightarrow x_0} f(x) = \ell$$

J'espère que vous avez apprécié le beau ℓ ; pour se fixer les idées sur les deux modes mathématiques, voici les mêmes formules mais incrustées dans le texte. Donc d'abord la sommation : $\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q}$, ensuite l'intégrale : $\int_1^x \frac{1}{t}dt = \ln(x)$, et enfin la limite : $\lim_{x\rightarrow x_0} f(x) = \ell$.

3.4 Des symboles les uns sur les autres

3.4.1 L'opérateur `not`

L'opérateur `\not` permet de produire la « négation » d'une relation :

```
Soit $x \not\in I$ un réel...
```

3.11

Soit $x \notin I$ un réel...

le résultat est donc un « slash » sur le symbole suivant. **Attention**, cet opérateur n'est pas très performant : `$\not\longrightarrow$` donne : $\not\longrightarrow$, mais est satisfaisant pour les symboles d'une largeur raisonnable.

3.4.2 Accents

Il est souvent utile,[3] d'accentuer les symboles en guise de notation particulière. Voici les accents disponibles :

`\hat{x}`	$\hat{x}$	`\check{x}`	$\check{x}$	`\breve{x}`	$\breve{x}$
`\acute{x}`	$\acute{x}$	`\grave{x}`	$\grave{x}$	`\tilde{x}`	$\tilde{x}$
`\bar{x}`	$\bar{x}$	`\dot{x}`	$\dot{x}$	`\ddot{x}`	$\ddot{x}$

3.4.3 Vecteurs

Il existe deux[4] façons d'obtenir un vecteur :
- `\vec` pour les petits symboles car `\vec` est une commande d'accentuation ;
- `\overrightarrow` dans les autres cas.

```
Soit $\overrightarrow{A\!B}$ défini
dans la base $(\vec{\imath},\vec{\jmath})$.
```

3.12

Soit $\overrightarrow{AB}$ défini dans la base $(\vec{\imath},\vec{\jmath})$.

Notez que `$\vec{A\!B}$` aurait donné : $\vec{AB}$ (voir aussi le paragraphe 3.5.1 pour la signification de la commande `\!`). Remarquez également les commandes `\imath` et `\jmath` qui fournissent les lettres « i » et « j » sans point : $\imath$ et $\jmath$.

[3] En réalité les mathématiciens dignes de ce nom raffolent de ce genre de petits chapeaux au dessus des symboles ; certains en superposent même deux, voire trois...

[4] Le package esvect d'Eddie SAUDRAIS permet de produire des vecteurs avec des flèches mieux dessinées que celles proposées ici.

TAB. 3.2 – Espacement en mode mathématique

`\!` □□	*(rien)* □□	`\,` □□	`\:` □□
`\;` □□	`\␣` □□	`\quad` □□	`\qquad` □□

3.4.4 Commande `stackrel`

La commande `\stackrel` permet de poser deux symboles l'un sur l'autre :

`\stackrel{`symb$_1$`}{`symb$_2$`}`

met le symb$_1$ sur symb$_2$. Par exemple :

```
x\stackrel{f}{\longmapsto}y
```

donne : $x \overset{f}{\longmapsto} y$.

3.5 Deux principes importants

Pour bien comprendre la manière dont LaTeX génère les formules, il faut saisir les deux principes suivants :

Espaces : LaTeX ignore les espaces entre les symboles mathématiques ; ainsi : `$x+1$` produira la même formule que `$x + 1$`. C'est LaTeX qui insère les espaces à l'endroit qu'il juge que le plus judicieux ;

Texte[5] **:** tout groupe de symboles est considéré comme un groupe de variables ou fonctions ; ainsi `$x=t avec t>0$` produira $x = tavect > 0$ et non ce que vous espériez : $x = t$ avec $t > 0$.

Une fois ces deux principes acquis, voyons comment on peut faire avec.

3.5.1 Espaces en mode mathématique

Tout d'abord, sachez que LaTeX fait un choix d'espacement qui est en général correct. Cependant le jour où vous aurez à jouer l'artiste de mouche, les commandes du tableau 3.2 vous permettront d'insérer un ou des espaces dans des formules. Dans ce tableau, on montre l'effet des commandes d'espacement entre deux symboles □.

Pour ce qui concerne les mouches, sachez que l'auteur de ce manuel a sournoisement inséré un certain nombre d'espacements au numérateur du calcul de la somme des termes de la suite géométrique (§ 3.3.2 page 37), pour aligner les deux q de la fraction. Voici ce que donnait la formule par défaut :

$$\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q}$$

et voyons si les histoires de q vous donnent le sens de l'observation.

[5] L'insertion de texte dans une formule ne devient un problème que dans un environnement de la famille `displaymath`, puisque vous pouvez toujours écrire « `$x=t$ avec $t>0$` », bien sûr !

3.5.2 Texte en mode mathématique

Le moyen le plus simple d'insérer du texte dans une formule est de de le mettre « en boîte » et d'insérer quelques espaces :

```
Soient les suites $(u_n)$ et $(v_n)$ :
\begin{displaymath}
  u_n=\ln n\quad
  \mbox{et}\quad v_n=(1+\frac{1}{n})^n
  \label{ex-maths-suite}
\end{displaymath}
```

3.13

Soient les suites (u_n) et (v_n) :

$$u_n = \ln n \quad \text{et} \quad v_n = (1+\frac{1}{n})^n$$

Vous trouverez des détails sur la commande `\mbox` à la section 4.4.1 page 57. Si vous avez pensé à mettre en route le package amsmath vous serez en mesure d'utiliser la commande `\text` en lieu et place de `\mbox`.

3.6 Array : simple et efficace

L'environnement `array` est un environnement qui vous permettra de produire la grande majorité de vos formules. Comme son nom l'indique il range des objets en ligne et colonne. En fait c'est le pendant de l'environnement `tabular` du mode texte. Et comme `tabular`, `array` ne passe pas à la ligne.

3.6.1 Comment ça marche

La syntaxe rappelle celle de `tabular` :

`\begin{array}[vpos]{format} ... \end{array}`

où format précise pour chaque colonne l'alignement : `c` pour centré, `l` pour aligné à gauche et `r` pour aligné à droite ; l'argument optionnel vpos spécifie quant à lui le positionnement vertical du tableau. Comme dans les tableaux, on notera l'utilisation des commandes :

- `&` comme séparateur de colonne ;
- `\\` pour passer à la ligne.

```
Soit $A=\begin{array}{rc}
  -1 & 1 \\
   3 & 4
\end{array}$ la matrice ...
```

3.14

Soit $A = \begin{array}{rc} -1 & 1 \\ 3 & 4 \end{array}$ la matrice ...

Voici un exemple utilisant les points de suspensions :

```
\begin{displaymath} A=\left[\begin{array}{ccc}
    a_{00} & \dots & a_{0n}\\
    \vdots & \ddots & \vdots\\
    a_{n0} & \dots & a_{nn}
  \end{array}\right]\end{displaymath}
```

3.15

$$A = \left[\begin{array}{ccc} a_{00} & \dots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{n0} & \dots & a_{nn} \end{array}\right]$$

3.6.2 Array et les délimiteurs

On utilise couramment l'environnement `array` pour produire des matrices. Il faut alors avoir recours à des *délimiteurs*. Ces délimiteurs sont de la famille des parenthèses et permettent d'englober un objet mathématique entre crochets, accolades, etc. La syntaxe est la suivante :

`\left`delim$_1$ mobjet `\right`delim$_2$

où delim$_1$ et delim$_2$ sont deux délimiteurs et mobjet un objet mathématique.

Parmi les délimiteurs, voici les plus usités :

`(` et `)`	(Π)	`[` et `]`	$[\Pi]$
`\{` et `\}`	$\{\Pi\}$	`\lfloor` et `\rfloor`	$\lfloor\Pi\rfloor$
`\lceil` et `\rceil`	$\lceil\Pi\rceil$	`\langle` et `\rangle`	$\langle\Pi\rangle$
`|`	$\|\Pi\|$	`\\|`	$\|\|\Pi\|\|$

L'intérêt des délimiteurs est qu'ils s'adaptent automatiquement à la taille des objets qu'ils entourent :

```
soit  $I=
\left[\begin{array}{cc}
       1&0 \\ 0&1
      \end{array}\right]$ la matrice identité.
```

3.16

soit $I = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right]$ la matrice identité.

On peut également reprendre l'exemple 3.13 page ci-contre avec des délimiteurs pour ajuster la taille des parenthèses :

```
Soient les suites $(u_n)$ et $(v_n)$ :
\begin{displaymath}
  u_n=\ln n\quad\mbox{et}
  \quad v_n=\left(1+\frac{1}{n}\right)^n
\end{displaymath}
```

3.17

Soient les suites (u_n) et (v_n) :

$$u_n = \ln n \quad \text{et} \quad v_n = \left(1+\frac{1}{n}\right)^n$$

Il doit toujours y avoir une commande `\right` pour une commande `\left`. Cependant, il n'est pas nécessaire d'avoir les mêmes symboles à droite et à gauche.

Voici un exemple où on utilise la commande `\right.` pour spécifier que l'on n'utilise pas de symbole à droite :

```
soit $ S_i=\left\{\begin{array}{rl}
    -1  & \mbox{si $i$ est pair}     \\
     1  & \mbox{sinon.}\end{array}\right.$
```

3.18

soit $S_i = \left\{\begin{array}{rl} -1 & \text{si } i \text{ est pair} \\ 1 & \text{sinon.} \end{array}\right.$

3.6.3 Pour vous simplifier la vie...

Le package amsmath permet de saisir plus simplement les matrices avec notamment deux environnements : `pmatrix` (p pour parenthèse) et `bmatrix` (b pour *bracket*).

```
\begin{displaymath}
  \bar{\bar{\sigma}}=\begin{bmatrix}
    \sigma_{11} & \sigma_{12} \\
    \sigma_{21} & \sigma_{22} \\
  \end{bmatrix}
\end{displaymath}
```

3.19

$$\bar{\bar{\sigma}} = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

3.7 Équations et environnements

Nous présenterons dans ce paragraphe trois environnements standard de LaTeX permettant de produire des formules.

3.7.1 L'environnement `displaymath`

Vous l'avez compris, si vous avez lu jusqu'ici, `displaymath` affiche une formule centrée, interrompant le paragraphe. Un raccourci agréable de :

```
\begin{displaymath}...\end{displaymath}
```

est : `\[...\]`. Ainsi :

```
Distance colorimétrique :\[
\Delta E=\sqrt{
  \Delta L^{*2}+ \Delta a^{*2}+\Delta b^{*2}}
\]
```

3.20

Distance colorimétrique :

$$\Delta E = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}}$$

3.7.2 L'environnement `equation`

L'environnement `equation` est l'équivalent du précédent, sauf qu'il numérote la formule.

```
À retenir : si $a>0$ et $b>0$,\begin{equation}
  \ln(ab)=\ln(a)+\ln(b)
\end{equation}
```

3.21

À retenir : si $a > 0$ et $b > 0$,

$$\ln(ab) = \ln(a) + \ln(b) \tag{3.1}$$

L'option de classe de document `leqno` met le numéro des équations à gauche. Et l'option `fleqn` aligne les équations à gauche, au lieu de les centrer.

3.7.3 Formules multi-lignes

Dans une précédente édition, nous finissions la présentation des environnements standard par l'environnement eqnarray qui permet de produire des formules de plusieurs lignes. **Sachez que c'est mal**. Il existe d'ailleurs des écrits à ce sujet (lire par exemple [5] ou [6]), vous expliquant comment produire des documents « propres ». Prenez bien conscience qu'utiliser eqnarray (et bien d'autres choses) **est un péché**, et que si vous cédez malgré tout à la tentation, l'inquisition vous retrouvera un jour ou l'autre par l'intermédiaire d'un moteur de recherche. Aucune confession ou indulgence ne pourra vous sortir de ce mauvais pas, vous être prévenu.

Nous vous présentons donc ici l'environnement `align` du package amsmath :

– \\ passe à la ligne ;

– chaque ligne est numérotée sauf si la commande `\nonumber` est présente dans la ligne ;
– on procède à l'alignement avec deux[6] opérateurs `&`.

```
\begin{align}
  (a+b)^2 & =  (a+b)(a+b)\nonumber\\
          & =  a^2+b^2+2ab
\end{align}
```

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2+b^2+2ab\end{aligned} \tag{3.2}$$

Il existe une forme « étoilée » de l'environnement : `align*` où aucune des lignes n'est numérotée. Si vous voulez faire référence à certaines lignes d'un `align`, il vous faudra poser autant de `\label` nécessaires sur chaque ligne correspondante.

Pour faire numéroter une équation s'étalant sur plusieurs lignes on peut utiliser l'environnement `split` (lui aussi fourni avec amsmath) :

```
\begin{equation}
  \begin{split}
    (a+b)^2 & =  (a+b)(a+b)\\
    & =  a^2+b^2+2ab
  \end{split}
\end{equation}
```

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2+b^2+2ab\end{aligned} \tag{3.3}$$

3.8 Changer le style en mode mathématique

3.8.1 Fontes

LaTeX fournit plusieurs commandes permettant de changer de fontes dans les modes mathématiques. Par défaut tout symbole ou suite de caractères (autre qu'une fonction) est produit en italique dans le document final. Or dans certains cas, il est utile de pouvoir forcer le style de fonte. Voici comment réaliser un tel exploit :

`Soit $\mathit{A\in\Phi}$`	Soit $\mathit{A\in\Phi}$
`Soit $\mathrm{A\in\Phi}$`	Soit $\mathrm{A\in\Phi}$
`Soit $\mathbf{A\in\Phi}$`	Soit $\mathbf{A\in\Phi}$
`Soit $\mathsf{A\in\Phi}$`	Soit $\mathsf{A\in\Phi}$
`Soit $\mathtt{A\in\Phi}$`	Soit $\mathtt{A\in\Phi}$
`Soit $\mathcal{A}\in\Phi$`	Soit $\mathcal{A}\in\Phi$

La commande `\mathcal` doit prendre exclusivement des lettres majuscules latines comme argument. Dans le cas contraire, les résultats seront farfelus. Par exemple, la séquence :

```
\mathcal{abcd\Gamma}
```

donne $\mathcal{abcd\Gamma}$.

3.8.2 Taille des symboles

LaTeX distingue quatre *styles* d'écriture des formules. Ces modes sont utilisés suivant la « situation » dans laquelle se trouve LaTeX lorsqu'il produit une partie d'une formule :

[6] Puisqu'il y a trois colonnes.

texte pour une formule insérée dans le texte courant ;
equation pour une formule sous forme d'*équation* ;
indice pour l'écriture des indices ;
sous-indice pour les indices d'indices
chacun de ces modes peut être enclenché explicitement par l'utilisateur grâce aux déclarations suivantes :

- `\textstyle` pour le mode texte ;
- `\displaystyle` pour le mode équation ;
- `\scriptstyle` pour le mode indice ;
- `\scriptscriptstyle` pour le mode indice d'indice

Voici deux exemples illustrant comment forcer le mode *texte* en mode *équation* et inversement :

```
deux produits : $\prod_{1}^{n}f_i$
et $\displaystyle\prod_{1}^{n}f_i$

et inversement :
\[ \prod_{1}^{n}f_i
\mbox{ et }\textstyle\prod_{1}^{n}f_i \]
```

3.24

deux produits : $\prod_1^n f_i$ et $\displaystyle\prod_1^n f_i$

et inversement :

$$\prod_1^n f_i \text{ et } \textstyle\prod_1^n f_i$$

3.8.3 Créer de nouveaux opérateurs

Imaginez que vous ayez besoin de créer un opérateur spécial nommé « burps ». Il suffira de procédé comme suit :

```
\newcommand{\burps}{%
  \mathop{\textrm{burps}}}
$x=\burps_i f(i)$
```

3.25

$x = \operatorname{burps}_i f(i)$

Un autre exemple, pour franciser la fonction « arcsinus » (produisant par défaut arcsin), on pourra écrire :

```
$\theta = \arcsin x$
\renewcommand{\arcsin}{%
  \mathop{\textrm{Arcsin}}\nolimits}
$\theta = \arcsin x$
```

3.26

$\theta = \arcsin x$ $\theta = \operatorname{Arcsin} x$

La commande `\nolimits` indique que l'opérateur concerné ne fera pas usage d'arguments en indice ou exposant comme le font les opérateurs `\lim`, `\int`, etc. En outre les deux exemples précédents utilisent les commandes `\newcommand` et `\renewcommand` dont il est question au paragraphe 4.5 page 62.

Enfin une autre voie possible si vous avez pris soin de charger le paquage amsmath est de déclarer dans le préambule :

```
\DeclareMathOperator*{\vlunch}{vlunch}
\DeclareMathOperator{\zirgl}{Zirgl}
```

```
\[x=\vlunch_i f(\theta)\]
où $\theta = \zirgl y$
```

3.27

$$x = \operatorname*{vlunch}_i f(\theta)$$

où $\theta = \operatorname{Zirgl} y$

Conclusion

Ce chapitre présente les fonctions de base pour produire des formules. Ces commandes suffisent pour la plupart des documents scientifiques. Si vous êtes amenés à rédiger des documents truffés de formules complexes, il est possible que les seules macros de LaTeX ne suffisent plus. C'est pourquoi la célèbre *American Mathematical Society* a conçu pour vous un package nommé $\mathcal{AMS}$TeX (mise en route : `\usepackage{amsmath}`) capable de générer des formules particulièrement « tordues. »

4 Un pas vers la sorcellerie

Sommaire

Et lorsque l'Agneau ouvrit le septième sceau
il se fit un silence dans le ciel,
environ une demi-heure...

L'Apocalypse Ap **8** 1.

AVANT DE CONTINUER l'exploration de ce système monstrueux et magnifique qu'est LaTeX, il est nécessaire de faire une pause et de prendre connaissance de quelques concepts importants. Il nous semble en effet fondamental d'assimiler ces notions pour pouvoir jouer les « Hercule Poirot » dans les nombreux fichiers qui forment le système. Nous présenterons dans ce chapitre les compteurs, les longueurs, les espaces et les boîtes. Ces quatre notions vous seront utiles pour utiliser LaTeX autrement qu'en acceptant docilement ce qu'il vous propose.

Ce chapitre traite de concepts assez subtils à saisir[1] ; nous vous conseillons donc vivement d'**expérimenter** car les outils présentés ici sont ceux qui offrent le plus de satisfaction mais qui entraînent aussi les plus grandes pertes de cheveux (essentiellement par arrachage).

4.1 Compteurs

Toute partie d'un document faisant l'objet d'une numérotation est gérée par un *compteur*. Ces compteurs peuvent être incrémentés ou décrémentés, remis à zéro, etc. On peut aussi en créer pour un usage personnel.

4.1.1 Compteurs disponibles

Les compteurs sont principalement liés aux titres, aux numéros de pages, aux environnements flottants (environnements `figure` et `table`), aux équations (environnement `equation`), aux notes de bas de page et aux items d'énumération (environnement `enumerate`).

Le tableau 4.1 page suivante vous donne le nom des principaux compteurs de LaTeX. Vous remarquerez qu'ils portent généralement le nom des objets auxquels ils

[1] L'auteur n'est lui-même pas sûr d'avoir tout compris...

TAB. 4.1 – Les compteurs de LaTeX

part	paragraph	figure	enumi
chapter	subparagraph	table	enumii
section	page	footnote	enumiii
subsection	equation	mpfootnote	enumiv
subsubsection			

sont associés. Les compteurs `enumi`, ..., `enumiv` sont associés aux items de niveaux 1 à 4 de l'environnement `enumerate`. Le compteur `mpfootnote` est le compteur de note de bas de page de l'environnement `minipage` dont il est question au paragraphe 4.4.3 .

4.1.2 Manipulation

Nous vous donnons, dans les paragraphes qui suivent, les outils de base pour manipuler les compteurs. Il est important de noter que les compteurs sont des variables *globales*. Ainsi les trois commandes décrites plus bas ont une portée globale. Il est également utile de noter que ces variables sont des *entiers*.

Création

On peut *créer* un nouveau compteur grâce à la commande :

```
\newcounter{cpteur}[cpt_maitre]
```

qui crée un nouveau compteur cpteur. Si l'argument optionnel cpt_maitre est présent, le compteur cpteur est remis à zéro à chaque fois que le compteur maître cpt_maitre est incrémenté.

Affectation

On affecte une valeur à un compteur de la manière suivante :

```
\setcounter{compteur}{valeur}
```

où compteur est le compteur que l'on veut modifier, et valeur la valeur que l'on veut lui affecter.

Incrémentation

On peut incrémenter ou décrémenter un compteur grâce à la commande :

```
\addtocounter{compteur}{valeur}
```

où valeur est un nombre positif (resp. négatif) pour réaliser une incrémentation (resp. décrémentation). Illustrons l'utilisation de cette commande en entrant la ligne suivante dans notre document :

```
\addtocounter{footnote}{357}
```

pour changer[359] la numérotation des notes de bas de page. Pour que tout rentre dans l'ordre, avec les notes de bas de page suivantes, nous avons préféré entrer dans notre source, la commande :

```
\addtocounter{footnote}{-357}
```

et normalement,[3] nous devrions avoir une numérotation correcte.

4.1.3 Affichage

Pour afficher un compteur on utilise la commande :

\thenom-du-compteur

En fait, toute commande ou environnement qui donne lieu à l'affichage d'un compteur fait appel à ce type de commande. Ainsi, on a par exemple :

- `\thepage` produit : « 49 » et est appelée notamment à chaque saut de page,
- `\thefootnote` produit : « 3 » et est appelée par `\footnote`,
- `\thesubsection` produit : « 4.1.3 » et est appelée par `\subsection`,
- ...

Les commandes de la famille `\the` sont généralement définies à partir des commandes de formatage suivantes :

- `\arabic{`compteur`}`,
- `\roman{`compteur`}` et `\Roman{`compteur`}`,
- `\alph{`compteur`}` et `\Alph{`compteur`}`

en voici quelques exemples :

- `\arabic{page}` produit : « 49 » ;
- `\alph{footnote}` produit : « c » et `\Alph{section}` produit : « A » ;
- `\Roman{subsection}` produit : « III » et `\roman{page}` produit : « xlix » ;
- ...

Il est courant de redéfinir les commandes de la famille `\the` pour personnaliser un document. Par exemple, dans la classe de document utilisée pour ce manuel, la commande `\thefigure` est définie comme suit :

```
\arabic{chapter}.\arabic{figure}
```

ce qui produit dans les légendes des figures une numérotation formée par : 1) le numéro du chapitre en chiffre arabe, 2) un point, et 3) le numéro de la figure en chiffre arabe. Il est possible de redéfinir cet affichage en définissant la commande `\thefigure` par exemple comme suit :

```
(\Roman{chapter}):\arabic{section}.\arabic{figure}
```

Ce qui permet d'obtenir un numéro de figure — relativement immonde — dans les légendes quelque peu différent du style prédéfini. Ici, on a donc redéfini la commande `\thefigure` pour produire une numérotation formée par le numéro du chapitre entre parenthèses et en chiffres romains, suivi du numéro de section et du numéro de la figure en chiffre arabe, séparés par un point. Le « Fig. » ainsi que le tiret qui

Fig. (IV):1.1 – La légende

[359] Même si ce changement est un peu ridicule...

[3] On croise les doigts !

suit le numéro de la figure sont quant à eux définis au niveau de la commande `\caption`...

4.2 Longueurs

Si les compteurs sont dédiés à la *numérotation* des objets d'un document, les longueurs définissent l'*encombrement* d'une entité. Il s'agit en quelque sorte, d'un type de donnée de LaTeX destiné à exprimer les dimensions d'un objet.

4.2.1 Unités

Toutes les dimensions doivent avoir une unité ; une dimension de type *rigide*[4] a la forme suivante :

nombreunité

où nombre est un nombre positif ou négatif avec éventuellement une partie décimale, et unité une unité de mesure reconnue par LaTeX. Voici une liste non exhaustive des unités *légales* :

cm pour *centimètre* ;

mm pour *millimètre* ;

in pour les allerg*inch* au système métrique (environ `2.54cm`) ;

pt pour *point* : couramment utilisé en typographie : $\frac{1}{72.27}$ inch ;

em : la largeur de la lettre 'M' de la fonte courante ;

ex : la hauteur de la lettre 'x' de la fonte courante

Notez que les unités `em` (resp. `ex`) sont généralement utilisées pour des dimensions horizontales (resp. verticales) et permettent de manipuler des dimensions dépendantes de la taille de la fonte courante. Voici quelques exemples de dimensions :

```
1cm    :  └────┘
1in    :  └──────────┘
3mm    :  └┘
2em    :  └──┘
10pt   :  └┘
```

4.2.2 Quelques longueurs de LaTeX

Il existe dans LaTeX et dans chaque extension des longueurs prédéfinies. Ces longueurs déterminent en général, les dimensions de certaines parties du document. Ainsi :

- `\parindent` est la dimension de l'indentation en début de paragraphe. Cette dimension est prédéfinie à `15pt` ;
- `\textwidth` et `\textheight` définissent la largeur (resp. la hauteur) du texte ;
- `\baselineskip` représente la distance entre la base de la ligne et la base de la ligne suivante (`10pt` dans ce document) ;

[4] On verra plus loin qu'il existe des dimensions *élastiques*.

– `\parskip` la distance séparant deux paragraphes ; cette distance est initialisée à `0pt plus 1pt`[5] ;
– ...

Il est important de comprendre qu'il est possible d'exprimer une dimension en fonction d'une de ces dimensions « internes ». Ainsi :

```
0.5\textwidth
```

représente la moitié de largeur de la page, et :

```
3\parindent
```

équivaut à trois fois l'indentation des paragraphes. Notez aussi que l'on peut écrire `-\baselineskip` pour : `-1\baselineskip`

4.2.3 Manipulation des longueurs

Comme pour les compteurs, il existe quelques commandes permettant de manipuler les dimensions.

Création

La commande suivante crée une nouvelle longueur :

```
\newlength{dim}
```

où dim est le nom de la nouvelle dimension initialisée à `0pt` (cf. exemple page suivante).

Attention, quel que soit l'endroit où intervient la commande `\newlength`, la longueur définie est toujours **globale**. De plus, déclarer deux fois la même longueur provoque une erreur. Par contre, la modification d'une longueur est locale au groupe (`{...}`) où elle survient.

Affectation

On peut affecter une valeur à une longueur avec la commande :

```
\setlength{dim}{val}
```

qui affecte la valeur val à la longueur dim.

Incrémentation

On incrémente une longueur comme suit :

```
\addtolength{dim}{val}
```

qui a pour effet d'augmenter la longueur dim de la valeur val.

Alors que vous lisez fébrilement ce paragraphe, nous nous sommes permis d'augmenter la longueur `\parindent` de 30 points avec :

```
\addtolength{\parindent}{30pt} Alors que vous
lisez fébrilement ce paragraphe...
```

pour illustrer l'utilisation de l'incrémentation des longueurs. Après ce paragraphe, on a écrit :

[5] Cf. les dimensions élastiques pour avoir la signification du `plus`.

```
\addtolength{\parindent}{-30pt}
```

pour que tout rentre dans l'ordre.

Obtenir les dimensions d'un objet

Comme il en a été vaguement question précédemment, au niveau de TEX, les différents objets qui composent le document sont assemblés dans des *boîtes*. Ces boîtes sont positionnées les unes par rapport aux autres en alignant leur *point de référence*. Ces points alignés forment une ligne imaginaire confondue avec la base de la ligne. Toute boîte est caractérisée par trois dimensions :

- sa *largeur* ;
- sa *hauteur* : du point de référence au haut de la boîte ;
- sa *profondeur* : du point de référence jusqu'au bas de la boîte.

Voici par exemple comment sont assemblées les boîtes du mot « Ingénierie » :

les symboles « ◦ » représentent les points de référence. On voit ici que toutes les boîtes ont une profondeur nulle sauf celle de la lettre 'g'.

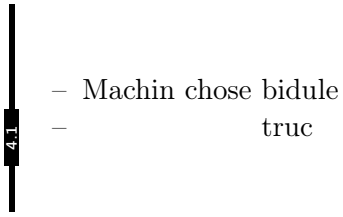
Mais fermons la parenthèse concernant les boîtes !

Il est donc possible d'extraire les caractéristiques d'un objet (une lettre, un mot, une boîte, etc.) à l'aide des commandes suivantes :

```
\settowidth{dim}{obj}
\settoheight{dim}{obj}
\settodepth{dim}{obj}
```

trois commandes qui affectent à la dimension dim respectivement la largeur, la hauteur et la profondeur de l'objet obj. Par exemple :

```
\newlength{\malongueur}
\settowidth{\malongueur}{Machin chose}
\begin{itemize}
\item Machin chose bidule
\item \hspace{\malongueur} truc
\end{itemize}
```



La longueur \malongueur contient alors la largeur de « Machin chose » et est utilisé pour insérer un blanc (voir le paragraphe sur les espaces).

4.2.4 Longueurs élastiques

Les dimensions présentées jusqu'ici sont des dimensions *rigides*,[6] il existe cependant des longueurs *élastiques* ou *ressort*. Au niveau de TEX, un grand nombre de dimensions sont définies comme suit :

val plus p_val minus m_val

[6] Sauf \parskip.

cette syntaxe permet de définir une longueur ayant la dimension val, mais pouvant selon les circonstances s'agrandir ou se rétracter. Et ainsi, si on appelle dim la dimension créée, on a :

$$val - m\_val \leqslant dim \leqslant val + p\_val$$

Par exemple, la longueur `\parskip` qui sépare deux paragraphes consécutifs, est fixée à :

```
0pt plus 1pt
```

ce qui signifie qu'au cas où la page est un peu lâche, LaTeX insérera entre les paragraphes un blanc vertical de 1 point. On ne peut pas directement[7] créer de telles longueur avec LaTeX. Cependant ces dimensions ont surtout un intérêt lors de la création de macros complexes ou d'extensions. Par contre, les utilisateurs de LaTeX auront la chance inouïe de pouvoir manipuler une autre famille de longueurs élastiques tout aussi intéressante. Cette famille possède les deux particularités suivantes :

1. une longueur **nulle** ;
2. la capacité de s'étirer **indéfiniment** avec une certaine force.

LaTeX 2ε dispose d'une commande permettant de spécifier une longueur élastique en précisant son degré d'élasticité :

```
\stretch{nbre}
```

où nbre est la force du ressort. Ce nombre peut être signé et avoir une partie décimale. Voici un exemple :

```
zéro\hspace{\stretch{1}}%
tiers\hspace{\stretch{2}}un
```

4.2 zéro tiers un

Ce code LaTeX introduit des espaces[8] de longueurs élastiques entre les mots « zéro tiers un ». Le deuxième ressort a une raideur deux fois plus importante que le premier. L'espacement est donc double. Vous noterez aussi que ces ressorts ont une élasticité *relative* mais *infinie* ; c'est pourquoi les mots « zéro » et « un » sont « poussés » contre les marges. Enfin, sachez que `\fill` est un raccourci agréable de `\stretch{1}`.

4.2.5 Affichage

Il est parfois utile d'afficher la valeur d'une longueur. Pour ce faire on peut avoir recours à la commande `\showthe` qui interrompt la compilation pour afficher la valeur de la longueur passée en paramètre. Ainsi :

```
\showthe\linewidth
```

afficher la valeur de la longueur `\linewidth` en interrompant la compilation. On aura sur le terminal quelque chose du genre :

```
[ ... ]                                  ← Le laïus initial
Document Class: book 2001/04/21 v1.4 Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/bk12.clo)) (./test.aux)
```

[7] Pas avec la commande `\setlength`.

[8] C'est la commande `\hspace` qui produit une espace horizontale de longueur définie par son argument.

```
> 17.62482pt.                    ← la valeur de la longueur
l.10 \showthe\parindent           ← la longueur à afficher

?
```

Lorsque la compilation est lancée dans un terminal de commande, une pression sur la touche `<Entrée>` fait reprendre la compilation.

Comme indiqué à la page 12, votre environnement de développement ne vous permet peut être pas directement d'avoir accès aux messages de LaTeX. À vous de chercher où se trouvent ces informations...

4.3 Espaces

On appelle *espaces* les blancs que l'on insère à divers endroits dans un document. Il existe des commandes permettant d'insérer des blancs de longueur prédéfinie ou choisie par l'utilisateur. Il s'agit bien sûr de longueur au sens de LaTeX.

4.3.1 Commandes de base

Pour insérer une espace[9] entre les objets, on dispose de commandes de la forme suivante :

`\dirspace{dim}`

où dim est une longueur rigide ou élastique, et dir vaut :
- `v` pour une espace verticale ;
- `h` pour une espace horizontale.

Ainsi :

```
un saut\hspace{1cm}de \texttt{1cm}

\vspace{2\baselineskip}

et deux lignes vierges.
```

un saut de `1cm`

et deux lignes vierges.

Dans certaines situations, TeX supprime les espaces. Il est alors nécessaire d'utiliser la forme « étoilée » des commandes d'espacement, à savoir `\hspace*` et `\vspace*`. Les situations en question sont :
- le début et la fin de page ;
- le début et la fin d'une ligne s'il ne s'agit pas de la première ou de la dernière ligne du paragraphe.

4.3.2 Quelques espaces prédéfinies

On dispose de plusieurs commandes d'espacement, regroupées en deux catégories selon le mode horizontal ou vertical.

[9] Nous utilisons ici le genre féminin du mot *espace* qui désigne alors les petites tiges métalliques utilisées autrefois en imprimerie pour séparer les mots et les lettres. Aujourd'hui, le genre féminin est utilisé dans le monde de la typographie et de la photo-composition.

Espaces horizontales

Voici quelques espaces rigides :

`\enspace`	: ␣	soit `0.5\quad`
`\quad`	: ␣␣	soit `1em`
`\qquad`	: ␣␣␣␣	soit `2\quad`

et quelques espaces élastiques :

`\hfill`	: soit `\hspace{\fill}`
`\hrulefill`	: comme `\hfill` mais trace une ligne
`\dotfill`	: comme `\hfill` mais trace des points

Voici quelques exemples montrant l'utilisation des espaces horizontales. Tout d'abord, notez que les espaces entourant la commande `\hspace` ne sont pas ignorées :

```
zéro\hspace{1cm}un\par
zéro \hspace{1cm}un\par
zéro \hspace{1cm} un\par
```

4.4

zéro un
zéro un
zéro un

Voici ensuite, les espaces élastiques de LaTeX :

```
zéro \hfill{} un\par
zéro \hrulefill{} un\par
zéro \dotfill{} un\par
```

4.5

zéro un
zéro _____ un
zéro ..... un

Et pour finir, la force relative des ressorts :

```
zéro \dotfill{} demi \hfill{} un\par
zéro \hrulefill{} tiers
\hspace{\stretch{2}} un\par
```

4.6

zéro ..... demi un
zéro _____ tiers un

Vous aurez donc compris que les « ressorts » prédéfinis de LaTeX (à savoir `\hfill`, `\hrulefill`, et `\dotfill`) ont une raideur de 1.

Espaces verticales

Voici trois grands classiques de la famille des espaces verticales :
- `\smallskip` pour un *petit* saut vertical ;
- `\medskip` pour un saut vertical *moyen* ;
- `\bigskip` pour un *grand* saut vertical.

Ces espaces s'utilisent comme la commande `\vspace`, avec pour effet :

défaut :	petite :	moyenne :	grande :
§ suivant...	§ suivant...	§ suivant...	§ suivant...

Il existe une espace verticale élastique prédéfinie : `\vfill` équivalent à :

```
\par\vspace{\fill}
```

c'est-à-dire, un saut de paragraphe, suivi d'une espace verticale de dimension `\fill`.

```
\hrulefill{}

haut

\vfill

\hfill{}fragile\hfill{}

\vspace{\stretch{2}}

\hfill{}bas

\hrulefill{}
```

haut

fragile

bas

Il est important d'utiliser la commande `\vspace` **entre deux paragraphes** au risque d'avoir des résultats surprenants. Il vaut donc mieux prendre l'habitude d'insérer un saut de paragraphe — une ligne vierge ou une commande `\par` — avant et/ou après `\vspace`.

4.4 Boîtes

La dernière section de ce chapitre sera dédiée aux *boîtes*, et vous verrez que le titre du présent chapitre sera amplement justifié ! Comme nous l'avons aperçu précédemment, les boîtes sont des entités qui contiennent d'autres éléments (une boîte pouvant en contenir une autre). Ces entités peuvent d'autre part être positionnées selon la fantaisie[10] de l'utilisateur.

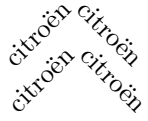
Il existe deux principaux types de boîtes (au niveau de TEX c'est un peu plus subtil) chacun d'eux ayant un comportement spécifique. Nous qualifierons ces deux catégories comme suit :

- boîte *simple*
- boîte *paragraphe*

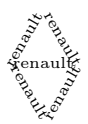
Nous verrons qu'une manipulation habile de ces boîtes permet de produire des mises en page sophistiquées particulièrement utiles notamment lors de la conception de transparents.

Voici un premier exemple avec des boîtes simples qui, j'en suis sûr, vous a sauté aux yeux, c'est le logo de `TeX` : TEX. Il s'agit en fait des trois lettres, T, E et X « mises en boîte » et assemblées avec des décalages horizontaux et verticaux :

notez que la boîte du 'E' est décalée vers le bas et que les trois boîtes se superposent. Un autre exemple :



et pour éviter les querelles culturelles :



ici chaque mot est dans une boîte. Chaque boîte est ensuite placée par rapport aux autres avec moult décalages et rotations. Pour en finir avec les exemples préliminaires, nous vous donnons ici deux exemples utilisant les boîtes *paragraphes* :

[10] Humm... ainsi qu'avec patience et bonne humeur...

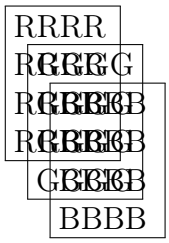
Ceci est une boîte paragraphe de 4cm de large. Une telle boîte est capable de contenir à peu près tout ce qu'on trouve dans un document LaTeX.

le texte continue, et,

Plaf! une formule mathématique dans une «boîboîte»

$$ax + b = c$$

Voici une image en couleur, avec ses 3 composantes :



4.4.1 Boîtes simples

La première catégorie — les boîtes simples — se comporte comme des mots dans un paragraphe. Voici leurs caractéristiques :

- on peut imposer sa *largeur*,
- sa *hauteur* est donnée par ce qu'elle contient,
- Elle **ne doit pas contenir** de saut de paragraphe

Sans bordure La commande `\makebox` permet de construire une boîte simple.

`\makebox[larg][pos]{contenu}`

où larg est la largeur désirée, pos la position (`c`=centré, `l`=aligné à gauche ou `r`=à droite) de contenu dans la boîte. Voici quelques exemples :

```
et \makebox[2cm][c]{hop !} une boîte\par
et \makebox[3cm][r]{rehop !} une autre
```

4.8

et hop ! une boîte
et rehop ! une autre

Les deux arguments larg et pos sont optionnels et s'ils sont omis, la largeur de la boîte est celle du texte. Le cas échéant on saisit :

`\mbox{texte}`

au lieu de `\makebox[][]{texte}`. On notera également que l'option `s` de la commande `\makebox` permet d'étirer le contenu pour qu'il fasse exactement la dimension imposée :

```
\makebox[5cm][s]{Ouaaahhh quelle fatigue !}
```

4.9

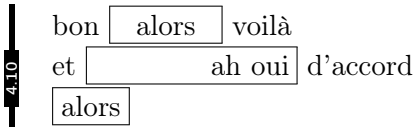
Ouaaahhh quelle fatigue !

Avec bordure On construit une boîte entourée par une bordure grâce à la commande `\framebox` qui suit la même syntaxe que `\makebox` :

`\framebox[larg][pos]{texte}`

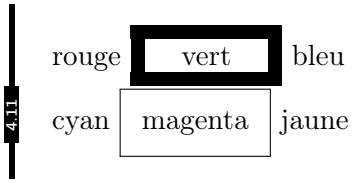
le raccourci `\fbox{texte}` existe comme pour les boîtes sans bordure. Ce qui donne :

```
bon \framebox[1.5cm][c]{alors} voilà\par
et \framebox[2.8cm][r]{ah oui} d'accord\par
 \fbox{alors}
```



Deux longueurs sont disponibles pour changer l'allure des `\framebox` :
- `\fboxsep` la distance entre la bordure et le texte,
- `\fboxrule` l'épaisseur du trait.

```
\setlength{\fboxrule}{5pt}
rouge \framebox[2cm]{vert} bleu\par
\setlength{\fboxrule}{0.4pt}
\setlength{\fboxsep}{8pt}
cyan \framebox[2cm]{magenta} jaune
```



Un particularité des boîtes simples contrairement aux boîtes paragraphes que l'on rencontrera un peu plus loin dans ce chapitre, est qu'elles n'effectuent pas de césure, ainsi :

```
=== \framebox[3cm]{Ça ne risque pas d'être coupé, ça...} ===
```

donnera :

=== Ça ne risque pas d'être coupé, ça... ===

On peut d'ailleurs exploiter cette fonctionnalité pour superposer du texte (cf. paragraphe sur les boîtes de largeur nulle page ci-contre).

4.4.2 Manipulation de boîtes simples

On peut avec un peu d'habitude faire subir aux boîtes des déplacements dans toutes les directions.

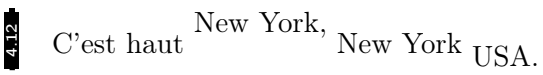
Translation verticale

La translation est permise grâce à la commande :

`\raisebox{trans}[prof][haut]{texte}`

où trans est le déplacement que vous voulez infliger à texte. Par exemple :

```
C'est haut \raisebox{8pt}{New York,}
New York \raisebox{-1ex}{USA.}
```



Les deux arguments optionnels prof et haut permettent de « faire croire » à LaTeX que la boîte résultant de la translation a une hauteur de haut et une profondeur de prof. L'exemple suivant illustre l'utilisation de la commande `\raisebox` avec ses arguments optionnels.

```
\begin{flushleft}
  ligne 1 : XXXXX\\
  ligne 2 :
  XX\raisebox{0.8\baselineskip}{O}XX\\
  ligne 3 : XXXXX\\
  ligne 4 : XXXXX\\
  ligne 5 :
  XX%
  \raisebox{0.8\baselineskip}[1ex][2ex]{O}XX\\
  ligne 6 : XXXXX\\
\end{flushleft}
```

ligne 1 : XXXXX
ligne 2 : XXOXX
ligne 3 : XXXXX
ligne 4 : XXXXX
ligne 5 : XXOXX
ligne 6 : XXXXX

On « soulève » un 'O' au milieu de la ligne 2. La bordure met en évidence la place occupée par la boîte soulevée.[11] Au milieu de la ligne 5, on soulève le même 'O' mais cette fois en imposant les dimensions (montrées par la bordure). LaTeX considère donc que la boîte résultant de la translation fait `1ex` de haut et `2ex` de profondeur, il effectue les sauts de lignes en conséquence.

Translation horizontale

Les translations horizontales ne sont pas à proprement parler des caractéristiques des boîtes, puisqu'on les obtient en insérant des espaces appropriées. Voici un exemple :

```
Non à la \makebox[1.5cm]{censure}%
\hspace{-1.5cm}\makebox[1.5cm]{///////}
sur Internet.
```

4.14

Non à la ///////// sur Internet.

Notez que ce n'est pas forcément la meilleure façon de « hachurer » un mot, mais cela illustre la manière de déplacer un boîte horizontalement, à l'aide d'un `\hspace` négatif.

Boîte simple de largeur nulle

Il est parfois utile de manipuler les boîtes de largeur nulle, par exemple dans le cas où l'on souhaite superposer des éléments. En imposant une dimension nulle en guise de premier argument optionnel de la commande `\makebox` :

```
\newcommand{\grogra}{\huge\bfseries}
avant\makebox[0cm][c]{\grogra C}après

avant\makebox[0cm][l]{\grogra G}après

avant\makebox[0cm][r]{\grogra D}après
```

4.15

avantCaprès
avantGaprès
avantDaprès

on produit bien une superposition mais l'alignement n'est pas exactement celui auquel on s'attendait ; en effet l'argument `l` met le contenu *à droite* du point d'insertion de la boîte, et *à gauche* pour l'argument `r`.

[11] Cette bordure est insérée ici pour la compréhension du mécanisme.

Rotation

Il existe plusieurs extensions de LaTeX pour faire subir des rotations à des éléments de texte ; nous avons choisi de vous présenter la commande `\rotatebox` de l'extension graphicx présentée au chapitre 5. La syntaxe en est la suivante :

```
\rotatebox{angle}{texte}
```

où angle est l'angle dans le sens trigonométrique, et texte l'élément de texte à faire tourner :

```
Attention \rotatebox{30}{virage} dangereux.
```

La version actuelle de xdvi n'est pas en mesure d'afficher les objets qui ont subi une rotation.[12] Cette lacune (avec quelques petites autres) sera peut-être corrigée dans les prochaines versions. La parade est de visualiser la sortie PostScript avec ghostview ou gv, ou la sortie Pdf.

4.4.3 Boîtes paragraphe

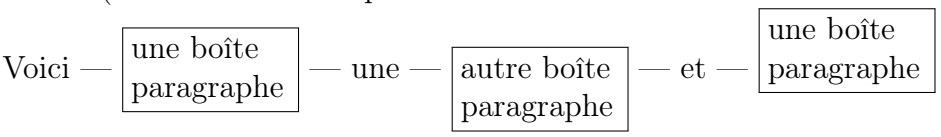
Les boîtes dites *boîtes paragraphe* ont la particularité de pouvoir contenir des sauts de ligne et des sauts de paragraphe (contrairement aux boîtes dites *simples*). Il existe deux manières de créer des boîtes paragraphe ; la première avec la commande `\parbox` :

```
\parbox[bpos][hauteur][tpos]{largeur}{contenu}
```

où contenu est l'élément de texte à mettre en boîte, largeur la largeur de la boîte à créer, bpos un argument optionnel précisant le point de référence. Cet argument optionnel est à rapprocher de celui de l'environnement `tabular`. Par exemple :

```
Voici  --- \parbox{2.1cm}{une boîte\\paragraphe}
--- une --- \parbox[t]{2.1cm}{autre boîte\\paragraphe}
--- et --- \parbox[b]{2.1cm}{une boîte\\paragraphe}
```

qui donne (avec des bordures pour mettre en évidence les dimensions des boîtes) :

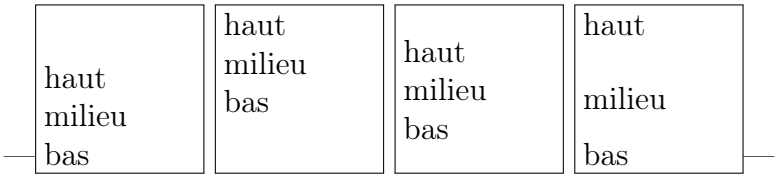


Pour construire une boîte paragraphe en imposant sa hauteur, on utilise l'argument optionnel hauteur. On peut alors éventuellement préciser la position verticale tpos du texte dans la boîte. Par défaut tpos vaut bpos, et il peut prendre les valeurs habituelles `c` pour centré, `t` et `b` pour haut et bas ; plus une valeur : `s` pour spécifier que le texte peut s'étirer (stretch) sur toute la hauteur de la boîte — dans ce cas c'est à l'utilisateur de positionner le texte. Par exemple :

```
---\parbox[b][2cm]{2cm}{haut\par milieu\par bas}}
   \parbox[b][2cm][t]{2cm}{haut\parmilieu\par bas}}
   \parbox[b][2cm][c]{2cm}{haut\par milieu\par bas}}
   \parbox[b][2cm][s]{2cm}{haut\par
     \vspace{\stretch{2}} milieu\par\vfill bas}}---
```

[12] L'objet est affiché mais sans la rotation.

donne avec les `\fbox` pour y voir un peu plus clair :



Pour créer une boîte paragraphe, il peut être utile d'utiliser l'environnement `minipage`, qui simule la création d'une page avec d'éventuelles notes de bas de page, tableaux, listes, etc.[13] La syntaxe est analogue à `\parbox` sauf qu'il s'agit d'un environnement :

```
\begin{minipage}[bpos][hauteur][tpos]{largeur}
  ... \marg{texte} ...
\end{minipage}
```

Voici un exemple :

Pour une raison x, y ou z, on peut vouloir raconter sa vie avec une `minipage`. Comme dans cet exemple là $\longrightarrow$

Ce que j'ai à dire[a] n'est pas à franchement parler :
– ni très intéressant ;
– ni particulièrement indispensable
mais bon, j'en parle quand même.
J'avais autre chose à raconter mais ça m'est sorti de la tête...

[a] C'est un bien grand mot.

Dans cet exemple on a créé une `minipage` faisant la moitié (55%) de la largeur du texte, et contenant un environnement `itemize` et une `\footnote`. La boîte ainsi créée est centrée par rapport au paragraphe « Pour une raison... » car l'argument optionnel pos est absent :

```
\parbox{0.40\textwidth}{...
    Comme dans cet exemple là $\longrightarrow$}\hfill
\begin{minipage}{0.55\textwidth}
  Ce que j'ai à dire\footnote{C'est un bien grand mot.} n'est pas à
  franchement parler :
  \begin{itemize}
  \item ni très intéressant ;
  \item ni particulièrement indispensable
  \end{itemize}
  mais bon, j'en parle quand même.

  J'avais autre chose à raconter mais ça m'est sorti de la tête...
\end{minipage}
```

Dans les boîtes paragraphe la longueur `\parindent` est mise à zéro. Ce qui explique que « J'avais autre chose ... » dans l'exemple précédent n'est pas indenté. Enfin, contrairement aux cas des `\parboxs`, lorsqu'on fait référence dans une `minipage` à la dimension `\textwidth`, il s'agit de celle de la boîte et non de celle du texte.

[13] Cet environnement ne peut toutefois pas contenir de flottants.

4.4.4 Petites astuces

Toutes les fonctions concernant les boîtes peuvent prendre en paramètre de longueur les dimensions suivantes :

- `\width` : la largeur du texte contenu,
- `\height` : la hauteur du texte contenu,
- `\depth` : la profondeur du texte contenu,
- `\totalheight` (hauteur + profondeur) du texte.

Il est alors possible de préciser les dimensions de la boîte relativement au texte qu'elle contient. Ce qui peut être utile dans certaines situations :

```
une \framebox[0.7\width]{boîte} à l'étroit.

une \framebox[1.8\width]{boîte} au large.

une \fbox{%
  \parbox[c][3\height]{1cm}{boîte\\vide.}}
```

4.17

une boîte à l'étroit.
une boîte au large.
une boîte vide.

4.4.5 Sauvegarde et réutilisation

Il est possible de stocker un extrait de code LaTeX dans une boîte pour le réutiliser — ceci par exemple lorsque ce code exige de LaTeX des ressources importantes ; dans ce cas on procède en 3 étapes :

1. déclaration d'une boîte avec la commande `\newsavebox`,
2. stockage avec `\sbox` ou `\savebox`,
3. réutilisation avec `\usebox`.

Par exemple voici une texture de Gnu :

```
\newsavebox{\gnu}
\sbox{\gnu}{\fbox{\textsc{Gnu}}}
\begin{center}
  \usebox{\gnu}\usebox{\gnu}\usebox{\gnu}\\
  \usebox{\gnu}\usebox{\gnu}\usebox{\gnu}\\
  \usebox{\gnu}\usebox{\gnu}\usebox{\gnu}
\end{center}
```

4.18

Gnu	Gnu	Gnu
Gnu	Gnu	Gnu
Gnu	Gnu	Gnu

On peut faire une analogie entre le couple de commande `\sbox` et `\savebox` et le couple `\mbox` et `\makebox` (cf. § 4.4.1).

4.5 Définitions

Une nouvelle fois, laissons parler le maître :

> *« . . . they have come to be known as macros because they are so powerful; one little macro can represent an enormous amount of material, so it has a sort of macroscopic effect. »*
>
> D. E. Knuth in the TeXbook

Lorsque dans un document, on peut définir une « entité » indépendante et que cette entité apparaît plus d'un « certain nombre de fois » il est nécessaire de se poser la question de savoir s'il n'est pas judicieux d'en faire une *macro*. Voilà une phrase vague ! Pour résumer, les macros sont là pour vous éviter de refaire x fois les mêmes choses. Avec un peu d'expérience, on peut définir des commandes très pratiques et avec le temps de plus en plus complexes.

4.5.1 Commandes

La commande `\newcommand` permet de définir une macro, son utilisation est très simple :

`\newcommand{`nomcom`}[`nargs`]{`code LaTeX`}`

où nargs est le nombre d'*arguments* — au sens arguments d'une fonction d'un langage de programmation — et code LaTeX le code définissant votre commande. Voici un exemple de macro, définissant le symbole d'un espace de représentation utilisé en colorimétrie :

```
\newcommand{\lab}{$L^*a^*b^*$}
Soit \lab{} l'espace...
```

4.19

Soit $L^*a^*b^*$ l'espace...

Notez que cette commande ne prend pas d'argument, il n'est donc pas nécessaire ici d'utiliser l'argument optionnel nargs. Pour améliorer un peu l'utilisation de cette commande, on peut la définir comme suit :

```
\newcommand{\Lab}{%
  \ensuremath{L^*a^*b^*}}
  L'espace \Lab{} et $\vec{c}\in\Lab$.
```

4.20

L'espace $L^*a^*b^*$ et $\vec{c} \in L^*a^*b^*$.

La commande `\ensuremath` permet de s'assurer que la commande sera utilisée dans un environnement mathématique, quel que soit le contexte, comme dans l'exemple ci-dessous.

Les macros ou commandes de LaTeX ne sont pas tout à fait des fonctions au sens d'un langage de programmation, elles s'apparentent plutôt au `#define` du C. Et en ce sens, elles suivent le mécanisme d'*expansion*. Ainsi, dans le premier exemple de la fonction `\Lab`, `\Lab` se « déploie » en `$L^* a^* b^*$`. On comprend donc pourquoi, `$...\Lab$` aurait généré une erreur de compilation.

Voici une commande utilisant un argument : elle permet de dessiner une touche de clavier :[14]

```
\newcommand{\Touche}[1]{\Ovalbox{#1}}
Appuyer sur \Touche{Tab}
   puis sur \Touche{Entrée}
```

4.21

Appuyer sur [Tab] puis sur [Entrée]

On voit donc que cette commande attend *un* argument (c'est le sens de « `[1]` ») et qu'on fait référence à cet argument dans la définition de la commande avec `#1`.

Si l'on souhaite définir une fonction avec plusieurs arguments (9 au maximum), *no problemo :*

[14] Cette commande fait appel à la commande `\Ovalbox` définie dans le package fancybox.

```
\newcommand{\fraction}[2]{%
  \raisebox{0.5ex}{#1}%
  \slash\raisebox{-0.5ex}{#2}}
\fraction{1}{2} et \fraction{3}{4} font
\fraction{5}{4}
```

4.22

$^1/_2$ et $^3/_4$ font $^5/_4$

On remarquera donc que :
- la macro `\fraction` prend 2 arguments,
- on fait référence au n^{e} argument avec `#n`,
- les caractères `%` s'il vous paraissent saugrenus, permettent d'insérer des sauts de ligne dans le code sans insérer d'espace dans le document (voir aussi le paragraphe 9.2.1 page 107 à ce sujet).

Il est également possible de définir une commande dont le premier argument est optionnel. La syntaxe est alors la suivante :

```
\newcommand{nomcom}[narg][arg defaut]{code LaTeX}
```

où narg est le nombre d'arguments, sachant que `#1` sera l'argument par défaut, arg defaut est la valeur que prend `#1` par défaut, et code LaTeX le code de la commande. Voici par exemple une autre approche de la commande vue précédemment, qui dessine une touche de clavier :

```
\newcommand{\Touche}[1][Entrée]{\Ovalbox{#1}}
Appuyer sur \Touche[Tab]{} puis sur \Touche{}
```

4.23

Appuyer sur Tab puis sur Entrée

On voit donc que l'argument 1 est facultatif et sa valeur par défaut est : « Entrée ». On notera également que l'utilisation de l'argument optionnel requiert des crochets et non des accolades.

On peut très bien imaginer que l'on ait à définir une commande ayant un argument optionnel et un ou plusieurs arguments obligatoires. Dans ce cas le premier argument obligatoire sera #2. D'autre part, notez qu'on ne peut rendre optionnel que le **premier argument** d'une commande.

4.5.2 Environnement

Il est possible de définir ses propres environnements de la manière suivante :

```
\newenvironment{nom env}[narg]{clause begin}{clause end}
```

où nom env est le nom de l'environnement ainsi défini, narg le nombre d'arguments, et clause begin et clause end les « pré » et « post » traitements de l'environnement. Il est pratique de définir des environnements à partir d'autres, par exemple les environnements de LaTeX :

```
\newenvironment{bonmot}%
{\small\slshape\begin{flushright}}%
{\end{flushright}\normalsize\upshape}
\begin{bonmot}
  L'homme a reçu de la nature une clef\\
  avec laquelle il remonte la femme\\
  toutes les vingt-quatre heures.
\end{bonmot}
```

4.24

L'homme a reçu de la nature une clef
avec laquelle il remonte la femme
toutes les vingt-quatre heures.

Il est vrai que ce « bon mot » serait un peu douteux si l'on ne citait son auteur. On peut y remédier en ajoutant à notre environnement un argument. Les arguments sont accessibles par # mais ne sont visibles que dans la clause `begin`. On contourne ceci en sauvant l'argument dans une boîte que l'on réutilise dans la clause `end` :

```
\newsavebox{\auteurbm}
\newenvironment{Bonmot}[1]%
  {\small\slshape%
  \savebox{\auteurbm}{\upshape\sffamily#1}%
  \begin{flushright}}
  {\\[4pt]\usebox{\auteurbm}
  \end{flushright}\normalsize\upshape}
\begin{Bonmot}{Victor Hugo}
  L'homme a reçu de la nature une clef\\
  avec laquelle il remonte la femme\\
  toutes les vingt-quatre heures.
\end{Bonmot}
```

L'homme a reçu de la nature une clef
avec laquelle il remonte la femme
toutes les vingt-quatre heures.

Victor Hugo

La citation n'en reste certes pas moins douteuse...

4.5.3 Redéfinitions

Il est possible de *redéfinir* commandes et environnements avec :

```
\renewcommand{nomcom}[nargs]{codeTEX}
```

pour les commandes et :

```
\renewenvironment{nom_env}[narg]{clause_begin}{clause_end}
```

pour les environnements. On redéfinit les commandes essentiellement pour *personnaliser* le comportement facétieux de LaTeX. On procède alors de la manière la plus naturelle qui soit, par exemple :

```
\renewcommand{\thepage}{\Roman{page}}
```

numérote les pages en chiffre romain majuscule.

La modification du comportement par défaut de LaTeX est un sujet très vaste qui dépasse le cadre de cette partie. Mais sachez que si vous modifiez une commande ou un environnement dont vous ne maîtrisez pas toutes les fonctionnalités, attendez vous à des résultats bizarres ! La lecture de la deuxième partie présente le moyen de redéfinir certaines commandes de LaTeX.

4.6 Mais encore ?

Si vous avez l'intention de créer des fichiers contenant des commandes de votre crû, vous devez ajouter la ligne :

```
export TEXINPUTS=$HOME/LaTeX/mesmacros//:
```

dans votre `.bash_profile` si vous utilisez bash, pour que LaTeX cherche aussi les fichiers dans le répertoire `$HOME/LaTeX/mesmacros` (c'est un exemple) et ses sous-répertoires. La ligne `\usepackage{moncru}` vous permettra alors d'utiliser votre

ensemble de commandes. LaTeX cherchera alors le fichier `moncru.sty`. Une autre solution est d'utiliser la commande `\input{moncru.sty}`.

Dans la plupart des distributions de LaTeX, un fichier nommé `texmf.cnf` définit un certain nombre de paramètres permettant de configuer le moteur LaTeX et notamment les chemins de recherche des fichiers. Sur mon système (TeTeX sous Debian) ce fichier contient notamment :

```
HOMETEXMF = $HOME/texmf
```

indiquant que le système cherchera les fichiers à inclure dans le répertoire `texmf` de votre répertoire privé. Vous devrez alors placer vos extensions « maison » dans le répertoire `~/texmf/tex/latex`.

Un dernier conseil : pour pouvoir définir vos commandes ou environnements de manière plus confortable, nous vous recommandons de jeter un petit coup d'œil sur :

- l'extension ifthen qui propose des structures de contrôle de type « si-alors-sinon » et « faire-tant-que »,
- le package calc qui permet d'effectuer des opérations arithmétiques sur les compteurs et les longueurs.
- enfin l'environnement `list` qui peut être un bon point de départ pour se définir un environnement de type liste.

Ces extensions et leur utilisation sont présentées en détail dans la deuxième partie de ce manuel.

5 Graphisme

Sommaire

Tu ne te feras aucune image sculptée de rien
qui ressemble à ce qui est dans les cieux là-haut [...]
Tu ne te prosterneras pas devant ces images ni ne les serviras.

Le Deutéronome Dt **5** 8.

Aujourd'hui il est tout à fait naturel d'insérer des dessins, figures et autres images dans un document. Ceci est dû aux imprimantes de plus en plus performantes et bon marché. Il faut cependant se replacer dans le contexte des années 80 à l'essor de TEX. C'est l'époque de l'apparition des imprimantes et le matériel de qualité professionnelle n'était pas accessible au particulier. Cependant beaucoup de solutions d'impression émergeront s'appuyant la plupart sur le langage PostScript devenu *ipso facto* un standard.

Il existe plusieurs solutions autour de LATEX pour insérer des graphiques dans un document. Parmi elles on notera l'utilisation de metafont (l'utilitaire qui gère les fontes de LATEX), la programmation d'un environnement `picture` ou la mise en œuvre d'un code PICTEX. Ces solutions ne seront pas décrites ici car nous considérons qu'elles sont d'une utilisation un peu déroutante au premier abord ; il est tout de même bon de connaître leur existence. L'approche adoptée dans ce manuel pour manipuler des graphiques est d'insérer dans le source LATEX un fichier au format PostScript encapsulé contenant le graphique en question, ce dernier ayant été créé par un logiciel de dessin tel que xfig, gnuplot, gimp, etc.

5.1 Apéritifs

Il n'est pas inutile de connaître la commande `\rule` qui permet de faire des traits :

`\rule[`hpos`]{`largeur`}{`hauteur`}`

où hpos impose une éventuelle translation verticale du trait, les deux autres arguments ont un nom suffisamment explicite :

```
Voici quelques \og traits \fg{} :
\begin{center}
  \rule[1ex]{1mm}{5mm}\quad\rule{1in}{0.4pt}
  \quad\rule[-0.5em]{1em}{1em}
\end{center}
```

Voici quelques « traits » :

5.2 Du format des fichiers graphiques

Pour inclure des dessins ou des images dans vos documents, il faut insérer un *fichier*. La configuration de LaTeX permet d'incorporer des fichiers de type PS pour PostScript et EPS pour Encapsuled PostScript. Ce fichier peut être généré par n'importe quel programme. Si le format PostScript vous semble contraignant, sachez que :

- tout *bon* logiciel de dessin « vectoriel » vous permet d'exporter vos schémas au format EPS. Ce format est devenu la référence en matière d'impression.
- toute image peut être convertie au format EPS. Sur un système Unix, le programme convert permet d'effectuer cette opération.[1] On pourra aussi recourir au logiciel gimp (logiciel libre de retouche et de création d'image numérique), présent également sur d'autres systèmes d'exploitation.

5.3 Le package graphicx

LaTeX, ou plutôt TeX, n'a pas été initialement conçu pour manipuler des graphiques (images, dessins,...). De ce fait, une multitude d'extensions ont été proposées, aucune n'ayant vraiment réussi à s'imposer ou à être vraiment indépendante des systèmes d'exploitation.

5.3.1 Un standard

Aujourd'hui, les concepteurs de LaTeX semblent s'être mis d'accord pour standardiser une extension *graphique*. Deux extensions ont donc vu le jour à fin de l'année 1994 :

- graphics l'extension « standard » ;
- graphicx l'extension « plus plus ».

Nous avons choisi de vous présenter graphicx. Il faut bien comprendre que si l'interface de cette extension est indépendante du système d'exploitation, la partie du code gérant les différent types de fichiers graphiques est dépendante du système sous-jacent. Aussi, est-il nécessaire de préciser un *driver* de package. Les drivers existants correspondent aux implantations connues de TeX sur des plate-formes diverses.[2] Sous Unix, le driver utilisé est généralement dvips, il est choisi par défaut dans la distribution teTeX, si bien que la ligne :

```
\usepackage{graphicx}
```

[1] Cherchez du côté de la suite « ImageMagick » pour obtenir ce programme s'il n'est pas présent sur votre système.

[2] On notera entre autres : xdvi et dvips pour le monde Unix, texture et OzTeX pour le Mac, emTeX et dviwin pour windows.

FIG. 5.1 – Robert (après quelques bières).

suffit pour mettre en route l'extension graphicx. La commande pour inclure un dessin ou une figure est la suivante :

```
\includegraphics[option]{fichier}
```

où fichier est un fichier contenant votre figure, et option est une liste d'options séparées par des virgules. La commande `\includegraphics` ne crée pas de mise en page particulière, elle insère juste une boîte contant le graphique dans le texte. Ainsi :

```
avant
\includegraphics{punch}
et après.
```

5.2

avant  et après.

Pour assurer la portabilité de vos sources et ainsi pouvoir insérer des fichiers graphiques dans des formats différents, il est impératif de *ne pas préciser l'extension du fichier dans la commande* `\includegraphics`.

En général, on combine `\includegraphics` avec un environnement `figure`. Par exemple, la figure 5.1 a été produite grâce au code suivant :

```
\begin{figure}
   \centering\includegraphics[width=5cm]{punch}
   \caption{Robert (après quelques bières).}
   \label{fig-exemple}
\end{figure}
```

Notez bien que l'environnement `figure` assure que le graphisme « flotte » dans la page et que ça n'est pas la commande `\includegraphics` qui assure ce rôle. Au cas où ça aurait échappé à certains :

L'environnement `figure` assure que le graphisme « flotte » dans la page ; ça n'est pas la commande `\includegraphics` qui assure ce rôle.

5.3.2 Options

Le package graphicx possède plusieurs options permettant de contrôler l'insertion des graphiques. Parmi les options disponibles voici les plus utilisées :

Changement d'échelle

Il existe trois manières d'agir sur la taille d'un graphique.

- `scale`=ratio, où ratio est un nombre positif ou négatif, permet de changer la taille globale de la figure ;
- `width`=dimen permet d'imposer la largeur du graphique ;
- `height`=dimen permet d'imposer la hauteur du graphique.

```
\begin{center}
  \includegraphics[scale=0.2]{magma}\\
  \includegraphics[width=8.5mm]{magma}\\
  \includegraphics[width=2cm,height=3mm]{magma}
\end{center}
```

5.3

Rotation

Vous pouvez si vous le désirez, faire effectuer une rotation à votre figure en utilisant l'option `angle`, dont la syntaxe est la suivante :

`angle`=ndegre

où ndegre est un angle précisé en degrés dans le sens trigonométrique.

```
\includegraphics[angle=45,scale=0.2]{magma}
```

5.4

On trouvera dans le fichier `grfguide.pdf`[3] une description détaillée de cette extension. On pourra également consulter le fichier `fepslatex.pdf`[4].

Mode brouillon

L'option `draft` permet de produire les figures en mode « brouillon » : seul un cadre avec le nom du fichier inclus est produit dans le document final.

```
avant
\includegraphics[draft,scale=.2]{punch}
et après.
```

5.5

avant pngs/punch.png et après.

Le mode `draft` est enclenché par défaut lorsque l'option de document `draft` est spécifiée. Si vous voulez contrer l'effet de l'option de document[5], il est possible d'utiliser l'option `final` de la commande `\includegraphics` ou au moment d'inclure l'extension avec `\usepackage`.

[3] Utiliser `locate` ou `find` pour le trouver sur votre système.

[4] http://tug.ctan.org/tex-archive/info/epslatex/french/fepslatex.pdf

[5] Par exemple, si vous voulez voir vos figures mais aussi les « OverfullBoxMark. »

5.4 Quelques extensions utiles

Voici dans les paragraphes qui suivent trois extensions utiles pour la production de documents contenant des graphiques.

5.4.1 subfig

Cette extension permet de gérer des figures comportant plusieurs sous-figures, avec numérotation automatique et possibilité de faire référence aux sous-figures elles-mêmes. Par exemple :

(a) Magma (b) UZMK

FIG. 5.2 – Uniweria Zëkt

```
\begin{figure}[htbp]
  \begin{center}
    \leavevmode
    \subfloat[Magma]{%
      \label{fig-uniweria-magma}
      \includegraphics[width=2cm]{magma}}
    \hspace{2cm}
    \subfloat[UZMK]{%
      \label{fig-uniweria-uzmk}
      \includegraphics[height=2cm]{uzmk}}
    \caption{Uniweria Zëkt}
    \label{fig-uniweria}
  \end{center}
\end{figure}
```

Pour ce qui concerne les références, on peut soit référencer la figure globale par `\ref{fig-uniweria}` qui donne : 5.2, soit les sous-figures par leur label respectif : `\ref{fig-uniweria-magma}` et `\ref{fig-uniweria-uzmk}` qui donnent : 5.2a et 5.2b.

Une manière élégante de gérer les `\subfigures` est d'encapsuler chacune d'elles dans un environnement `minipage`. On trouvera dans le fichier `subfig.pdf` accompagnant la distribution, comment personnaliser l'environnement `subfigure`, notamment les espaces inter-légendes.

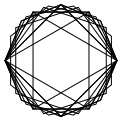
5.4.2 Le package wrapfig

Le package wrapfig propose l'environnement `wrapfigure` permettant de faire flotter une figure dans un paragraphe. Il ne s'agit pas d'un environnement flottant au sens de l'environnement `figure` de LaTeX puisqu'on spécifie la position de la figure dans le paragraphe. La syntaxe est la suivante :

```
\begin{wrapfigure}{position}{largeur}
  ...
\end{wrapfigure}
```

où position est la position de la figure (`l` ou `r`) et largeur la largeur de la figure à insérer. Voici un exemple :

```
\begin{wrapfigure}{r}{1.5cm}
  \includegraphics[width=1cm]{polygons}
\end{wrapfigure}
Le package \ltxcom{wrapfig} n'est ---~à ma connaissance~--- pas
documenté sous la forme d'un fichier \texttt{dvi} ; par contre
il est possible...
```



Le package `\wrapfig` n'est — à ma connaissance — pas documenté sous la forme d'un fichier `dvi` ; par contre il est possible de trouver des informations très détaillées dans le fichier `.sty` lui-même qui se trouve dans l'arborescence TEX dans : `[...]/misc/wrapfig.sty`. On notera au passage — car il faut parler pour faire un paragraphe un peu long — que la règle veut que tout package soit « auto-documenté » grâce à une extension connue sous le nom de docstrip. Ainsi toute extension — *package* en anglais — contient aussi bien le code que la documentation. Une procédure d'installation permet d'extraire l'un et l'autre. L'auteur de wrapfig n'a vraisemblablement pas suivi cette règle, tant pis...

5.4.3 Le package psfrag

Une autre extension intéressante est l'extension psfrag. Elle a pour but de pouvoir réunir la puissance d'un fichier PostScript et la beauté des équations de LATEX. Un problème se pose en effet lorsque l'on veut intégrer des formules à un dessin, car la génération d'équations n'est pas prévue dans la plupart de ces logiciels. La solution adoptée par les auteurs de psfrag est d'utiliser la commande `\psfrag` pour insérer les formules à la place de chaînes de caractères présentes dans le dessin. Ainsi, pour avoir la figure 5.3b au lieu de la figure 5.3a, on a procédé comme suit :

1. ajout avant le `\includegraphics{courbe}` la ligne :

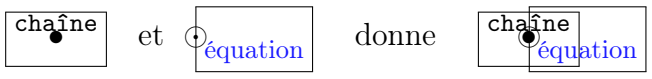
   ```
   \psfrag{exp(-x)*sin(10*x)}[r][r]{$e^{-x}\sin(10x)$}
   ```

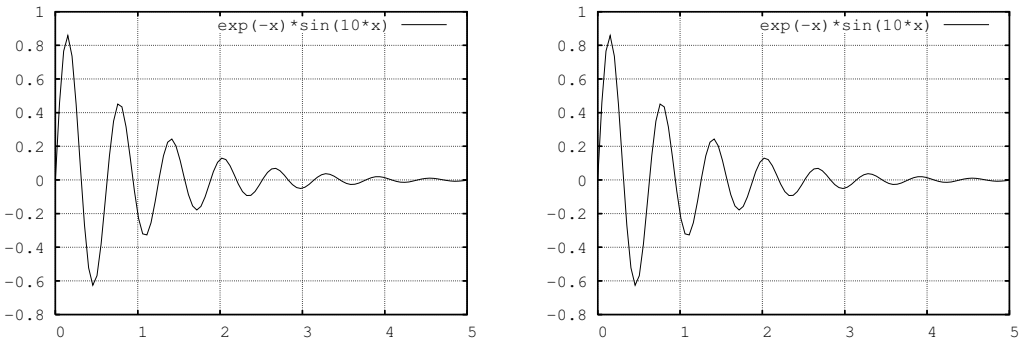
   qui permet de remplacer la chaîne de caractères faisant office de légende par une belle équation ;
2. le résultat n'est pas visible dans le fichier `.dvi`, par contre dvips se charge d'exploiter les instructions précédentes pour modifier le fichier PostScript généré.

Le positionnement de la formule se fait en faisant correspondre deux points de référence, l'un appartenant à l'équation, l'autre à la chaîne de caractères à remplacer. C'est à l'utilisateur d'indiquer où se trouve ces points de référence, par l'intermédiaire de deux arguments optionnels à la commande `\psfrag`. Supposons qu'on définisse ces points de référence comme suit :

```
\psfrag{chaîne}[l][c]{equation}
```

On aura alors l'assemblage suivant :





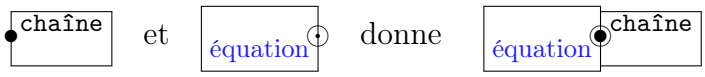
(a) avant (b) après

FIG. 5.3 – Utilisation de psfrag, à gauche la figure originale, à droite la figure avec un remplacement par une équation LaTeX.

De même en écrivant :

```
\psfrag{chaîne}[r][l]{equation}
```

on aura :



Dans l'exemple de la figure 5.3b, on a fait correspondre le côté droit de l'équation (1er argument optionnel `r)` avec le côté droit de la chaîne (2e argument optionnel `r`). La documentation du package est très instructive à ce sujet...

Attention, si on génère un document pdf à partir du source LaTeX, on ne peut utiliser le package psfrag qu'au prix de manipulations un peu tordues.

5.4.4 Le package xcolor

L'extension xcolor est mise au point par l'équipe qui a développé le package graphicx. Il peut être intéressant — par exemple pour produire des transparents — de générer du texte en couleur. Le package xcolor permet les constructions suivantes :

```
Du texte {en \color{red}rouge} et
\textcolor{cyan}{en cyan}.

Une boîte \colorbox{green}{Verte}.

Une \fcolorbox{blue}{yellow}{autre boîte}.
```

Du texte en rouge et en cyan.
Une boîte Verte.
Une autre boîte.

5.6

On aura compris qu'on dispose pour le texte :

- de la déclaration :

  ```
  \color{couleur}
  ```

- et de la commande :

  ```
  \textcolor{couleur}{texte}
  ```

et pour les boîtes :

5

– sans bordure :

```
\colorbox{couleur du fond}{contenu}
```

– avec bordure :

```
\fcolorbox{couleur bordure}{couleur fond}{contenu}
```

Les deux commandes pour les boîtes en couleur sont sensibles à la longueur `\fboxsep`. « *Quid* des couleurs qui n'ont pas de nom ? » vous entends-je marmonner *in petto*... Ce à quoi je réponds sur le champ :

```
Voici un
{\color[rgb]{.2,.4,.5}\bfseries bleu gris}...
```

5.7 Voici un **bleu gris**...

Il est aussi possible de donner un « petit nom » à cette dernière couleur :

```
\definecolor{bleugris}{rgb}{.2,.4,.5}
Voici un
{\color{bleugris}\bfseries bleu gris}...
```

5.8 Voici un **bleu gris**...

Vous noterez qu'en lieu et place du modèle de couleur « `rgb` » il est possible d'utiliser le modèle `gray` de manière à définir des nuances de gris. De même, en utilisant le modèle `html`, on pourra utiliser la syntaxe du langage Html pour spécifier les couleurs.

5.5 Utiliser make

Ce paragraphe est destiné aux utilisateurs d'un système d'exploitation disposant de l'utilitaire Gnu make (pour de plus amples informations sur cet utilitaire, n'hésitez pas lire à l'incontournable [7]). Les autres peuvent passer leur chemin...

Voici donc une idée de makefile qui vous permettra d'automatiser la génération :

– des fichiers au format Eps à partir des images « bitmaps » ;
– des fichiers au format Eps à partir de fichiers stockés dans le format d'un logiciel de dessin vectoriel.

On souhaite pour ce faire, élaborer une cible que l'on précisera en ligne de commande :

```
make figs
```

On suppose que les images et les fichiers de dessins sont respectivement stockés dans les sous-répertoire `Imgs` et `Figs` du répertoire contenant le document maître et que les fichiers Eps fabriqués seront également stockés dans un sous-répertoire `Epss` (voir la figure 5.4 page ci-contre). On commencera donc par définir un ensemble de variables précisant les différents répertoires à utiliser :

```
FIGSDIR=Figs
EPSSDIR=Epss
IMGSDIR=Imgs
```

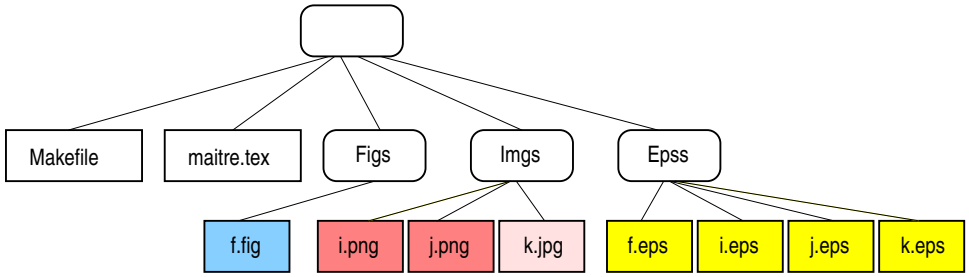


FIG. 5.4 – Proposition d'arborescence pour stocker les fichiers graphiques : un répertoire pour les images et un un répertoire pour les graphiques vectoriels. Les fichiers au format Eps sont enregistrés dans un répertoire à part.

5.5.1 Convertir les images

Tout d'abord on fait la liste des fichiers au format Jpeg et Png (c'est un exemple) en stockant cette liste dans deux variables comme suit :

```
PNGS=$(notdir $(wildcard $(IMGSDIR)/*.png))
JPGS=$(notdir $(wildcard $(IMGSDIR)/*.jpg))
```

La fonction `wildcard` permet d'obtenir la liste des fichiers du répertoire `$(IMGSDIR)` tandis que la fonction `notdir` supprime la partie « répertoire » de chacun des fichiers. Finalement la variable `PNGS` contiendra :

```
i.png j.png
```

et `JPGS` :

```
k.jpg
```

On peut ensuite à partir de ces deux variables créer la liste des fichiers Eps à fabriquer (rappelez-vous qu'ils doivent résider dans un répertoire à part) :

```
IMGS2EPSS=$(patsubst %,$(EPSSDIR)/%,\
            $(PNGS:.png=.eps) $(JPGS:.jpg=.eps))
```

L'expression à droite de l'affectation permet de changer l'extension en `eps` dans les deux listes précédentes et de préfixer chaque nom par le répertoire de stockage des fichiers Eps. `IMG2EPSS` contiendra donc :

```
Epss/i.eps Epss/j.eps Epss/k.eps
```

Cette liste constitue les « prérequis » (au sens de make) pour fabriquer les images. On définit donc la cible `figs` comme suit :

```
figs : $(IMGS2EPSS)
```

Il faudra également expliquer à make comment on peut fabriquer un fichier au format postscript encapsulé à partir d'une image. Ceci pourra s'écrire à l'aide de la règle suivante :

```
$(EPSSDIR)/%.eps : $(IMGSDIR)/%.png
⟼ convert $<  EPS:$@
$(EPSSDIR)/%.eps : $(IMGSDIR)/%.jpg
⟼ convert $<  EPS:$@
```

qui précise qu'on utilisera l'utilitaire convert[6] pour convertir un fichier Png ou Jpeg en Eps.

5.5.2 Convertir les fichiers de dessin

La conversion des fichiers de dessins suit exactement le même principe. Supposons qu'on dispose de sources au format Fig provenant de xfig et au format Svg provenant de Inkscape. On aura alors dans le `makefile` :

```
FIGS=$(notdir $(wildcard $(FIGSDIR)/*.fig))
SVGS=$(notdir $(wildcard $(FIGSDIR)/*.svg))
FIGS2EPSS=$(patsubst %,$(EPSSDIR)/%,\
          $(FIGS:.fig=.eps) $(SVGS:.svg=.eps))
```

Les règles de conversion sont bien évidemment différentes puisqu'elles font appel l'une à fig2dev (utilitaire connexe à xfig) et à Inkscape lui-même pour l'autre :

```
$(EPSSDIR)/%.eps : $(FIGSDIR)/%.fig
⟼ fig2dev -L eps $< > $@
$(EPSSDIR)/%.eps : $(FIGSDIR)/%.svg
⟼ inkscape -E $@ $<
```

La cible permettant de fabriquer les fichiers de dessins et les images devient finalement :

```
figs : $(IMGS2EPSS) $(FIGS2EPSS)
```

5.6 À part ça

On peut trouver un grand nombre d'extensions permettant de produire des graphiques correspondant à un besoin particulier (arbres, circuits électroniques, histogrammes,...). Vous pouvez en effet avoir besoin, un jour, de générer des graphiques de manière automatique à partir d'une commande. Jetez alors un coup d'oeil sur les différentes extensions disponibles (pstricks, METAPOST,...) ainsi que sur l'environnement `picture` et ses extensions epic et eepic. Bon courage !

[6] Du package ImageMagick disponible à http://www.imagemagick.org, qui convertit à peu près tout format d'images

6 Documents scientifiques

Sommaire

Les sages thésaurisent la science
mais la bouche du fou
est un danger permanent.

Les proverbes Pr **10** 14.

VOICI venu le moment de vous parler des quelques caractéristiques des documents dits *scientifiques*. Si le problème des formules et autres équations a été abordé avec brio au chapitre 3, il reste tout de même un gros morceau à avaler : la bibliographie. Sachez quand même que si l'ingurgitation peut être difficile, la suite vous permettra de vous simplifier grandement le travail. Nous profiterons également de ce chapitre pour expliquer le principe de la génération d'index.

Nous vous parlerons donc des quelques particularités de la rédaction d'article, ensuite viendra un exposé sur la génération d'une bibliographie, la génération d'index, enfin la méthode utile à connaître pour diviser un gros document en petites parties.

6.1 Article

Pour rédiger un article, rien de bien nouveau, tout ce qui a été vu jusqu'ici s'applique. On notera juste l'utilisation dans le préambule, des commandes :

- `\title` pour définir le titre,
- `\date` pour définir la date,
- `\author` pour définir les auteurs,
- `\thanks` pour spécifier l'affiliation des auteurs.

Pour insérer le titre à partir de ces définitions, il est nécessaire d'ajouter la commande `\maketitle` **après** le `\begin{document}` :

```
\documentclass{article}
\title{Le seuillage à 128 : une révolution !}
\author{M. C. Orlanrien\\
        Institut du Pixel\\
        42007 Saint-Etienne---FRANCE}
\date{2 Avril 1927}

\begin{document}
\maketitle% c'est  ici  qu'est  inséré  le  titre
```

```
...
\end{document}
```

Nous répétons :[1] c'est la commande `\maketitle` qui génère et insère le titre et non les définitions du préambule.

En règle générale, les conférences ou revues qui fournissent un fichier de style, proposent quelques variantes, par exemple une commande `\address` pour séparer les auteurs et leur adresse respective, etc. Mais l'idée de base reste la même.

6.2 Bibliographie

Il existe deux manières de rédiger une bibliographie avec LaTeX : l'une que l'on peut qualifier de « manuelle » consiste à insérer un environnement `thebibliography` dans le document, l'autre que nous allons décrire ici, utilise le programme BibTeX. Voici le principe :

1. on crée un ou plusieurs fichiers de données contenant une description de chaque entrée de bibliographie (article, conférence,...) au format BibTeX. C'est l'inévitable tâche de *saisie*,
2. dans le document, on fait référence aux entrées par la commande `\cite`,
3. la bibliographie sera formatée automatiquement selon un style particulier que vous choisirez.

L'avantage de cette méthode est que vous saisissez une fois pour toute les entrées de votre bibliographie. De plus, vous n'avez pas à vous soucier de sa mise en page, dans la mesure où vous utilisez des *fichiers de style* ; il en existe plusieurs dizaines correspondant à toutes sortes de standards, revues et autres conférences. On trouve aussi sur internet beaucoup de bases de données bibliographiques au format BibTeX que l'on peut utiliser directement dans ses documents.

Nous répétons qu'il existe des standards en matière de bibliographie, mais que malheureusement certaines revues prennent un malin plaisir à pondre leur propre style de bibliographie. Le jour où vous publierez dans ce genre de revue, vous aurez à créer ou adapter un fichier de style. Pour ce faire, cherchez du coté de l'utilitaire makebst.

6.2.1 Fichier `.bib`

La première opération est de constituer[2] le fichier de bibliographie qui doit de préférence porter l'extension `.bib`. Ce fichier doit suivre une syntaxe particulière. Tout d'abord, il faut savoir que BibTeX distingue chaque entrée par son *type*. Ainsi, chaque entrée correspond à un type de document : livre, article, conférence, rapport technique,... En tout plus d'une douzaine de types de document différents.

Accompagnant les distributions LaTeX, on trouve normalement un fichier nommé BibTeXing (généralement sous le nom `btxdoc.dvi`) écrit par Oran Patashnik il y a une vingtaine d'années et contenant une source importante d'information sur la manière de constuire un fichier au format BibTeX.

Chaque *type* d'entrée contient à son tour un certain nombre de *champs* décrivant l'entrée. La structure d'une entrée de bibliographie est la suivante :

[1] Parce qu'il paraît qu'enseigner c'est répéter

[2] Le module AucTeX d'Emacs possède un mode BibTeX très pratique.

```
@entree{clef,
  champ1 = {...},
  champ2 = {...},
  ...
  champn = {...}
  }
```

où entree est le type de document (`article`, `inproceedings`,...) et $champ_1$, $champ_2$,...,$champ_n$ sont les différents champs de l'entrée de bibliographie. Ces différents mots réservés de BibTeX peuvent être saisis en majuscules ou en minuscules.

Le symbole clef doit identifier le document de manière univoque. Ce symbole est à rapprocher du symbole identifiant une étiquette avec `\label`. Pour vous permettre de commencer à utiliser rapidement BibTeX nous vous donnons, ici, un exemple pour les trois principales entrées que vous serez amené à utiliser :

Article dans une revue

Un article dans une revue doit être saisi comme suit :

```
@article{qtz:UchArb,
  author ={Uchiyama, Toshio and Arbib, Michael A.},
  title = {Color Image Segmentation Using Competitive Learning},
  journal=pami,
  volume =16, number=2, pages={1197--1206},
  month=dec, year=1994}
```

Notez que :

1. les champs `author`, `title`, `journal`, `year` sont obligatoires ;
2. pour les auteurs,[3] il est impératif de suivre l'ordre nom, prénom et de séparer **tous** les auteurs par `and` ;
3. pour les auteurs ayant un nom composé ou à particule, on saisira :

   ```
   author="de la Motte Beuvron, Alain"
   ```

   donc en respectant l'ordre particule, nom, prenom et en utilisant la virgule en guise de séparateur comme indiqué ci-dessus ;
4. tous les mois de l'année peuvent être produits grâce aux chaînes : `jan`, `feb`, `mar`, etc.

Nous avons créé par commodité l'*abréviation* `pami` qui est définie au début de notre fichier `.bib` par :

```
@string{pami="IEEE transactions on Pattern Analysis and Machine
             Intelligence"}
```

Article dans une conférence

Eh oui, BibTeX distingue un article dans une *revue*, et un article dans une *conférence*. La structure est sensiblement la même, si ce n'est qu'on utilise le champ `booktitle` pour le titre de la conférence, à la place du titre de la revue :

[3] Ces remarques concernant les auteurs sont valables pour les autres entrées (conférences, livres, etc.

```
@Inproceedings{qtz:BouOrch,
  author={Bouman, Charles A. and Orchard, Michael T.}
  title={Color Image Display with a Limited Palette Size},
  booktitle={SPIE Conference on Visual Communications
             and Image Processing},
  volume=1199,pages={522--533},
  year=1989}
```

Ici les champs `author`, `title`, `booktitle`, `year` sont obligatoires, et l'on doit choisir entre `volume` et `number`.

Un extrait de livre

On cite souvent un extrait—chapitre(s), ou page(s)—d'un livre plutôt que le livre lui-même :

```
@inBook{col:McA,
  author =    {MacAdam, David L.},
  title =     {Color Measurement},
  chapter =   4,
  pages   = {48--49},
  publisher = {Springer-Verlag},
  year =      1985}
```

Sont obligatoires : `author`, `title`, `chapter` ou `pages`, `publisher` (l'éditeur) et `year`.

Encore une fois nous ne saurons trop vous conseiller d'exploiter le mode BibTeX du module AucTeX d'Emacs. Ce mode vous propose notamment un menu contenant tous les types d'entrée. La sélection d'un item de ce menu insère un « squelette » d'entrée dans votre fichier `.bib`. Ce module est téléchargeable à `ftp://ftp.lip6.fr/pub/TeX/CTAN/support/auctex` et également disponible sous la forme d'un paquet Debian.

6.2.2 Citation

Une fois le (ou les) fichier(s) de bibliographie constitué(s), on peut faire référence aux entrées par l'intermédiaire des clefs, avec la commande `\cite` :

```
\cite{clef}
```

La commande `\cite` a pour effet :

1. d'insérer un renvoi dont la forme dépend du style choisi ([2], [Loz95],...),
2. d'ajouter l'article cité dans la bibliographie de votre document.

Un article—au sens large du terme—n'apparaît dans la bibliographie que s'il fait l'objet d'une commande `\cite`. Pour qu'un article apparaisse sans pour autant être cité, il faut utiliser la commande `\nocite{clef}`. L'article référencé par clef sera alors inséré dans la bibliographie. Par ailleurs, la commande `\nocite{*}` insère *toutes* les entrées de votre fichier de biblio.

Avant de passer à l'étape de génération proprement dite, il est nécessaire d'insérer à la fin du document LaTeX un appel à la commande `\bibliographystyle` pour stipuler un style de bibliographie, puis un appel à la commande `\bibliography` pour insérer effectivement la bibliographie. Pour le style :

```
\bibliographystyle{style}
```

Les trois styles prédéfinis :[4] de LaTeX sont :

- `plain` les citations sont sous la forme [2], et la bibliographie est classée par auteur,
- `unsrt` idem mais pas de tri, les documents apparaissent dans l'ordre où ils sont cités ; très utilisé pour les actes de conférences.
- `alpha` les citations sont sous la forme « auteurs abrégés + année ».

Il faut ensuite spécifier quels sont les fichiers contenant les informations bibliographiques sur lesquelles « pointent » les commandes `\cite` de votre document :

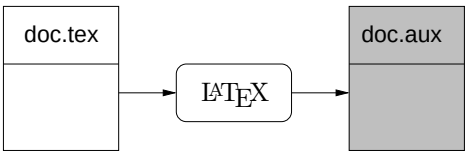
```
\bibliography{fichier1,fichier2,...}
```

indiquera à BibTeX de considérer les fichiers `fichier1.bib`, `fichier2.bib`,... lors de son traitement.

6.2.3 Génération

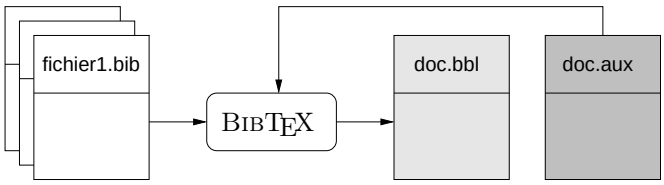
Pour générer la bibliographie :

1. effectuer une première compilation avec LaTeX pour que le fichier auxiliaire `doc.aux` contienne les informations de *citations* :

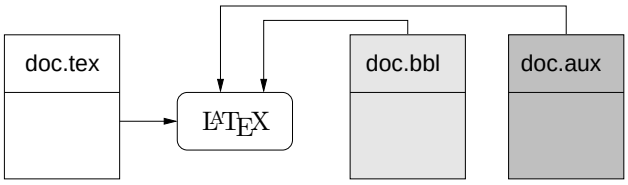


2. lancer BibTeX pour générer la bibliographie dans le fichier `doc.bbl` :

```
bibtex doc
```



3. effectuer une deuxième compilation avec LaTeX pour insérer la bibliographie :



4. résoudre les références croisées par une troisième compilation.

Si vous êtes curieux, vous verrez que le fichier `doc.bbl` contient un environnement `thebibliography` prêt à l'emploi[5] et que le fichier `doc.blg` est l'équivalent du `.log` : un fichier « log » contenant les éventuelles erreurs ou warnings de la dernière utilisation de BibTeX.

Le programme BibTeX est sensible à la variable d'environnment `BIBINPUTS`. Il peut donc parfois être nécessaire d'ajouter la ligne :

[4] Cherchez sur les sites CTAN, dans le répertoire `biblio/bibtex/contrib` il y a plusieurs dizaines d'autres styles disponibles.

[5] C'est-à-dire celui que vous auriez dû vous palucher si vous n'utilisiez pas BibTeX.

```
export BIBINPUTS=$HOME/LaTeX/biblio//:
```

dans votre `.bash_profile` pour que BibTeX cherche vos fichiers de bibliographie dans le répertoire `$HOME/LaTeX/biblio` (c'est un exemple).

6.3 Index

La génération d'index s'appuie sur deux concepts :

1. l'ajout de commandes `\index` dans le document LaTeX pour ajouter des entrées dans l'index ;
2. l'utilisation du programme makeindex qui va trier et mettre en page l'index proprement dit.

C'est la commande `\printindex` qui insère l'index dans le document. Cette commande est analogue à la commande `\tableofcontents`.

6.3.1 Ce qu'il faut faire

Voici un petit mémo pour faire un index.

1. rajouter deux commandes dans le document maître :

```
\makeindex                    ← pour dire à LaTeX de générer les index
\begin{document}
... le document ...
\printindex                   ← Pour réellement l'insérer dans le document

\end{document}
```

2. ajouter une entrée dans l'index :

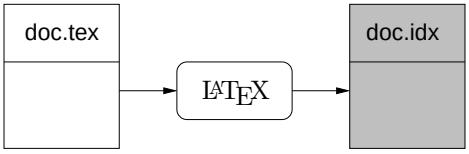
```
\index{bidule}                ← insère « bidule » dans l'index
```

3. pour générer l'index pour le document `doc.tex`, on lancera successivement les trois commandes suivantes :

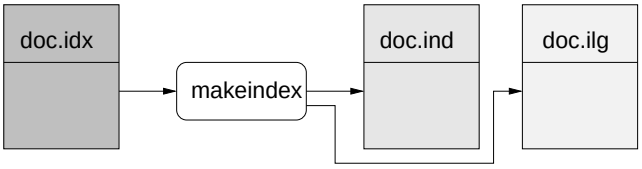
```
latex doc
makeindex doc
latex doc
```

6.3.2 Détail du fonctionnement

La première compilation du document `doc.tex` (à la condition que la séquence de contrôle `\makeindex` soit présente dans son préambule) génère un fichier `doc.idx` contenant les entrées de l'index « en vrac » :



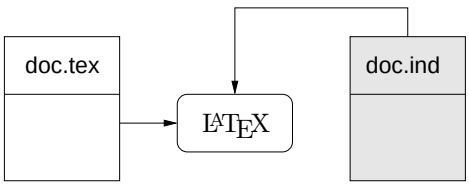
On utilise ensuite makeindex pour classer et supprimer les doublons dans ce fichier `doc.idx`, le résultat est mis dans `doc.ind` ; une trace de l'exécution est stockée dans `doc.ilg` :



makeindex est bavard sur le terminal ; voici ce qu'il dit pour générer l'index de ce document :

```
This is makeindex, version 2.13 [07-Mar-1997] (using kpathsea).
Scanning input file guide.idx....done (982 entries accepted, 0 rejected).
Sorting entries..........done (11254 comparisons).
Generating output file guide.ind....done (745 lines written, 0 warnings).
Output written in guide.ind.
Transcript written in guide.ilg.
```

Il faut donc veiller aux éventuels rejets ou avertissements (*warnings*) et se reporter au fichier log `doc.ilg` le cas échéant. La deuxième compilation avec LaTeX permet d'insérer l'index formaté (fichier `doc.ind`) à l'endroit spécifié par la commande `\printindex` dans `doc.tex` :



L'utilitaire makeindex reconnaît l'option `-s` qui permet de spécifier un *style* pour l'index. Ces styles—définis dans des fichiers portant l'extension `.ist`—changent la mise en page de l'index. On utilise un fichier style de la manière suivante :

```
makeindex -s fichier-style document-maitre
```

Cherchez sur votre distribution quels sont les fichiers de styles et testez-les.

6.3.3 Différents types d'entrée d'index

On peut utiliser des entrées un peu plus sophistiquées que la forme vue jusqu'à maintenant (*<mot> et <page>*). Il existe au moins trois autres entrées :

1. les entrées hiérarchiques :

   ```
   \index{bidule!chouette}
   ```

   insère une sous-entrée 'chouette' à 'bidule'

2. les entrées à cheval sur plusieurs pages :

   ```
   \index{bidule|(}          ← à la page i
   \index{bidule|)}          ← à la page j
   ```

   génère une entrée de type : bidule i–j

3. les entrées symboliques :

   ```
   \index{alpha@\alpha}
   ```

   ajoute la lettre grecque α et la classe à l'entrée 'alpha'. De même :

   ```
   \index{eplucher@éplucher}
   ```

ajoute le mot « éplucher » et le classe parmi les mots commençant par « e »

Cette dernière forme peut également être utilisée pour mettre dans l'index une entrée avec une mise en forme particulière, par exemple :

```
\index{bonjour@\textbf{bonjour}}
```

ajoute **bonjour** (bonjour en gras) dans l'index, et classe cette entrée à 'bonjour'. Enfin on peut vouloir afficher les numéros de pages avec une mise en évidence particulière. On utilisera alors la forme :

```
\index{entrée|commande de mise en forme}
```

Par exemple :

```
\index{bidule|textbf}
```

affichera le numéro de la page où apparaît "bidule" en gras (notez qu'il n'y a pas de caractère \ pour la commande de mise en forme).

6.3.4 Glossaire

On a parfois besoin de préciser la signification de certains termes d'un document ; la partie d'un manuel qui regroupe l'explication de ces termes s'appelle un *glossaire*. Pour en générer, il faut procéder de manière analogue à un index avec quelques petites variations présentées au paragraphe 11.6 page 171.

6.4 Diviser votre document

Lorsqu'on manipule un gros document, on peut le diviser naturellement en chapitres ou parties. Il est alors conseillé de créer un document *maître* chargé d'inclure ces chapitres ou parties. Le document maître a l'allure suivante :

```
\documentclass{book}

\begin{document}
\frontmatter % tout ce qui est introductif
\include{preface}
\tableofcontents
\mainmatter % le « corps » du document
\include{chapitre1}
\include{chapitre2}
\backmatter % tout ce qui vient en fin de document
\bibliographystyle{plain}
\bibliography{machin,bidule,truc}
\end{document}
```

L'intérêt des commandes `\include` réside dans le fait qu'elles vous permettent de travailler sur un nombre réduit de chapitres à la fois, tout en gardant l'intégrité du document. On utilise pour cela la commande `\includeonly` :

```
\includeonly{preface,savoir}
```

dans le préambule, permet de compiler uniquement la préface (fichier `preface.tex`) et le chapitre contenu dans le document `savoir.tex`.

Chaque commande `\include` commence une nouvelle page. Et il n'y a apparemment pas moyen de passer outre. Vous aurez donc compris que `\include` est à utiliser avec les commandes de section qui sautent une page (`\chapter` et cie). Pour insérer un fichier sans saut de page, utilisez la commande `\input`. Par contre, vous ne bénéficierez pas du mécanisme de document maître.

Enfin, il est utile de noter que les commandes `\frontmatter`, `\mainmatter` et `\backmatter` ne sont pas indispensables, mais permettent automatiquement d'adopter une numérotation en roman pour les pages introductives et d'autres petites choses (elles ne sont cependant accessibles que dans la classe `book`).

7 Des documents en français

Sommaire

L'homme répondit : c'est la femme
que tu as mise auprès de moi
qui m'a donné de l'arbre, et j'ai mangé !

La Genèse Gn **3** 12.

La composition d'un document en français suit des règles qu'il est bon de connaître. Ces règles ne sont pas à proprement parler des directives dont on ne peut se soustraire, il s'agit la plupart du temps de règles d'usage, qu'il est conseillé de suivre pour rendre un document lisible ne perturbant pas le lecteur. Ces conseils d'usage donnent généralement un aspect sérieux voire professionnel à un document. Il existe plusieurs ouvrages traitant de la typographie française, je citerais ici le lexique de l'imprimerie nationale [8] et le manuel d'Yves Peyrousseaux [9].

Ce chapitre contient des informations sommaires sur la manière dont sont codées les fontes dans LaTeX pour obtenir les accents de la langue française. Suivent quelques règles de typographie et une présentation du package babel permettant de simplifier la saisie de documents en français. Ce chapitre se termine sur une présentation d'une classe de document `lettre` ayant pour but de composer des lettres et des fax.

7.1 Le problème des lettres accentuées

Il y a quelques années, lorsque TeX a été conçu, les fontes utilisées ne comportaient pas de lettres accentuées. Chacune de ces fontes était codée à l'aide de 7 bits par caractère, et donc contenaient quelques 128 caractères codables. Provenant des États-Unis, ces 128 caractères ne comportaient évidemment pas les caractères accentués de la langue française. C'est la raison pour laquelle, pendant un long moment de valeureux utilisateurs francophones de TeX et de LaTeX étaient contraints de saisir leur document en `fran\c{c}ais` avec des `caract{\'e}res` assez `p{\'e}nibles {\`a}` taper.

Aujourd'hui, ces petits désagréments ne sont plus qu'un mauvais souvenir. Depuis 1990 un codage des fontes tenant compte de caractères accentués de plusieurs langues a été adopté et porte le nom de *Cork encoding* ou de *codage T1*. Le tableau 7.1a page suivante donne à titre indicatif les caractères correspondants à chaque valeur. Dans ces tableaux, les cases sont numérotées à partir de 0, les valeurs augmentent de droite à gauche et de haut en bas.

`	´	^	~	¨	˝	°	ˇ	˘	¯	˙	¸	˛	,	‹	›
“	”	„	«	»	–	—		0	ı	j	ff	fi	fl	ffi	ffl
␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	-
Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§
ă	ą	ć	č	ď	ě	ę	ğ	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
ř	ś	š	ş	ť	ţ	ű	ů	ÿ	ź	ž	ż	ij	¡	¿	£
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	œ	ø	ù	ú	û	ü	ý	þ	ß

(a) Codage Cork (T1)

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL	VT	NP	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SP	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

(b) Caractères iso-latin1

TAB. 7.1 – Codage des fontes T1 et codage de caractères Iso-latin1. Le tableau du bas montre un codage de caractères (incluant les lettres accentuées de plusieurs langues européennes) aujourd'hui très répandu : le codage iso-latin1.

On peut remarquer que le codage des fontes est différent de celui des caractères ; le tableau 7.1b montre le codage dit *iso-latin1* qui fait maintenant office de standard pour le codage des caractères de la plupart des langues européennes. Les packages LaTeX contiennent donc une opération de « traduction » du codage des caractères (par ex. iso-latin1) en codage des fontes (par ex: codage T1).

7.2 Rédiger un document en français avec LaTeX

Il existe deux packages LaTeX permettant de « franciser » un document : le package french et le package babel. Pour des raisons tout à fait partisanes, nous nous intéresserons au deuxième. On active le package babel comme suit :

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

Cet ordre dans le préambule met en route cinq fonctionnalités qui sont :

Césure : babel gère la césure des paragraphes en tenant compte de la langue française,[1] et plus particulièrement des mots accentués du français ;

Typographie : les règles de typographie française sont appliquées notamment en ce qui concerne les guillemets et les signes de ponctuation ;

Mise en page : il s'agit essentiellement de réintroduire l'indentation du début de paragraphe qui suit un titre de section,[2] changement du symbole et d'espacement pour les environnements de types listes, ...

Traduction : tous les mots susceptibles d'être traduits (« Chapitre », « Table des matières », etc.) sont traduits en français ;

Macros : un ensemble de macros sont disponibles avec le package babel, ces macros permettent de saisir correctement certaines constructions courantes en français, telles que n° , 1er, 2° , 37° C, ...

7.3 Le package babel et la typographie

L'ensemble des « règles » liées à la typographie française dépasse largement le cadre de ce chapitre. Heureusement le package babel permet pratiquement de les utiliser sans les connaître. Il suffit simplement de respecter quelques règles de *saisie* du document LaTeX pour que la composition respecte les règles de typographie les plus courantes. Ainsi, à titre d'exemple, babel insérera un quart de cadratin insécable avant le point virgule ; ce qui est une pratique courante en typographie française.

Si cette insertion automatique ne vous convenait pas, il est possible d'appeler la commande `\NoAutoSpaceBeforeFDP`. Vous serez alors responsable de l'insertion ou non de l'espace avant les signes de ponctuations.

7.3.1 Ponctuation

Les règles à connaître pour la ponctuation peuvent se résumer aux deux propositions suivantes :

[1] On ne coupe pas de la même manière les mots anglais et les mots français.
[2] Ce qui n'est pas le cas en typographie anglaise.

1. un espace doit apparaître avant et après tous les signes de ponctuation *doubles*, c'est-à-dire les signes ; : ! ? « et »
2. on saisit un espace après (et pas avant) les signes de ponctuation *simples*, c'est-à-dire les signes . , (et)

Le respect de cette saisie permet à babel d'insérer les espaces nécessaires avant et après les signes de ponctuation. À ce sujet, il est intéressant de remarquer que les espaces avant les points d'interrogation et d'exclamation sont des espaces fines :

```
fouilla ! et \selectlanguage{english}
fouilla !\selectlanguage{french}
```

7.1 fouilla ! et fouilla !

7.3.2 L-a, e dans l'a, t-i, t-i, a !

Je cite Serge GAINSBOURG en guise de titre de ce paragraphe sur les deux « jolies » ligatures de la langue française : 'æ' et 'œ'. Au sujet de la saisie de ces ligatures, on peut au choix : saisir `\oe` et `\ae` (`\AE` et `\OE` en majuscules) :

```
L\ae titia va au Sacré-C\oe ur.
```

7.2 Lætitia va au Sacré-Cœur.

Ou saisir directement 'æ' sur le clavier s'il le permet. À titre indicatif, `AltGr+a` donne l'e dans l'a sur un système Linux digne de ce nom. Et pour une histoire compliquée l'e dans l'o ne se trouvant pas dans la norme iso-latin1, mon clavier n'est pas en mesure de fournir la ligature 'œ'.

7.3.3 Outils du package babel

Un grand nombre de « petites choses » restent toujours très floues quant à la manière correcte de les « typographier ». Je pense à toutes ces abréviations courantes telles que : Monsieur, Madame, premier, deuxième, primo, etc. Heureusement le package babel répond à certaines de nos interrogations.

`1\ier`	1^{er}
`3\ieme`	3^{e}
`37\degres{} C`	37° C
`\primo`, `\secundo`, `\tertio`, `\quarto`	1^{o} , 2^{o} , 3^{o} , 4^{o}
`\no 4`	n^{o} 4
`\No 4`	N^{o} 4

Lettrine

ON TROUVE dans certains documents des *lettrines* comme celle en début de ce paragraphe. Le package french de Bernard GAULLE et le package lettrine définissent une telle commande. Nous vous proposons dans la deuxième partie de ce document un exemple de code LaTeX permettant de générer une telle commande.

Sommaire

Dans un document français, on insère généralement la table des matières en fin de document et le sommaire, qui est une table des matières résumée, en début de

document. Le package french propose la commande `\sommaire` qui permet — comme son nom l'indique — un sommaire dans le document. Encore une fois, nous vous proposons dans la deuxième partie de ce document d'étudier une manière de générer un tel sommaire.

7.3.4 Recommandations d'usage

Les recommandations qui suivent ne sont pas à proprement parler des fonctionnalités du package babel ; ces recommandations sont des conseils que vous pourrez retrouver dans des ouvrages ayant trait à la typographie :

- les guillemets à la française se saisissent soit avec « et » si votre clavier permet de les générer, soit avec les signes inférieurs et supérieurs : `<<` et `>>`, soit avec les commandes `\og` et `\fg`. Les guillemets "à l'anglaise" se saisissent avec les quote et backquote : ` `` ` et `''` ; dans tous les cas, utiliser `"` pour les guillemets n'est pas recommandé ;
- les locutions latines se saisissent *a priori* en italique ;
- on abrège :

et cætera	avec	`etc.`	etc. et non pas « etc... »
Monsieur	avec	`M.`[3]	M. Machin
Messieurs	avec	`MM.`	MM. Machin et Bidule
Madame	avec	`M\up{me}`	M$^{\text{me}}$ Machin
Mademoiselle	avec	`M\up{lle}`	M$^{\text{lle}}$ Machin
kilomètre(s)	avec	`km`	25 km (pas de 's')
kilogramme(s)	avec	`kg`	25 kg

- le séparateur de partie décimale et partie entière est la *virgule* en français, et le point en anglais. On « doit » donc écrire : $123,54$.
- on insère un quart de cadratin tous les milliers, et millièmes :

```
\nombre{12345678,23434}
```

7.3

12 345 678,234 34

- il est d'usage d'écrire les noms propres en petites capitales, comme ceci : John Coltrane. Ici on a utilisé la commande `\textsc{Coltrane}` ; le package babel contient la macro `\bsc` : `\bsc{COLTRANE}`, `\bsc{Coltrane}` et `\bsc{coltrane}` donnent le résultat escompté ;
- on écrit les sigles sans point et en lettre capitales (RATP, SNCF, ENISE). Certains sigles qui « se prononcent bien » peuvent même s'écrire en minuscules : Assedic, Inserm, etc.

7.3.5 Le cas de l'euro

Le symbole de l'euro peut être produit à l'aide de la commande `\texteuro` du package textcomp. On obtient alors le caractère : € ou € en utilisant la fonte sans sérif. Une autre approche consiste à utiliser la package eurosym fournissant les commandes :

- `\euro{}` : €
- `\EUR{35}` : 35 €

[3] Et non pas « Mr » qui est l'abréviation anglaise de *mister*.

7.3.6 Au sujet des majuscules

En dehors des cas bien connus où l'on doit mettre ou ne pas mettre de majuscule (il faut en mettre en début de phrase, ne pas en mettre pour commencer une parenthèse, selon le contexte après les deux points, etc.), voici trois points importants au sujet des majuscules (capitales comme disent les typographes).

Tout d'abord **les majuscules doivent être accentuées** (je ne m'énerve pas, j'explique) lire à ce sujet ce que dit Yves Perrousseaux dans son manuel. Il y est expliqué que les accents présents sur les majuscules depuis le XVI^{e} siècle ont disparu avec l'arrivée des machines à écrire et de composition typographique d'origine anglo-saxonne. On peut également trouver dans tous les bons ouvrages de typographie des exemples de phrases ambiguës lorsque les accents ne sont pas mis.

Ensuite, *dans un titre*, on ne mettra une majuscule qu'à la première lettre (contrairement à l'anglais où on met une majuscule à chaque mot). Enfin, il faut insister sur le fait que l'usage des majuscules est un domaine dont les nuances sont assez subtiles à saisir. Notons ici quelques points pour appréhender ces « règles » :

- on écrit maître de conférences (donc sans majuscule) ;
- l'université Jean Monnet (pas de majuscule à université) ;
- mais l'Université lorsqu'on parle de la structure en tant qu'entité propre ;
- le ministre de l'Intérieur ;
- l'académie de Lyon ;
- l'Assemblée nationale et le Sénat parce que ce sont des organismes uniques ;
- les Espagnols (pour le peuple) et le français (pour la langue)

Je ne résiste pas à l'envie de citer Jacques André :

> *« [...] Voici typiquement le genre de phrase que l'on trouve dans notre rapport d'activité :*
>
> > *Jean Transent, Maître de Conférence en Analyse de Données à l'Université de Nancy(Bien connue de la Communauté Scientifique Internationale) a donné, lors du séminaire de Biologie Informatique de Mardi 23 Juin, une conférence sur les Applications de l'Intelligence Artificielle à l'emploi de la Télévision Haute Définition en Robotique Avancée.*
>
> *Dans cette phrase, il y a 23 majuscules. Il ne devrait y en avoir que trois (Jean, Transent et Nancy). si si... »*
>
> Jacques André [10]

et Yves Perrouseaux :

> *« Les dénominations d'une dignité, d'une charge, d'un grade ou d'une fonction* ***sont des noms communs*** *:*
>
> *...*
> – *le président du conseil général, etc.* *C'est un nom commun, au même titre que le concierge ou les femmes de ménage du conseil général.*
>
> *»*
>
> Yves Perrouseaux [9]

7.4 Courrier et fax

Le noyau de LaTeX comprend une classe de document pour rédiger des lettres. Cependant cette classe n'est pas très souple et mal adaptée au français.[4] Pour les lettres françaises, nous conseillons l'utilisation de la classe `lettre` de Denis Mégévand de l'observatoire de Genève. La classe et sa documentation peuvent se trouver à : `ftp://obsftp.unige.ch/pub/tex/macros/lettre` et dans le paquet `tetex-frogg` de la distribution Debian Sarge.

7.4.1 Commandes disponibles

Voici quelques unes des entités que l'on peut définir dans la classe lettre :

Adresse de l'expéditeur en utilisant la commande `\address` ;

Ville originaire `\lieu` permet d'écrire en haut à droite, l'endroit d'où l'on écrit la lettre ;

Téléphone et fax sont précisés avec les commandes `\telephone` et `\fax` respectivement ;

Signature à l'aide de la commande `\signature` ;

Objet de la lettre avec la commande `\conc` (pour concernant) ;

Pièces jointes grâce à la commande `\encl` (de l'anglais *enclosed*)

7.4.2 Structure d'un document basé sur la classe `lettre`

Nous donnons à la figure 7.1 page suivante l'« ossature » d'un document LaTeX basé sur la classe `lettre`. Les commandes `\opening` et `\closing` sont obligatoires et ont pour respectivement pour objet d'introduire les formules de politesse de début et de fin de lettre.

7.4.3 Fichiers « instituts »

La classe `lettre` est livrée avec un fichier `default.ins` qui définit par défaut l'adresse de l'observatoire de Genève. L'administrateur du système LaTeX que vous utilisez devra donc adapter ce fichier à votre organisation.

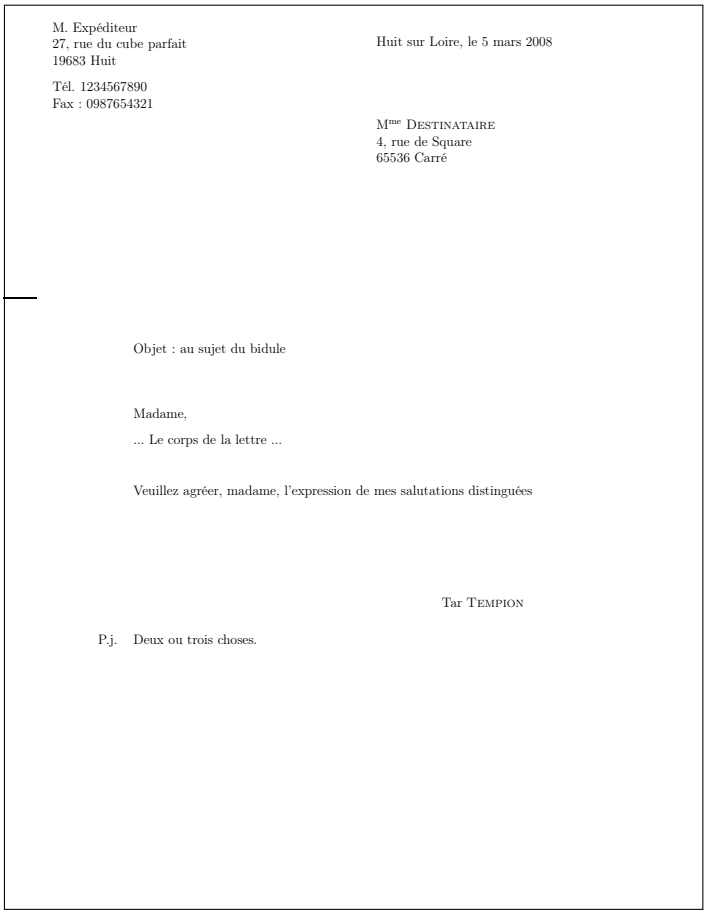
On peut cependant définir son propre fichier « institut » et l'inclure dans ses lettres. Lorsqu'on veut envoyer des lettres à titre personnel,[5] il est en effet plus logique d'utiliser ses propres coordonnées ; on pourra alors définir un fichier nommé `moi.ins` contenant par exemple :

```
\address{%
  M. Expéditeur\\
  27, rue du cube parfait\\
  19683 Huit}
\lieu{Huit sur Loire}
\telephone{1234567890}
\fax{0987654321}
\signature{Tar \textsc{Tempion}}
```

[4] Appréciation personnelle sur la version 1.2z de la classe `letter` du 9 février 1999.

[5] Ce qui n'a pas véritablement lieu d'être puisque vous êtes tout de même au boulot pour bosser et non pas pour envoyer du courrier personnel.

```
\documentclass[12pt]{lettre}
\usepackage[francais]{babel}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}

\begin{document}

\begin{letter}{%
    M\up{me} \textsc{Destinataire}\\
    4, rue de Square\\
    65536 Carré
}
\address{%
  M. Expéditeur\\
  27, rue du cube parfait\\
  19683 Huit}

\lieu{Huit sur Loire}
\telephone{1234567890}
\fax{0987654321}
\signature{Tar \textsc{Tempion}}

\conc{au sujet du bidule}

\opening{Madame,}

... Le corps de la lettre ...

\closing{Veuillez agréer, madame,
  l'expression de mes salutations
  distinguées}
\encl{Deux ou trois choses.}
\end{letter}
\end{document}
```


M. Expéditeur
27, rue du cube parfait
19683 Huit

Tél. 1234567890
Fax : 0987654321

Huit sur Loire, le 5 mars 2008

Mme Destinataire
4, rue de Square
65536 Carré

Objet : au sujet du bidule

Madame,

... Le corps de la lettre ...

Veuillez agréer, madame, l'expression de mes salutations distinguées

Tar Tempion

P.j. Deux ou trois choses.

FIG. 7.1 – Ossature d'un document basé sur la classe `lettre`.

Il suffit alors d'insérer dans le préambule du document, la commande `\institut` qui cherche un fichier portant l'extension `.ins` :

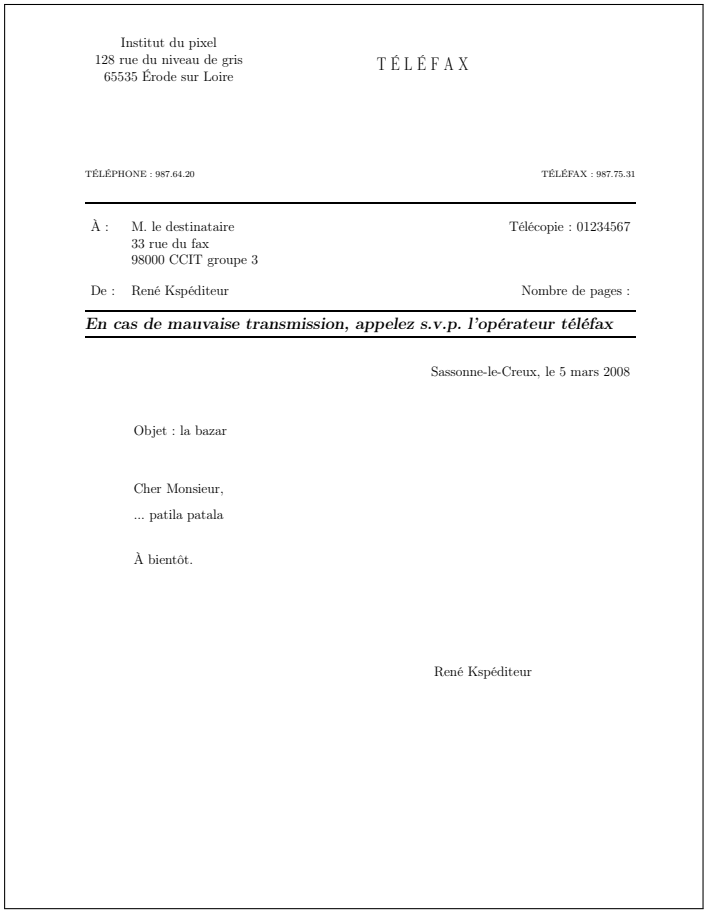
```
\institut{moi}
```

7.4.4 Fax

La classe lettre contient également un environnement pour préparer un fax avec un en-tête correspondant à votre organisation. Le principe général et les mots clés sont les mêmes à condition d'utiliser l'environnement `telefax` en lieu et place de l'environnement `letter`.

Notez qu'on peut ici aussi utiliser les fichier « instituts ». Enfin la commande `\addpages` permet de gérer le cas où vous joignez un document déjà imprimé à votre fax. Par exemple si vous avez à envoyer n pages à votre fax initial, il faudra ajouter la commande `\addpages{n}`. La figure 7.2 page ci-contre montre le document minimal pour créer un fax.

```
\documentclass[12pt]{lettre}
\usepackage[latin1]{inputenc}
\usepackage[french]{babel}
\usepackage[T1]{fontenc}

\begin{document}


\begin{telefax}{01234567}{% numéro de fax
    M. le destinataire\\
    33 rue du fax\\
    98000 CCIT groupe 3}

\address{\centering
  Institut du pixel\\
  128 rue du niveau de gris\\
  65535 Érode sur Loire}

\name{René Kspéditeur}

\conc{la bazar}

\opening{Cher Monsieur,}

... patila patala

\closing{À bientôt.}

\end{telefax}
\end{document}
```



Institut du pixel
128 rue du niveau de gris
65535 Érode sur Loire

T É L É F A X

TÉLÉPHONE : 987.64.20 TÉLÉFAX : 987.75.31

À : M. le destinataire
33 rue du fax
98000 CCIT groupe 3

Télécopie : 01234567

De : René Kspéditeur

Nombre de pages :

En cas de mauvaise transmission, appelez s.v.p. l'opérateur téléfax

Sassonne-le-Creux, le 5 mars 2008

Objet : la bazar

Cher Monsieur,

... patila patala

À bientôt.

René Kspéditeur

FIG. 7.2 – Ossature d'un document « fax »

8 À vous de jouer !

Sommaire

Tu ne coucheras pas avec un homme
comme on couche avec une femme.
C'est une abomination.

Le Lévitique Lv **18** 22.

S'il est vrai que LaTeX permet de faire à peu près tout ce que l'on veut, il est souvent difficile de savoir comment le lui demander. Nous tenterons ici de vous donner quelques points d'entrée pour chercher plus de documentation sur le monstre.

8.1 Livres et autres manuels

La documentation « standard » concernant TeX et LaTeX est constituée des ouvrages suivants :

- *LaTeX: A Document Preparation System* de L. Lamport [2] ;
- *The LaTeX Companion* [4] de M. Goossens, F. Mittelbach et A. Samarin
- *The LaTeX Graphics Companion* [11] des mêmes auteurs ;
- *The TeXbook* [3] de Knuth ;
- La FAQ française de LaTeX disponible à l'url suivante :
  http://www.grappa.univ-lille3.fr/FAQ-LaTeX.
  On y trouve aujourd'hui environ 70 questions répondant essentiellement à des problèmes classiques de mise en page. Ce document est actuellement maintenu par B. Bayard

Deux ouvrages d'initiation en français sont parus récemment:

- *LaTeX* de D. Bitouzé et J.C. Charpentier [12] ;
- *LaTeX pour l'impatient* de C. Chevalier *et al.* [13].

Les documents en ligne référencés par le LaTeX navigator (cf. plus bas) traitant de LaTeX 2ε en français, sont les suivants :

- *Apprends LaTeX !* de M. Baudoin ;
- *Joli manuel pour LaTeX 2ε* de B. Bayart
- *Aide mémoire pour LaTeX* de C. Willems et F. Geraerds ;
- *Guide d'introduction français au traitement de texte LaTeX* écrit par F. Geraerds ;
- *Une courte (?) introduction a LaTeX 2ε* de T. Oetiker, H. Partl, I. Hyna, E. Schlegl, traduit de l'allemand.

Ces documents sont disponibles au format `dvi` et/ou PostScript et parfois `pdf`.

8.2 Local

Avant de vous jeter sur ce pauvre réseau international, sachez que si vous avez la chance d'utiliser la distribution teTEX, vous trouverez sous l'arborescence TEX (`/usr/share/texmf/doc`[1]), un répertoire `doc` contenant un certain nombre de documentations intéressantes :

- `latex` : la doc de toutes les extensions installées au format `dvi` ;
- `fonts` : des docs sur les fontes disponibles ;
- ...

D'autre part, vous trouverez dans le menu aide d'Emacs un guide de référence au format info très pratique ; avec notamment la syntaxe de toutes les commandes et environnements standard de LATEX.

Enfin, n'hésitez pas à utiliser les commandes de recherche de fichiers disponibles sur votre système pour essayer de trouver des informations sur un package ou une fonte à partir de son nom.

8.3 EffTépé, Ouèbe et niouses

Comme pour la plupart des logiciels gratuits, on trouve une foultitude d'informations sur internet.

8.3.1 Sites FTP

Il a été créé une archive *standard* pour TEX et Cie portant le doux nom de CTAN pour Comprehensive TeX Archive Network. Cette archive est copiée sur plusieurs sites français, donc inutile d'aller encombrer les quelques lignes qui traversent l'océan, vous trouverez votre bonheur dans :

- `ftp://ftp.lip6.fr/pub/TeX/CTAN` à ma connaissance un serveur français très complet, avec des « miroirs » de beaucoup de sites.
- `ftp://ftp.inria.fr/pub/TeX/CTAN` une autre réplique CTAN.

L'arborescence du CTAN est la même quel que soit le site, et on notera entre autres :

- `macros/latex` la racine de LATEX et ses extensions,
- `graphics` la racine de ce qui touche de près ou de loin au graphisme,
- `support` les logiciels qui tournent autour de LATEX : correcteurs orthographiques, convertisseurs, éditeurs,...
- ...

8.3.2 Sites Web

Encore une fois, ne cherchez pas plus loin :

`http://tex.loria.fr`

Même si ce site ne semble plus être maintenu, il contient de très nombreux liens vers des documents de référence. On pourra également se référer au site :

`http://www.latex-project.org`

pour avoir des infos sur les modifications du format LATEX 2_ε et de l'avancement du projet LATEX3. Notez toutefois que ces sites sont en langue anglaise.

[1] Ou quelque chose d'approchant suivant l'installation effectuée.

8.3.3 Les newsgroups

Il existent deux groupes de discussion sur TEX et LATEX :

- `comp.text.tex` environ 150 messages par jour ! Mais on peut y apprendre des choses, et
- `fr.comp.text.tex` beaucoup moins de transit et en français.

Ces groupes de discussion constituent une source d'information extraordinaire si on veut bien faire l'effort de trier un peu les messages parfois un peu trop nombreux. Vous pouvez *en dernier recours* — c.-à-d., après vous être documenté — poser une question sur le groupe. Si vous êtes clair et concis, la réponse ne se fait en général guère attendre.

*
* *

À vous de jouer !

II

Tout ce que vous avez toujours voulu savoir sur (Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander) sans jamais oser le demander

Introduction

Que tes pieds sont beaux dans ta chaussure, fille de prince!
Les contours de ta hanche sont comme des colliers, Œuvre des mains d'un artiste.
Ton sein est une coupe arrondie, Où le vin parfumé ne manque pas
Ton corps est un tas de froment, Entouré de lis.[2]

Le Cantique des cantiques Ct **7** 2.

Cette partie s'intitule « Tout ce que vous avez toujours voulu savoir sur (Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander) sans jamais oser le demander ». Elle a pour but d'expliquer comment les chapitres précédents ont été produits, et donc présente les différentes commandes et environnements qui ont été définis pour générer le manuel que vous avez sous les yeux. Mais son objectif est plus large puisque nous espérons fournir ici au courageux lecteur, une base solide pour la création de ses propres styles...

L'idée a germé dans mon esprit d'écrire les chapitres suivants après avoir eu plusieurs questions de lecteurs me demandant s'ils pouvaient réutiliser tel ou tel aspect du style de ce document. Le chantier que représente la rédaction des pages qui suivent a été pour moi titanesque dans la mesure où j'ai dû présenter des aspects de LaTeX qui ne font plus partie des connaissances de base et qui sont donc à ce titre plus difficiles à expliquer.[3] Enfin — et non des moindres — ce qui reste généralement de l'ordre du bazar privé a dû ici être « rationnalisé » pour être présentable. Ce qui n'a pas été une mince affaire.

J'ai voulu dans cette partie présenter la démarche que j'ai adoptée pour générer ce document. Je ne prétends pas qu'il s'agit du seul moyen possible pour obtenir la mise en page que vous avez sous les yeux. Par exemple, certaines parties de ce document auraient pu être produites à l'aide de paquets existants fournissant des fonctionnalités analogues ou même meilleures que celles des outils développés ici.

L'idée sous-jacente à cette partie est donc bien de guider l'utilisateur curieux vers des pistes d'exploration de LaTeX, de montrer comment on peut à l'aide de quelques outils, mettre au point des commandes originales correspondant exactement à ses propres besoins. Ces pistes sont suffisament générales pour être suivies telles quelles ou adaptées pour des cas similaires ou non. Il s'agit donc de découvrir les grands classiques des fonctionnalités internes de « LaTeX » et d'avoir la satisfaction — sans pour autant prôner la « ré-invention » de la roue — de créer ses propres outils

J'ai tenté, autant que possible, de présenter des commandes n'utilisant que LaTeX. Il a cependant parfois été nécessaire d'utiliser certaines des fonctionnalités de TeX ce qui a donné l'occasion de les présenter ici. Cette partie se compose donc de trois chapitres :

Outillage nécessaire qui présente les commandes à connaître permettant de s'équiper pour la suite. On y trouve par exemple quelques pistes sur la structure des fichiers d'une distribution LaTeX, des idées pour changer les fontes d'un docu-

[2] Les épigraphes de cette partie sont toutes tirées du Cantiques des cantiques et n'ont jamais de lien avec le titre du chapitre.

[3] D'autant plus, je l'écris en petit, que je ne les maîtrise pas du tout.

ment, et une présentation détaillée de la création de nouveaux environnements basés sur les listes ;

Cosmétique qui présente les outils qui ont été mis en œuvre pour changer l'allure des titres, des en-têtes et pieds de page, des marges, et quelques autres petites choses ;

De nouveaux jouets qui est l'occasion d'expliquer la création des onglets de ce manuel, du glossaire, des exemples, du sommaire, des lettrines, des notas, et de quelques autres bricoles.

Certaines explications données dans les chapitres qui suivent sont tellement fumeuses que même l'auteur ne les comprend pas. Certaines solutions apportées aux problèmes ne sont que partielles. Enfin, certaines choses restent mystérieuses pour votre serviteur ; dans ces situations, un panneau « dos d'âne » est inséré dans le paragraphe.

9 Outillage nécessaire

Sommaire

Que tu es belle, que tu es agréable,
Ô mon amour, au milieu des délices !
Ta taille ressemble au palmier,
Et tes seins à des grappes.

Le Cantique des cantiques Ct **7** 7.

Dans ce chapitre nous présentons les outils pré requis permettant de créer des commandes et des environnements plus complexes que ceux exposés au chapitre 4. Nous profitons d'ailleurs de cette introduction pour dire que le chapitre 4 auquel nous faisons référence ici doit être correctement digéré pour commencer la lecture de cette partie. Quelques mécanismes autour des fontes sont également présentés ainsi que quelques pistes pour fouiller dans les sources de LaTeX.

9.1 Hercule Poirot

9.1.1 Fouiller dans les fichiers

Tout d'abord, pour personnaliser un document écrit avec LaTeX, il est nécessaire de connaître la manière dont sont organisés les fichiers qui composent la distribution du système TeX/LaTeX que vous utilisez. Votre serviteur utilise la distribution TeXLive pour Unix (http://www.tug.org/texlive). Dans cette distribution on pourra dans un premier temps compulser les documentations des packages se trouvant dans le répertoire :

```
/usr/share/texmf-texlive/doc/latex/
```

Ce répertoire contient d'autres sous-répertoires, généralement un par package, et donc la documentation est sous la forme d'un fichier dvi ou PostScript. Dans certaines situations, il est nécessaire d'aller scruter le source des packages. Dans la distribution teTeX ces sources se trouvent dans :

```
/usr/share/texmf-texlive/tex/latex
```

et là aussi, on trouvera généralement un répertoire par package, contenant les sources dans un fichier au format texte, portant l'extension `.sty` et éventuellement des fichiers connexes. Enfin, pour comprendre le comportement par défaut de LaTeX, indépendamment des packages que l'on peut inclure, on pourra avoir recours au source de LaTeX dans :

```
/usr/share/texmf-texlive/tex/latex/base/latex.ltx
```

et aux sources des classes de documents dans :

```
/usr/share/texmf-texlive/tex/latex/base/book.cls
```

pour la classe `book`.

9.1.2 Examiner les macros

Un moyen pratique de trouver la définition d'une commande consiste à le demander à LaTeX lors d'une session interactive. On lance directement dans un terminal de commande du système d'exploitation :

```
latex
```

Mon système me répond froidement :

```
This is e-TeXk, Version 3.14159-2.1 (Web2C 7.4.5)
 %&-line parsing enabled.
**
```

À l'invite de ce prompt spartiate (`**`) qui est le cri du « TeX tout nu », je réponds bravement `&latex` pour demander à charger le format LaTeX. La réponse ne tarde pas :

```
**&latex
entering extended mode
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french loaded.
*
```

Notez que le prompt a perdu une étoile. À partir de maintenant on peut écrire un document LaTeX interactivement. Ce qui a certes peu d'intérêt dans l'absolu, mais peut s'avérer très utile pour obtenir la définition d'une commande avec la syntaxe. On pourra par exemple taper :

```
*\showcommande
```

pour avoir la définition de commande. Par exemple :

```
*\show\mbox
> \mbox=\long macro:
#1->\leavevmode \hbox {#1}.        ← c'est la définition
<*> \show\mbox
```

nous montre la définition de la commande `\mbox`. On remarque que cette commande lorsqu'elle est appelée, se transforme en un appel à `\leavevmode` et `\hbox`. Notre esprit de curiosité nous pousse donc à écrire :

```
*\show\hbox
> \hbox=\hbox.                    ← c'est une primitive
<*> \show\hbox
```

On constate ici que `\hbox` n'est pas définie à partir d'une autre commande. Il s'agit donc de ce que TEX appelle une primitive. L'exploration peut être poursuivie :

```
*\show\leavevmode
> \leavevmode=macro:
->\unhbox \voidb@x .            ← définition de \leavevmode
<*> \show\leavevmode
```

et ainsi de suite...

9.2 Outils de bas niveaux

9.2.1 Pour qui sont ces pourcents ?

Vous avez peut-être déjà remarqué que le code LATEX contient parfois le caractère % en fin de ligne. La présence de ce % s'explique par le fait qu'un saut de ligne dans le code insère un espace dans le texte. Ainsi la commande :

```
\newcommand{\beurk}{bidule}
```

peut s'écrire pour des raisons de lisibilité :

```
\newcommand{\beurk}{
  bidule
}
==(\beurk)==
```

9.1

==(bidule)==

On constate donc qu'il y a deux espaces non désirés autour du mot « bidule ». On peut éviter cela en écrivant :

```
\newcommand{\ahhh}{%
  bidule%
}
==(\ahhh)==
```

9.2

==(bidule)==

Il existe une autre situation où les espaces peuvent s'immiscer pernicieusement dans le texte. Définissons un environnement :

```
\newenvironment{hyperimportant}{%
  \bfseries\itshape}{%
  \upshape\mdseries}
```

```
Il est impératif
\begin{hyperimportant}
  de multiplier les sauvegardes
\end{hyperimportant}
de vos documents personnels
```

9.3

Il est impératif ***de multiplier les sauvegardes*** de vos documents personnels

Si vous regardez attentivement le texte produit, vous noterez qu'il y a deux espaces de chaque côté de la séquence mise en italique gras « ***de ... sauvegardes*** » :

– deux espaces avant « ***de*** » introduits par le saut de ligne à la fin de « `est impératif` » et celui à la fin de `\begin{hyperimportant}` ;
– deux espaces après « ***sauvegardes*** » induits par le saut de ligne à la fin de « `sauvegardes` » et par celui à la fin de `\end{hyperimportant}`.

La preuve, si on supprime ces sauts de ligne :

```
Il est impératif\begin{hyperimportant} de
multiplier  les
sauvegardes\end{hyperimportant} de vos
documents personnels
```

9.4

Il est impératif ***de multiplier les sauvegardes*** de vos documents personnels

Pour éviter d'avoir à se soucier de ce genre de problème on a généralement recours à deux commandes permettant de supprimer ces espaces doubles. On fait appel, pour éliminer ceux situés *avant* la séquence à `\ignorespaces` et pour ceux situés *après*, à la commande `\unskip`.

La commande `\ignorespaces`

Cette commande procède à l'expansion des commandes qui suivent en ignorant tous les espaces qui la suivent :

```
\newcommand{\truc}{ }\newcommand{\bidule}{ }

a\truc\bidule b\par
a\ignorespaces\truc\bidule b
```

9.5

a b
ab

Dans l'exemple ci-dessus, les commandes `\truc` et `\bidule` ont pour seul but de produire un espace lorsqu'elles seront appelées. Par conséquent, la ligne :

```
a\truc\bidule b
```

produira 'a␣{␣}b' c'est-à-dire les deux lettres a et b séparées par deux espaces. L'appel avec la commande `\ignorespaces` ignore — comme son nom l'indique — les deux espaces produits par les commandes `\truc` et `\bidule`. On peut donc utiliser cette commande dans notre exemple précédent :

```
\newenvironment{hyperimportant}{%
  \bfseries\itshape\ignorespaces}{\upshape\mdseries}
```

qui devrait supprimer un espace :

```
Il est impératif
\begin{hyperimportant}
  de multiplier les sauvegardes
\end{hyperimportant}
de vos documents personnels.
```

9.6

Il est impératif ***de multiplier les sauvegardes*** de vos documents personnels.

La commande `\unskip`

Si vous êtes attentif, vous noterez que deux espaces entre « ***sauvegardes*** » et « de » résistent à nos assauts. C'est là qu'intervient la primitive TEX `\unskip` qui enlève le dernier espace inséré :

```
\newcommand{\truc}{ }\newcommand{\bidule}{ }

a\truc\bidule b\par
a\truc\bidule\unskip b
```

9.7

a b
a b

Finalement la définition « correcte » de notre environnement est la suivante :

```
\newenvironment{hyperimportant}{%
  \bfseries\itshape\ignorespaces}{\unskip\upshape\mdseries}
```

qui devrait supprimer tous les espaces indésirables :

```
Il est impératif
\begin{hyperimportant}
  de multiplier les sauvegardes
\end{hyperimportant}
de vos documents personnels.
```

9.8

Il est impératif ***de multiplier les sauvegardes*** de vos documents personnels.

9.2.2 Le caractère @

Lorsque vous vous lancerez dans l'exploration des sources des packages vous remarquerez que le nom d'une grande partie des commandes qui y sont définies contient le caractère `@`. Or dans un document `.tex`, il n'est pas autorisé d'exécuter une commande dont le nom contient ce dernier. Ceci permet de protéger ou de limiter la portée des commandes des packages. Par exemple la commande `\cb@defpoint`, définie dans le package changebar, ne peut pas être appelée par un utilisateur du package. Pour redéfinir ces commandes internes, il est nécessaire d'effectuer la petite manipulation suivante :

```
\makeatletter
% ici on peut bidouiller
\renewcommand{\@ttention}{oulala...}
\makeatother
% ici on ne peut plus
```

La commande hypothétique `\@ttention` peut uniquement être manipulée si le caractère `@` est une lettre. C'est le rôle de la commande `\makeatletter` qui transforme le caractère `@` en une lettre comme les autres, tandis que `\makeatother` lui réaffecte sa fonction spéciale.

Cette manipulation n'est pas nécessaire dans les fichiers de styles inclus avec la commande `\usepackage` pour lesquels la lettre `@` peut être utilisé comme un caractère.

La manière dont TEX peut changer la catégorie des caractères est expliquée au chapitre suivant au paragraphe 10.5.1 page 142.

9.2.3 Le `\let` de TEX

Il est parfois utile de modifier une commande interne de LATEX pour ajouter une fonctionnalité à son comportement par défaut. Par exemple pour modifier la commande interne `\bidule`,[1] on peut procéder comme suit :

[1] Oui oui ça n'est pas une commande interne, mais un exemple idiot de nom de commande qui n'existe pas...

1. sauvegarder la commande grâce à l'instruction `\let` de TEX :

   ```
   \let\biduleORIG\bidule
   ```

2. redéfinir la commande `\bidule` en se basant sur la définition intiale :

   ```
   \renewcommand{\bidule}{quelque chose en plus\biduleORIG}
   ```

3. si nécessaire, revenir à la définition initiale grâce à :

   ```
   \let\bidule\biduleORIG
   ```

9.3 Structures de contrôle et tests

Les structures introduites par le package ifthen suivent la syntaxe :

```
\ifthenelse{ expression booléenne }
{ ... code LaTeX si vrai ... }
{ ... code LaTeX si faux ... }
```

et :

```
\whiledo{ expression booléenne }
{ ... code LaTeX tant que c'est vrai ... }
```

L'expression booléenne peut être constituée selon le contexte de différentes commandes du package ifthen, et parmi elles :

- l'expression nombre$_1$>nombre$_2$, l'expression nombre$_1$<nombre$_2$ ou l'expression nombre$_1$=nombre$_2$ permettant de comparer les deux valeurs numériques nombre$_1$ et nombre$_2$;
- `\equal{`C_1`}{`C_2`}` qui renvoie vrai ou faux selon que la chaîne de caractère C_1 est égale à la chaîne C_2 ;
- `\isodd{`nombre`}` qui renvoie vrai si le nombre est impair, faux sinon ;
- `\value{`compteur`}` qui renvoie la valeur d'un compteur sous la forme d'un nombre exploitable dans les conditions booléennes ;
- `\lengthtest{`test longueur`}` qui renvoie l'évaluation de test longueur, test contenant les opérateurs <, > ou = et des longueurs LaTeX comme opérandes.

On notera également qu'on pourra utiliser les connecteurs logiques `\OR`, `\AND` et `\NOT` qui jouent le rôle qu'on attend d'eux dans une expression booléenne. On peut grouper des expressions avec les opérateurs `\(` et `\)`.

9.3.1 Booléens et opérateurs associés

Le package ifthen propose à ses vaillants utilisateurs la possibilité de manipuler des booléens. On peut en déclarer un avec la commande `\newboolean` :

```
\newboolean{id booléen}
```

qui définit une « variable » booléenne identifiée par id booléen. On pourra ensuite lui affecter la valeur `true` ou `false` avec la commande `\setboolean` :

```
\setboolean{id booléen}{valeur}
```

Et bien sûr on pourra utiliser le booléen ainsi créé avec les structures de contrôle, par exemple comme ceci :

```
\ifthenelse{\boolean{id booléen}}
{  code LaTeX si id booléen vaut vrai }
{  code LaTeX s'il vaut faux               }
```

Il est bon de connaître la version TEX de ce qui précède. On trouve en effet dans les packages LATEX du code écrit en TEX, et en particulier l'utilisation de la structure de contrôle « Si Alors Sinon ». Voici un exemple pour définir un nouveau booléen avec monsieur TEX :

```
\newif\ifimprimantecouleur
```

On le positionne à faux avec :

```
\imprimantecouleurfalse
```

et à vrai avec :

```
\imprimantecouleurfalse
```

On peut ensuite exploiter ce booléen dans une structure « Si Alors Sinon » à la mode TEX comme suit :

```
\ifimprimantecouleur
... % code si on a une imprimante couleur
\else
... % code si c'est une imprimante noir et blanc
\fi
```

9.3.2 Exemples

On souhaite écrire une commande pour produire le développement de la fonction factorielle,[2] de manière à pouvoir écrire :

```
On peut exprimer la factorielle de 10
comme suit :
\begin{displaymath}
  10!=\itfactorielle{10}
\end{displaymath}
```

On peut exprimer la factorielle de 10 comme suit :

$$10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$$

Une façon de résoudre le problème est d'écrire une commande contenant une boucle `\whiledo` :

```
\newcommand{\itfactorielle}[1]{%
  \setcounter{cptfact}{#1}% on stocke l'argument dans un compteur
  \whiledo{\value{cptfact}>1}{% tant qu'il est > 1
    \thecptfact\times% on l'affiche avec un ×
    \addtocounter{cptfact}{-1}}% on décrémente le compteur
  1}% on affiche 1 à la fin
```

Il faudra bien sûr déclarer le compteur :

```
\newcounter{cptfact}
```

On notera que dans la condition booléenne de la boucle « TantQue », on fait appel à la commande `\value` pour comparer la valeur du compteur avec la valeur 1. Un peu plus tordu : on peut implémenter cette commande de manière récursive :

[2] Y en a qui n'ont pas grand chose à faire de leur journée...

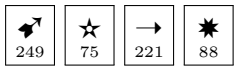
```
\newcommand{\recfactorielle}[1]{% version récursive :
\setcounter{cptfact}{#1}% on affecte l'argument au compteur
 \ifthenelse{#1>1}{% si cette valeur est  supérieure  à  1
   \thecptfact\times% on l'affiche  suivie  de  ×
   \addtocounter{cptfact}{-1}% on décrémente le compteur
   \recfactorielle{\thecptfact}}% on fait un appel récursif
 {1}}% sinon (valeur=1) on  affiche  1
```

Cette commande produit évidemment le même résultat que la précédente. On notera que dans la condition du `\ifthenelse` on compare un nombre (`#1`) avec un autre (1). Enfin on pourra remarquer que la présence de la commande `\times` impose le mode mathématique pour exécuter ces commandes. On peut contourner le problème, si nécessaire, avec la commande `\ensuremath`.

Le `\whiledo` et le `\ifthenelse` ont été utilisés dans le document que vous avez sous les yeux pour générer les tableaux de symboles à la page 209 et 210, ainsi que les tableaux sur le codage à la page 88 du chapitre sur les documents en francais. Nous avons tout d'abord créé une commande permettant d'afficher un symbole :

```
\affsymb{pzd}{249} \affsymb{pzd}{75}
\affsymb{pzd}{221} \affsymb{pzd}{88}
```

9.10



Cette commande est la suivante :

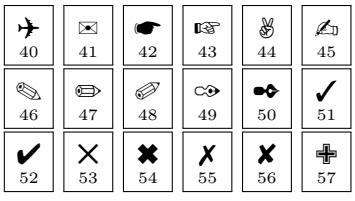
```
\newcommand{\affsymb}[2]{%
  \framebox{% un cadre
    \parbox[][16pt][b]{1em}{% autour d'une boite paragraphe
      \centering% de 16 pt de hauteur, 1em de large,
      \Pisymbol{#1}{#2}\\% dont le contenu centré
      \tiny#2}}}% est composé du symbole et de son numéro
```

L'argument `#1` est le nom de la police (`pzd` ou `psy`), et l'argument `#2` est le numéro de symbole. Sinon, rien de particulier dans cette commande, si vous avez suivi jusqu'ici (notamment en lisant le chapitre 4 et plus particulièrement le paragraphe 4.4 page 56)... Nous avons ensuite défini une commande permettant d'afficher un série de symboles :

```
Voici les symboles Zapf Dingbats, à partir
du \No 40,  sur 3 lignes et 6 colonnes :
\begin{center}
  \symboles[40]{pzd}{3}{6}
\end{center}
```

9.11

Voici les symboles Zapf Dingbats, à partir du N° 40, sur 3 lignes et 6 colonnes :



Voici le code de la commande `\symboles` :

```
\newcommand{\symboles}[4][0]{%
  \setcounter{clig}{0}% Mise à zéro des compteurs de ligne
  \setcounter{ccol}{0}% et de colonne
  \setcounter{cligmax}{#3}% arguments 3 et 4 pour fixer
  \setcounter{ccolmax}{#4}% le nombre max de colonnes et de lignes
  % Pour chaque ligne  :
```

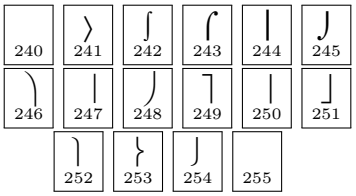
```
\whiledo{\value{clig}<\value{cligmax}}{%
  \setcounter{ccol}{0}% remise à zéro du compteur de colonne
  % et pour chaque colonne :
  \whiledo{\value{ccol}<\value{ccolmax}}{%
    % on calcule le numéro du symbole
    \setcounter{csym}{%
      \value{clig}*\value{ccolmax}+\value{ccol}+#1}
    % si sa valeur est inférieure à 256
    \ifthenelse{\value{csym}<256}{%
      \affsymb{#2}{\thecsym}}{% on l'affiche
      \mbox{}}% sinon on créé un boîte vide
    \stepcounter{ccol}}% on passe à la colonne suivante
  \stepcounter{clig}% on passe à la ligne suivante
  % on saute une ligne , sauf à la fin
  \ifthenelse{\value{clig}<\value{cligmax}}{\\}{}}}
```

Il faudra bien sûr déclarer les cinq compteurs avec la commande `\newcounter`.

```
Et je sais que vous êtes tout particulièrement
curieux de voir ce que fait cette commande
lorsque le compteur  dépasse les bornes :
\begin{center}
  \symboles[240]{psy}{3}{6}
\end{center}
```

9.12

Et je sais que vous êtes tout particulièrement curieux de voir ce que fait cette commande lorsque le compteur dépasse les bornes :



9.3.3 Tester la parité des pages

C'est une pratique courante — on le verra par la suite — de créer des commandes ayant un comportement différent selon la parité de la page. À l'entrée « Finding if you're on an odd or an even page » de la Faq anglaise [14] est expliqué que le naïf :

```
\ifthenelse{\isodd{\value{page}}}
{ ... la page est impaire ... }
{ ... la page est paire ... }
```

peut ne pas donner le résultat escompté. Le compteur de page lorsqu'il est examiné *à la frontière entre deux pages* peut ne pas être à jour : le compteur de page s'il est interrogé au début d'une page renverra le numéro de la page précédente... Ceci est dû à la manière dont TEX procède pour effectuer les sauts de page. Pour contourner ce problème plusieurs solutions sont possibles. Celle adoptée dans ce document consiste à utiliser le package chngpage qui insère artificiellement un label à l'endroit où l'on veut tester la parité de la page.

Par conséquent dans le cas où le test de parité à réaliser peut être évalué à la frontière entre deux pages, il faudra écrire :

```
\checkoddpage%
\ifcpoddpage
   ... la page est impaire ...
\else
```

```
    ... la page est paire...
\fi
```

9.4 Fontes

9.4.1 Le jeu des « trois » familles

Pour conserver une homogénéité dans l'allure des caractères dans un document LaTeX, sont définies trois familles :

1. la famille roman celle que vous êtes en train de lire ;
2. la famille sans sérif que vous êtes également en train de lire à l'instant même ;
3. et la famille machine à écrire, également appellée « typewriter » lorsqu'on est anglophone, que — cela ne vous aura sans doute pas échappé — `vous êtes en train de lire`.

Il est important de noter que ces trois familles de fontes sont par défaut trois familles de la police baptisée par son auteur (Knuth lui-même) « Computer Modern ». Elles sont conçues pour s'harmoniser au sein d'un même document. Dans cet ordre d'idée, il faudra toujours veiller à ce que ces trois familles (roman, sans sérif, et machine à écrire) soient visuellement « compatibles » entre elles. Les distributions de LaTeX proposent généralement des packages permettant d'utiliser les fontes PostScript dans un document, avec notamment le célèbre[3] package times utilisant :

1. Times pour la famille roman celle que vous êtes en train de lire ;
2. Helvetica pour la famille sans sérif que vous êtes également en train de lire à l'instant même ;
3. `Courrier pour la famille machine à écrire.`

De même le package newcent utilise :

1. **NewCentury pour la famille roman celle que vous êtes en train de lire** ;
2. AvantGarde pour la famille sans sérif que vous êtes également en train de lire à l'instant même ;
3. `Courrier pour la famille machine à écrire.`

9.4.2 Désignation des fontes et de leurs attributs

Une fonte[4] ou police de caractères est définie dans LaTeX par plusieurs caractéristiques dont il a été question au paragraphe 2.1 page 15. De manière à désigner la fonte à l'aide des commandes que nous allons découvrir dans ce paragraphe, on utilisera :

- un codage qui sera à quelques exceptions près le codage T1 ;
- une série de caractères identifiant la famille : `cmr` pour « computer modern roman », `ptm` pour « PostScript times », etc.
- une série de caractères désignant la « graisse » de la fonte, `m` pour « médium », `b` pour « bold » (gras), `bx` pour « bold extended » (gras étendu, c'est-à-dire gras avec des caractères plus large), etc.

[3] Mais obsolète. Aujourd'hui il est conseillé d'utiliser le package mathptmx

[4] Terme faisant référence au plomb de l'imprimerie...

– une série de caractères définissant l'allure (*shape* en anglais) de la fonte : `n` pour « normal », `it` pour « italique », `sl` pour « slanted » (penché), etc.

Fontes « computer modern »

Il s'agit d'un ensemble de fontes dessinées par Donald KNUTH. et utilisées par défaut dans LaTeX. Les commandes `\emph`, `\textbf`, etc. sélectionnent donc automatiquement ces polices.

Computer Modern roman (cmr)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	bx	n	gras étendu normal
machin Bidule Chouette chose	bx	it	gras étendu italique
machin Bidule Chouette chose	bx	sl	gras étendu penché
machin Bidule Chouette chose	b	n	gras normal

Computer Modern sans sérif (cmss)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	sl	penché
machin Bidule Chouette chose	bx	n	gras étendu normal
machin Bidule Chouette chose	sbc	n	semi gras condensé normal

Computer Modern typewriter (cmtt)	Codage T1		
`machin Bidule Chouette chose`	m	n	normal
`machin Bidule Chouette chose`	m	it	italique
`machin Bidule Chouette chose`	m	sl	penché
`MACHIN BIDULE CHOUETTE CHOSE`	m	sc	petites capitales

Computer Modern fibonacci (cmfib)	Codage T1		
machin Bidule Chouette chose	m	n	normal

Computer Modern funny roman (cmfr)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique

Computer Modern dunhil (cmdh)	Codage T1		
machin Bidule Chouette chose	m	n	normal

Fontes en béton

Elles ont été dessinées par KNUTH pour son ouvrage intitulé « Mathématiques concrètes ». Le package beton[5] permet de les charger dans un document.

[5] Notez ici le jeu de mots désopilant, *concrete* veut aussi dire « béton » en langue anglaise.

Concrete fonts (ccr)	Codage T1		
machin Bidule Chouette chose	m	n	normal
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	m	sl	penché
machin Bidule Chouette chose	m	it	italique

Fontes « gothiques »

Les fontes ci-dessous sont dites de la famille gothique et ne sont à utiliser que dans un contexte bien précis faute de rendre le texte parfaitement illisible, comme celui que vous êtes en train de lire actuellement, d'ailleurs vous avez probablement déjà arrêté, donc je peux dire des gros mots : caca…

Gothique (ygoth)	Codage U		
machin Bidule Chouette chose	m	n	

Fraktur (yfrak)	Codage U		
machin Bidule Chouette chose	m	n	

Schwabacher (yswab)	Codage U		
machin Bidule Chouette chose	m	n	

Fontes PostScript

Les fontes ci-dessous sont généralement disponibles gratuitement et résident la plupart du temps dans les imprimantes.

Times (ptm)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

Palatino (ppl)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

charter (bch)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

New century (pnc)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

Bookman (pbk)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	it	italique
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

Helvetica (phv)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras
machin Bidule Chouette chose	bc	n	gras condensé

Avantgarde (pag)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

Courier (pcr)	Codage T1		
machin Bidule Chouette chose	m	n	normal
machin Bidule Chouette chose	m	sl	penché
MACHIN BIDULE CHOUETTE CHOSE	m	sc	petites capitales
machin Bidule Chouette chose	b	n	gras

Zapf Chancery (pzc)	Codage T1		
machin Bidule Chouette chose	m	n	normal

9.4.3 Changer de fontes

Globalement

On peut changer de police de caractères en utilisant des packages plus ou moins standard de la distrubtion LaTeX :

▲ mathptmx : pour utiliser le « vilain » Times New Roman ;

▲ newcent : pour le New Century ;

▲ mathpazo : pour le Palatino ;

▲ ... : et d'autres, n'avez qu'à fouiller votre distribution...

Si l'on examine le contenu du fichier `newcent.sty` on trouve simplement :

```
\renewcommand{\rmdefault}{pnc}
\renewcommand{\sfdefault}{pag}
\renewcommand{\ttdefault}{pcr}
```

ce qui signifie que — comme expliqué au § 9.4.1 page 114 — on a redéfini les trois familles « roman », « sans sérif » et « machine à écrire » en les nommant par leur nom LaTeX standardisé : `pcn` pour PostScript NewCentruy, `pag` pour PostScript AvantGarde etc. Les noms standardisés sont donnés dans les tableaux de la section précédente.

Localement

Il est toujours possible de changer localement de fonte dans un texte en spécifiant les paramètres nécessaires :

```
{\fontfamily{cmfr}\selectfont
  On passe en ``Funny Roman'' et même qu'on
  peut faire de l'\emph{italique}...
  c'est dingue !} Et hop nous voila de nouveau
en \verb+\rmdefault+
```

9.13

On passe en "Funny Roman" et même qu'on peut faire de l'*italique*... c'est dingue ! Et hop nous voila de nouveau en `\rmdefault`

Les appels qu'il est possible de faire avant la commande `\selectfont` sont :

- `\fontencoding` pour le codage ;
- `\fontfamily` avec comme argument la famille (`cmr` pour Computer Modern, `ptm` pour PostScript Times, etc.) ;
- `\fontseries` pour préciser la graisse (argument `b` pour gras, `m` pour la graisse moyenne, etc.) ;
- `\fontshape` pour l'allure de la fonte (argument `n` pour normal, `sl` pour penché, etc.) ;
- `\fontsize` avec deux arguments : la taille des caractères et l'espace entre deux lignes consécutives.

Voici un autre exemple :

```
{\fontfamily{ppl}\fontseries{b}%
  \fontsize{2cm}{2.5cm}\selectfont
  gros !}

Et nous voila de nouveau en \verb+\rmdefault+
```

9.14

gros !

Et nous voila de nouveau en `\rmdefault`

Finalement pour utiliser à plusieurs endroits dans un document, une fonte dont tous les attributs sont fixes, on pourra avoir recours à la commande `\DeclareFixedFont` prenant six arguments (nom, codage, famille, graisse, allure, taille) et permettant d'être ensuite utilisé comme une déclaration :

```
\DeclareFixedFont{\toupiti}{T1}{pag}{m}{n}{3pt}
Avant {\toupiti bon bé là à moins d'avoir une
bonne loupe vous ne serez pas capable de lire
ce texte} après.
```

9.15

Avant bon bé là à moins d'avoir une bonne loupe vous ne serez pas capable de lire ce texte après.

9.5 Listes et nouveaux environnements

À plusieurs reprises dans ce document, nous avons eu recours à l'environnement `list` permettant de créer des environnements basés sur le principe des listes (numérotés, de descriptions, etc.). Nous donnons ici les bases nécessaires pour pouvoir utiliser cet environnement en s'appuyant sur des exemples.

9.5.1 Principe

Pour définir un environnement basé sur les listes, on utilisera la syntaxe suivante :

```
\newenvironment{maliste}%
{\begin{list}%
    { ... code pour l'item par défaut ... }
    { ... caractéristiques de la liste ... }
}%
{\end{list}}
```

L'environnement `list` prend donc *deux* arguments. Le premier permet de définir l'allure de l'étiquette (ou item) par défaut. Le second permet de définir la liste elle-même et en particulier :

Sa géométrie : les marges, les espaces entre les paragraphes composant la liste, les espaces entre la liste et l'environnement dans lequel elle est insérée, etc.

la production de l'étiquette : c'est-à-dire la manière dont on va effectivement produire le titre de chaque entrée de la liste.

La liste suivante tente d'illustrer les différentes dimensions que l'on peut modifier pour définir sa propre liste :

Horizontales: (1) la dimension `\itemindent` (1) permet d'introduire une indentation pour le premier paragraphe d'une entrée de liste.

(3)

(4) la marge de gauche peut être est définie par la dimension `\leftmargin` (4) et celle de droite par `\rightmargin` ;

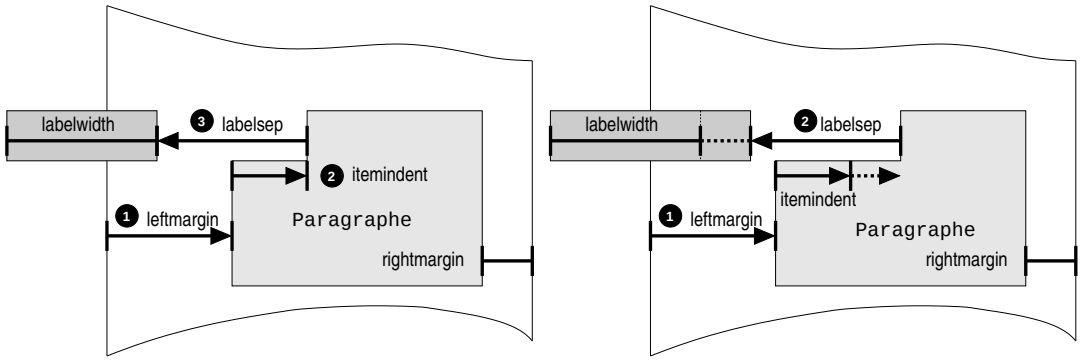
Étiquette: (2) la dimension `\labelsep` (2) détermine la dimension séparant l'étiquette du début du paragraphe. La dimension (3) `\labelwidth` définit quant à elle la largeur de la boîte contenant l'étiquette.

(3)

(5) Si on commence un nouveau paragraphe dans une entrée de liste, ce paragraphe est indenté de `\listparindent` (5) qui vaut 0 point par défaut.

Remarque « assez » importante: (2) si la largeur du texte de l'étiquette est inférieure à `\labelwidth` alors ce texte est inséré dans une boîte de largeur `\labelwidth`. Dans le cas contraire, comme ici, le texte de l'étiquette sera inséré dans une boîte de la largeur nécessaire et le paragraphe sera indenté en conséquence.

La commande `\makelabel` prenant un argument, permet de produire l'étiquette. Ainsi lorsqu'on entre la commande `\item[texte étiquette]`, il est fait appel à la commande `\makelabel{texte étiquette}`.



(a) Cas standard (b) Largeur supérieure à `\labelwidth`

FIG. 9.1 – Positionnement de l'entrée de liste

9.5.2 Réglage de l'étiquette

Pour comprendre le fonctionnement de l'environnement `list` et plus particulièrement le principe du positionnement relatif de l'étiquette et du paragraphe adjacent, on peut imaginer que les éléments sont positionnés dans l'ordre suivant :

1. le paragraphe est d'abord positionné par rapport à la marge de gauche à l'aide de la longueur `\leftmargin` ;
2. la première ligne du paragraphe est ensuite indenté à l'aide de la longueur `\itemindent` ;
3. puis l'étiquette est positionnée *relativement* au début du paragraphe ainsi indenté à l'aide de la longueur `\labelsep`.

Il découle de ceci que l'entrée de liste (ou étiquette) peut être produite dans la marge de gauche... La figure 9.1 illustre le positionnement de l'entrée de liste par rapport au paragraphe dans les deux situations suivantes :

- cas de la figure 9.1a où la largeur de l'entrée de liste est inférieure à la dimension `\labelwidth`. Dans ce cas l'entrée de liste est positionnée à une distance de `\labelsep` du paragraphe, lui même étant indenté de `\itemindent` et positionné par rapport à la marge gauche à l'aide de `\leftmargin` ;
- cas de la figure 9.1b : la largeur de l'entrée de liste est supérieure à la dimension `\labelwidth`. Dans ce cas l'entrée de liste est toujours positionnée à la distance `\labelsep` du paragraphe, mais celui-ci est indenté d'une valeur supérieure à `\itemindent`.

9.5.3 Réglages verticaux

On peut également régler les blancs verticaux dans l'environnement `list`. Ces paramètres permettent notamment de définir les espaces entre les paragraphes constituant les entrées de liste, mais également les blancs que l'on veut insérer avant ou après la liste. Il s'agit de :

- `\itemsep` : l'espace entre chaque entrée de liste ;
- `\parsep` : l'espace entre deux paragraphes à l'intérieur d'une entrée de liste ;
- `\topsep` : le blanc inséré avant et après l'environnement créé, auquel est ajouté `\partopsep` si celui-ci commence un nouveau paragraphe.

9.5.4 Valeurs par défaut

Tous les paramètres de l'environnement `list` ont des valeurs par défaut. Les longueurs pour les réglages horizontaux sont par défaut les suivantes sur le système de votre serviteur :

dimension	valeur par défaut
`\itemindent`	0pt
`\listparindent`	0pt
`\rightmargin`	0pt
`\leftmargin`	25pt
`\labelwidth`	20pt
`\labelsep`	5pt

Et pour les réglages verticaux :

dimension	valeur par défaut
`\itemsep`	4.0pt plus 2.0pt minus 1.0pt
`\parsep`	4.0pt plus 2.0pt minus 1.0pt
`\topsep`	8.0pt plus 2.0pt minus 4.0pt
`\partopsep`	2.0pt plus 1.0pt minus 1.0pt

La commande `\makelabel` est quant à elle définie par :

```
\hfil #1
```

par conséquent, dans la boîte de largeur `\labelwidth`, le contenu de l'étiquette est poussé à droite. Ainsi si on définit une liste simple avec :

```
\newenvironment{listebasique}
{\begin{list}{}{}}
{\end{list}}
```

On aura :

```
Avant avant avant avant avant avant avant
avant avant avant avant avant avant avant
\begin{listebasique}
  \item[X]  o o o o o o o o o o o o o o o o
    o o o o o o o o o o o o o o o o o o o o

    u u u u u u u u u u u u u u u u u u u u
  \item[Machin] v v v v v v v v v v v v v v
\end{listebasique}
Après après après après après après après
après après après après après après après
```

9.16

Avant avant avant avant avant avant avant avant avant avant avant avant avant avant

X o o o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o

u u u u u u u u u u u u u u u u u u u u

Machin v v v v v v v v v v v v v v

Après après après après après après après après après après après après après après

avec des étiquettes, et sans étiquette :

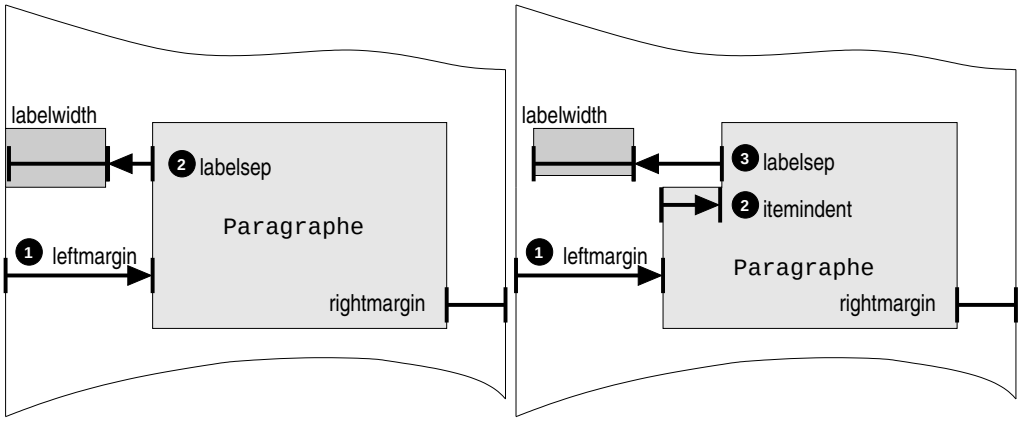
```
Avant avant avant avant avant avant avant
avant avant avant avant avant avant avant
\begin{listebasique}
  \item[] e e e e e e e e e e e e e e e e e
    e e e e e e e e e e e e e e e e e e e e
\end{listebasique}
Après après après après après après après
après après après après après après après
```

9.17

Avant avant avant avant avant avant avant avant avant avant avant avant avant avant

e e e e e e e e e e e e e e e e e e e e e e e
e e e e e e e e e e e e e e

Après après après après après après après après après après après après après après



(a) Liste « fichiers auxiliaires » (b) Liste « questions »

FIG. 9.2 – Positionnement des étiquettes dans les listes exemples.

9.5.5 Exemples

La liste décrivant les fichiers auxiliaires de LaTeX située à la page 196 a été produite avec le code suivant :

```
\begin{ficaux}
\item[tex] fichier source \LaTeX{} ; blabla
  blabla blabla blabla blabla blabla blabla
  blabla blabla blabla blabla
\item[aux] fichier auxiliaire ...
\item[log] le fichier de trace ...
\item[dvi]  fichier \emph{device
        independant},...
\end{ficaux}
```

tex fichier source LaTeX ; blabla blabla blabla blabla blabla blabla blabla blabla blabla blabla blabla blabla

aux fichier auxiliaire ...

log le fichier de trace ...

dvi fichier *device independant*,...

9.18

L'environnement `ficaux` a lui été conçu comme suit :

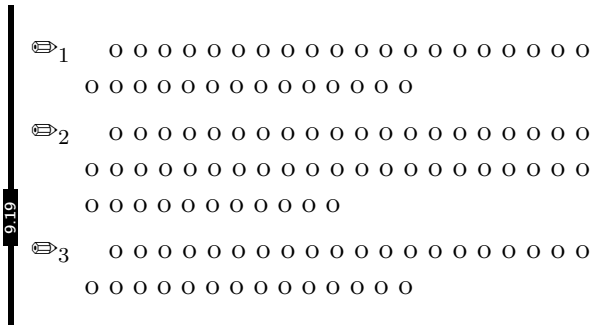
```
\newenvironment{ficaux}{%
  \begin{list}{}{%
     \setlength{\labelwidth}{1cm}% largeur de la boîte englobant le label
     \setlength{\labelsep}{8pt}% espace entre paragraphe et l'étiquette
     \setlength{\leftmargin}{\labelwidth+\labelsep}% marge de gauche
     \renewcommand{\makelabel}[1]{% production de l'étiquette :
       \framebox[\labelwidth]{\texttt{##1}}}}}{\end{list}}
```

Dans cet exemple la relation :

```
\leftmargin=\labelwidth+\labelsep
```

permet de positionner l'entrée de liste comme indiqué à la figure 9.2a.

Un autre exemple : la création d'un environnement de liste numérotée pour produire des questions dans un énoncé de travaux pratiques ou autre devoir surveillé...



```
\begin{question}
\item o o o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o
\item o o o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o
\item o o o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o
\end{question}
```

9.19

✏$_1$ o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o

✏$_2$ o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o

✏$_3$ o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o

Cet environnement a été produit par le code suivant :

```
\newenvironment{question}{\begin{list}{}{%
      \usecounter{cptquestion}%
      \setlength{\labelwidth}{2em}%
      \setlength{\labelsep}{1em}\setlength{\itemindent}{15pt}%
      \setlength{\leftmargin}{.8cm}\setlength{\rightmargin}{10pt}
      \renewcommand{\makelabel}[1]{\etiquettequestion{##1}}}}
  {\end{list}}
```

Le positionnement correspondant est montré à la figure 9.2b page précédente. On notera que dans la définition de la liste est fait usage de la commande `\usecounter` permettant de créer une liste numérotée et de préciser quel compteur est utilisé. On devra donc déclarer le compteur en question :

```
\newcounter{cptquestion}
```

Enfin, chaque entrée de liste composée du numéro de la question et d'un «joli» crayon est produite par la commande :

```
\newcommand{\etiquettequestion}[1]{%
  \makebox[\labelwidth]{\Pisymbol{pzd}{47}$_\thecptquestion$}}
```

Finalement la commande :

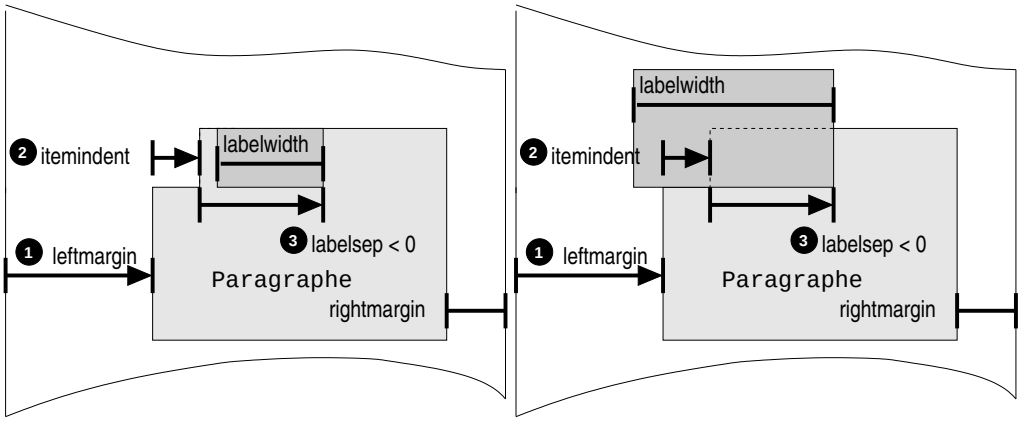
```
\renewcommand{\makelabel}[1]{\etiquettequestion{##1}}}}
```

redéfinit la commande `\makelabel` comme faisant appel à notre «joli» crayon. Le premier et unique argument est passé à `\etiquettequestion` à l'aide de l'expression `##1`, car `#1` désignerait, dans le contexte de la définition de environnement `question`, le premier argument de celui-ci.

Dans ce manuel, on trouve dans le mémento une liste de packages (page 195) produite par le code suivant :

```
\newenvironment{packages}{\begin{list}{}{%
      \setlength{\labelwidth}{2.5cm}%
      \setlength{\itemindent}{0pt}%
      \setlength{\leftmargin}{\labelwidth+\labelsep}%
      \renewcommand{\makelabel}[1]{%
        $\blacktriangle$ \ltxpack{##1} \hfil:}}}{\end{list}}
```

La commande `\ltxpack` est définie au paragraphe 11.1.2 page 154. Pour information, l'environnement ci-dessus donne :



(a) Utiliser un `\labelsep` négatif.

(b) Créer une boîte paragraphe pour superposer l'étiquette et le paragraphe.

FIG. 9.3 – Postionnement de l'étiquette dans la liste « glossaire ».

```
\begin{packages}
  \item[bidule] cette extension permet d'insérer
    des bidules dans son document sans avoir
    à savoir s'il s'agit de machin ou de truc.
\end{packages}
```

▲ bidule 9.20 : cette extension permet d'insérer des bidules dans son document sans avoir à savoir s'il s'agit de machin ou de truc.

9.5.6 Un exemple un peu plus tordu...

Nous allons détailler dans cette section la manière dont la liste composant le glossaire de la page 217 a été générée. Cette liste dont l'allure est donnée ci-dessous exploite deux idées :

1re idée

La longueur `\labelsep` peut être négative ce qui permet de superposer l'entrée de liste avec le paragraphe. Ceci est illustré à la figure 9.3a ;

Deuxième idée

On peut créer une boîte paragraphe pour la boîte produisant l'étiquette. La boîte ainsi créée pourra donc être composée de deux lignes : celle du haut contenant le texte de l'étiquette, celle du dessous étant vide, se superpose avec le paragraphe (figure 9.3b).

Le code permettant de générer la liste ci-dessus et celle du glossaire de ce manuel est donc :

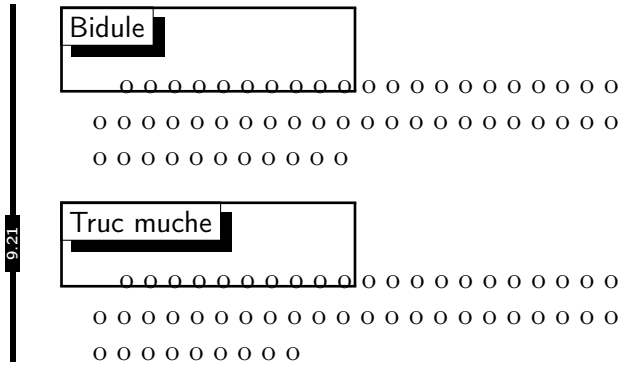
```
\newenvironment{glossaire}{\begin{list}{}{%
    \setlength{\labelwidth}{.5\textwidth}%
    \setlength{\labelsep}{-.8\labelwidth}%
    \setlength{\itemindent}{\parindent}%
    \setlength{\leftmargin}{25pt}%
    \setlength{\rightmargin}{0pt}%
    \setlength{\itemsep}{.8\baselineskip}%
    \renewcommand{\makelabel}[1]{\boiteentreeglossaire{##1}}}}
{\end{list}}
```

La valeur `.8\baselineskip` pour `\itemsep` permet d'aérer le glossaire en insérant des blancs entre chaque entrée. La commande permettant de générer la boîte contenant l'entrée de liste à superposer est :

```
\newcommand{\boiteentreeglossaire}[1]{%
  \parbox[b]{\labelwidth}{%
    \setlength{\fboxsep}{3pt}%
    \setlength{\fboxrule}{.4pt}%
    \shadowbox{\sffamily#1}\\\hfill\mbox{}}}
```

Pour comprendre comment est positionnée cette boite, nous avons quelque peu modifié la commande précédente pour dessiner un cadre autour :

```
\begin{leglossaire}
  \renewcommand{\boiteentreeglossaire}{%
    \fboiteentreeglossaire}%
\item[Bidule] o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o o o
\item[Truc muche] o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o o o
  o o o o o o o o o o o o o o o o o o o o
\end{leglossaire}
```



9.6 Des environnements qui mettent en boîte

L'environnement `lrbox` qui fait l'objet de ce paragraphe permet de stocker le « contenu » d'un environnement dans une boîte. C'est un outil très pratique qui est souvent une solution adaptée à des problèmes courants comme l'encadrement d'objets. Nous allons voir sur un exemple comment on peut utiliser cette construction.

9.6.1 Principe

On déclare une boîte comme expliqué au paragraphe 4.4.5 page 62 :

```
\newsavebox{maboite}
```

On peut ensuite écrire :

```
\begin{lrbox}{maboite}...contenu...\end{lrbox}
```

qui est parfaitement équivalent à :

```
\savebox{maboite}{...contenu...}
```

Bon mais alors si c'est la même chose, à quoi ça sert ?? Ça sert pour la *définition d'un environnement*.

9.6.2 Exemple

Supposons, par exemple l'existence d'une environnement :

```
\newenvironment{remarque}{%
  % clause begin
  \begin{center}\begin{minipage}{.8\textwidth}}{%
```

```
    % clause end
    \end{minipage}\end{center}}
```

Et :

```
On peut faire la remarque (édifiante) suivante :
\begin{remarque}
  Jazz is not dead. It just smells funny.
  \hfill Frank \textsc{Zappa}
\end{remarque}
Et on reprend notre propos.
```

9.22

On peut faire la remarque (édifiante) suivante :

Jazz is not dead. It just smells funny.
Frank Zappa

Et on reprend notre propos.

À la question : *comment pourrait-on faire pour encadrer cette remarque ?* nous répondons sans hésiter : « avec l'environnement `lrbox` ». Dans la mesure où il n'y a pas de moyen avec LaTeX de commencer une `\fbox` dans la clause `begin` et de la finir dans la claude `end`, on utilisera le principe suivant :

1. stocker le contenu à encadrer à l'aide de l'environnement `lrbox`
2. l'encadré en réutilisant la boîte ainsi construite.

On aura donc :

```
\newsavebox{\boiteremarque}
\newenvironment{remarque}{%
  % clause begin
  \begin{lrbox}{\boiteremarque}% début mise en boîte
  \begin{minipage}{.8\textwidth}}{%
  % clause end
  \end{minipage}
  \end{lrbox}% fin mise en boîte
  % production de la boîte encadrée
  \begin{center}
    \fbox{\usebox{\boiteremarque}}
  \end{center}}
```

e

```
On peut faire la remarque (édifiante) suivante :
\begin{remarque}
  Jazz is not dead. It just smells funny.
  \hfill Frank \textsc{Zappa}
\end{remarque}
Et on reprend notre propos.
```

9.23

On peut faire la remarque (édifiante) suivante :

Jazz is not dead. It just smells funny.
Frank Zappa

Et on reprend notre propos.

Cette idée est utilisée à plusieurs reprises dans les deux chapitres qui suivent.

Il faut noter que la boîte que définit l'environnement `lrbox` est une boîte simple à l'instar des boîtes créées par les commandes de la famille `\mbox` et par conséquent ne peut contenir de saut de paragraphe.

10 Cosmétique

Sommaire

Je me dis: Je monterai sur le palmier, J'en saisirai les rameaux !
Que tes seins soient comme les grappes de la vigne,
Le parfum de ton souffle comme celui des pommes,
Et ta bouche comme un vin excellent qui coule aisément pour mon bien-aimé,
Et glisse sur les lèvres de ceux qui s'endorment !

Le Cantique des cantiques Ct **7** 10.

L'IDÉE GÉNÉRALE de ce chapitre, laissant présager des macros parfumées, est de présenter les outils standard de LaTeX qui ont été personnalisés pour produire certaines parties du document. Ces personnalisations s'entendent à plusieurs niveaux : en utilisant des options de packages (par exemple pour les en-têtes de page), ou parfois en « mettant le nez » dans la définition des macros, comme pour l'allure des chapitres et des sections, ou en modifiant plus en profondeur ces macros comme dans le cas de la minitable des matières. Une partie du chapitre est consacrée aux outils que l'on peut mettre au point à partir du package fancyvrb. Enfin un combat en règle contre les guillemets français est mené en guise de clôture de ce chapitre.

10.1 Allure de l'index

Pour changer l'allure de l'index, il faut comprendre que lorsqu'on tape fébrilement avec ses petits doigts la commande :

```
makeindex document
```

on génère alors un fichier document`.ind` contenant quelque chose ressemblant à :

```
\begin{theindex}                    ← préambule

\item Cosmic debries, 12,34

\indexspace                         ← espace inter-groupe
```

```
\item Debra kadabra, 23                    ← entrée, séparateur, page

\end{theindex}                             ← postambule
```

En réalité, ce code est généré à partir d'entités génériques ayant des valeurs prédéfinies et pouvant être modifiées. Pour s'en convaincre, il suffit de savoir que le programme makeindex peut générer un fichier `.ind` contenant autre chose que du code LaTeX. Pour comprendre cette affaire d'entités génériques, on pourrait décrire le travail de makeindex comme suit :

1. Écrire le préambule en examinant la valeur de l'entité `preamble` ;
2. Pour chaque entrée du fichier `.idx`:
   (a) écrire le contenu de l'entité `item_0` ;
   (b) écrire l'entrée (« Cosmic debries » dans l'exemple précédent) ;
   (c) écrire le séparateur (valeur de l'entité `delim_0`) ;
   (d) écrire le numéro de page
3. À chaque fin de groupe (changement de lettre) écrire le contenu de l'entité `group_skip` ;
4. Écrire le postambule en examinant la valeur de l'entité `postamble`.

Les valeurs des entités auxquelles il est fait allusion sont par défaut les suivantes :

`preamble`	`"\\begin{theindex}\n"`
`item_0`	`"\n  \\item"`
`delim_0`	`", "`
`group_skip`	`"\n\n  \\indexspace\n"`
`postamble`	`"\n\n\\end{theindex}\n"`

Ces valeurs peuvent être modifiées par l'intermédiaire d'un fichier de style auquel on met généralement l'extension `.ist` et que l'on utilisera lors de l'appel à makeindex de la manière suivante :

```
makeindex -s style.ist document
```

Ainsi pour produire l'index de ce document, nous avons dans un premier temps redéfini les séparateurs de niveau 1 et 2 :

```
delim_0 " \\dotfill \ "
delim_1 " \\dotfill \ "
```

on remplace donc la virgule qui sépare par défaut l'entrée d'index et son numéro de page par des points de suspension. Ensuite, en lisant scrupuleusement la documentation makeindex,[1] on apprend qu'écrire :

```
headings_flag 1
```

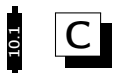
est la manière polie de demander à makeindex de produire entre les groupes d'entrées la lettre correspondant au groupe. Cette lettre sera (en majuscule) et encadrée par les contenus respectifs des entités `heading_prefix` et `heading_suffix`. Qu'à cela ne tienne, pour produire nos jolies boîtes ombrées, nous écrivons dans le fichier de style :

```
heading_prefix "{\\large\\sffamily\\bfseries\\shadowbox{"
heading_suffix "}\\hfill}\\nopagebreak\n"
```

[1] Voir également le nota qui suit pour les références bibliographiques utiles.

Ce qui vous l'avez compris, produira par exemple pour la lettre « c » :

```
{\large\sffamily\bfseries%
  \shadowbox{C}\hfil}\nopagebreak
```



Cette commande sera précédée par le contenu de `group_skip` qui, nous l'avons dit un peu plus haut, vaut par défaut `\indexspace`. Nous avons après quelques mois de recherche,[2] déniché la définition de cette commande dans `book.cls` et l'avons modifiée pour augmenter légèrement l'espace entre les groupes :

```
\renewcommand\indexspace{\par \vskip 20pt plus5pt minus3pt\relax}
```

Ce paragraphe ne donne bien évidemment qu'un aperçu très succinct des fonctionnalités proposées par makeindex. Outre les informations que l'on peut trouver dans le LaTeX companion, la page de manuel de cet utilitaire dans un environnement Debian donne une liste exhaustive des entités génériques que l'on peut définir. Un fichier nommé `ind.dvi` écrit par P. Chen et M. A. Harrinson constitue également un bon point de départ pour la personnalisation de l'index.

10.2 Allures des titres

Nous proposons ici d'exposer la manière dont on a modifié l'allure des titres standard (partie, chapitre, section, etc.) de LaTeX.

10.2.1 Numérotation des titres et table des matières

Avant toute chose il faut savoir qu'on peut agir sur la table des matières à l'aide de deux compteurs :

1. `secnumdepth` (*section numbering depth*) stipulant la profondeur de la numérotation des titres dans le document ;
2. `tocdepth` (*tab le of contents depth*) définissant quel est le niveau (ou profondeur) de titre maxi dans la table des matières.

Pour utiliser ces deux compteurs, il faut en outre avoir connaissance de la manière dont LaTeX associe une profondeur à chaque titre. Et bien réjouissez-vous, le tableau suivant vous fournit cette information :

Titre	profondeur	Titre	profondeur
`part`	-1		
`chapter`	0	`subsubsection`	3
`section`	1	`paragraph`	4
`subsection`	2	`subparagraph`	5

Ainsi, affecter 1 à `secnumdepth` et 2 à `tocdepth` numérotera les titres jusqu'aux `\sections` et insérera dans la table des matières, toutes les titres jusqu'aux `\subsection`...

[2] Je plaisante, juste quelques semai... minutes veux-je dire.

10.2.2 Sections et niveaux inférieurs

Dans le fichier `book.cls` du système TEX, on trouve le code suivant :[3]

```
\newcommand{\section}{%
  \@startsection%
  {section}% nom du titre
  {1}% niveau de titre
  {0pt}% indentation
  {-3.5ex plus -1ex minus -.2ex}% espace vertical avant
  {2.3ex plus.2ex}% espace vertical après
  {\normalfont\Large\bfseries}} % allure du titre
```

Ce code permet de définir comment sera produit le titre d'une section. On constate que la commande `\section` fait appel à la commande `\@startsection`, cette dernière attendant six arguments :

- le nom du titre : `section`, `subsection`, etc.
- son niveau : 1 pour `section`, 2 pour `subsection`, 3 pour `subsubsection`, etc.
- son indentation ;
- le blanc vertical avant le titre ;
- le blanc vertical après le titre ;
- un ensemble de *déclarations* pour formater le titre lui-même.

On pourra donc noter que la mise en page par défaut de LATEX pour les sections dans la classe `book` est la suivante :

- pas d'indentation (`0pt`)
- espace avant le titre de `3.5ex` avec un tolérance de plus `-1ex` et moins `-.2ex` ;
- espace après le titre de `2.3ex` avec une tolérance de plus `.2ex` ; on pourra noter que si l'espace est négatif, le paragraphe commence juste après le titre, et non sur un nouveau paragraphe ;
- les titres sont en gros et en gras dans la fonte « normale ».

Pour définir l'allure des sections de ce document, nous avons introduit trois longueurs pour l'indentation de `section`s, `subsection`s et `subsubsection`s :

```
\newlength{\sectiontitleindent}
\newlength{\subsectiontitleindent}
\newlength{\subsubsectiontitleindent}
```

Ces longueurs ont pour valeur :

```
\setlength{\sectiontitleindent}{-1cm}
\setlength{\subsectiontitleindent}{-.5cm}
\setlength{\subsubsectiontitleindent}{-.25cm}
```

D'autre part, nous avons défini une fonte particulière pour les titres, définie comme suit :

```
\newcommand{\sectionfont}{%
  \fontencoding{\encodingdefault}%
  \fontfamily{pag}%
  \fontseries{bc}%
  \fontshape{n}%
  \selectfont}
```

[3] Légèrement simplifié...

Cette commande permet de sélectionner la fonte PostScript Avant-Garde en gras condensé (cf. 9.4 page 114). Finalement, pour définir l'allure de nos sections on utilisera :

```
\renewcommand{\section}{%
  \@startsection%
  {section}%
  {1}%
  {\sectiontitleindent}%
  {-3.5ex plus -1ex minus -.2ex}%
  {2.3ex plus.2ex}%
  {\sectionfont\Large}}
```

Des commandes équivalentes ont été écrites pour les titres de niveaux inférieurs.

10.2.3 Chapitres

C'est en fouillant dans le fichier `book.cls` qu'on peut trouver des informations sur la manière dont LaTeX produit les en-têtes de chapitres.

Principe

Dans le fichier `book.cls`, on trouve la commande :

```
\newcommand{\chapter}{%
  ...
  \thispagestyle{plain}%
  ...
  \secdef\@chapter\@schapter} % la ligne qui nous intéresse
```

La commande `\chapter` fait donc elle-même appel à deux commandes distinctes :

1. `\@chapter` pour les titres de chapitre qui sont numérotés ;
2. `\@schapter` pour les titres de chapitre non numérotés (`s` pour *star* ou étoile faisant référence à la commande `\chapter*`).

En cherchant vaillamment la définition de ces deux commandes (toujours dans le fichier `book.cls`), on trouve quelque chose du genre :

```
\def\@chapter[#1]#2{%
  ...
  \refstepcounter{chapter}%
  \typeout{\@chapapp\space\thechapter.}% message sur le terminal
  \addcontentsline{toc}{chapter}% ajout du titre dans la toc
  ...
  \if@twocolumn
  ...
  \else% le cas d'un document à une colonne
  \@makechapterhead{#2}% la ligne qui nous intéresse
  \fi}
```

ce qui nous met sur la voie... en effet la commande `\@makechapterhead` (qu'on peut traduire littéralement par « faire l'en-tête de chapitre ») est la commande qu'il va nous falloir redéfinir pour changer l'allure des en-têtes. Une recherche supplémentaire

nous met également sur la piste de la commande `\@makeschapterhead` produisant l'en-tête d'un chapitre non numéroté. Ces deux commandes attendent un argument qui est le titre du chapitre.

Petits outils nécessaires

Nous avons défini un environnement `cadrechap` dont le propos est simplement d'élargir la marge de droite de deux centimètres :

```
\newenvironment{cadrechap}%
{\begin{list}{}{%
      \setlength{\leftmargin}{0pt}%
      \setlength{\rightmargin}{-2cm}% on se met au large
      \setlength{\itemindent}{0pt}%
      \setlength{\labelsep}{0pt}%
    }\item}%
{\end{list}}
```

Il sera également fait usage du booléen `@mainmatter` permettant de savoir si on se trouve dans la partie « centrale » du document. C'est le cas lorsque la commande `\mainmatter` a été appelée (cf. 6.4 page 84).

En-tête des chapitres à proprement parler

Pour ce manuel, nous avons redéfini la commande `\@makechapterhead` comme suit :

```
\renewcommand{\@makechapterhead}[1]{%
  \begin{cadrechap}
    \if@mainmatter
    {\chapnumfont\thechapter}\hfill{\chapfont#1}
    \else
    \mbox{}\hfill{\chapfont#1}
    \fi
  \end{cadrechap}
  \vspace{1cm}}
```

Par conséquent, si on est dans le « `\mainmatter` » :

- on affiche le numéro du chapitre (`\thechapter`) dans une fonte particulière (`\chapnumfont`) ;
- on insère un ressort horizontal (`\hfill`) ;
- on affiche le titre du chapitre (`#1`) dans une police particulière (`\chapfont`).

Si on n'est pas dans la partie centrale, on n'affiche pas de numéro de chapitre, mais juste le titre poussé à droite. Les commandes permettant de mettre en évidence le numéro du chapitre et son titre sont respectivement :

```
\DeclareFixedFont{\chapnumfont}{T1}{phv}{bc}{n}{80pt}
\DeclareFixedFont{\chapfont}{T1}{phv}{b}{n}{24.88pt}
```

Finalement, si on écrit :

```
\begin{cadrechap}
  {\chapnumfont7}\hfill{\chapfont Le titre de la mort qui tue}
\end{cadrechap}
```

On obtient :

7 Le titre de la mort qui tue

10.2.4 Parties

Dans le fichier `book.cls` on trouve la définition de la commande `\part` :

```
\newcommand\part{%
   \cleardoublepage
  \thispagestyle{plain}
  [...]
  \null\vfil
  \secdef\@part\@spart}
```

qui nous informe qu'à l'instar des chapitres, la commande `\part` fait appel à deux commandes distinctes pour produire les parties numérotées et non numérotées (grâce à un appel aux commandes `\@part` et `\@spart` respectivement). Dans un premier temps nous avons imposé que le style de page pour les débuts de partie soit vide (c'est-à-dire sans numéro de page ni en-tête etc.), nous avons donc écrit :

```
\newcommand\part{%
  \cleardoublepage
  \thispagestyle{empty}% à la place de plain par défaut
  [...]
  \null\vfil% un boîte vide et un ressort vertical
  \secdef\@part\@spart}
```

On peut ensuite examiner la définition de la commande `\@part` qui produit la page de partie :

```
\def\@part[#1]#2{%
  [...]
  {\centering % centrage
    [...]
    \huge\bfseries \partname\nobreakspace\thepart
     \par
     \vskip 20\pt
    [...]
   \Huge \bfseries #2\par}%
  \@endpart}
```

En examinant ce code on constate que la page de partie est constituée d'une ligne en gros caractères gras, du nom « Partie » suivie du numéro de la partie :[4]

```
\huge\bfseries \partname\nobreakspace\thepart
```

suivie 20 points plus bas du titre de la partie (contenu dans l'argument `#1`). Pour ce manuel, nous avons redéfini la commande `\@part` comme suit :

[4] En réalité, après avoir enclenché le package `babel` et l'option `french` ces deux commandes sont redéfinies pour produire quelque chose du style : « Première partie »

```
\def\@part[#1]#2{%
  [...]
  {\centering
   \interlinepenalty \@M
   \normalfont
   [...]
   \partnumfont \thepart  % juste le numéro de la partie
   \par
   \vskip 50\p@% 50 point au lieu de  20...
   \partfont #2\par}% le titre avec une fonte  personnalisée
  \@endpart}
```

Pour garder une homogénéité avec les en-têtes de chapitre on a défini les commandes de fontes comme suit :

```
\newcommand{\partfont}{%
  \fontencoding{\encodingdefault}\fontfamily{phv}%
  \fontseries{bc}\fontshape{n}%
  \fontsize{32}{34}%
  \selectfont}
\DeclareFixedFont{\partnumfont}{T1}{phv}{bc}{n}{80}%
```

On notera également que la commande `\@part` se termine par l'appel à une autre commande : `\@endpart`. En examinant le fichier `book.cls` on pourra se rendre compte que cette dernière permet de s'opposer au ressort vertical de la commande `\part` et de sauter une page blanche...

10.3 Géométrie

Les différentes dimensions de chaque page de ce document ont été définies à l'aide du package geometry et de la commande :

```
\geometry{%
  a4paper,
  body={150mm,250mm},
  left=25mm,top=25mm,
  headheight=7mm,headsep=4mm,
  marginparsep=4mm,
  marginparwidth=27mm}
```

qui définit respectivement (voir aussi figure 10.1 page suivante) :

- un corps de texte faisant 150 mm de largeur par 250 mm ;
- le positionnement du corps du texte dans la page, à 25 mm du bord gauche du papier, et 25 mm du bord supérieur du papier ;
- la hauteur de l'en-tête (7mm) et l'espace entre l'en-tête et le texte lui-même (4 mm) ;
- la taille du papier : standard A4 ;
- la largeur de la marge pour les notes de marges (2.7 cm).

De manière générale, comme le montre la figure 10.1 page ci-contre, le package geometry permet de définir un certain nombre de dimensions que l'on peut passer soit en option à la commande `\usepackage` soit à l'aide de la commande `\geometry`.

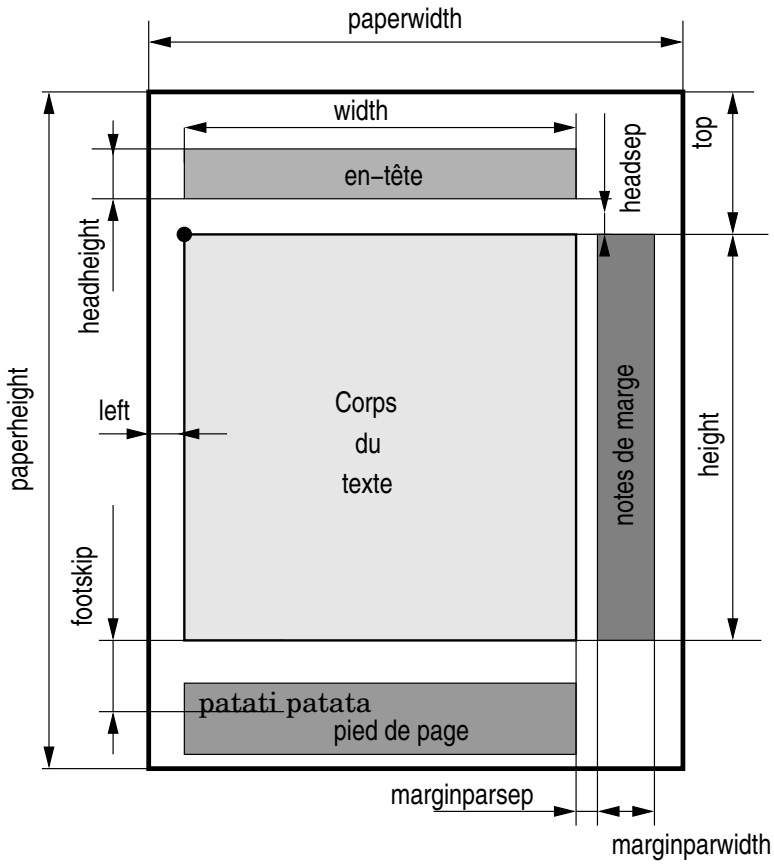


FIG. 10.1 – Quelques unes des dimensions pour définir la géométrie d'un document.

Dimension du papier :
- `a4paper`, `a5paper`, etc. pour utiliser un format de papier prédéfini,
- `paperwidth=`=dim et `paperheight=`dim pour spécifier une dimension de papier libre, par exemple pour un document qui sera massicoté.

Texte :
- soit avec : `body={`largeur`,`hauteur`}`
- soit avec : `width=`largeur et `height=`hauteur.
- le texte est positionné à l'intérieur de la page par rapport à un point de référence spécifié avec `top=`pos_vert et `left=`pos_horiz

Haut et bas de page :
- la hauteur de la surface réservée à l'en-tête peut être définie à l'aide de la formule magique `headheight=`hauteur et sa position par rapport au corps du texte à l'aide de `headsep=`espace.
- la position du pied de page peut être définie à l'aide de `footskip=`espace qui impose l'espace entre le bas du corps du texte et la première ligne du contenu du pied de page.

Note de marge dans le même esprit la largeur et la position de la surface réservée aux notes de marge peuvent être définies à l'aide de `marginparwidth=`largeur et `marginparsep=`espace.

Dans le package geometry les dimensions concernant l'en-tête, le pied de page et la zone pour les notes de marge, sont par défaut comptabilisées *en plus* du corps du texte. Des options permettent d'inclure l'une ou l'intégralité de ces dimensions dans

le corps du texte pour le calcul, en disant par exemple : « je veux que la largeur soit de 10 centimètres, notes de marge comprises. » (voir la documentation du package pour les détails).

10.4 En-tête et pied de page

Les zones au-dessus et en dessous du corps du texte appelées en-tête et pied de page peuvent être personnalisées à l'aide du package fancyhdr. Le principe de base est simple,[5] il suffit d'utiliser la commande :

```
\pagestyle{fancy}
```

pour spécifier qu'on veut utiliser des en-têtes et des pieds de page définis grâce au package fancyhdr. Par défaut le package produit des traits horizontaux en dessous de l'en-tête et au-dessus du pied de page dont les épaisseurs sont définies par les commandes `\footrulewidth` et `\headrulewidth`. On peut ensuite utiliser les commandes :

- `\fancyhead` pour définir l'en-tête ;
- `\fancyfoot` pour définir le pied de page ;

Ces deux commandes peuvent prendre un argument optionnel constitué d'une ou deux séquences des caractères suivants :

- `E` ou `O` pour spécifier la parité de la page (paire=*even*, impaire=*odd*) ;
- `R`, `L`, `C` pour spécifier où l'on veut produire l'information : respectivement à droite, à gauche ou au centre ;

Voici un exemple :

```
\fancyhf{} % on efface tout et on recommence
% EN TÊTE :
% initiales à droite sur page paire, à gauche sur page impaire :
\fancyhead[RE,LO]{VL}
% numéro de page au centre :
\fancyhead[C]{\thepage}
% numéro de section à droite sur page impaire, à gauche sur page paire :
\fancyhead[LE,RO]{\thesection}
% PIED DE PAGE :
% une image à droite sur page impaire, à gauche sur page paire :
\fancyfoot[RO,LE]{\includegraphics[height=4ex]{punch}}
% titre à gauche sur page impaire, à droite sur page paire :
\fancyfoot[LO,RE]{%
  Tout ce que vous avez toujours voulu savoir sur \LaTeX{}}
% épaisseur des traits
\renewcommand{\footrulewidth}{3pt}
```

10.4.1 Cas de la première page des chapitres

Dans la classe `book`, LaTeX fait automatiquement appel au style `plain` pour les premières pages de chapitre. Pour demander au package fancyhdr de définir un style particulier pour ces pages, on écrit :

[5] Vous ne serez sans doute pas tout à fait d'accord avec le terme « simple » après avoir lu la suite...

```
% le cas de la première page d'un chapitre
\fancypagestyle{plain}{%
  \fancyhf{}% on efface tout
  \fancyfoot[C]{\thepage}% numéro en bas de la page
  % on efface tous les traits
  \renewcommand{\headrulewidth}{0pt}%
  \renewcommand{\footrulewidth}{0pt}}
```

Vérifiez maintenant que les pages 3, 15, 33, 67, etc. ont ce style...

10.4.2 Pages vierges avant le début d'un chapitre

Dans la classe `book` en mode recto-verso (c'est le cas de ce document), LaTeX commence par défaut un chapitre sur une page impaire — appelée dans le jargon typographique la « belle page ». Pour ce faire LaTeX fait appel dans différentes commandes internes, à la commande `\cleardoublepage` qui permet d'insérer si nécessaire une page blanche avant le début du chapitre. Cette page reçoit par défaut le style des en-tête et pied en cours. Dans le manuel que vous avez sous les yeux, nous avons imposé un style « vide » à ces pages en modifiant la définition de la commande `\cleardoublepage` du fichier `latex.ltx` :

```
\renewcommand{\cleardoublepage}{% redéfinition de la commande
  \clearpage\ifodd\c@page\else
  \hbox{}
  \vspace*{\fill}
  \thispagestyle{empty}% ligne ajoutée
  \newpage
  \fi}
```

Feuilletez le manuel et cherchez si les pages vierges avant le début des chapitres sont bien vides...

10.4.3 Mécanisme de marqueurs

Vous aurez sans doute remarqué que dans ce manuel, les en-têtes des pages contiennent des informations qui dépendent du contexte. Sont en effet insérés sur les pages paires (page de gauche) le titre du chapitre, et sur les pages impaires (page de droite) le titre de la dernière section de la page. Il est possible de produire ce genre d'en-têtes car LaTeX dispose d'un mécanisme de *marqueurs* que nous allons tenter d'expliquer ici.

Il n'est pas inutile de préciser maintenant que lorsque TeX et LaTeX produisent une page, ils vont garnir l'en-tête et le pied en fonction d'information collectées le long de la page en question. La production de l'en-tête et du pied est donc postérieure à la composition de la page.

Les commandes `\markboth` et `\markright`

Soient les commandes :

```
\markboth{texte_g}{texte_d}
```

ou :

`\markright{`texte`}`

Nous allons imaginer que les arguments texte$_x$ sont stockés dans une pile et une file. Dans cet ordre d'idée :

- `\markboth` empile texte$_g$, et stocke texte$_d$ dans la file ;
- `\markright` stocke texte dans la file.

Ces deux commandes de « marquage » peuvent être appelées plusieurs fois ou jamais, sur une même page. Les données de la pile et de la file seront exploitées au moment de générer les en-têtes et pied de page, lorsque TEX achève la mise en forme de la page, et ceci grâce aux commandes :

- `\leftmark` renvoie le sommet de la pile, c'est-à-dire texte$_g$ du *dernier appel* à `\markboth` ;
- `\rightmark` renvoie le début de la file, c'est-à-dire texte$_d$ du *premier appel* à `\markboth` ou texte du *premier appel* à `\markright`.

Une petite subtilité au sujet de la « file » que nous présentons ici : tant qu'aucune commande de « marquage » n'ajoute de données au cours d'une page, la file contiendra *la dernière information insérée* dans les pages précédentes. La « file » est vidée dès qu'une commande `\markboth` où `\markright` survient.

Un autre moyen de comprendre ce mécanisme de marqueurs pourrait être de dire :

- `\leftmark` contient la dernière information que j'ai insérée sur la pile (à l'aide du premier argument de `\markboth`) ;
- `\rightmark` contient la première information de la « file », si on en a mis une sur cette page, ou la dernière qui a été enfilé (à l'aide du deuxième argument de `\markboth` ou de l'argument de `\markright`.

Il peut être utile de savoir que l'auteur a utilisé ces commandes pour la production d'un trombinoscope composé de plusieurs dizaines de noms et photos par page. L'idée était d'exploiter le mécanisme de marquage pour faire apparaître dans l'en-tête le premier et le dernier nom de la page, comme dans un dictionnaire. Il suffit pour cela d'appeler pour chaque personne (nom et photo) la commande :

```
\markboth{nom du gugusse}{nom du gugusse}
```

puis d'insérer dans les en-têtes la commande `\rightmark` sur les pages de gauche (impaires) et `\leftmark` sur les pages de droite (paires)...

Interactions avec les commandes de paragraphe

À chaque début de chapitre, de section, de sous-section, etc. une commande interne de LATEX fait appel aux commandes de marquages présentées au paragraphe précédent, pour stocker des informations susceptibles d'enrichir l'en-tête ou le pied de page. Ces commandes se nomment :

- `\chaptermark` pour les chapitres ;
- `\sectionmark` pour les sections ;
- ...

elles attendent un argument qui contient le titre du chapitre ou du paragraphe. Dans ce manuel, les deux commandes précédentes ont été définies comme suit :

```
\renewcommand{\sectionmark}[1]{% #1 contient le titre de la section
  \markright{\sectionfont\thesection\ #1}}
\renewcommand{\chaptermark}[1]{% #2 contient le titre du chapitre
  \markboth{\sectionfont#1}{}}
```

Puis :

```
\fancyhead[LE,RO]{\thepage}
\fancyhead[LO]{\rightmark}
\fancyhead[RE]{\leftmark}
```

Par conséquent :

- à droite des pages paires, on trouve (`\leftmark`) le dernier titre de chapitre rencontré ;
- à gauche des pages impaires, on trouve (`\rightmark`) le numéro et le titre de la première `\section` de cette page, ou le numéro et le titre de la dernière `\section` rencontrée...

Si vous ne me croyez pas voyez par vous-même le haut de cette page.

10.4.4 Organisation du document

Il est nécessaire de savoir que dans un document tel que celui que vous lisez, il existe trois parties qui sont reconnues par LaTeX : le *front matter*, le *main matter* et le *back matter* désignant respectivement le début du document (comportant généralement la préface et le sommaire), la partie principale, et la partie clôturant le document (comportant généralement la table des matières, le ou les index, la ou les bibliographies, le glossaire, etc). On doit alors explicitement écrire un document LaTeX comme suit :

```
\documentclass{classe du document}
\begin{document}
\frontmatter % introduction
[...]
\mainmatter % partie principale
[...]
\backmatter % pour clore le document
[...]
\end{document}
```

Nous verrons dans la suite de ce chapitre que nous serons amenés à modifier les trois commandes permettant de passer d'une partie à une autre. Pour l'instant, il faut savoir que la classe `book` définit un booléen :

```
\newif\if@mainmatter
```

utilisé par défaut dans LaTeX pour savoir si on se trouve dans le « main matter » ou pas. Nous avons en outre défini pour notre document un autre booléen :

```
\newif\if@frontmatter
```

qui nous permettra d'effectuer des traitements particuliers lorsqu'on sera dans la partie introductive du document. Les trois commandes délimitant les trois parties sont définies par :

```
\renewcommand\frontmatter{%
  \cleardoublepage
  \@frontmattertrue
  \@mainmatterfalse
  \pagenumbering{roman}% numérotation en romain
```

```
}
\renewcommand\mainmatter{%
  \cleardoublepage
  \@mainmattertrue
  \@frontmatterfalse
  \pagenumbering{arabic}% numérotation en chiffres arabes
}
\renewcommand\backmatter{%
  \cleardoublepage
  \@frontmatterfalse
  \@mainmatterfalse
}
```

En farfouillant dans le code de LaTeX on peut comprendre que `\pagenumbering`, la commande permettant de changer la numérotation, réinitialise le compteur de page à 1.

10.4.5 Numéroter l'introduction en roman « petites capitales »

Votre serviteur a tenu à ce que les pages de l'introduction soient numérotées en chiffres romains et petites capitales. On ne peut malheureusement pas écrire :

```
\renewcommand{\thepage}{\textsc{\roman{page}}}
```

Puisque cela provoque une incompatibilité avec la gestion de l'index. L'idée retenue est de procéder comme suit :

1. utiliser la numérotation en chiffre romain minuscule ;
2. dans le pied de page afficher `\textsc{\thepage}` ;
3. modifier la commande `\index` pour que les numéros de pages s'affiche en petites capitales.

D'où dans la définition de `\frontmatter` on ajoutera :

```
\let\indexORI\index% sauvegarde de la définition initiale
\renewcommand{\index}[1]{\indexORI{##1|textsc}}
\fancyfoot{}
\fancyhead[LE,RO]{\textsc{\thepage}}
```

Et dans la définition de `\mainmatter` :

```
\let\index\indexORI% pour revenir à la définition initiale
```

Pour être parfaitement rigoureux on va modifier l'allure des premières pages de chapitre :

```
\fancypagestyle{plain}{%
  \fancyhf{}
  \if@frontmatter% introduction
     \fancyfoot[C]{\textsc{\thepage}}
  \else
     \fancyfoot[C]{\thepage}
  \fi
\renewcommand{\headrulewidth}{0pt}
\renewcommand{\footrulewidth}{0pt}}
\makeatother
```

10.4.6 Interaction avec index, bibliographie et table des matières

Dans la classe `book`, deux environnements sont définis :
- `thebibliography` permettant de produire la bibliographie ;
- `theindex` pour produire l'index ;

et la commande :
- `\tableofcontents` pour produire la table des matières.

Ces environnements et cette commande sont conçus pour produire des en-têtes avec le numéro de la page et le nom du chapitre en majuscules à savoir `\bibname`, `\indexname` et `\contentsname`. Voici par exemple un extrait de `\tableofcontents` :

```
\newcommand\tableofcontents{%
  [...]
  \chapter*{\contentsname
      \@mkboth{%
         \MakeUppercase\contentsname}%
        {\MakeUppercase\contentsname}}%
  \@starttoc{toc}%
  [...]
  }
```

J'ai souhaité que dans ce manuel les en-têtes ne soient pas en majuscules. Deux solutions sont possibles :

1. utiliser la commande `\nouppercase` du package fancyhdr et écrire dans la définition de `\backmatter` :

   ```
   \fancyhead[LO]{\nouppercase\rightmark}% en−tête en minuscule
   \fancyhead[RE]{\nouppercase\leftmark}%
   ```

2. recopier la définition de `\tableofcontents` provenant de `book.cls`, et la modifier pour supprimer les commandes `\MakeUppercase`. Faire la même chose pour l'index et la bibliographie.

Dans ce document, c'est la deuxième solution qui a été adoptée. On en a également profité pour insérer l'index et la bibliographie dans la table des matières, ce qui n'est pas le comportement par défaut de LaTeX et de la classe `book`. Nous avons donc pour l'environnement `theindex` :

```
\renewenvironment{theindex}
{%
  [...]
  % insertion dans la table des matières
  \addcontentsline{toc}{chapter}{\indexname}
  \@mkboth{\indexname}{\indexname}% suppression de \MakeUppercase
  \thispagestyle{plain}
  [...]
{\if@restonecol\onecolumn\else\clearpage\fi}
```

10.5 Environnements avec caractères spéciaux

Les packages fancyvrb et listings ont tous deux la particularité de produire du texte avec des caractères spéciaux. Le premier permet de produire des environnements de type `verbatim` avec beaucoup plus de souplesse. Il permet notamment

de personnaliser d'éventuelles bordures, les marges, et surtout on peut « s'échapper vers LaTeX » au beau milieu de l'environnement, ou comme disent les anglophones : *to escape to LaTeX*. En d'autres termes, bien qu'étant dans un environnement où les caractères `\`, `{` et `}` sont sans effet, il est malgré tout possible de faire appel à des commandes LaTeX.

Le second (listings) a pour objectif de produire des extraits de langage de programmation. Il propose également parmi un grand nombre de fonctionalités, la possibilité de s'échapper vers LaTeX. Nous vous proposons donc ici de découvrir ces deux environements à l'aide d'exemples « en grandeur nature » utilisés dans ce manuel.

10.5.1 Digression vers les caractères...

Il peut ne pas être inutile[6] de faire ici une petite digression sur la manière dont TeX mange et digère les caractères qu'on lui fournit. Il faut savoir que les caractères peuvent entrer dans seize catégories différentes. Chaque caractère peut n'appartenir qu'à une catégorie à la fois. Chacune de ces catégories permet de basculer TeX vers un traitement particulier. Par exemple lorsque le caractère `\` est rencontré, TeX va lire les caractères qui suivent pour connaître le nom de la commande (ou *séquence de contrôle*), lorsqu'il rencontre le caractère `{`, TeX va ouvrir un nouveau groupe, lorsque que le caractère `%` est lu, TeX va ignorer les caractères jusqu'à la fin de la ligne, c'est-à-dire jusqu'à ce qu'il rencontre un caractère catégorisé « fin de ligne », etc. Parmi les catégories reconnues par TeX :

Catégorie 0 caractère de contrôle (`\` dans LaTeX) ;

Catégorie 1 début de groupe (`{` dans LaTeX) ;

Catégorie 2 fin de groupe (`}` dans LaTeX) ;

Catégorie 11 lettre ;

Catégorie 14 commentaire (`%` dans LaTeX) ;

On peut donc s'amuser à changer le contenu de chaque catégorie. Dans l'exemple ci-dessous, on a transformé les caractères `\`, `{`et `}` en lettres, et on a décidé que les caractères `/`, `(` et `)` appartiendraient respectivement aux catégories : caractère de contrôle, début de groupe et fin de groupe. Le caractère `#` a également été changé de catégorie, c'est désormais un caractère de commentaire.

```
{ \catcode`\/=0 \catcode`\(=1 \catcode`\)=2
  \catcode`\#=14
  \catcode`\{=11 \catcode`\}=11 \catcode`\\=11
  # ça on devrait pas le voir...
  \bidule{truc muche} /textbf(en gras)
  )\par
  on retourne dans le monde \LaTeX{}...
```

10.2

\bidule{truc muche} **en gras**
on retourne dans le monde LaTeX...

En outre, il est intéressant de savoir que TeX peut rendre actifs certains caractères (qui entrent alors dans la catégorie 13). Ces caractères peuvent alors être définis comme des commandes voici un exemple idiot :

[6] Les français sont parait-il des spécialistes de la litote. Mais ne nous égarons pas...

```
\catcode`\+=13
\newcommand{+}{plus}
3 + 4 = 7
```

10.3

3 plus 4 = 7

Dans cet exemple on a rendu « actif » le caractère +, puis on l'a défini comme une commande. Vous noterez qu'ici on a pu créer une commande que l'on utilise sans faire appel au caractère \.

Il peut être utile de savoir que lorsqu'on charge le package babel et l'extension française, les caractères de ponctuations double sont également rendus actifs notamment pour empêcher la césure avant ceux-ci. En outre le caractère ~ est dans la catégorie des caractères actifs dans LaTeX. On peut d'ailleurs voir sa définition dans une session LaTeX interactive :

```
*\show~
> ~=macro:
->\nobreakspace {}.
<*> \show~
```

10.5.2 Environnements maison basés sur fancyvrb

Les environnements du type de `verbatim` ont pour but de changer l'appartenance des caractères à leur catégorie respective. En outre, le package fancyvrb permet de définir quels caractères permettent de repasser le contrôle à LaTeX. Dans ce document, l'environnement `unixcom` a été défini comme suit :

```
\DefineVerbatimEnvironment{unixcom}{Verbatim}{%
  commandchars=¢« »,
  frame=single, framerule=.4pt, framesep=1.5mm,
  gobble=2,
  xleftmargin=15pt}
```

Cet environnement est donc de type `verbatim` mais dans lequel on peut « exécuter » des commandes LaTeX à l'aide des caractères ¢ de catégorie 0, « de catégorie 1 et » de catégorie 2 — on peut bien évidemment choisir n'importe quel caractère pour ce faire. Il faut cependant garder à l'esprit que ceux-ci doivent être à la fois lisibles pour l'utilisateur et peu utilisés à d'autres fins que de repasser le contrôle à LaTeX.

```
Pour afficher le contenu d'une variable :
\begin{unixcom}
  echo ${¢marg« nom variable »}
\end{unixcom}
```

Pour afficher le contenu d'une variable :

```
echo ${nom variable}
```

La commande \marg permet de produire son argument entre chevrons simples et en penché. Les autres paramètres de la commande `\DefineVerbatimEnvironment` ont pour but de préciser l'allure de la bordure (paramètres `frame...`), la marge de gauche (paramètre `xleftmargin`) et le fait que les premiers caractères de chaque ligne seront systématiquement ignorés (`gobble`). Comme le montre la documentation du package fancyvrb beaucoup d'autres options sont disponibles.

Un autre environnement de ce genre a été créé pour saisir les commandes d'Emacs dans l'annexe consacrée à AucTeX. L'environnement en question (`emacscom`) a été créé comme suit :

```
\DefineVerbatimEnvironment{emacscom}{Verbatim}{%
commandchars=¢« »,
```

```
  frame=leftline, framerule=1mm, framesep=2mm,
  gobble=2, xleftmargin=15pt}
```

qui donne par exemple :

```
Pour jouer à Tetris dans \soft{Emacs} :
\begin{emacscom}
  M-x tetris
\end{emacscom}
```

Pour jouer à Tetris dans Emacs :

```
M-x tetris
```

10.5.3 Environnements pour les langages de programmation

Le package listings reconnaît la syntaxe d'un grand nombre de langage de programmation. Une manière simple d'utiliser ce package consiste à créer son propre environnement à l'aide d'une commande analogue à `\newenvironment` :

```
\lstnewenvironment{C}{\lstset{language=C}}{}
```

On pourra alors simplement écrire :

```
\begin{C}
  /* hello world en C */
  int main()
  {
    printf("bonjour monde\n");
    return 0;
  }
\end{C}
```

```
/* hello world en C */
int main()
{
  printf("bonjour monde\n");
  return 0;
}
```

Bien évidemment une foultitude d'option de configuration permet d'adapter cet environement à vos besoins. Le plus simple et le plus efficace est sans doute de lire la documenation accompagnant le package. À titre d'exemple, il faut savoir que l'on peut changer la mise en évidence des mots réservés et des commentaires du langage considéré. Ainsi, en écrivant :

```
\lstnewenvironment{Cbis}{%
  \lstset{language=C,
    basicstyle=\rmfamily\slshape,
    commentstyle=\rmfamily\upshape,}}{}
```

On aura :

```
\begin{Cbis}
  /* hello world en C */
  int main()
  {
    printf("bonjour monde\n");
    return 0;
  }
\end{Cbis}
```

```
/* hello world en C */
int main()
{
  printf("bonjour monde\n");
  return 0;
}
```

Et puisqu'il est question des caractères spéciaux et de l'« échappement vers LaTeX », il faut savoir qu'à l'instar de fancyvrb, le package listings permet de spécifier un caractère permettant cet échappement. Ainsi :

```
\lstnewenvironment{Cter}{\lstset{language=C, escapechar=@}}{}
```

Permet d'insérer des commandes LaTeX dans le listing :

```
\begin{Cter}
  int main()
  {
    printf("bonjour monde\n");
    return @\fbox{code de retour}@;
  }
\end{Cter}
```

```
int main()
{
  printf("bonjour monde\n");
  return code de retour;
}
```

10.6 About those so called “french guillemets”

Un des plaisirs de la typographie française est sans aucun doute l'utilisation de ces merveilleux guillemets « à la française »...[7] Cependant le package babel ne gère pas la césure correctement s'il on saisit dans un document :

```
\begin{minipage}{5cm}
  Cette courte phrase dans une boîte a pour unique
  but de montrer que ces symboles ne se comportent
  pas comme de « gentils » guillemets.
\end{minipage}
```

on aura la minipage suivante :

Cette courte phrase a pour unique but de montrer que ces petits symboles ne se comportent pas comme de « gentils » guillemets.

Ce qui est pour le moins gênant... Il est bien sûr possible de saisir ces guillemets avec les commandes \og et \fg fournies par le package babel, mais cela est au goût de votre serviteur trop contraignant dans la mesure où ce caractère est directement accessible depuis le clavier.[8] Il existe une solution — qui avait été adopté par le package french — palliant le problème de la césure qui consiste à rendre actifs les caractères ‘«’ et ‘»’. Nous avons donc écrit :

```
\catcode`\«=13
\catcode`\»=13
```

puis défini les deux commandes suivantes :

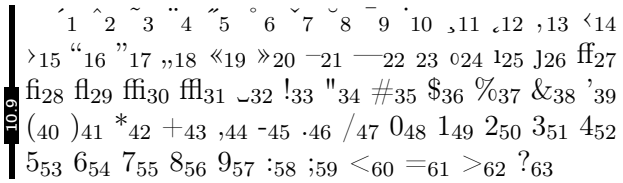
```
\newcommand{\fermerguillemets}{\unskip\kern.15em\symbol{20}}
\newcommand{\ouvrerguillemets}{\symbol{19}\ignorespaces\kern.15em}
```

On notera l'utilisation de la commande TeX \kern permettant d'insérer un blanc insécable d'une longueur donnée, de la commande \unskip et enfin de la commande \symbol qui insère ici les 19e et 20e caractères de la fonte courante :

[7] On notera d'ailleurs que la mode qui consiste aujourd'hui à faire subir à ses majeurs et index des mouvements d'oreilles de lapin, pendant qu'on parle pour dire « entre guillemets » sans le dire, est clairement emprunter aux américains. Je propose donc ici publiquement que l'on se force à utiliser plutôt l'index et le pouce pour faire ce geste, il faudra par contre se munir de deux autres bras pour être en mesure d'imiter les deux chevrons « et », ou peut-être qu'un habile tortillage de quatre doigts d'une main peut donner un résultat...

[8] Alt-Gr-z et Alt-Gr-x.

```
\setcounter{car}{1}
\whiledo{\value{car}<64}{%
  \symbol{\value{car}}$_{\thecar}$
  \stepcounter{car}}
```



Enfin, on a affecté aux caractères les deux commandes précédentes :

```
\let »=\fermerguillemets
\let« =\ouvrerguillemets
```

Cette façon de faire a trois inconvénients mineurs que je suis bien incapable de résoudre aujourd'hui. Tout d'abord, les moteurs sachant gérer le codage Utf 8 et permettant à TEX de rendre actif le caractère « (alors codé sur 2 octets), sont aujourd'hui peu répandus et j'avoue humblement que je ne les ai pas encore testés. Par conséquent la manipulation proposée ici est limitée aux codages utilisant 1 octet par caractère . Ensuite on ne peut utiliser ces guillemets dans un titre au risque d'avoir des artéfacts dans les « signets/*bookmarks* » d'un fichier pdf. Enfin, ces guillemets ne fonctionnent pas avec l'environnement `ltxexemple` défini à la fin du chapitre suivant. Un drame quoi !

10.7 Une boîte pour la minitable des matières

Le package mini-toc permet — comme son nom l'indique — de produire des « minitables des matières » que l'on peut insérer dans un document à un endroit donné, généralement en début de chapitre. Après avoir utilisé l'ordre `\dominitoc` dans le préambule, on fait ensuite appel à la commande `\minitoc` pour insérer cette minitable des matières à l'endroit voulu. La documentation du package explique tout cela en détail et présente notamment les différents styles que l'on peut utiliser. Pour ce manuel, j'ai trouvé l'idée d'une table des matières en début de chapitre séduisante, mais les styles proposés par le package ne me convenaient pas. En fait je souhaitais pouvoir mettre les titres de sections dans un boîte comme ceci :

Sommaire
x.1 Le premier titre
x.2 Le deuxième titre
x.3 etc.

C'est-à-dire une boîte avec un titre — ici le titre est « Sommaire ». À ma connaissance, LATEX ne propose pas de telles boîtes et suite à une question posée sur les forums de discussions, une bonne âme — en l'occurrence Benjamin BAYART — me propose un code TEX répondant au cahier des charges. Je vous propose dans ce paragraphe, une version[9] LATEX d'une boîte avec titre...

10.7.1 L'interface de la commande

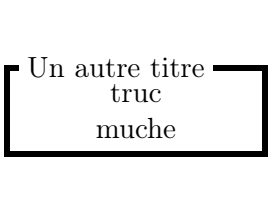
Plusieurs solutions sont possibles pour créer une telle commande. En s'inspirant de l'interface des boîtes de LATEX, on peut créer une macro dont la syntaxe d'utilisation serait :

[9] Tout à fait discutable et limitée dans ses fonctionnalités comme tout « logiciel » pondu par un bricoleur...

```
\titlebox{\footnotesize Un titre}{%
  Le contenu de la boîte}
```

ou encore :

```
\setlength{\fboxsep}{5pt}
\setlength{\fboxrule}{2pt}
\titlebox{Un autre titre}{%
  \begin{minipage}{3cm}\begin{center}
      truc\\ muche
    \end{center}\end{minipage}}
```



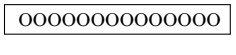
10.7.2 Quand même un peu de TEX

La primitive de TEX `\leaders` permet de remplir un espace élastique avec ce qui vous passe par la tête. Sa syntaxe :

`\leaders`ce qui vous chanteespace

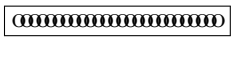
permet donc de remplir l'espace avec ce qui vous chante. Par exemple :

```
\framebox[3cm]{%
  \leaders\hbox{o}\hfill}
```



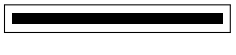
La primitive `\hbox` de TEX (utilisée par `\mbox` et `\makebox`) permet de créer des boîtes horizontales :

```
\framebox[3cm]{%
  \leaders\hbox to 3pt{o}\hfill}
```



Les `\leaders` peuvent également être utilisés avec la primitive `\hrule` de TEX permettant de dessiner des traits :

```
\framebox[3cm]{%
  \leaders\hrule height 4pt\hfill}
```

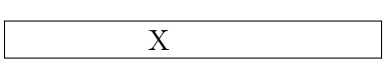


Ici, le ressort `\hfill` s'étire les trois centimètres de la `\framebox` et est rempli par un trait de hauteur quatre points.

```
\framebox[3cm]{%
  \leaders\hbox to5pt{%
    \leaders\hrule width1pt\hfill%
    \kern2pt}\hfill}
```

Dans l'exemple ci-dessus, l'espace de trois centimètres est rempli par des boites de cinq points de large, contenant chacunes d'elles des `\leaders` comme dans l'exemple précédent, et un blanc de deux points de large. Avec TEX, on peut régler la raideur du ressort de la manière suivante :

```
\framebox[5cm]{%
  \hskip0pt plus 2fill X\hskip0pt plus 3fill}
```



La dimension :

```
\hskip 0pt plus nfill
```

permet de définir une longueur élastique de raideur relative n. Dans l'exemple précédent la lettre 'X' se trouve donc au $^2/_5$ de la boîte... En utilisant ce type de blanc élastique et des `\leaders`, on peut définir la commande suivante :

```
\newcommand{\traitressort}[2][1]{%
  \leaders\hrule height#2\hskip0pt plus #1fill\relax}
```

pouvant être par exemple utilisée comme suit :

```
\framebox[5cm]{%
  \traitressort[2]{2ex}X\traitressort{2pt}}
```

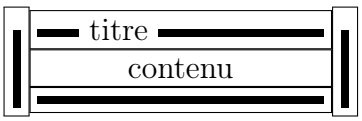
on a donc dans la boîte de cinq centimètres :

- un blanc élastique de raideur 2, rempli d'un trait de quatre points de hauteur ;
- la lettre X ;
- un blanc élastique de raideur 1, rempli d'un trait de deux points de hauteur.

Nous allons bien entendu nous servir de cette commande pour la suite...

10.7.3 Conception de la boîte

Pour concevoir la boîte à proprement parler, nous allons créer trois `\parbox` comme suit :



Il y a :

- deux `\parbox`s pour contenir les deux traits verticaux à droite et à gauche ;
- une `\parbox` pour le centre, contenant le trait du haut interrompu par le titre, le contenu, et en bas un trait horizontal.

Nous allons voir maintenant comment on peut construire ces trois boîtes et les positionner correctement les unes par rapport aux autres.

10.7.4 Le code

Nous allons avoir besoin d'une boîte pour stocker la `\parbox` centrale :

```
\newsavebox{\boitetitre}
```

et de deux dimensions :

```
\newlength{\largeurboitetitre}
\newlength{\hauteurboitetitre}
```

qui portent un nom suffisamment explicite m'évitant ainsi des phrases alambiquées expliquant la signification de telle ou telle variable. La première tâche que l'on va demander à la commande `\titlebox` est de stocker son contenu et de le mesurer :

```
\newcommand{\titlebox}[2]{%
  \begin{lrbox}{\boitetitre}% stockage du contenu
    \kern\fboxsep#2\kern\fboxsep
  \end{lrbox}
```

```
  % mesure de la largeur de la parbox centrale
  \settowidth{\largeurboitetitre}{\usebox{\boitetitre}}%
  % mesure de la hauteur de la parbox centrale
  \settoheight{\hauteurboitetitre}{\usebox{\boitetitre}}%
  \settodepth{\tempdim}{\usebox{\boitetitre}}%
  \addtolength{\hauteurboitetitre}{\tempdim+2\fboxrule+2\fboxsep}%
  ... }
```

`\kern` est une commande TEX permettant d'insérer un blanc insécable, ici de largeur `\fboxsep`. Notez que pour mesurer la hauteur totale on a recours à une longueur temporaire qui nous permet de faire la somme de la hauteur (*height*) et la profondeur (*depth*). On ajoute ensuite à cette hauteur totale deux fois l'épaisseur du trait et deux fois l'espace `\fboxsep`. Vous vous souvenez sans doute que les dimensions `\fboxrule` et `\fboxsep` définissent respectivement l'épaisseur du trait et l'espace entre la bordure et le contenu d'une boîte simple. Par conséquent, on a :

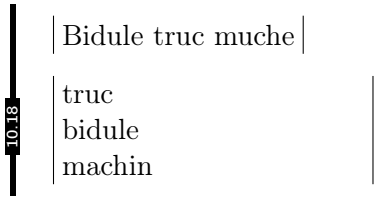
- `\largeurboitetitre` correspond à la largeur de la `\parbox` centrale augmentée de deux fois `\fboxsep` ;
- `\hauteurboitetitre` correspond à la hauteur totale augmentée de l'espace occupée par les deux traits horizontaux : `2(\fboxsep+\fboxrule)`.

On peut donc construire une première version de la commande :

```
\newcommand{\titleboxI}[2]{%
  ...
  \parbox{\fboxrule}{% le trait de gauche
    \rule{\fboxrule}{\hauteurboitetitre}}%
  \parbox{\largeurboitetitre}{% la boîte centrale
    \begin{flushleft}
      \usebox{\boitetitre}
    \end{flushleft}}%
  \parbox{\fboxrule}{% le trait de droite
    \rule{\fboxrule}{\hauteurboitetitre}}}}
```

Ce qui donnera pour l'instant :

```
\titleboxI{titre}{Bidule truc muche}

\titleboxI{encore}{%
  \parbox{4cm}{truc\\bidule\\machin}}
```



Il reste donc à modifier le contenu de la `\parbox` centrale pour ajouter les deux traits horizontaux, celui du bas, et celui du haut coupé par le titre. L'idée est d'entasser trois boîtes :

1. une boîte contenant le titre et des « traits ressorts » :
2. la boîte stockant le contenu (`\boitetitre`) ;
3. un trait de largeur `\largeurboitetitre`.

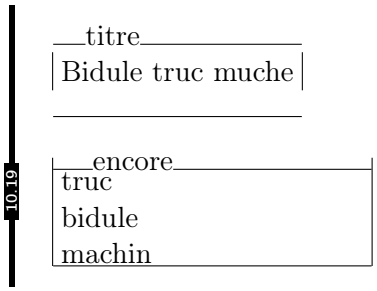
Voici une première approche :

```
\newcommand{\titleboxII}[2]{%
  ...
  \parbox{\largeurboitetitre}{% la boîte centrale
```

```
      \begin{flushleft}
        \makebox[\largeurboitetitre]{%
          \traitressort{\fboxrule}#1\traitressort[5]{\fboxrule}}\\
        \usebox{\boitetitre}\\
        \rule{\largeurboitetitre}{\fboxrule}
      \end{flushleft}}
    ...}
```

qui donnera :

```
\titleboxII{titre}{Bidule truc muche}

\titleboxII{encore}{%
  \parbox{4cm}{truc\\bidule\\machin}}
```



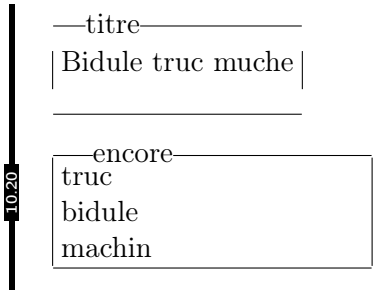
On dirait que c'est pas « tout à fait ça ». Il faudrait penser à faire en sorte que la commande \\ effectue un saut vertical équivalent à la dimension \fboxsep. On en profite au passage pour faire subir au titre une translation verticale vers le bas :

```
\newcommand{\titleboxIII}[2]{%
  ...
  \parbox{\largeurboitetitre}{% la boîte centrale
    \begin{flushleft}
      \makebox[\largeurboitetitre]{%
        \traitressort{\fboxrule}%
        \raisebox{-.5ex}[0pt][0pt]{#1}%
        \traitressort[5]{\fboxrule}}\\[\fboxsep]
      \usebox{\boitetitre}\\[\fboxsep]
      \rule{\largeurboitetitre}{\fboxrule}
    \end{flushleft}}
  ...}
```

ce qui donnera :

10

```
\titleboxIII{titre}{Bidule truc muche}

\titleboxIII{encore}{%
  \parbox{4cm}{truc\\bidule\\machin}}
```



On dirait que ça n'a pas arrangé grand chose... Il faut savoir que lorsque TEX entasse des boîtes en mode vertical, il insère de lui même des espaces entres ces boîtes de manière à ce que les lignes soit espacées de la longueur \baselineskip. On trouve dans le TEXbook, à la page 79 du chapitre traitant des *glues* :

> « Exception : no interline glue is inserted before or after a rule box. You can also inhibit interline glue by saying \nointerlineskip between two boxes. »

L'ordre `\nointerlineskip` résout donc le problème :

```
\newcommand{\titleboxIV}[2]{%
  ...
  \parbox{\largeurboitetitre}{% la boîte centrale
    \begin{flushleft}
      \makebox[\largeurboitetitre]{%
        \traitressort{\fboxrule}%
        \raisebox{-.5ex}[0pt][0pt]{#1}%
        \traitressort[5]{\fboxrule}}\\[\fboxsep]\nointerlineskip
      \usebox{\boitetitre}\\[\fboxsep]\nointerlineskip
      \rule{\largeurboitetitre}{\fboxrule}
    \end{flushleft}}
  ...}
```

ce qui donnera :

```
\titleboxIV{titre}{Bidule truc muche}

\titleboxIV{encore}{%
  \parbox{4cm}{truc\\bidule\\machin}}
```

titre
Bidule truc muche

encore
truc
bidule
machin

10.21

Ce qui répond au cahier des charges.

D'autres améliorations — laissées en guise d'exercice — peuvent être apportées à cette commande. On pourra par exemple définir un argument optionnel permettant de régler l'abaissement du titre (on a mis ici `-.5ex` en dur). Il est également possible de régler le rapport des traits entourant le titre. Enfin, il est tout à fait envisageable de régler l'espace autour du titre (ici il n'y en a pas).

10.7.5 Utilisation avec package minitoc

L'utilisation de la commande `\titlebox` précédemment définie, dans le package minitoc se fait simplement en revêtant le chapeau de monsieur Poirot. En inspectant à la loupe le fichier de style, on trouve la définition d'une commande nommée `\minitoc@`. J'ai simplement recopié le code de cette macro et inséré un appel à la merveilleuse commande `\titlebox`.

11 De nouveaux jouets

Sommaire

Je suis à mon bien-aimé, Et ses désirs se portent vers moi.
Viens, mon bien-aimé, sortons dans les champs, Demeurons dans les villages!
Dès le matin nous irons aux vignes, Nous verrons si la vigne pousse,
Si la fleur s'ouvre, Si les grenadiers fleurissent.
Là je te donnerai mon amour.

Le Cantique des cantiques Ct **7** 11.

NOUS PRÉSENTONS dans ce chapitre les outils qui ont été créés spécialement pour ce manuel. Pour comprendre la plupart des commandes et environnements définis ici, il est impératif d'avoir lu les deux précédents chapitres... Il est question dans ce chapitre de la manière dont le nota avec panneau danger a été créé, des lettrines apparaissant en début de chapitres, du sommaire, du glossaire, des onglets contenant le numéro du chapitre courant, et enfin de l'environnement permettant de produire du code LaTeX et son interprétation côte à côte.

11.1 Quelques bricoles

11.1.1 Arguments et convention typographique

Dans un document parlant de langage informatique, il est important de faire ressortir clairement les arguments de commande ou de fonction. Par exemple on écrira :

```
Pour compiler le fichier \bwarg{fichier} :
\begin{flushleft}
  \ttfamily latex \bwarg{fichier}
\end{flushleft}
```

Pour compiler le fichier ⟨*fichier*⟩ :

```
latex ⟨fichier⟩
```

La commande `\bwmarg` écrit son argument en *fonte penchée*, entre les symboles ⟨ et ⟩ produits respectivement par les commandes `\langle` et `\rangle` en mode mathé-

matique. De plus vous aurez sans doute remarqué qu'on peut utiliser une notation indicée comme dans l'exemple ci-dessous :

```
Pour copier des fichiers :
\begin{flushleft}\ttfamily
  cp \bwarg[1]{fichier} ... \bwarg[n]{fichier}
  \bwarg[dst]{fichier}
\end{flushleft}
```

Pour copier des fichiers :
cp ⟨*fichier*$_1$⟩ ... ⟨*fichier*$_n$⟩ ⟨*fichier*$_{dst}$⟩

La commande \bwarg[1] est définie comme suit :

```
\newcommand{\marg}[2][]{%
  {\normalfont%
    \textsl{$\langle$#2%
      \ifthenelse{\equal{#1}{}}{}% si l'argument optionnel est présent
      {$_\mathit{#1}$}% on l'affiche en  indice
      $\rangle$}}}%
```

La commande \normalfont permet de revenir à la fonte par défaut dans le document. Ce qui explique que « ⟨*fichier*⟩ » n'apparaît pas en fonte machine à écrire dans l'exemple 11.1.

Dans la version électronique du document — version lue sur un écran — il a été décidé, pour la mise en évidence, d'utiliser la *couleur* plutôt que les caractères ⟨ et ⟩. Ainsi on peut définir la commande \colarg :

```
\newcommand{\colarg}[2][]{{%
    \normalfont\color{blue!90}#2% un bleu
    \ifthenelse{\equal{#1}{}}{}{$_\mathit{#1}$}}}
```

On pourra ensuite grâce à un booléen habilement positionné, définir une commande générique \marg faisant appel à l'une ou l'autre des versions (noir & blanc ou couleur) :

```
\ifversionenligne
   \let\marg\colarg
\else
   \let\marg\bwarg
\fi
```

Cette construction fait appel à la commande \let de TEX présentée de manière lumineuse à la section 9.2.3 page 109.

11.1.2 Autour de la génération de l'index

Lorsque dans le texte du présent manuel, il est question d'une commande, d'un environnement, d'un package, d'une classe de document, etc. il est fait appel à une commande particulière insérant automatiquement une entrée dans l'index. Ainsi par exemple :

```
Le package \ltxpack{varioref} permet
d'utiliser la commande \ltxcom{vref}...
```

Le package varioref permet d'utiliser la commande \vref...

La commande \ltxpack est définie comme suit. Tout d'abord :

[1] Pour « black & white » argument...

```
\newcommand{\ltx@pack}[1]{%
  \upshape\textsf{#1}}
```

définissant la commande `\ltx@pack` permettant simplement de produire le nom du package en sans sérif. On définit ensuite :

```
\newcommand{\ltxpack}[1]{%
  \ltx@pack{#1}%
  \protect\index{extensions!\protect\texttt{#1}}%
  \protect\index{#1@\protect\textsf{#1 extension}}}
```

qui appelle la commande précédente, et qui insère deux entrées dans l'index. Une de la forme « nom du package extension » et l'autre comme sous-entrée de « extensions ». La commande `\protect` permet ici d'éviter les ennuis si la commande `\ltxpack` est elle-même en argument d'une autre commande. Dans un même ordre d'idée, la commande `\ltxcom` est définie tout d'abord par :

```
\newcommand{\ltx@com}[1]{%
  \texttt{\symbol{92}#1}}
```

permettant de produire en fonte machine à écrire, le nom de la commande précédé du caractère \. La commande `\symbol` est une commande LaTeX permettant d'insérer ici le 92e caractère de la fonte sélectionnée (en l'occurrence le backslash). On peut alors définir la commande finale :

```
\newcommand{\ltxcom}[1]{%
  \ltx@com{#1}%
  \index{#1@\protect\texttt{\symbol{92}#1}}}
```

qui appelle la commande précédente et introduit une entrée dans l'index. L'idée à retenir, c'est qu'il est peut être utile de définir des commandes pour insérer automatiquement des entrées dans l'index. On pourrait par exemple définir une commande :

```
\newcommand{\jargonanglais}[1]{%
  \emph{#1}%
  \index{#1}}
```

permettant à la fois de formater les mots du jargon en anglais, et de les insérer dans l'index, voire dans un index spécial. De même si un mot revient souvent dans un document on peut définir une commande pour le produire et l'insérer dans l'index. Par exemple dans ce manuel, on a défini :

```
\newcommand{\postscript}{%
  PostScript%
  \protect\index{PostScript}}
```


11.1.3 Des renvois

La version « papier » de ce document est parsemée de renvois comme celui-ci parlant de ▶glossaire◀ qui n'a strictement rien à voir avec le propos du moment[2] si ce n'est qu'il s'agit d'un renvoi. La commande mise en œuvre pour les renvois a été baptisée `\voir` et attend deux arguments :

§ D.3.6 p. 217 ◀

[2] Je veux dire que nous ne sommes pas en train de parler de glossaire, ou alors vous ne suivez pas du tout...

```
\voir{label cible}{texte objet du renvoi}
```

Par exemple, le renvoi précédent a été produit par :

```
\voir{chap-glossaire}{glossaire}
```

Toute la « difficulté » de la conception de cette commande réside dans l'orientation des triangles qui dépend de la parité de la page. Cette difficulté peut être levée grâce à l'utilisation du package chngpage comme expliqué au paragraphe 9.3.3 page 113.

Le reste concerne la mise en page des triangles. Pour cela deux commandes ont été définies, produisant chacunes les renvois dans la marge et les marques dans le texte. D'où la forme de la commande \voir :

```
\newcommand{\voir}[3][\S]{%
  \checkoddpage%
  \ifcpoddpage
    \v@irpageimpaire{#1}{#2}{#3}}{% renvoi sur page impaire
  \else
    \v@irpagepaire{#1}{#2}{#3}} % renvoi sur page paire
  \fi
```

On notera qu'outre les deux arguments obligatoires, la commande accepte un argument optionnel défini par défaut comme étant le caractère de paragraphe (§). Les deux commandes \v@irpageimpaire et \v@irpagepaire sont symétriques l'une de l'autre et ont pour objet :

1. d'encadrer le texte objet du renvoi par des triangles correctement orientés ;
2. de produire une note marginale avec la cible du renvoi.

Les triangles sont obtenus à l'aide de symboles contenus dans le package amssymb :

```
Oh les \og joulis\fg{} triangles :
$\blacktriangleleft$ et
$\blacktriangleright$ !
```

11.4 Oh les « joulis » triangles : ◀ et ▶ !

Le petit triangle à plat est obtenu grâce à la commande suivante :

```
\newcommand{\petit@triangle}{%
  \makebox[2.5pt]{\tiny$\blacktriangle$}}
```

On peut alors écrire :

```
Mais, il sont tous \petit@triangle{}petits
ces \petit@triangle{}triangles...
```

11.5 Mais, il sont tous ▲petits ces ▲triangles...

Il n'est pas inutile de noter l'usage de la boîte de deux points et demi de large pour faire croire à ce naïf de LaTeX que c'est la largeur effective de \blacktriangle. Ce qui n'est bien entendu pas le cas, mais contribue à coller ce « petit triangle » au mot qui le suit. Je sens que vous n'êtes pas convaincu, donc :

```
Mais, il sont tous {\tiny$\blacktriangle$}%
petits ces {\tiny$\blacktriangle$}triangles...
```

11.6 Mais, il sont tous ▲petits ces ▲triangles...

Voici finalement la commande permettant de faire un renvoi dans le cas d'une page paire :

```
\newcommand{\v@irpagepaire}[3]{%
  \petit@triangle#3{\small$\blacktriangleleft$}% le renvoi est à gauche
  \marginpar{{\small% on écrit en petit dans la marge
      % le numéro et la page de la cible du renvoi ...
      $\blacktriangleright$ #1~\ref{#2} p.~\pageref{#2}}}}
```

Pour la page impaire il faut faire la même chose que pour la page paire, mais en tenant compte que c'est une page impaire :-)

Dans la version « en ligne » les renvois apparaissent comme un lien hypertexte. Ceci est facilement réalisable grâce à la commande `\hyperref` du package éponyme. On aura donc quelque chose du genre :

```
\newcommand{\voir}[3][\S]{%
  \hyperref[#2]{#3}}
```

L'argument optionnel ne sert ici à rien d'autre qu'à assurer la compatibilité avec la version « papier » de la commande `\voir`.

11.1.4 Changement de marges

À plusieurs reprises dans ce document, j'ai eu recours à des changement de marges provisoires. C'est le cas par exemple des exemples de code LaTeX avec le résultat en face, ou pour les épigraphes. Pour ce faire, Marie-Paul Kluth[3] qui maintient la Faq française de LaTeX avait suggéré un environnement ressemblant à celui-ci :

```
\newenvironment{changemargin}[2]%
{\begin{list}{}{%
  \setlength{\listparindent}{\parindent}%
  \setlength{\itemindent}{\parindent}%
  \setlength{\leftmargin}{#1}%
  \setlength{\rightmargin}{#2}%
}\item }%
{\end{list}}
```

L'idée est donc de définir une liste dont on change les marges. Les deux arguments qu'attend cet environnement correspondent respectivement aux dimensions des marges gauche et droite. Une idée intéressante serait celle d'un environnement dans lequel les marges ont des dimensions différentes selon la parité de la page. Un tel environnement peut être défini comme suit :

```
\newenvironment{agrandirmarges}[2]{%
\begin{list}{}{%
    \setlength{\topsep}{0pt}%
    \setlength{\listparindent}{\parindent}%
    \setlength{\itemindent}{\parindent}%
    \setlength{\parsep}{0pt plus 1pt}%
    \checkoddpage%
    \ifcpoddpage
      \setlength{\leftmargin}{-#1}\setlength{\rightmargin}{-#2}
    \else
      \setlength{\leftmargin}{-#2}\setlength{\rightmargin}{-#1}
```

[3] Qui possède un lom prédestilé...

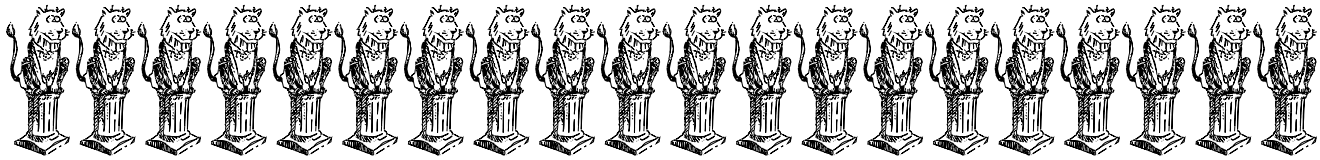


FIG. 11.1 – Une figure qui ne sert à rien si ce n'est à montrer qu'on peut momentanément changer les marges gauche et droite quand on a besoin de place...

```
    \fi}\item }%
{\end{list}}
```

Notez qu'on utilise ici la commande `\isodd` pour tester la parité de la page. La figure 11.1 montre un exemple d'utilisation de cet environnement avec le code suivant :

```
\begin{figure}[tb]
  \begin{agrandirmarges}{1cm}{2cm}
    % ici on a 1cm de plus côté « reliure »
    %        et 2cm de plus côté « bord »
    ...
    \caption{Une figure qui ne sert à rien...}
  \end{agrandirmarges}
\end{figure}
```

11.2 Des nota

Les pictogrammes[4] ont été «conçus» avec le logiciel xfig et sont représentés à la figure 11.2 en trois centimètres de large. Les «nota» insérés ça et là dans le

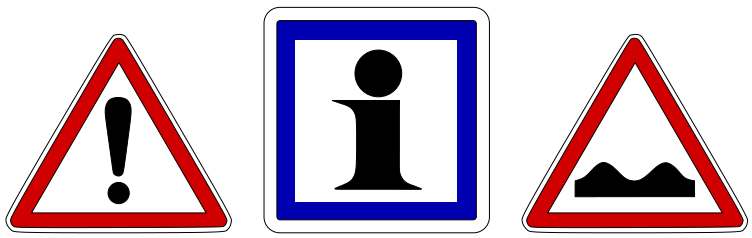


FIG. 11.2 – Les pictogrammes du manuel

document, ont été produits par un environnement défini par votre serviteur, basé sur une fonctionnalité de niveau TEX découverte lors de mes laborieuses lecture du TEXBook : la commande `\parshape`. Cette commande permet de donner une forme arbitraire à un paragraphe :

[4] L'idée de ces pictogrammes a été inspirée par la lecture de TEXbook, comme je l'explique en introduction.

```
\parshape=5
2cm 3cm
2cm 3cm
1cm 2cm
1cm 2cm
0pt \textwidth
a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a a a a a a a a
```

11.7

```
        a a a a a a a a a
      a a a a a a a a a a
   a a a a a a a
   a a a a a a a
a a a a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a
```

Le nombre suivant le signe ‘=’ permet de spécifier le nombre de lignes auxquelles on imposera une déformation. Suivent ensuite des couples de dimensions indiquant le retrait et la longueur de chaque ligne déformée. Dans l’exemple ci-dessus :

– les deux premières lignes auront un retrait de deux centimètres et mesureront chacune trois centimètres ;
– les deux lignes suivantes seront indentées d’un centimètre et mesureront deux centimètres ;
– la cinquième et dernière spécification détermine l’allure de toutes les lignes restantes : retrait de zéro centimètre et longueur de la ligne égale à la longueur prédéfinie `\textwidth`.

Pour insérer un nota dans un paragraphe, on va donc déplacer les deux premières lignes à l’aide de cette commande.

```
\setlength{\larnota}{.9cm}
\setlength{\largligne}{\textwidth-\larnota}
\parshape=3
\larnota\largligne
\larnota\largligne
0pt\textwidth
\noindent Attention ce paragraphe a uniquement
pour but de montrer que l’on peut décaler deux
lignes dans un paragraphe et ensuite continuer
comme si de rien n’était...
```

11.8

Attention ce paragraphe a uniquement pour but de montrer que l’on peut décaler deux lignes dans un paragraphe et ensuite continuer comme si de rien n’était...

Bon, il reste à essayer de mettre l’image dans le « trou » laissé par la commande `\parshape`. Essayons simplement :

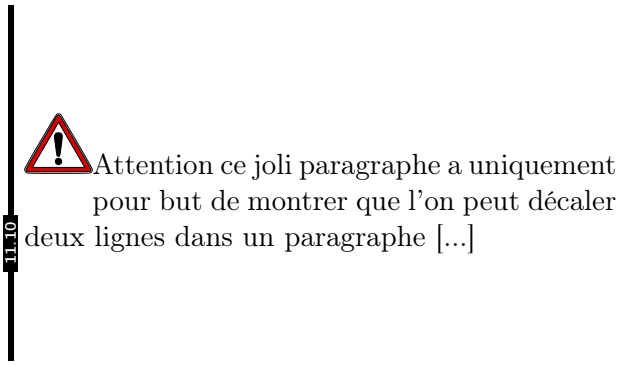
```
\setlength{\larnota}{.9cm}
\setlength{\largligne}{\textwidth-\larnota}
\parshape=3
\larnota\largligne\larnota\largligne
0pt\textwidth\noindent%
\includegraphics[width=\larnota]{\ficnota}
Attention ce paragraphe a uniquement
pour but de montrer que l’on peut décaler deux
lignes dans un paragraphe [...]
```

11.9

⚠ Attention ce paragraphe a uniquement pour but de montrer que l’on peut décaler deux lignes dans un paragraphe [...]

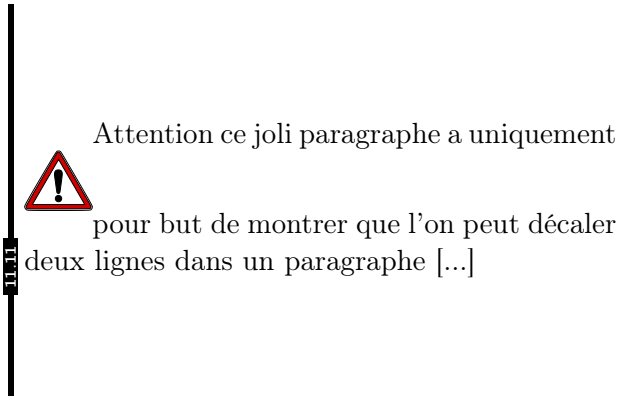
Évidemment, l’image se pose sur la ligne comme n’importe quel autre caractère. On la met dans une boîte de largeur nulle dont le contenu est aligné à droite :

```
\setlength{\larnota}{.9cm}
\setlength{\largligne}{\textwidth-\larnota}%
\parshape=3
\larnota\largligne\larnota\largligne%
0pt\textwidth\noindent%
\makebox[0pt][r]{%
  \includegraphics[width=\larnota]{\ficnota}}%
Attention ce joli paragraphe a uniquement
pour but de montrer que l'on peut décaler deux
lignes dans un paragraphe [...]
```



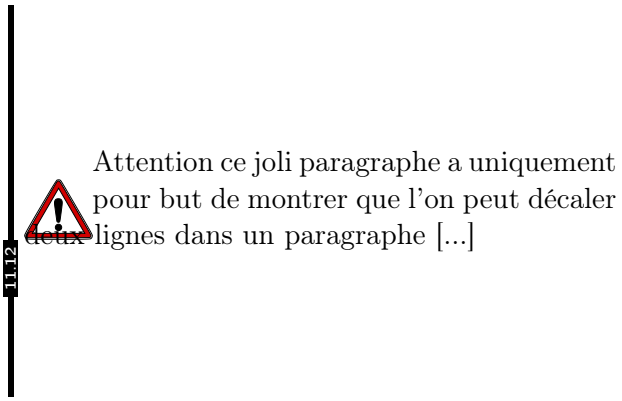
Il reste à faire subir au pictogramme, une translation verticale (le pictogramme est pratiquement carré, on se sert donc de la dimension `\indnota`) :

```
\setlength{\larnota}{.9cm}
\setlength{\largligne}{\textwidth-\larnota}
\parshape=3
\larnota\largligne\larnota\largligne
0pt\textwidth\noindent%
\raisebox{-\larnota}{%
  \makebox[0pt][r]{%
  \includegraphics[width=\larnota]{\ficnota}}}%
Attention ce joli paragraphe a uniquement
pour but de montrer que l'on peut décaler deux
lignes dans un paragraphe [...]
```



Bon, encore raté, il faut faire croire à LaTeX que la boîte qu'on translate verticalement est de taille nulle :

```
\setlength{\larnota}{.9cm}
\setlength{\largligne}{\textwidth-\larnota}
\parshape=3
\larnota\largligne\larnota\largligne
0pt\textwidth\noindent%
\raisebox{-\larnota}[0pt][0pt]{%
  \makebox[0pt][r]{%
  \includegraphics[width=\larnota]{\ficnota}}}%
Attention ce joli paragraphe a uniquement
pour but de montrer que l'on peut décaler deux
lignes dans un paragraphe [...]
```



On y est presque. Deux ajustements sont nécessaires :

- la boîte est un peu trop basse, puisque la ligne de référence est le bas de la ligne du texte. Par conséquent on peut enlever `1ex` (la hauteur d'un caractère) à la translation ;
- il serait bon d'ajouter un espace entre le pictogramme est le texte. On définit pour cela une longueur `\padnota`.

```
\setlength{\padnota}{5pt}
\setlength{\larnota}{.9cm}
\setlength{\indnota}{\larnota+\padnota}
\setlength{\largligne}{\textwidth-\indnota}
\parshape=3
\indnota\largligne\indnota\largligne
0pt\textwidth\noindent%
\raisebox{-\larnota+2.2ex}[0pt][0pt]{%
  \makebox[0pt][r]{%
  \includegraphics[width=\larnota]{\ficnota}%
  \hspace*{\padnota}}}%
Attention ce joli paragraphe a uniquement
pour but de montrer   qu'on peut décaler deux
lignes dans un paragraphe [...]
```

Attention ce joli paragraphe a uniquement pour but de montrer qu'on peut décaler deux lignes dans un paragraphe [...]

11.13

Et ouala ! Il ne reste « qu'à » modifier ce code pour créer un nouvel environnement. La technique choisie ici est de se baser sur l'environnement `list` présenté au paragraphe 9.5 page 119. Le code complet de l'environnement est le suivant :

```
\newenvironment{pictonote}[1]{% on passe le nom du fichier en argument
  \begin{list}{}{%
      \setlength{\labelsep}{0pt}%
      \setlength{\rightmargin}{15pt}}
  \item%
    \setlength{\indentationnota}{%
      \@totalleftmargin+\largeurnota+\paddingnota}%
    \setlength{\largeurlignenota}{%
      \linewidth-\largeurnota-\paddingnota}%
    \parshape=3%
    \indentationnota\largeurlignenota%
    \indentationnota\largeurlignenota%
    \@totalleftmargin\linewidth%
    \raisebox{-\largeurnota+2.2ex}[0pt][0pt]{%
      \makebox[0pt][r]{%
        \includegraphics[width=\largeurnota]{#1}%
        \hspace{\paddingnota}}}%
    \ignorespaces}{%
  \end{list}}
```

On notera l'utilisation de la dimension `\@totalleftmargin` permettant d'obtenir la largeur de la marge de gauche dans une liste, en tenant compte d'éventuelles imbrications. En effet la dimension `\leftmargin` dans une liste correspond à la marge de gauche *relativement* à l'environnement contenant la dite liste.

On pourra ensuite utiliser cet environnement en lui passant en paramètre un fichier particulier, ou mieux définir un nouvel environnement par exemple :

```
\newenvironment{nota}{%
  \begin{pictonote}{votre fichier}}{\end{pictnote}}
```

Pour terminer cette section sur les nota il faut savoir que cet environnement a un défaut : Si un nota contient un saut de paragraphe, un espace est de nouveau laissé libre pour un pictogramme

Oui oui comme dans celui-ci, vous voyez bien qu'il y a un emplacement prévu pour le pictogramme, alors qu'ici on n'a pas vraiment l'intention d'en mettre un, non ? Pour éviter ce genre de désagrément il faut réinitialiser les retraits au niveau TEX grâce aux incantations vaudoues suivantes :

```
\par\parshape=1\@totalleftmargin\linewidth
```

qui pourra faire l'objet d'une commande à insérer manuellement au début de chaque nouveau paragraphe, comme je viens de le faire ici :-)

11.3 Des citations

11.3.1 Épigraphes

Les épigraphes provocatrices de ce manuel ont été produites par un environnement que j'ai nommé `epigraphe`. Par exemple au début du code LATEX du chapitre 2, on trouve :

```
\chapter{Ce qu'il faut savoir}
\label{chap-savoir}
\begin{epigraphe}{Les proverbes Pr \textbf{21} 11}
  Quand on châtie le railleur, le simple s'assagit ;\\
  quand on instruit le sage, celui-ci gagne en savoir.
\end{epigraphe}
```

L'environnement `epigraphe` est défini comme suit :

```
\newenvironment{epigraphe}[1]
{% clause begin
  \vspace*{-1.5cm}%
  \small\sffamily% mise en évidence
  \savebox{\nomepigraphe}{#1}% une boîte pour sauvegarder
                             % l'origine de la citation
  \slshape% tout est penché
  \begin{changemargin}{0pt}{-2cm}% on se met au large
    \begin{flushright}}% tout est poussé à droite
    {% clause end
      \\[4pt]\usebox{\nomepigraphe}.% insertion de l'origine
    \end{flushright}%
  \end{changemargin}\par\vspace*{0.6cm}}
```

Il faut bien entendu déclarer la boîte qu'on utilise pour sauvegarder l'origine de notre citation :

```
\newsavebox{\nomepigraphe}
```

Les blanc verticaux (`\vspace`) insérés avant et après cet environnement permettent de caler correctement l'épigraphe entre le début du chapitre et la minitable des matières.

11.3.2 Citations

Quelques citations parsèment le manuel que vous avez sous les yeux. Elles ont été produites avec un environnement fait maison :

```
xxxxxxxxxxxxx
\begin{unecitation}[Georges \textsc{Bataille}]
  La vieillesse renouvelle la terreur à
  l'infini. Elle ramène l'être sans finir au
  commencement. Le commencement qu'au bord
  de la tombe j'entrevois est le \emph{porc}
  qu'en moi la mort ni l'insulte ne peuvent
  tuer. La terreur au bord de la tombe est
  divine et je m'enfonce dans la terreur dont
  je suis l'enfant.
\end{unecitation}
xxxxxxxxxxxxxx
```

xxxxxxxxxxxxx

1114

« La vieillesse renouvelle la terreur à l'infini. Elle ramène l'être sans finir au commencement. Le commencement qu'au bord de la tombe j'entrevois est le porc *qu'en moi la mort ni l'insulte ne peuvent tuer. La terreur au bord de la tombe est divine et je m'enfonce dans la terreur dont je suis l'enfant. »*

Georges Bataille

xxxxxxxxxxxxxx

Nous allons créer pas à pas cet environnement de manière à mettre le doigt sur quelques problèmes classiques auxquels on peut être confronté avec monsieur LaTeX. On va définir l'environnement citation en s'appuyant sur celui du paragraphe 9.6 page 125 (permettant de faire une remarque encadrée) et sur celui du § 4.5.2 page 64 qui produit une citation avec son auteur :

```
% un boite pour l'auteur de la  citation
\newsavebox{\auteurcitation}
\newsavebox{\boitecitation}
\newenvironment{citationi}[1]{% clause begin
  % on sauve l'argument 1 pour l'auteur
  \savebox{\auteurcitation}{#1}%
  \begin{lrbox}{\boitecitation}
    \begin{minipage}{.8\linewidth}
      \small\slshape}% on passe en petit et penché
  {% clause end : on pousse l'auteur de la  citation  à  droite
    \par\mbox{}\hfill\usebox{\auteurcitation}
    \end{minipage}
  \end{lrbox}
  \begin{center}
    \usebox{\boitecitation}
  \end{center}
```

Ce qui donne :

```
Avant avant avant avant avant avant avant
avant avant avant avant avant avant avant
\begin{citationi}{%
    Michel \textsc{Bakounine}, 1845}
  Dans presque tous les pays les femmes
  sont esclaves ; tant qu'elles ne seront
  pas complètement émancipées, notre propre
  liberté sera impossible.
\end{citationi}
Après après après après après après après
après après après après après après après
```

Avant avant avant avant avant avant avant avant avant avant avant avant avant avant

1115

Dans presque tous les pays les femmes sont esclaves ; tant qu'elles ne seront pas complètement émancipées, notre propre liberté sera impossible.

Michel Bakounine, 1845

Après après après après après après après après après après après après après après

11

On peut également changer l'indentation du paragraphe dans la minipage (qui vaut par défaut `0pt` dans cet environnement) en modifiant dans l'environnement la valeur de la longueur `\parindent` :

```
...
\begin{lrbox}{\boitecitation}
  \begin{minipage}{.8\linewidth}%
    \setlength{\parindent}{10pt}%      ← pour indenter la 1re ligne
...
```

On insère ensuite des guillemets en début et fin de citation, avec le code suivant :

```
...
\begin{minipage}{.8\linewidth}%
  \setlength{\parindent}{10pt}%
  \small\slshape« \ignorspaces}% on passe en petit et penché
{\unskip »
  \par\mbox{}\hfill\usebox{\auteurcitation}
...
```


On se réfèrera aux paragraphes 9.2.1 et 9.2.1 page 108 pour la signification des commandes `\ignorespaces` et `\unskip`.

Ce qui donne :

```
Avant avant avant avant avant avant avant
avant avant avant avant avant avant avant
\begin{citationiii}{%
    Michel \textsc{Bakounine}, 1845}
  Dans presque tous les pays les femmes
  sont esclaves ; tant qu'elles ne seront
  pas complètement émancipées, notre propre
  liberté sera impossible.
\end{citationiii}
Après après après après après après après
après après après après après après après
```

Avant avant avant avant avant avant avant avant avant avant avant avant avant avant

« Dans presque tous les pays les femmes sont esclaves ; tant qu'elles ne seront pas complètement émancipées, notre propre liberté sera impossible. »

Michel Bakounine, 1845

Après après après après après après après après après après après après après après

Ensuite — on y est presque — on se propose de rendre l'argument « auteur de la citation » optionnel de manière à pouvoir produire une citation sans auteur si besoin est. L'idée est de déclarer un booléen de manière à mémoriser le fait qu'un auteur est présent ou pas :

```
\newboolean{auteurcitationpresent}
```

On modifie ensuite la définition de l'environnement comme suit :

```
\newenvironment{unecitation}[1][] {% argument optionnel vide par défaut
  % Clause begin :
  % on note si on a un auteur pour la citation ou pas
  \ifthenelse{\equal{#1}{}}{%
    \setboolean{auteurcitationpresent}{false}}{%
    \setboolean{auteurcitationpresent}{true}%
    \savebox{\auteurcitation}{#1}}% on le sauve si nécessaire
...
```

Puis on modifie la clause `\end` de l'environnement en insérant l'auteur uniquement s'il est présent en argument :

```
{»% clause end de l'environnement
  % s'il y a un auteur on le met poussé tout à droite
  \ifthenelse{\boolean{auteurcitation}}%
  {\par\nopagebreak\hfill\usebox{\auteurcitation}}
  {}% sinon on ne fait rien ...
```

Enfin, on met la citation sur un fond. On peut par exemple déclaré cette couleur grâce au package xcolor :

```
\definecolor{coulcitation}{rgb}{0.60,0.70,0.90}%
```

Il suffit alors de remplacer la commande `\usebox` de la clause end par quelque chose du genre :

```
\setlength{\fboxsep}{10pt}%
\colorbox{coulcitation}{\usebox{\boitecitation}}
```

Ce qui donnera finalement :

```
avant avant avant avant avant avant avant
\begin{citationiv}[Pierre \textsc{Desproges}]
  Il était tellement obsédé qu'à la fin
  il sautait même des repas.
\end{citationiv}
Après après après après après après après
```

avant avant avant avant avant avant avant

> *« Il était tellement obsédé qu'à la fin il sautait même des repas. »*
> Pierre Desproges

Après après après après après après après

11.4 Des lettrines

Les documents soignés font souvent appel aux *lettrines* qui permettent, selon les règles d'usage en typographie, de produire la première lettre du chapitre en gros, ainsi que le mot ou groupe de mots qui suit. Par exemple :

```
\lettrine{Les documents} soignés font souvent
appel aux lettrines, qui selon les règles
d'usage en typographie...
```

Les documents soignés font souvent appel aux lettrines, qui selon les règles d'usage en typographie...

Il y a deux difficultés dont une a déjà été surmontée :

- comment appliquer un traitement à une seule lettre d'un argument d'une commande ;
- comment décaler les lignes pour laisser la place à la lettrine. On utilisera pour cela la commande `\parshape` comme pour le nota (§ 11.2 page 158).

11.4.1 La commande `\glurps` ou un pas vers TEX

Au niveau de LaTeX c'est la commande `\newcommand` qui permet de créer de nouvelles commandes. Une des limitations de LaTeX pour ce qui concerne la création de commandes réside dans le fait que les délimiteurs d'arguments sont toujours les caractères `{` et `}`. Comme nous le verrons un peu plus bas, au niveau de TeX, cette contrainte n'existe pas. En effet, pour créer une commande avec TeX, on peut écrire :

```
\def\bidule{machin truc}
\bidule{} et \bidule
```

machin truc et machin truc

Avec un ou plusieurs arguments, on écrira

```
\def\bidule#1#2{(1=#1) et (2=#2)}

\bidule{ab}{cd}
```

(1=ab) et (2=cd)

On note qu'en TeX, on écrit dans la définition les arguments dans l'ordre. Ce qui est intéressant et que l'on va exploiter pour notre lettrine, peut être illustré par les exemples suivants d'utilisation de la commande `\bidule` :

```
\def\bidule#1#2{(1=#1) et (2=#2)}

\bidule abcd

\bidule ab cd
```

(1=a) et (2=b)cd
(1=a) et (2=b) cd

On remarque donc que si on ne délimite pas explicitement les arguments avec les caractères `{` et `}`, le premier argument (`#1`) est remplacé par le premier caractère rencontré, le deuxième argument (`#2`) par le deuxième caractère, etc. Encore plus intéressant, on peut définir très souplement le format d'appel de la commande, par exemple :

```
\def\bidule#1#2/{(1=#1) et (2=#2)}
```

Ici, on indique que pour appeler la commande `\bidule` il faut lui faire suivre deux arguments suivis du caractère `/`.

```
\bidule abc/d

\bidule ab/cd
```

11.22

(1=a) et (2=bc)d
(1=a) et (2=b)cd

Par conséquent cette dernière commande prendra comme premier argument, le premier caractère rencontré, et comme deuxième : tout ce qu'elle trouve jusqu'au caractère `/`. On peut donc créer une ébauche de commande pour une lettrine :

```
\def\glurps#1#2/{{\Huge#1}\textsc{#2}}
\newcommand{\lettrinedev}[1]{\glurps#1/}
\lettrinedev{Bon bé} ouala le travail
```

11.23

BON BÉ ouala le travail

On va même pousser le vice jusqu'à mettre la grosse lettre un peu plus bas :

```
\def\glurps#1#2/{%
  {\Huge#1}%
  \raisebox{\baselineskip}{\textsc{#2}}}
\newcommand{\lettrinedev}[1]{\glurps#1/}
\lettrinedev{Bon bé} ouala le travail
```

11.24

B
ON BÉ
ouala le travail

11.4.2 Insertion de la lettrine dans un paragraphe

Pour insérer la lettrine dans un paragraphe on aura recours à la commande `\parshape`. La figure 11.3 montre que l'on doit définir deux dimensions pour insérer la lettrine :

1. l'indentation de la première ligne, correspondant à la largeur de la « grosse lettre » plus celle de la suite de la lettrine ;
2. l'indentation de la deuxième ligne, correspondant à la largeur de la « grosse lettre » éventuellement augmentée d'une espace pour aérer un peu.

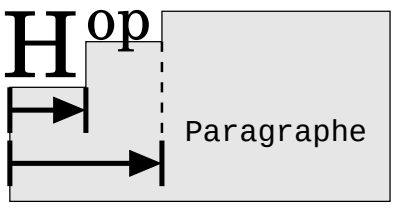


FIG. 11.3 – Insertion de la lettrine dans un paragraphe

On définit les dimensions suivantes :

– `\indletH` et `\larligH` respectivement l'indentation et la largeur de la première ligne (la ligne « du Haut ») du paragraphe contenant une lettrine ;
– `\indletB` et `\larligB` la même chose pour la deuxième ligne (la ligne « du Bas »)

Ceci permet d'écrire quelque chose du genre :

```
\setlength{\indletB}{.8cm}% au pif
\setlength{\larligB}{\textwidth-\indletB}
\setlength{\indletH}{1.5cm}% au pif
\setlength{\larligH}{\textwidth-\indletH}
\parshape=3
\indletH\larligH
\indletB\larligB
0pt\textwidth
\noindent Ce paragraphe est prêt à recevoir
une jolie lettrine qui occupera deux lignes
environ voire même exactement...
```

11.25

Ce paragraphe est prêt à recevoir une jolie lettrine qui occupera deux lignes environ voire même exactement...

L'emplacement est prêt, il reste à insérer la « grosse lettre » et ce qui suit à la bonne place. On commence par créer une commande pour la fonte à utiliser :

```
\DeclareFixedFont{%
  \lettrinefont}{T1}{ppl}{m}{n}{2\baselineskip}
{\lettrinefont Pour écrire les grosses lettres}
```

11.26

Pour écrire les grosses lettres

Ensuite on va modifier la commande `\glurps`[5] et la commande `\lettrine` pour qu'elles calculent elles-mêmes les dimensions définies ci-avant (`\indletH`, `\larligH`, ...).

```
\newsavebox{\lalettrine}% une boîte pour la lettrine
\def\creerlettrine#1#2/{%
  \savebox{\lalettrine}{%
    {\lettrinefont#1}\raisebox{\baselineskip}{\textsc{#2}}}%
  \settowidth{\indletB}{{\lettrinefont#1}}%
  \settowidth{\indletH}{\usebox{\lalettrine}}}
```

La commande `\creerlettrine` (digne héritière de `\glurps`) sauve donc la lettrine dans une boîte et en profite pour sauvegarder la largeur de la « grosse lettre » dans la dimension `\indletB`, et la dimension de l'ensemble dans `\indletH`. Tentons maintenant, une première version de la lettrine :

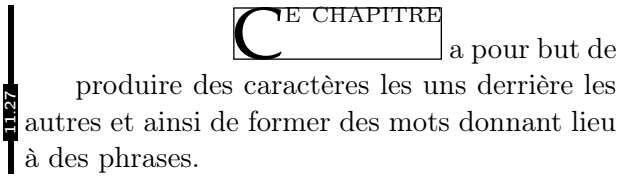
```
\newcommand{\lettrineI}[1]{%
  \creerlettrine#1/%
  \setlength{\larligH}{\textwidth-\indletH}%
  \setlength{\larligB}{\textwidth-\indletB}%
  \parshape=3\indletH\larligH\indletB\larligB%
  0cm\textwidth%
  \noindent\usebox{\lalettrine}}
```

Qui donne :[6]

[5] Vous pourrez notez qu'il est tout à fait inadmissible d'utiliser des noms aussi ridicules pour les commandes que vous serez amené à définir...

[6] Des traits ont été ajoutés pour bien situer la boîte englobant la lettrine.

```
\lettrineI{Ce chapitre} a pour but de produire
des caractères les uns derrière les autres et
ainsi de former des mots donnant lieu à des
phrases.
```

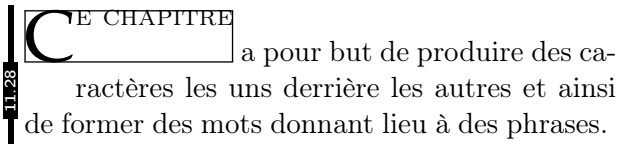


Les lecteurs très attentifs auront noté que ce nota n'est pas à la bonne place. Il semblerait que des translations horizontales et verticales soient nécessaires. On commencera ici par se décaler vers la gauche. À la dernière ligne de la définition de la commande `\lettrine`, on écrira donc :

```
\noindent\hspace{-\indletH}% décalage équivalent à la largeur totale
```

qui produira :

```
\lettrineII{Ce chapitre} a pour but de produire
des caractères les uns derrière les autres et
ainsi de former des mots donnant lieu à des
phrases.
```

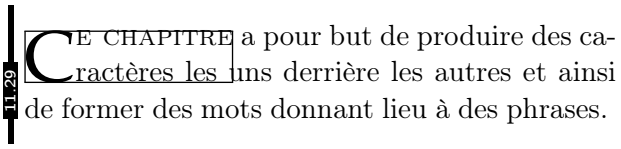


Puis on intègre la translation verticale :

```
\noindent\hspace{-\indletH}%
\raisebox{-\baselineskip}[0pt][0pt]{\usebox{\lalettrine}}
```

qui produira :

```
\lettrineIII{Ce chapitre} a pour but de
produire des caractères les uns derrière les
autres et ainsi de former des mots donnant
lieu à des phrases.
```



Les lecteurs encore éveillés auront remarqué que la « grosse lettre » est un peu trop rapprochée du texte. On peut donc augmenter légèrement la dimension `\indletB`. Voici finalement le code de la lettrine :

```
\newcommand{\lettrine}[1]{%
  \creerlettrine#1/%
  \addtolength{\indletB}{3pt}% pour avoir un peu d'espace
  \setlength{\larligH}{\textwidth-\indletH}%
  \setlength{\larligB}{\textwidth-\indletB}%
  \parshape=3%
  \indletH\larligH\indletB\larligB0cm\textwidth%
  \noindent\hspace{-\indletH}%
  \raisebox{-\baselineskip}[0pt][0pt]{\usebox{\lalettrine}}}
```

11.5 Un sommaire

C'est une pratique courante que le sommaire d'un document rédigé en français soit placé en tête de document et contienne un condensé de la table des matières. Cette dernière est quant à elle généralement insérée à la fin. En fouillant dans le fichier `book.cls` on trouve une instruction commune aux commandes `\tableofcontents` et `\listoffigures` : la commande `\@starttoc`.

```
\@starttoc{toc}
```

pour la commande `\tableofcontents` et :

```
\@starttoc{lof}
```

pour la commande `\listoffigures`. La commande interne de LaTeX `\@starttoc` permet de commencer un table (matières, figures, etc.) à partir d'un fichier auxiliaire portant l'extension donné en argument, ici `toc` pour la table des matières et `lof` pour la liste des figures. Nous avons donc créé dans un premier temps la commande `\sommaire` comme suit :

```
\newcommand{\sommaire}{%
  \chapter*{Sommaire}
  \@starttoc{som}}
```

Le fichier portant l'extension `som` contiendra les entrées du sommaire. L'étape suivante consiste à remplir le fichier `som`. Pour cela, il faut savoir que les commandes `\chapter`, `\section`, etc. font toutes appel à une commande pour insérer une entrée dans la table des matières. Ainsi quand on écrit :

```
\section{Bidule truc muche}
```

il sera fait appel à la commande :

```
\addcontentsline{toc}{section}{Bidule truc muche}
```

pour insérer le titre dans la table des matières. De même lorsqu'on écrit :

```
\chapter{Machin chose}
```

il sera fait automatiquement appel à :

```
\addcontentsline{toc}{chapter}{Machin chose}
```

Cette dernière commande écrit dans le fichier de table des matières (portant donc l'extension `toc`) la ligne :

```
\contentsline{chapter}{\numberline {1}Machine chose}{3}{chapter.1}
```

Et (Hercule Poirot a eu du pain sur la planche), il se trouve que cette dernière commande, fait finalement appel à une commande de la forme :

```
\l@type d'entrée
```

pour produire l'entrée dans la table. Par exemple la commande précédente fait appel à `l@chapter`. Ceci parce qu'on trouve la définition suivante :

```
\def\contentsline#1{\csname l@#1\endcsname}
```

dans `latex.ltx`. Nous passons volontairement sous silence le détail de ces commandes, et on se contentera ici de savoir que la commande `\l@section` produit une entrée de table de matières pour les sections, `\l@subsection` produit une entrée pour les subsections, etc. Pour finir notre sommaire, il reste à écrire dans le fichier de sommaire. Nous avons choisi d'insérer dans le sommaire les titres de parties, chapitres et sections. Pour arriver à nos fins, nous avons ajouté cette insertion à la commande `\addcontentsline` originale :

```
% sauvegarde de l' originale
\let\aclORIG\addcontentsline
% redéfinition
```

```
\renewcommand{\addcontentsline}[3]{%
  \aclORIG{#1}{#2}{#3}% appel de l'originale
  \ifthenelse{% on insère sections, chapitres et parties
    \equal{#2}{section} \or \equal{#2}{chapter}
                        \or \equal{#2}{part}}{%
    \aclORIG{som}{#2}{#3}}{}}
```

Il faut noter que notre sommaire sera mis en page exactement de la même manière que la table des matières, puisque tous deux font appel à la même commande interne :

```
\aclORIG{som}{section}{...}
```

c'est donc la commande `\l@section` qui sera appelée pour mettre en page l'entrée. Pour mettre en page le sommaire différemment, il aurait fallu écrire :

```
\aclORIG{som}{somsection}{...}
```

puis définir la commande `\l@somsection` pour mettre en page les entrées de sections dans le sommaire...

11.6 Un glossaire

Il est souvent utile d'agrémenter un document d'un glossaire ayant pour but de rendre moins mystérieux les termes du jargon qu'on appelle aussi vocabulaire technique. Nous proposons ici une méthode s'appuyant sur le programme makeindex présenté au paragraphe 6.3 page 82 et sur le paragraphe 10.1 page 127 présentant les outils permettant de changer l'allure de l'index.

11.6.1 Tordre le cou à makeindex

De même que pour la création d'un index, on doit mettre dans le préambule du document la commande `\makeglossary`. Cette commande demande à monsieur LaTeX de bien vouloir créer un fichier portant l'extension `glo`, réceptacle des entrées de glossaire. On pourra écrire :

```
\glossary{machin chouette chose}
```

pour ajouter « machin chouette chose » dans le glossaire. En réalité cette commande a pour effet d'ajouter dans le fichier d'extension `glo`, la ligne :

```
\glossaryentry{machin chouette chose}{n° page}
```

La phase suivante consiste à produire le glossaire proprement dit grâce au programme makeindex et au fichier d'entrées de glossaire :

```
makeindex -sstyle glossaire document.glo -odocument.glx
```

qui produit, en utilisant comme fichier de style `gglo.ist` le glossaire suivant :

```
\begin{theglossary}
\item machin chouette chose\pfill 27
\end{theglossary}
```

On note donc que le glossaire — qu'il va falloir insérer explicitement dans notre document — est constitué par l'environnement `theglossary` et que chaque entrée donne lieu à un `\item` et son numéro de page (ici 27 pour l'exemple). Pour arriver à nos fins, il nous faudra :

1. définir l'environnement `theglossary` car rien n'est fait par défaut dans LaTeX ;
2. produire les entrées avec un terme suivi de sa définition, sous la forme :

```
\begin{theglossary}
\item[terme] blabla de définition
\end{theglossary}
```

3. supprimer les numéros de pages puisqu'ils n'apparaissent pas dans un glossaire.

Ces trois tâches font l'objet des paragraphes suivants.

11.6.2 Un environnement pour le glossaire

Au paragraphe 9.5.6 page 124 du chapitre 9, nous avons proposé une liste particulière permettant de présenter les entrées dans une boîte avec une ombre. La définition était la suivante :

```
\newenvironment{leglossaire}{\begin{list}{}{%
      \setlength{\labelwidth}{.5\textwidth}%
      \setlength{\labelsep}{-.8\labelwidth}%
      \setlength{\itemindent}{\parindent}%
      \setlength{\leftmargin}{25pt}%
      \setlength{\rightmargin}{0pt}%
      \setlength{\itemsep}{.8\baselineskip}%
      \renewcommand{\makelabel}[1]{\boiteentreeglossaire{##1}}}}
  {\end{list}}
```

où `\boiteentreeglossaire` est :

```
\newcommand{\boiteentreeglossaire}[1]{%
  \parbox[b]{\labelwidth}{%
    \setlength{\fboxsep}{3pt}%
    \setlength{\fboxrule}{.4pt}%
    \shadowbox{\sffamily#1}\\\hfill\mbox{}}}
```

On se reportera au paragraphe mentionné ci-dessus pour les explications de ces commandes. En tous les cas on aura :

```
\begin{leglossaire}
\item[Sphère] Patate bien régulière.
\end{leglossaire}
```

Sphère

Patate bien régulière.

11.6.3 Produire le fichier `.glx`

De manière à ce que makeindex produise le fichier d'extension `glx` avec cet environnement on doit écrire le fichier de style suivant :

```
preamble "\n\\begin{leglossaire}"
postamble "\n\\end{leglossaire}"
```

Doit également apparaître dans ce fichier de style — que l'on nommera par exemple `glossaire.ist` — le mot clef désignant les entrées de glossaire dans le fichier `.glo` :

```
keyword "\\glossaryentry"
```

Pour que chaque entrée soit constituée d'un terme et de sa définition, nous avons défini la commande suivante :

```
\newcommand{\entreeglossaire}[2]{\glossary{[#1] #2}}
```

De telle sorte que :

```
\entreeglossaire{Sphere}{Patate bien régulière.}
```

aura pour effet d'écrire :

```
\glossary{[Sphère] Patate bien régulière.}
```

dans le fichier `.glo`. Après appel de makeindex :

```
makeindex -s glossaire.ist document.glo -odocument.glx
```

on aura dans le fichier `.glx` :

```
\begin{leglossaire}
\item [Sphère] Patate bien régulière., n° de page
\end{leglossaire}
```

Si vous m'avez bien suivi jusqu'ici, vous devriez râler et me dire : « faudrait p'têt' voir à enlever cette vilaine virgule et ce numéro de page... » Certes. La virgule est automatiquement mise par makeindex comme délimiteur « de premier niveau » entre une entrée d'index et le numéro de page où apparaît cette entrée. Pour utiliser autre chose qu'une virgule (ici rien en l'occurrence) on écrit dans le fichier de style :

```
delim_0 ""
```

Il reste à régler le cas du numéro de page. La solution que j'ai adoptée consiste à utiliser une commande « absorbante » en guise de mise en forme du numéro de page (cf. § 6.3.3 page 83). Voici la commande`\entreeglossaire` modifiée :

```
\newcommand{\pasdenumerodepage}[1]{}% « mange » l'argument
\newcommand{\entreeglossaire}[2]{%
  \glossary{[#1] #2|pasdenumerodepage}}
```

ceci permet de créer le fichier `.glx` de la forme :

```
\begin{leglossaire}
\item [Sphère] Patate bien régulière.\pagedenumerodepage{27}
\end{leglossaire}
```

qui enverra le numéro de page (ici 27 pour l'exemple) dans le vide intersidéral.

11.6.4 Recollons les morceaux

Pour en finir avec le glossaire, il nous faut créer une commande suffisamment souple qui aura essentiellement pour but de produire un chapitre contenant le glossaire, c'est-à-dire d'insérer dans le document le fichier `.glx`. On peut décomposer les tâches à effectuer par la commande en question, comme suit :

- créer un nouveau chapitre (avec comme titre « glossaire ») ;
- lire les entrées de glossaire (commande `\entreeglossaire`) depuis un fichier ;
- insérer le fichier `.glx`

Voici la commande effectuant ces traitements :

```
\newcommand{\printglossary}[1][glossaire.tex]{%
  \chapter*{Glossaire}
  \label{chap-glossaire}
  \markboth{Glossaire}{Glossaire}% entête de page
  % insertion dans la table des matières :
  \addcontentsline{toc}{chapter}{Glossaire}
  % insertion des entrées de glossaire :
  \InputIfFileExists{#1}{%
    \typeout{Données du glossaire}}{%
    \typeout{Pas de fichier glossaire.tex}}
  % insertion du fichier .glx
  \InputIfFileExists{\jobname.glx}{%
    \typeout{Glossaire trié}}{%
    \typeout{Pas de fichier \jobname.glx}}
}
```

On notera les points suivants concernant cette commande :

- le fichier d'entrée de glossaire est par défaut le fichier nommé `glossaire.tex` ;
- on a décidé d'insérer le glossaire dans la table des matières ;
- pour insérer un fichier, on fait appel à la commande `\InputIfFileExists` définie dans le format LaTeX ayant la forme suivante :

  ```
  \InputIfFileExists{fichier à inclure}
  {code LaTeX si le fichier existe}
  {code LaTeX si le fichier n'existe pas}
  ```

- Enfin `\jobname` contient le nom du fichier (ou document maître) en cours de compilation et la commande `\typeout` affiche un message sur la console de compilation.

Enfin, le fichier `glossaire.ist` contient finalement :

```
delim_0 ""
preamble "\n\\begin{leglossaire}"
postamble "\n\\end{leglossaire}"
keyword "\\glossaryentry"
```

11.7 Des onglets

J'avais initialement prévu de mettre en page ce document sur du papier plus petit que le standard A4 pour ensuite le massicoter. D'où l'idée de créer des onglets, petits carrés colorés se retrouvant au ras de la feuille après massicotage. Le fait que l'établissement dans lequel je travaille s'est séparé de son massicot hydraulique d'une part, et que d'autre part tout le monde n'a pas facilement accès à ce type de matériel m'a fait changer d'avis quant au bien fondé que ce document devait impérativement être massicoté. Les onglets sont malgré tout restés dans le document, bien que leur place n'est plus tout à fait au ras de la feuille. Voici comment ils ont été générés...

11.7.1 Idée retenue

Le cahier des charges est le suivant :
- les onglets apparaissent sur chaque page du coté opposé à la reliure ;
- ils sont produits « à l'envers » sur les pages paires ;
- ils doivent être à une hauteur proportionnelle au numéro du chapitre.

D'où l'idée :
- d'utiliser les fonctionnalités du package fancyhdr pour produire les onglets ;
- utiliser des translations verticales pour les positionner ;

11.7.2 Les boîtes dans la marge

De manière à produire les numéros de chapitres dans la marge, j'ai simplement créé des boîtes paragraphe de largeur et hauteur imposées :[7]

```
\newlength{\ongletwidth}
\newlength{\ongletheight}
\setlength{\ongletheight}{32pt}
\setlength{\ongletwidth}{.96cm}
```

Voici la commande produisant la boîte :

```
\newcommand{\b@iteonglet}{%
  \colorbox[gray]{.7}{% une boîte avec un fond gris contenant
    % la boîte paragraphe de largeur et hauteur fixée :
    \parbox[t][\ongletheight][s]{\ongletwidth}{%
      \vfill%
      \centering%
      % on applique un effet miroir selon la parité de la page
      \ifthenelse{\isodd{\value{page}}}{%
        \ongletfont\thechapter}{%
        \reflectbox{\ongletfont\thechapter}}%
      \vfill}}}
```

On notera l'utilisation de la commande `\reflectbox` du package graphicx. On pourra également remarquer que le contenu de la boîte paragraphe est centré en hauteur grâce à deux ressorts verticaux et qu'on insère finalement dans cette boîte le numéro du chapitre dans une mystérieuse fonte enclenchée par la commande `\ongletfont`.

```
Et voici la boîte : \b@iteonglet...
```

Et voici la boîte : 11...

Il est important de noter que les boîtes produites par `\colorbox` sont assujetties à la dimension `\fboxsep` :

```
\setlength{\fboxsep}{15pt}
En voici une autre : \b@iteonglet...
```

En voici une autre : 11...

[7] On remarquera cette maladie des informaticiens de nommer les variables avec des noms à moitié en français et en anglais...

11.7.3 Position des onglets

Pour positionner les onglets, il faut effectuer deux translations :
- une horizontale pour pousser l'onglet jusqu'au bout de la zone de note marginale ;
- une verticale en fonction du numéro de chapitre.

On utilisera le mécanisme de définition des en-têtes de page du package fancyhdr pour positionner les onglets. L'idée est simple, on écrit :

```
\fancyhead[RO]{\bfseries\thepage\onglet}
\fancyhead[LE]{\onglet\bfseries\thepage}
```

pour dire qu'en plus du numéro de page en gras, on mettra dans l'en-tête le résultat de la commande `\onglet` (à droite sur les pages impaires et à gauche sur les pages paires). Nous allons voir comment construire pas à pas cette commande.

Postionnement horizontal

Dans le cas des pages impaires, il faut dans un premier temps pousser la boîte de l'onglet vers la droite, et vers la gauche pour les pages paires. Qu'à cela ne tienne :[8]

```
\newcommand{\ongletI}{%
  \ifthenelse{\isodd{\value{page}}}{%
    % page impaire
    \hspace*{\marginparwidth}\hspace*{\marginparsep}}{%
    % page paire
    \hspace*{-\marginparwidth}\hspace*{-\marginparsep}}%
  \b@iteonglet}
```

Voici ce que donnerait cette commande sur cette page :

176

Pour placer correctement le numéro de la page dans l'entête, il faut donc « leurrer » LaTeX en utilisant une boîte de largeur nulle contenant la boîte de l'onglet et les espaces de translation horizontale :

```
\newcommand{\ongletII}{%
  \makebox[0pt][l]{%
  \ifthenelse{\isodd{\value{page}}}{%
    \hspace*{\marginparwidth}\hspace*{\marginparsep}}{%
    \hspace*{-\marginparwidth}\hspace*{-\marginparsep}}%
    \b@iteonglet}}
```

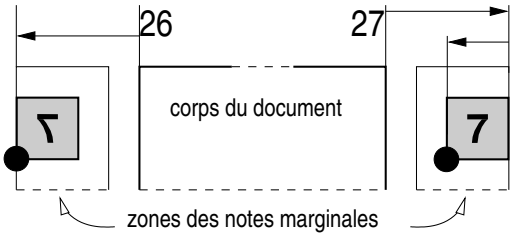
qui donnerait sur cette page :

176

---

[8] Je suis toujours très ému de commencer une phrase par le beau 'Q' de la police Computer Modern...

Pour les pages impaires, s'il on veut pousser la boîte de l'onglet contre le bord de la zone réservée aux notes marginales, on doit également tenir compte de la largeur totale de cette boîte, comme le montre les points de références dans le schéma ci-dessous :



Par conséquent, le positionnement horizontal est obtenu grâce à la commande :

```
\newcommand{\ongletIII}{%
  \makebox[0pt][l]{%
  \ifthenelse{\isodd{\value{page}}}{% page impaire
    \hspace*{\marginparwidth}\hspace*{\marginparsep}%
    \hspace*{-\ongletwidth}\hspace{-2\fboxsep}}{% page paire
    \hspace*{-\marginparwidth}\hspace*{-\marginparsep}}%
    \b@iteonglet}}
```

puisqu'à la largeur de la boîte de l'onglet, il faut rajouter deux fois la dimension `\fboxsep` induite par la commande `\colorbox`.

Positionnement vertical

Reste maintenant à traiter le problème du positionnement vertical... La commande `\raisebox` va nous permettre de positionner verticalement l'onglet. De plus avec la forme suivante :

```
\raisebox{déplacement}[0pt][0pt]{objet à déplacer}
```

on fait croire à LaTeX que l'objet à déplacer a une hauteur nulle. Il n'y aura par conséquent pas de déplacement des objets aux alentours et en particulier ceux constituant l'en-tête de la page. Par exemple, en écrivant :

```
\newcommand{\ongletIV}{%
  \makebox[0pt][l]{%
  \ifthenelse{\isodd{\value{page}}}{%
    \hspace*{\marginparwidth}\hspace*{\marginparsep}%
    \hspace*{-\ongletwidth}\hspace{-2\fboxsep}}{%
    \hspace*{-\marginparwidth}\hspace*{-\marginparsep}}%
    \raisebox{-5cm}[0pt][0pt]{\b@iteonglet}}}
```

On place à partir d'ici un onglet à cinq centimètres vers le bas :

La méthode choisie pour définir le placement de la boîte est la suivante : pour le chapitre de numéro c :

– se déplacer vers le bas d'une dimension fixe donnée d_f;

– ajouter à ce déplacement un déplacement proportionnel à c. Le déplacement de l'onglet pour le chapitre c peut s'écrire :

$$c \times \textit{hauteuronglet} \times \alpha$$

où α est un facteur permettant d'espacer les onglets. Si $\alpha = 1$, les onglets seront espacés d'exactement la longueur de la boîte, avec $\alpha = 2$ ils seront séparés par un espacement égal à deux fois la hauteur de l'onglet, etc.

Voici pour le premier déplacement :

```
% position de la première étiquette
\newlength{\ongletvshift}
\setlength{\ongletvshift}{2cm}
```

Puis pour le facteur alpha :

```
\newcommand{\ongletsep}{1.37}
```

On déclare une dimension permettant de positionner l'onglet :

```
\newlength{\ongletpos}
```

On peut maintenant écrire, la commande `\onglet` :

```
\newcommand{\onglet}{%
  \makebox[0pt][l]{%
    \ifthenelse{\isodd{\value{page}}}{%
      \hspace*{\marginparwidth}\hspace*{\marginparsep}%
      \hspace*{-\ongletwidth}\hspace*{-2\fboxsep}%
    }{%
      \hspace*{-\marginparwidth}\hspace*{-\marginparsep}}%
    % calcul de la position verticale
    \setlength{\ongletvpos}{%
      -\ongletvshift
      -\ongletheight*\real{\thechapter}*\real{\ongletsep}}%
    % positionnement de l'onglet
    \raisebox{\ongletvpos}[0pt][0pt]{\b@iteonglet}}}
```

Ouf ! ça marche... Oui pour le chapitre ne faisant pas partie des annexes. Mais pour ceux en faisant partie, numérotés A, B, etc. ça ne collera pas. Puisqu'on ne pourra pas calculer la position verticale en fonction de ces numéros. En petit coup d'œil dans `book.cls` nous confirme cela :

```
\newcommand\appendix{\par
\setcounter{chapter}{0}%
\setcounter{section}{0}%
[...]
\gdef\thechapter{\@Alph\c@chapter}}
```

on peut comprendre ici que lorsqu'on commence les annexes avec la commande `\appendix`, le compteur de chapitre est remis à zéro, et produit en lettre majuscule. Il faut donc trouver une parade pour que l'onglet continue à se décaler même après les annexes. La solution que j'ai adoptée consiste simplement à utiliser un nouveau compteur permettant de numéroter les chapitres et les annexes. Pour cela, on le déclare :

```
\newcounter{chapitre}
```

On précise tout de suite qu'on veut le produire en chiffre arabe :

```
\renewcommand{\thechapitre}{\arabic{chapitre}}
```

On ajoute la mise à zéro de ce compteur dans la commande `\frontmatter` :

```
\renewcommand\frontmatter{%
  \cleardoublepage
  \setcounter{chapitre}{1}
  [...]
  }
```

Puis à chaque appel de la commande `\chapter`, on incrémente le compteur :

```
\renewcommand{\chapter}{%
\cleardoublepage
\stepcounter{chapitre}
\thispagestyle{plain}
[...]
}
```

11.8 Exemples LaTeX

```
Pour clore ce chapitre, nous présenterons
l'environnement \ltxenv{ltxexemple} permettant
d'introduire des exemples de code  \LaTeX{}
et le résultat dans le document...
```

11.33

Pour clore ce chapitre, nous présenterons l'environnement `ltxexemple` permettant d'introduire des exemples de code LaTeX et le résultat dans le document...

11.8.1 Outils nécessaires

Le package fancyvrb propose deux fonctionnalités axées autour des fichiers. Tout d'abord, la commande `\VerbatimInput` permet d'insérer le contenu d'un fichier. Par exemple :

```
\VerbatimInput[lastline=4]{corps/jouets.tex}
```

donne (`jouets.tex` est le fichier source de ce chapitre) :

```
\chapter{De nouveaux jouets}
\label{chap-jouets}

\begin{epigraphe}{Le Cantique des cantiques Ct \textbf{7} 11}
```

Ensuite, l'environnement `VerbatimOut` réalisant la tâche inverse :

```
\begin{VerbatimOut}{fichier}
  code LaTeX
\end{VerbatimOut}
```

stockera le code LaTeX dans le fichier nommé fichier.

Le deuxième outil nécessaire est défini en s'inspirant de la définition des commandes `\settowidth`, `\settoheight`, etc. dans le fichier `latex.ltx`. Il s'agit d'une commande permettant de récupérer la hauteur totale d'un objet, c'est-à-dire sa hauteur additionnée de sa profondeur :

```
\newcommand{\hauteurtotale}[2]{%
  \setbox\@tempboxa\hbox{{#2}}% sauvegarde de l'objet à mesurer
  \setlength{#1}{\ht\@tempboxa}% récupération de la hauteur
  \addtolength{#1}{\dp\@tempboxa}% à laquelle on ajoute la profondeur
  \setbox\@tempboxa\box\voidb@x}% vidange de la boîte temporaire
```

On notera l'utilisation des commandes TEX, `\ht` et `\dp` renvoyant respectivement la hauteur et la largeur d'une boîte. La commande `\setbox` et l'équivalent TEX de `\savebox`. La boîte `\@tempboxa` est une boîte temporaire utilisée par LATEX et enfin, la boîte `\voidb@x` est une boîte vide.

11.8.2 Le principe de l'environnement `ltxexemple`

L'environnement `ltxexemple` que nous présentons en début de section, est un peu plus complexe que ce que nous avons rencontré jusqu'à maintenant. En effet lorsqu'on écrit :

```
\begin{ltxexemple} contenu \end{ltxexemple}
```

il faut d'une part pouvoir produire contenu tel quel (en verbatim), et d'autre part pouvoir l'interpréter c'est-à-dire le traiter comme LATEX le ferait. Finalement, on se rend compte qu'il faudrait demander à LATEX de traiter *deux fois* contenu ce qui n'est pas envisageable. Une solution pour contourner ce problème est précisément de sauvegarder contenu dans un fichier pour pouvoir le réutiliser, soit en tant que verbatim, soit pour être interpréter par LATEX. La première difficulté à surmonter est donc de créer un environnement sauvegardant son contenu :

```
\newenvironment{ltxexemple}{%
  \VerbatimEnvironment
  \begin{VerbatimOut}{\jobname.exa}}{% clause begin
  \end{VerbatimOut}}% clause end
```

Ne me demandez pas à quoi sert la commande `\VerbatimEnvironment` non documentée dans le package, mais nécessaire au bon fonctionnement de l'environnement défini ci-dessus.

Tout cela est bien joli, mais cet environnement ne fait que sauvegarder son contenu dans un fichier. Il faut donc écrire dans la clause « end » de l'environnement :

```
\end{VerbatimOut}%
\VerbatimInput{\jobname.exa}% le contenu tel quel
\input{\jobname.exa}% le contenu interprété par LaTeX
```

Ainsi :

```
\begin{ltxexemplei}
  du code \LaTeX{}...
\end{ltxexemplei}
```

donne :

```
  du code \LaTeX{}...
```

du code LATEX...

Il reste donc à modifier la mise en page des deux parties. Ce qui fait l'objet du paragraphe suivant.

11.8.3 Mises en boîte

L'idée de base consiste à mettre dans deux boîtes :

```
\newsavebox{\b@iteentree}
\newsavebox{\b@itesortie}
```

pour stocker l'« entrée » (le code) et la « sortie » (le code interprété par LaTeX). On écrira donc dans la clause « end » de l'environnement :

```
[...]
\end{VerbatimOut}%
\begin{ltxexempleenv}% pour agrandir les marges
  \savebox{\b@iteentree}{% sauvegarde du code en verbatim
    \begin{minipage}{.53\linewidth}
      \VerbatimInput{\jobname.exa}
    \end{minipage}}%
  \savebox{\b@itesortie}{% code interprété
    \begin{minipage}{.44\linewidth}
      \setlength{\parindent}{10pt}% par défaut 0pt
      \input{\jobname.exa}
    \end{minipage}}%
  \usebox{\b@iteentree}
  \usebox{\b@itesortie}
 \end{ltxexempleenv}
```

L'environnement `ltxexempleenv` est analogue à celui que nous avons présenté au paragraphe concernant l'agrandissement des marges. La seule différence est qu'il bascule en `\small` et effectue quelques réglages sur les blancs verticaux. On notera que la boîte « entrée » occupera 53% de la largeur de la page et, la boîte « sortie » 44%. Ainsi, le code :

```
\begin{ltxexempleii}
  du code \LaTeX{}...
  \par\noindent
  et c'est tout.
\end{ltxexempleii}
```

Donnera :

```
du code \LaTeX{}...
\par\noindent
et c'est tout.
```

du code LaTeX...
et c'est tout.

Nous avons mis des bordures aux ras des deux boîtes « entrée » et « sortie » pour mettre en évidence leurs dimensions. On peut également noter ici que la clause « begin » de l'environnement est en réalité définie comme suit :

```
\pagebreak[3]% on suggère de changer de page ici
\VerbatimEnvironment%
\begin{VerbatimOut}[gobble=2]{\jobname.exa}
```

L'option `[gobble=2]` permet de « manger » systématiquement deux caractères au début de chaque ligne car tout bon éditeur[9] ajoute des espaces pour l'indentation du source. Ainsi le code :

[9] Je veux bien sûr parler de ŷĭ, pardon, Emacs...

```
\begin{ltxexempleii}
du code \LaTeX{}...
\par\noindent
et c'est tout.
\end{ltxexempleii}
```

donnerait :

```
 code \LaTeX{}...
ar\noindent
 c'est tout.
```

code LaTeX... arc'est tout.

La suite de la mise en page consiste en la création du trait central. Ceci fait l'objet du paragraphe 11.8.5 page 184. Avant cela nous nous attarderons sur la numérotation des exemples.

11.8.4 Numérotation des exemples

Pour numéroter les exemples, il est nécessaire de déclarer un compteur :

```
\newcounter{c@exemple}[chapter]
```

qui sera donc remis à zéro à chaque chapitre. On précise la manière dont il s'affichera lorsqu'on y fera référence :

```
\renewcommand{\thec@exemple}{\thechapter.\arabic{c@exemple}}
```

À chaque appel à l'environnement `ltxexemple`, on fera appel à :

```
\refstepcounter{c@exemple}
```

pour incrémenter le compteur et mettre à jour le système de référence. La petite boîte noire que vous avez pu apercevoir au milieu des exemples a pour largeur `\l@rgeurnumex` (définie à 16 points) et est produite par :

```
\newcommand{\affichenumex}{%
  \raisebox{-1.7pt}[0pt][0pt]{%
    \setlength{\fboxsep}{.7pt}%
    \colorbox{black}{%                    ← la boîte noire
      \makebox[\l@rgeurnumex]{%           ← une boîte de largeur fixée (16 pt)
        \color{white}%                    ← le contenu en blanc
        \tiny\textsf{\thec@exemple}}}}}
```

Ainsi :

```
La petite boîte (\affichenumex)
```

La petite boîte (11.36)

Un nouvelle version de l'environnement `ltxexemple` pourrait donc être :

```
\newenvironment{ltxexempleiii}{%
\VerbatimEnvironment%
\begin{VerbatimOut}[gobble=2]{\jobname.exa}}{%
\end{VerbatimOut}%
\begin{ltxexempleenv}%
  \refstepcounter{c@exemple}%         ← incrémentation du compteur
  \savebox{\b@iteentree}{ [...] }%
  \savebox{\b@itesortie}{ [...] }%
```

```
    \usebox{\b@iteentree}%
    \kern2pt%
    \parbox{3pt}{\rotatebox{90}{\affichenumex}}%
    \kern2pt%
    \usebox{\b@itesortie}
  \end{ltxexempleenv}}%
```

qui donnera :

```
\setlength{\fboxsep}{-2pt}
\setlength{\fboxrule}{.5mm}
Ceci est un \fbox{EXEMPLE} idiot...
```

11.37

Ceci est un EXEMPLE idiot...

La dernière difficulté est de gérer le système de référencement. En effet, on ne peut pas écrire :

```
Voici un exemple.\label{monexemple}.
```

Puisque la séquence « `\label{monexemple}` » apparaîtra dans l'exemple :

```
Voici un exemple.\label{monexemple}
```

11.38

Voici un exemple.

Ce qui n'est pas souhaitable... La solution adoptée ici a été de passer l'éventuelle étiquette de `\label` à l'environnement par le truchement d'une commande. On a défini :

```
\newcommand{\l@belex}{} % la valeur courante du label
\newcommand{\labelexemple}[1]{% commande pour la mettre à jour
  \renewcommand{\l@belex}{#1}}
```

Ainsi avant l'utilisation d'un environnement `ltxexemple` il suffira d'appeler :

```
\labelexemple{étiquette}
```

pour pouvoir ensuite y faire référence avec `\ref{étiquette}` où une commande équivalente. Dans la définition de l'environnement `ltxexemple`, on a ajouté :

```
\ifthenelse{\equal{\l@belex}{}}{}{% si le label courant n'est pas vide
  \label{\l@belex}}% on pose un label
```

qui signifie en français : « si la commande `\l@belex` est définie à une valeur non vide, on pose une étiquette (commande `\label`) avec cette valeur ». Il faut ensuite repositionner la commande `\l@belex` à une valeur vide car dans le cas contraire l'étiquette serait définie plusieurs fois (message « Label 'xxx' multiply defined » de LaTeX). On pourrait donc écrire :

```
\ifthenelse{\equal{\l@belex}{}}{}{% si le label courant n'est pas vide
  \label{\l@belex}% on pose un label
  \renewcommand{\l@belex}{}}
```

ce qui est correct du point de vue de la syntaxe. Cependant la portée de la commande `\renewcommand` est ici locale au groupe dans lequel le test `\ifthenelse` intervient. Pour contourner ce problème on utilisera la construction TeX :

```
\global\def\l@belex{}
```

en lieu et place du `\renewcommand` pour effectuer une redéfinition à portée globale...

11.8.5 Le trait central

Il reste donc à traiter la partie ayant pour but de tracer le vilain trait entre les deux boîtes. La première chose à faire est de mesurer la hauteur totale des deux boîtes, et de conserver la plus importante. Ceci est réalisé grâce au code suivant :

```
% mesure de la boîte d'entrée
\hauteurtotale{\tempodim}{\usebox{\b@iteentree}}%
% mesure de la boîte de sortie
\hauteurtotale{\hauteurdutrait}{\usebox{\b@itesortie}}%
% on garde la plus grande
\ifthenelse{\hauteurdutrait>\tempodim}{%
  \setlength{\tempodim}{\hauteurdutrait}}{}
```

On aura bien sûr préalablement déclaré les dimensions `\hauteurdutrait` et `\tempodim`. La commande `\settotoalheight` est quant à elle présentée au paragraphe 11.8.1.

Le trait noir à tracer entre l'entrée et la sortie mesure exactement `\hauteurtrait` moins `\l@rgeurnumex` (la largeur de la boîte contenant le numéro de l'exemple). On stocke cette dimension :

```
% hauteur du trait sans la boîte du numéro
\setlength{\hauteurdutrait}{\tempodim-\l@rgeurnumex}
```

Nous avons décidé après un vote à bulletin secret en assemblée générale, de dessiner 70% du trait au dessus du numéro et 30% en dessous. Par conséquent le trait central est produit dans un `\parbox` comme suit :

```
% le trait central
\parbox{3pt}{%
  \begin{center}
    \rule{3pt}{.7\hauteurdutrait}\\\nointerlineskip% 70% au dessus
    \rotatebox{90}{\affichenumex}\\\nointerlineskip%
    \rule{3pt}{.3\hauteurdutrait}% 30% en dessous
  \end{center}}
```

La commande `\nointerlineskip` permet de supprimer tout blanc vertical supplémentaire qui pourrait être inséré par la commande `\\`. Plus de détails sont donnés à la section présentant l'implémentation de la boîte avec titre pour la minitable des matières.

```
Voilà, c'est tout pour ce merveilleux
environnement \ltxenv{ltexexemple}. Notez que la
dimension \texttt{3pt} pourrait faire l'objet
de la définition d'une longueur...
```

Voilà, c'est tout pour ce merveilleux environnement `ltexexemple`. Notez que la dimension `3pt` pourrait faire l'objet de la définition d'une longueur...

III

Annexes

A Générer des « pdf »

Sommaire

CETTE ANNEXE présente un moyen de générer des documents au format Pdf (portable document format). Ce format créé par la société Adobe présente l'avantage d'être effectivement portable d'un ordinateur à un autre, et de manière plus générale, d'un système d'exploitation à un autre. Il est donc intéressant aujourd'hui de pouvoir générer de tels fichiers à partir d'un source LaTeX.

A.1 Principe général

Il y a au moins trois façons de générer des fichiers au format Pdf à partir d'un document LaTeX :

1. à l'aide de pdflatex qui s'utilise en lieu et place du programme latex pour traduire le source LaTeX en Pdf ;
2. à l'aide de dvipdf permettant de traduire le fichier Dvi en Pdf ;
3. à l'aide de ps2pdf pour traduire une sortie PostScript en Pdf.

Votre serviteur qui a une certaine expérience de la première solution s'attardera sur pdflatex. Un des pré-requis pour une utilisation correcte de ce logiciel est
– soit l'utilisation du package lmodern ;
– soit l'installation de l'extension « CM-Super font » de Vladimir VOLOVICH. La distribution Etch de la Debian contient un paquet prêt à l'emploi. On peut également trouver des documentations sur le ouèbe permettant d'installer cette extension sur une distribution Debian Sarge avec teTeX (http://sravier.free.fr/linux/debian_latex_cm-super.html).

A.2 Ce qui change

Pour compiler le fichier source LaTeX et produire un fichier au format Pdf, on utilisera le programme pdflatex :

```
pdflatex monfichier.tex
```

commande qui, si le document source ne contient pas d'erreur, créera le fichier nommé `monfichier.pdf`. Voici ensuite quelques remarques importantes :

Graphiques : ils devront être inclus au format Png ou Jpeg pour les images et Pdf pour les dessins.[1]

Liens : à condition d'inclure le package hyperref, le document Pdf contiendra automatiquement des liens à chaque occurrence de la commande `\ref`, dans la table des matières, dans l'index, etc. De plus une table des matières déroulante sera générée pour le programme acrobat reader.

A.3 Trucs et astuces

Puisqu'on génère souvent du Dvi ou du Pdf à partir du même source et que l'on doit inclure des fichiers graphiques à des formats différents selon la situation, on utilisera le package ifpdf et l'astuce suivante :

```
\ifpdf
% un truc spécifique à la sortie en pdf
\else
% un machin spécifique à la sortie en dvi
\fi
```

A.3.1 Gestion des graphiques

On pourra écrire ensuite quelque chose du genre :

```
\ifpdf
\graphicspath{{pngs/}{pdfs/}}
\else
\graphicspath{{epss}}
\fi
```

Si on a pris soin de ranger les fichiers graphiques dans les répertoires `pngs`, `pdfs` et `epss`... Ce nouveau « if » permet également des constructions du style :

```
\ifpdf
\includegraphics[pdftex]{graphicx}
\else
\includegraphics{graphicx}
\fi
```

Ce qui ne doit pas être nécessaire avec les dernières moutures de LaTeX.

A.3.2 Vignettes

Les versions récentes de pdflatex permettent de créer des vignettes (*thumbnail*) pour les visualiseurs evince et Acrobat reader pour ne citer qu'eux. Auparavant, il était nécessaire d'utiliser le package thumbpdf :

```
\usepackage{thumbpdf}
```

[1] Les fichiers du logiciel Xfig peuvent être convertis en Pdf

puis exécuter :

```
thumbpdf monfichier.pdf
```

Cette commande crée un fichier nommé `monfichier.tpt` qui sera inclus à la compilation suivante avec pdflatex.

A.3.3 Pagination

Pour faire apparaître les numéros de pages du document dans le navigateur Acrobat Reader, il est nécessaire d'ajouter l'option `pdfpagelabels` à l'inclusion du package hyperref (cf. § suivant).

A.3.4 Signets

Les signets (*bookmarks* en anglais) des visualiseurs de fichier pdf sont des sortes d'« explorateurs de table des matières » qui donnent accès directement à une section d'un niveau déterminé. Il y a eu deux difficultés à contourner pour produire ce manuel :

1. faire en sorte que le contenu du « backmatter » (biblio, glossaire, index) soit au même niveau hiérarchique que les « parties » dans cet explorateur. Par défaut, ces informations se trouvent en effet « cachées » dans la partie des annexes, car à la même profondeur que les `\chapter` ;
2. faire en sorte que le lien vers l'index dans les signets pointe effectivement sur l'index...

Pour régler le premier problème, il suffit de faire croire à LaTeX qu'à partir du « backmatter » les *chapitres* sont d'un même niveau de profondeur dans la table des matières, que les *parties*. L'incantation vaudou correspondante est :

```
\renewcommand{\toclevel@chapter}{-1}
```

à placer à un endroit judicieux dans un fichier de style. L'endroit où l'on redéfinit le `\backmatter` est sans nul doute un bon choix.

Pour en finir avec les *bookmarks*, afin que celui de l'index pointe effectivement sur l'index (!), on devra cette fois avoir recours à une incantation chamanique :

```
\let\printindexORIG\printindex
\renewcommand{\printindex}{%
  \cleardoublepage
  \phantomsection% création d'une fausse section
  \addcontentsline{toc}{chapter}{Index}
  \printindexORIG}
```

permettant de surcharger la commande `\printindex` en y ajoutant une fausse section à l'aide de la commande `\phantomsection` fournie avec le package hyperref. Ne m'en demandez pas plus :-)

A.4 Hyperliens

Le package `hyperref` permet d'insérer dans les fichiers `.dvi` et `.pdf` des commandes spéciales qui pourront être exploitées par les navigateurs (xdvi et Acrobat

reader entre autres). On pourra alors cliquer sur le texte produit par les commandes telles que `\ref` pour se rendre automatiquement à la zone référencée. Dans le document que vous avez sous les yeux, la version électronique possède des liens sur lesquels on peut cliquer pour :

- toutes les références générées par `\ref`, `\pageref` et `\vref` ;
- les notes de bas de page ;
- les url produites par la commande `\url` ;
- les renvois bibliographiques ;
- les pages pour chaque entrée d'index.

Pour mettre en branle ce système d'hyperliens, on écrira :

```
\ifpdf
\usepackage[pdftex=true,
            hyperindex=true,
            colorlinks=true]{hyperref}
\else
\usepackage[hypertex=true,
            hyperindex=true,
            colorlinks=false]{hyperref}
\fi
```

Les liens seront mis en couleur uniquement pour la version Pdf. Dans le cas contraire la version à imprimer apparaîtra en gris plus ou moins clair sur imprimante laser noir et blanc...

L'ordre dans lequel on inclura les différents packages pour un document influera sur le bon fonctionnement de l'extension hyperref. Il arrive même que l'endroit choisi pour l'inclusion provoque une erreur de compilation. À vous de trouver la bonne séquence :-)

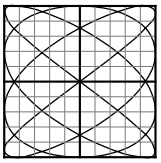
A.5 Interaction avec psfrag et pstricks

A.5.1 pstricks

To trick en anglais, ou « tricher » en français... En gros en écrivant ça :

```
Soit :
\begin{pspicture}[](-1,-1)(1,1)
  \parametricplot[linewidth=.5pt,plotstyle=ccurve]%
  {0}{360}{4 t mul sin 3 t mul sin}
  \psgrid[gridlabels=0pt](-1,-1)(1,1)
\end{pspicture}
\quad le tracé de $x=\sin(4t), y=\sin(3t)$...
```

On obtient ça :

Soit :  le tracé de $x = \sin(4t), y = \sin(3t)$...

Dingue, non ? Certes. Le principe de l'extension pstricks est d'insérer du code PostScript dans le fichier `.dvi`, code qui pourra être également traité par le programme dvips. Là où ça se corse c'est lorsque l'on veut utiliser ces bestioles avec pdflatex. En effet ce dernier créant directement un fichier `.pdf` à partir du `.tex`, insérer du PostScript dans le fichier au format Pdf n'aura aucun effet... Il est malgré tout possible de contourner le problème :

1. d'abord générer un document LaTeX minimal contenant les commandes pstricks ;
2. compiler ce document avec LaTeX pour générer un `.dvi` ;
3. demander à dvips de créer un fichier au format PostScript encapsulé avec l'option `-E` ;
4. convertir ce fichier au format Pdf ;
5. inclure ce fichier au moment d'utiliser pdflatex.

Tout cela est évidemment un peu tordu mais peut être automatisé à l'aide d'un Makefile, d'un petit script Unix, et d'une commande... Tout d'abord :

```
\newcommand{\includepstricksgraphics}[1]{%
  \ifpdf\includegraphics{#1}\else\input{#1}\fi}
```

L'idée est donc d'extraire la portion de code contenant des commandes pstricks pour les stocker dans un fichier `bidule.tex`, puis lorsqu'on écrit :

```
\includepstricksgraphics{bidule}
```

on inclura `bidule.pdf` si on utilise pdflatex et `bidule.tex` si on utilise LaTeX. Ensuite, le « petit » script Unix qu'on peut adapter à ses besoins :

```
#!/bin/sh
# on enlève l'extension du 1er argument
FILE=${1%.*}
# création d'un fichier temporaire psttemp.tex
cat > psttemp.tex <<EOF
\documentclass{manuel}          ← mettre la classe et les packages adéquats
\thispagestyle{empty}
\begin{document}
\input{$FILE}
\end{document}
EOF
# Création du fichier dvi
latex psttemp
# Création du fichier eps
dvips -E  $TMPFILE.dvi -o psttemp.eps
# Création du fichier pdf
epstopdf psttemp.eps --debug --outfile=$FILE.pdf
# effacement des fichiers temporaires
rm -f psttemp.*
```

Ce script sauvé sous le nom `pstricks.sh` pourra être invoqué comme suit :

```
./pstricks.sh bidule.tex
```

et devrait créer le fichier `bidule.pdf` que pdflatex aura la sagesse d'inclure grâce à la commande `\includepstricksgraphics` dont le code est donné plus haut. Pour ce qui est du Makefile, il n'est pas très difficile à partir du script précédent de définir une règle ayant pour but de transformer un fichier `.tex` en un fichier `.pdf`. Avec le version Gnu de make, on aura quelque chose du genre :

```
%.pdf : %.tex
        ./pstricks.sh $<
```

Le programme dvips n'est pas toujours en mesure de calculer correctement la boîte englobante pour le PostScript encapsulé. En particulier la section 41 de la documentation de pstricks[2] indique que dvips n'est pas capable de tenir compte du code postscript généré pour estimer cette boîte englobante. Dans ce cas, il est conseillé soit d'ajouter du texte autour du graphique et dvips arrive à s'en sortir, soit d'utiliser l'environnement `TeXtoEPS`. Le document temporaire du script précédent devient alors :

```
cat > psttemp.tex <<EOF
\documentclass{manuel}
\usepackage{pst-eps}
\thispagestyle{empty}
\begin{document}
\begin{TeXtoEPS}                ← Pour aider PSTricks à calculer la boîte
\input{$FILE}
\end{TeXtoEPS}
\end{document}
EOF
```

A.5.2 psfrag

La limitation et le principe sont les mêmes que pour pstricks. Pour utiliser psfrag avec pdflatex, il est nécessaire de procéder comme suit :

1. d'abord générer un document LaTeX minimal contenant des commandes psfrag ;
2. compiler ce document avec LaTeX pour générer un `.dvi` ;
3. demander à dvips de créer un fichier au format PostScript encapsulé avec l'option `-E` ;
4. convertir ce fichier au format Pdf ;
5. inclure ce fichier au moment d'utiliser pdflatex.

Il y a cependant un petit « hic » car la figure dont on calcule la boîte englobante avec dvips contient du texte généré par psfrag indiquant les remplacements qui seront effectués. On doit en tenir compte. Dans le script shell, on crée une fonction :

```
function genere_eps
{
  cat > $TMPFILE.tex <<EOF
\documentclass{manuel}          ← mettre la classe et les packages qui vont
                                  bien
\thispagestyle{empty}
\begin{document}
  \input{$1}
```

[2] http://tug.org/PSTricks

```
\end{document}
EOF
  echo "Création du fichier dvi"
  latex $TMPFILE > $LOGFILE
  echo "Création du fichier $TMPFILE.eps"
  dvips -E  $TMPFILE.dvi -o $TMPFILE.eps >> $LOGFILE 2>&1
}
```

On utilise ensuite cette fonction par deux fois comme suit, dans le script :

```
FILE=${1%.*}
TMPFILE=truc
LOGFILE=truc.log
sanspsfrag=$TMPFILE-sanspsf.tex

# on enlève les lignes contenant la commande \psfrag
# et on récupère la boite englobante du fichier eps sans les psfrag
grep -v \\\\psfrag $FILE.tex > $sanspsfrag
genere_eps $sanspsfrag
bonnebb=$(grep "^%%BoundingBox" $TMPFILE.eps | head -1)

# on récupère la boite englobante du fichier eps avec psfrag
genere_eps $FILE
mauvaisebb=$(grep "^%%BoundingBox" $TMPFILE.eps | head -1)

# on remplace la boite englobante par la bonne
sed  -i "s/$mauvaisebb/$bonnebb/" $TMPFILE.eps

echo "Création du fichier pdf"
epstopdf $TMPFILE.eps --debug --outfile=pdfs/${FILE##*/}.pdf >>
$LOGFILE 2>&1

# un petit coup de toilette
rm -f $TMPFILE.* $LOGFILE $sanspsfrag
```

Ce script a plusieurs limitations. Parmi elles : il échouera si une commande \psfrag s'étend sur plusieurs lignes. On demande en effet à grep d'enlever les lignes contenant \psfrag sans vérifier que la commande ne se termine pas une ou plusieurs lignes plus bas...

B Mémento

Sommaire

VOUS TROUVEREZ ici quelques informations « en vrac » sur les extensions de LaTeX, une liste assez complète des fichiers auxiliaires qui gravitent autour de votre fichier source. Suivent quelques explications succinctes sur la merveilleuse extension AucTeX d'Emacs. Enfin pour ceux qui ont la chance de travailler dans un environnement UNIX, cette annexe s'achève sur la configuration d'Emacs pour travailler avec le correcteur orthographique aspell.

B.1 Extensions

Comme il en est question dans la préface de ce document, TeX et LaTeX sont des systèmes *ouverts*. Autour du noyau LaTeX gravitent un certain nombre de packages standard qui constituent la base du système. Mais tout utilisateur peut faire évoluer LaTeX en lui rajoutant des fonctionnalités diverses. On trouve donc une multitude d'outils sous formes d'extensions (ou *packages* en anglais) ou sous forme de classe de documents. Certaines sont devenues des standards, « toutes » sont disponibles sur les serveurs dédiés à la distribution de LaTeX (cf. chapitre 8) ou sur des pages personnelles, d'autres sont fournies avec les Call for papers et autres author's guides.

Nous vous donnons ici une liste de packages « classiques » et vous invitons à vous référer à la documentation qui est généralement jointe au package. Notez que le site du loria propose une liste des packages « généraux » à `http://www.loria.fr/services/tex/packages.html#latex` ; il existe d'autre part un moteur de rercherche contenant plus de mille extensions référencées, à `ftp://ftp.loria.fr/pub/unix/tex/ctan/help/Catalogue/catalogue.html`.

- ▲ french : utilisé pour « franciser » les documents. Ne coupe pas une phrase entre un mot et une double ponctuation. Propose aussi quelques commandes axées sur la typographie française (voir chapitre 7) ;
- ▲ amsmath : le package pour faire des formules et équations perfectionnées ;
- ▲ array : améliore l'utilisation de `tabular` ;
- ▲ hhline : étend les bordures de tableaux de base de LaTeX ;
- ▲ fancyhdr : permet de personnaliser en-tête et pied de page. Jetez un coup d'œil sur ceux de ce manuel ;

▲ varioref : propose la commande `\vref` à la place de `\ref`. Celle-ci ajoute « page suivante », « page 12 », ou rien du tout selon où se trouve l'objet référencé par rapport à la position du renvoi ;

▲ ifthen : fournit deux *structure de contrôle* un « if then else » et un « do while ». Ce qui permet de faire des commandes un peu plus évoluées ;

▲ chapterbib : permet d'insérer une bibliographie à chaque fin de chapitre ;

▲ overcite : écrit les citations bibliographiques sous forme d'exposant ;

▲ bibunits : permet de produire des bibliographies composées de plusieurs unités ;

▲ fancybox : propose quatre variantes de `\fbox` : `\shadowbox`, `\doublebox`, `\ovalbox` et `\Ovalbox`.

▲ algorithms : pour écrire des pseudo-algorithmes (facilement « francisable ».) sous la forme d'un environnement qui peut être flottant ou non ;

▲ geometry : une extension permettant de changer les marges et la plupart des dimensions intervenant au niveau de la page, de manière assez souple ;

▲ url : permet d'écrire des adresses sous forme d'une url, la césure est gérée « pour le mieux » ;

▲ fancyvrb : propose une version améliorée de l'environnement `verbatim` ;

B.2 Les fichiers auxiliaires

Voici la liste des fichiers que vous pourrez trouver sur votre disque à côté de votre document source. Ces fichiers portent tous une extension de trois lettres, les voici :[1]

`tex` fichier source LaTeX ;

`aux` fichier auxiliaire que LaTeX utilise pour résoudre les références, entre autres ;

`log` le fichier de trace (dit *log file* en anglais) contenant les infos de la compilation ;

`dvi` fichier *device independant*, qui va pouvoir être affiché ou imprimé selon la situtation ;

`toc` fichier contenant la table des matières (initiales de *table of contents*) ;

`lof` fichier contenant la liste des figures (*list of figure*);

`lot` fichier contenant la liste des tables ;

`bib` fichier source BibTeX contenant des entrées de bibliographie ;

`bbl` fichier contenant la bibliographie, peut être généré à partir de BibTeX ;

`blg` fichier trace de BibTeX ;

`idx` fichier des entrées d'index non triées ;

`ind` fichier contenant l'index, est généralement généré par makeindex ;

`ilg` fichier trace de makeindex ;

[1] Certains packages créent leur propres fichiers auxiliaires comme le package minitoc et la classe `lettre` ; ils ne sont pas mentionnés dans cette liste.

`sty` fichier contenant des définitions de commandes modifiant la mise en page, ou fournissant des outils particuliers ;

`cls` un fichier definissant une classe. standard ;

Pour archiver un document LaTeX,

Vous pouvez effacer : tous les fichiers auxiliaires, les fichiers *log*, ainsi que les fichiers de tables des matières et listes de figures et tables.

Vous pouvez aussi effacer : le fichier `bbl` si vous êtes capable de le générer à partir d'un fichier `bib` et BibTeX. Les fichiers d'index peuvent généralement être effacés puisqu'ils sont en principe produits par makeindex. Le fichier `dvi` n'est pas indispensable puisque vous êtes censé avoir le source LaTeX

Vous devez garder : le source LaTeX et les éventuels fichiers de styles que vous avez définis (`sty` et `cls`) ; mais si vous en êtes au stade de la définition de classe, le conseil est probablement un peu saugrenu...

B.3 AucTEX

AucTeX est un module d'Emacs qui permet de faciliter la saisie de documents LaTeX. Ce module est automatiquement chargé lorsqu'on ouvre un document portant l'extension `.tex`, `.sty` ou `.cls`. On peut distinguer trois types d'aide dans AucTeX :

1. l'aide au formatage du source (couleur, indentation,...)
2. les raccourcis clavier pour insérer des commandes ou des environnements,
3. l'aide à la compilation.

B.3.1 Formatage du source

Les couleurs et la touche tab jouent le même rôle que dans un buffer C ou C++. On notera que `M-q` « formate » un paragraphe, c.-à-d., découpe automatiquement le paragraphe en lignes de longueurs à peu prés égales.

B.3.2 Raccourcis

fontes

`C-c C-f` (Changer Fonte) suivi de :
- `C-e` insère `\emph{}`
- `C-b` insère `\textbf{}`
- `C-t` insère `\texttt{}`
- `C-s` insère `\textsl{}`
- `C-c` insère `\textsc{}`
- ...

Section

`C-c C-s` insère une Section en vous demandant son niveau, son titre et son label dans le minibuffer.

Commandes et Environnement

`M-Tab` tente de compléter le nom en cours (*automatic completion*).
`C-c RET` insère une commande.
`C-c C-e` insère un Environnement.[2]
`C-u C-c C-e` change un environnement.
`C-c ]` ferme un environnement en ajoutant la commande `\end` qui manque.

B.3.3 Compilation

`C-c C-c` tente de suivre de cycle de compilation d'un document, en lançant suivant la situation, LaTeX, BibTeX, xdvi,... Notez aussi que AucTeX permet de gérer le mécanisme du document maître (cf. 6.4). Pour cela il vous demandera de saisir le nom du document maître lorsque vous ouvrirez un nouveau document dans Emacs. Dans le cas contraire il faudra expliquer gentiment à AucTeX qui est le document maître avec :

> M-x TeX-master-file-ask ou C-c _

vous devrez alors saisir le nom du fichier maître. En agissant ainsi, lorsque vous lancerez une compilation avec `C-c C-c` sur un des documents « esclaves », c'est sur le *master* qu'elle agira.

B.4 Aspell

Aspell est un correcteur orthographique multilingue qu'on peut interfacer avec l'éditeur de texte à tout faire Emacs. Pour l'utiliser dans Emacs, deux commandes à connaître :

> M-x ispell-change-dictionary

selectionne la langue du dictionnaire (`francais` ou `english`), et :

> M-x ispell-buffer

commence une session de correction sur le buffer. Il est également possible de vérifier l'orthographe d'un seul mot avec la commande :

> M-x ispell-word ou M-$

À l'heure où j'écris ces lignes, la distribution Debian ne configure pas Emacs pour utiliser le programme Aspell par défaut. Il faut donc ajouter dans votre fichier `.emacs`, la ligne :

```
(setq-default ispell-program-name "aspell")
```

Il est particulièrement utile de noter que l'on peut configurer le programme Aspell pour lui demander explicitement d'ignorer ou non les arguments des commandes LaTeX. On pourra par exemple ne pas vérifier l'argument d'une commande ne contenant pas de français. Ainsi, si l'on définit la commande :

[2] Pour certains environnements et certaines commandes dont la syntaxe est connue par AucTeX il vous sera demandé quelques précisions (valeurs des arguments, légendes, format du tableau,...).

```
\newcommand{\bidule}[2]{% commandes prenant 2 arguments
  ... }
```

Il suffira d'écrire dans son fichier `~/.aspell.conf` :

```
add-tex-command bidule pP
```

pour demander à Aspell de vérifier le second paramètre (`P`), mais d'ignorer le contenu du premier (`p`). Pour ignorer les deux, on aurait écrit :

```
add-tex-command bidule pp
```

Enfin, Emacs dispose également d'un mode de correction de mot « à la volée »[3] qu'on peut activer ou désactiver avec la commande :

```
M-x flyspell-mode
```

[3] D'aucuns diront « à la ouôrde »...

C Symboles

Sommaire

Vous trouverez dans cette annexe, une liste de « tous » les symboles mathématiques disponibles dans LaTeX. Nous avons séparé ces symboles en quatre catégories :

- les symboles standard du tableau C.1 au tableau C.10 ;
- les symboles de LaTeX disponibles avec le package latexsym donnés par le tableau C.11 ;
- les symboles de l'*American Mathematical Society* disponibles avec le package amssymb, du tableau C.12 au tableau C.19 ;
- les symboles disponibles avec le package textcomp (tableaux C.20 et C.21);
- les symboles des polices PostScript bien connues ZapfDingbats et Symbol. Les symboles de ces fontes sont accessibles en incluant le package pifont et en utilisant la commande :

  `\Pisymbol{pzd}{`numéro`}`

  pour les symboles de la police Zapf, et :

  `\Pisymbol{psy}{`numéro`}`

  pour ceux de la police Symbol. Le nombre numéro est le numéro de la case correspondant au symbole choisi dans la table C.22 page 209 ou C.23.

C.1 Symboles standard

TAB. C.1 – Les lettres grecques.

\alpha	α	\beta	β	\gamma	γ	\delta	δ
\epsilon	ϵ	\varepsilon	ε	\zeta	ζ	\eta	η
\theta	θ	\vartheta	ϑ	\iota	ι	\kappa	κ
\lambda	λ	\mu	μ	\nu	ν	\xi	ξ
o	o	\pi	π	\varpi	ϖ	\rho	ρ
\varrho	ϱ	\sigma	σ	\varsigma	ς	\tau	τ
\upsilon	υ	\phi	ϕ	\varphi	φ	\chi	χ
\psi	ψ	\omega	ω				
\Gamma	Γ	\Delta	Δ	\Theta	Θ	\Lambda	Λ
\Xi	Ξ	\Pi	Π	\Sigma	Σ	\Upsilon	Υ
\Phi	Φ	\Psi	Ψ	\Omega	Ω		

TAB. C.2 – Les opérateurs binaires.

\pm	$\pm$	\cdot	$\cdot$	\setminus	$\setminus$	\ominus	$\ominus$
\mp	$\mp$	\cap	$\cap$	\wr	$\wr$	\otimes	$\otimes$
\times	$\times$	\cup	$\cup$	\diamond	$\diamond$	\oslash	$\oslash$
\div	$\div$	\uplus	$\uplus$	\bigtriangleup	$\bigtriangleup$	\odot	$\odot$
\ast	$\ast$	\sqcap	$\sqcap$	\bigtriangledown	$\bigtriangledown$	\bigcirc	$\bigcirc$
\star	$\star$	\sqcup	$\sqcup$	\triangleleft	$\triangleleft$	\dagger	$\dagger$
\circ	$\circ$	\vee	$\vee$	\triangleright	$\triangleright$	\ddagger	$\ddagger$
\bullet	$\bullet$	\wedge	$\wedge$	\oplus	$\oplus$	\amalg	$\amalg$

TAB. C.3 – Les symboles de tailles variables.

\sum	$\sum$	\prod	$\prod$	\coprod	$\coprod$	\int	$\int$	\oint	$\oint$
\bigcap	$\bigcap$	\bigcup	$\bigcup$	\bigsqcup	$\bigsqcup$	\bigvee	$\bigvee$	\bigwedge	$\bigwedge$
\bigodot	$\bigodot$	\bigotimes	$\bigotimes$	\bigoplus	$\bigoplus$	\biguplus	$\biguplus$		

TAB. C.4 – Les points.

\ldots	$\ldots$	\cdots	$\cdots$	\vdots	$\vdots$	\ddots	$\ddots$

TAB. C.5 – Les relations.

\leq	$\leq$	\geq	$\geq$	\equiv	$\equiv$	\models	$\models$
\prec	$\prec$	\succ	$\succ$	\sim	$\sim$	\perp	$\perp$
\preceq	$\preceq$	\succeq	$\succeq$	\simeq	$\simeq$	\mid	$\mid$
\ll	$\ll$	\gg	$\gg$	\asymp	$\asymp$	\parallel	$\parallel$
\subset	$\subset$	\supset	$\supset$	\approx	$\approx$	\bowtie	$\bowtie$
\subseteq	$\subseteq$	\supseteq	$\supseteq$	\cong	$\cong$	\smile	$\smile$
\sqsubseteq	$\sqsubseteq$	\sqsupseteq	$\sqsupseteq$	\neq	$\neq$	\frown	$\frown$
\in	$\in$	\ni	$\ni$	\doteq	$\doteq$		
\vdash	$\vdash$	\dashv	$\dashv$	\propto	$\propto$		

TAB. C.6 – Les flèches.

\leftarrow	$\leftarrow$	\longleftarrow	$\longleftarrow$	\uparrow	$\uparrow$
\Leftarrow	$\Leftarrow$	\Longleftarrow	$\Longleftarrow$	\Uparrow	$\Uparrow$
\rightarrow	$\rightarrow$	\longrightarrow	$\longrightarrow$	\downarrow	$\downarrow$
\Rightarrow	$\Rightarrow$	\Longrightarrow	$\Longrightarrow$	\Downarrow	$\Downarrow$
\leftrightarrow	$\leftrightarrow$	\longleftrightarrow	$\longleftrightarrow$	\updownarrow	$\updownarrow$
\Leftrightarrow	$\Leftrightarrow$	\Longleftrightarrow	$\Longleftrightarrow$	\Updownarrow	$\Updownarrow$
\mapsto	$\mapsto$	\longmapsto	$\longmapsto$	\nearrow	$\nearrow$
\hookleftarrow	$\hookleftarrow$	\hookrightarrow	$\hookrightarrow$	\searrow	$\searrow$
\leftharpoonup	$\leftharpoonup$	\rightharpoonup	$\rightharpoonup$	\swarrow	$\swarrow$
\leftharpoondown	$\leftharpoondown$	\rightharpoondown	$\rightharpoondown$	\nwarrow	$\nwarrow$

TAB. C.7 – Divers.

\aleph	$\aleph$	\prime	$\prime$	\forall	$\forall$	\infty	∞
\hbar	$\hbar$	\emptyset	$\emptyset$	\exists	$\exists$	\triangle	$\triangle$
\imath	$\imath$	\nabla	∇	\neg	$\neg$	\clubsuit	$\clubsuit$
\jmath	$\jmath$	\surd	$\surd$	\flat	$\flat$	\diamondsuit	$\diamondsuit$
\ell	ℓ	\top	$\top$	\natural	$\natural$	\heartsuit	$\heartsuit$
\wp	$\wp$	\bot	$\bot$	\sharp	$\sharp$	\spadesuit	$\spadesuit$
\Re	$\Re$	\\|	$\|$	\backslash	$\backslash$		
\Im	$\Im$	\angle	$\angle$	\partial	∂		

TAB. C.8 – Les fonctions.

\arccos	\cos	\csc	\exp	\ker	\limsup	\min	\sinh
\arcsin	\cosh	\deg	\gcd	\lg	\ln	\Pr	\sup
\arctan	\cot	\det	\hom	\lim	\log	\sec	\tan
\arg	\coth	\dim	\inf	\liminf	\max	\sin	\tanh

TAB. C.9 – Les délimiteurs.

\uparrow	$\uparrow$	\Uparrow	$\Uparrow$	\downarrow	$\downarrow$	\Downarrow	$\Downarrow$
\{	$\{$	\}	$\}$	\updownarrow	$\updownarrow$	\Updownarrow	$\Updownarrow$
\lfloor	$\lfloor$	\rfloor	$\rfloor$	\lceil	$\lceil$	\rceil	$\rceil$
\langle	$\langle$	\rangle	$\rangle$	/	/	\backslash	$\backslash$
|	$\|$	\\|	$\|\|$				

TAB. C.10 – Les grands délimiteurs.

\rmoustache	$\rmoustache$	\lmoustache	$\lmoustache$	\rgroup	$\rgroup$	\lgroup	$\lgroup$
\arrowvert	$\arrowvert$	\Arrowvert	$\Arrowvert$	\bracevert	$\bracevert$		

TAB. C.11 – Les symboles de latexsym

\lhd	$\lhd$	\rhd	$\rhd$	\unlhd	$\unlhd$	\unrhd	$\unrhd$
\sqsubset	$\sqsubset$	\sqsubset	$\sqsubset$	\Join	$\Join$	\mho	$\mho$
\Box	$\Box$	\Diamond	$\Diamond$	\leadsto	$\leadsto$		

C.2 Symboles de l'$\mathcal{AMS}$

TAB. C.12 – Les flèches de l'$\mathcal{AMS}$

\dashrightarrow	$\dashrightarrow$	\dashleftarrow	$\dashleftarrow$	\leftleftarrows	$\leftleftarrows$
\leftrightarrows	$\leftrightarrows$	\Lleftarrow	$\Lleftarrow$	\twoheadleftarrow	$\twoheadleftarrow$
\leftarrowtail	$\leftarrowtail$	\looparrowleft	$\looparrowleft$	\leftrightharpoons	$\leftrightharpoons$
\curvearrowleft	$\curvearrowleft$	\circlearrowleft	$\circlearrowleft$	\Lsh	$\Lsh$
\upuparrows	$\upuparrows$	\upharpoonleft	$\upharpoonleft$	\downharpoonleft	$\downharpoonleft$
\multimap	$\multimap$	\leftrightsquigarrow	$\leftrightsquigarrow$	\rightrightarrows	$\rightrightarrows$
\rightleftarrows	$\rightleftarrows$	\rightrightarrows	$\rightrightarrows$	\rightleftarrows	$\rightleftarrows$
\twoheadrightarrow	$\twoheadrightarrow$	\rightarrowtail	$\rightarrowtail$	\looparrowright	$\looparrowright$
\rightleftharpoons	$\rightleftharpoons$	\curvearrowright	$\curvearrowright$	\circlearrowright	$\circlearrowright$
\Rsh	$\Rsh$	\downdownarrows	$\downdownarrows$	\upharpoonright	$\upharpoonright$
\downharpoonright	$\downharpoonright$	\rightsquigarrow	$\rightsquigarrow$		

TAB. C.13 – Les relations de l'$\mathcal{AMS}$

\leq	$\leq$	\leqslant	$\leqslant$	\eqslantless	$\eqslantless$
\lesssim	$\lesssim$	\lessapprox	$\lessapprox$	\approxeq	$\approxeq$
\lessdot	$\lessdot$	\lll	$\lll$	\lessgtr	$\lessgtr$
\lesseqgtr	$\lesseqgtr$	\lesseqqgtr	$\lesseqqgtr$	\doteqdot	$\doteqdot$
\risingdotseq	$\risingdotseq$	\fallingdotseq	$\fallingdotseq$	\backsim	$\backsim$
\backsimeq	$\backsimeq$	\subseteqq	$\subseteqq$	\Subset	$\Subset$
\sqsubset	$\sqsubset$	\preccurlyeq	$\preccurlyeq$	\curlyeqprec	$\curlyeqprec$
\precsim	$\precsim$	\precapprox	$\precapprox$	\vartriangleleft	$\vartriangleleft$
\trianglelefteq	$\trianglelefteq$	\vDash	$\vDash$	\Vvdash	$\Vvdash$
\smallsmile	$\smallsmile$	\smallfrown	$\smallfrown$	\bumpeq	$\bumpeq$
\Bumpeq	$\Bumpeq$	\geqq	$\geqq$	\geqslant	$\geqslant$
\eqslantgtr	$\eqslantgtr$	\gtrsim	$\gtrsim$	\gtrapprox	$\gtrapprox$
\gtrdot	$\gtrdot$	\ggg	$\ggg$	\gtrless	$\gtrless$
\gtreqless	$\gtreqless$	\gtreqqless	$\gtreqqless$	\eqcirc	$\eqcirc$
\circeq	$\circeq$	\triangleq	$\triangleq$	\thicksim	$\thicksim$
\thickapprox	$\thickapprox$	\supseteqq	$\supseteqq$	\Supset	$\Supset$
\sqsupset	$\sqsupset$	\succcurlyeq	$\succcurlyeq$	\curlyeqsucc	$\curlyeqsucc$
\succsim	$\succsim$	\succapprox	$\succapprox$	\vartriangleright	$\vartriangleright$
\trianglerighteq	$\trianglerighteq$	\Vdash	$\Vdash$	\shortmid	$\shortmid$
\shortparallel	$\shortparallel$	\between	$\between$	\pitchfork	$\pitchfork$
\varpropto	$\varpropto$	\blacktriangleleft	$\blacktriangleleft$	\therefore	$\therefore$
\backepsilon	$\backepsilon$	\blacktriangleright	$\blacktriangleright$	\because	$\because$

C

TAB. C.14 – Négations de flèches de l'$\mathcal{AMS}$

`\nleftarrow`	$\nleftarrow$	`\nrightarrow`	$\nrightarrow$	`\nLeftarrow`	$\nLeftarrow$
`\nRightarrow`	$\nRightarrow$	`\nleftrightarrow`	$\nleftrightarrow$	`\nLeftrightarrow`	$\nLeftrightarrow$

TAB. C.15 – Lettres grecques et hébraïques de l'$\mathcal{AMS}$

`\digamma`	$\digamma$	`\varkappa`	$\varkappa$	`\beth`	$\beth$	`\daleth`	$\daleth$	`\gimel`	$\gimel$

TAB. C.16 – Délimiteurs de l'$\mathcal{AMS}$

`\ulcorner`	$\ulcorner\alpha$	`\urcorner`	$\urcorner\alpha$	`\llcorner`	$\llcorner\alpha$	`\lrcorner`	$\lrcorner\alpha$

TAB. C.17 – Négations de relations de l'$\mathcal{AMS}$

`\nless`	$\nless$	`\nleq`	$\nleq$	`\nleqslant`	$\nleqslant$
`\nleqq`	$\nleqq$	`\lneq`	$\lneq$	`\lneqq`	$\lneqq$
`\lvertneqq`	$\lvertneqq$	`\lnsim`	$\lnsim$	`\lnapprox`	$\lnapprox$
`\nprec`	$\nprec$	`\npreceq`	$\npreceq$	`\precnsim`	$\precnsim$
`\precnapprox`	$\precnapprox$	`\nsim`	$\nsim$	`\nshortmid`	$\nshortmid$
`\nmid`	$\nmid$	`\nvdash`	$\nvdash$	`\nvDash`	$\nvDash$
`\ntriangleleft`	$\ntriangleleft$	`\ntrianglelefteq`	$\ntrianglelefteq$	`\nsubseteq`	$\nsubseteq$
`\subsetneq`	$\subsetneq$	`\varsubsetneq`	$\varsubsetneq$	`\subsetneqq`	$\subsetneqq$
`\varsubsetneqq`	$\varsubsetneqq$	`\ngtr`	$\ngtr$	`\ngeq`	$\ngeq$
`\ngeqslant`	$\ngeqslant$	`\ngeqq`	$\ngeqq$	`\gneq`	$\gneq$
`\gneqq`	$\gneqq$	`\gvertneqq`	$\gvertneqq$	`\gnsim`	$\gnsim$
`\gnapprox`	$\gnapprox$	`\nsucc`	$\nsucc$	`\nsucceq`	$\nsucceq$
`\succnsim`	$\succnsim$	`\succnapprox`	$\succnapprox$	`\ncong`	$\ncong$
`\nshortparallel`	$\nshortparallel$	`\nparallel`	$\nparallel$	`\nvDash`	$\nvDash$
`\nVDash`	$\nVDash$	`\ntriangleright`	$\ntriangleright$	`\ntrianglerighteq`	$\ntrianglerighteq$
`\nsupseteq`	$\nsupseteq$	`\nsupseteqq`	$\nsupseteqq$	`\supsetneq`	$\supsetneq$
`\varsupsetneq`	$\varsupsetneq$	`\supsetneqq`	$\supsetneqq$	`\varsupsetneqq`	$\varsupsetneqq$

TAB. C.18 – Opérateurs binaires de l'$\mathcal{AMS}$

\dotplus	$\dotplus$	\smallsetminus	$\smallsetminus$	\Cap	$\Cap$
\Cup	$\Cup$	\barwedge	$\barwedge$	\veebar	$\veebar$
\doublebarwedge	$\doublebarwedge$	\boxminus	$\boxminus$	\boxtimes	$\boxtimes$
\boxdot	$\boxdot$	\boxplus	$\boxplus$	\divideontimes	$\divideontimes$
\ltimes	$\ltimes$	\rtimes	$\rtimes$	\leftthreetimes	$\leftthreetimes$
\rightthreetimes	$\rightthreetimes$	\curlywedge	$\curlywedge$	\curlyvee	$\curlyvee$
\circleddash	$\circleddash$	\circledast	$\circledast$	\circledcirc	$\circledcirc$
\symcenterdot	$\symcenterdot$	\intercal	$\intercal$		

TAB. C.19 – Symboles divers de l'$\mathcal{AMS}$

\hbar	$\hbar$	\hslash	$\hslash$	\vartriangle	$\vartriangle$
\triangledown	$\triangledown$	\square	$\square$	\lozenge	$\lozenge$
\circledS	$\circledS$	\angle	$\angle$	\measuredangle	$\measuredangle$
\nexists	$\nexists$	\mho	$\mho$	\Finv	$\Finv$
\Game	$\Game$	\Bbbk	$\Bbbk$	\backprime	$\backprime$
\varnothing	$\varnothing$	\blacktriangle	$\blacktriangle$	\blacktriangledown	$\blacktriangledown$
\blacksquare	$\blacksquare$	\blacklozenge	$\blacklozenge$	\bigstar	$\bigstar$
\sphericalangle	$\sphericalangle$	\complement	$\complement$	\eth	$\eth$
\diagup	$\diagup$	\diagdown	$\diagdown$		

C.3 Symboles du package textcomp

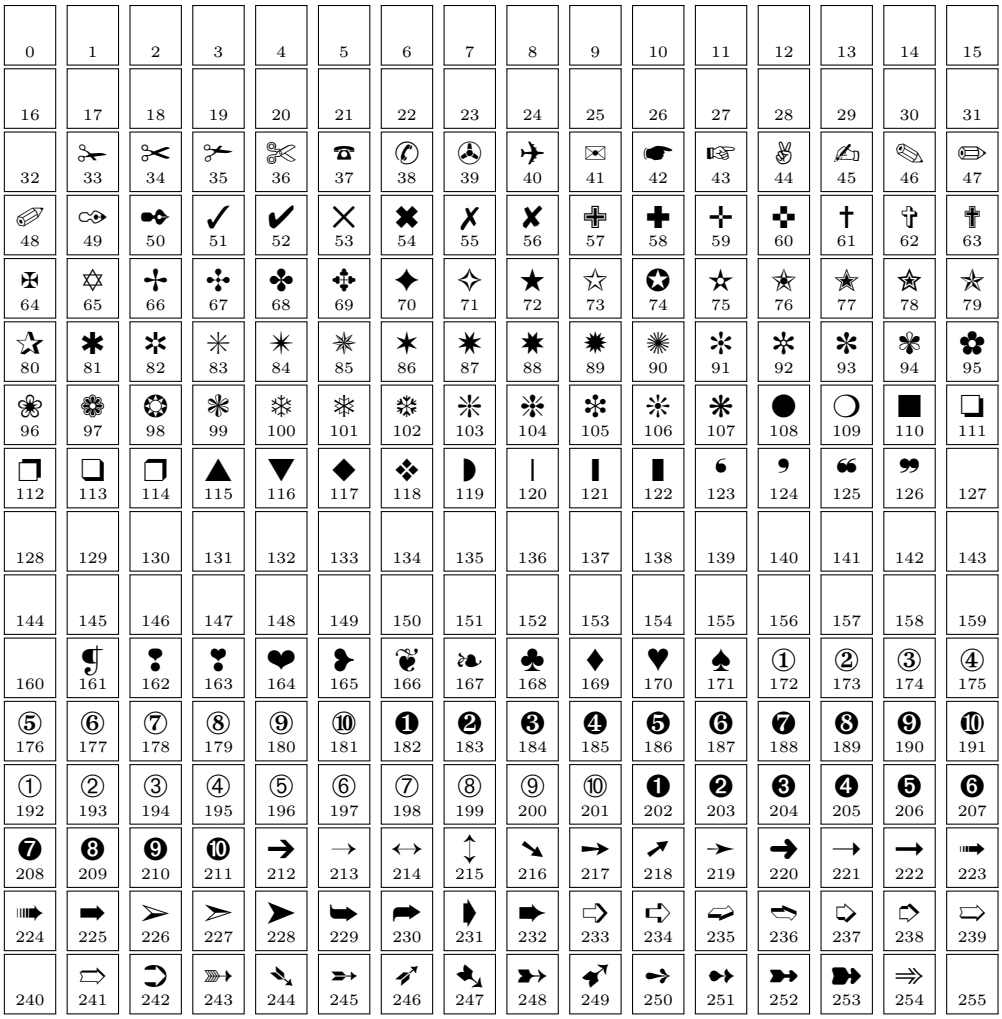
TAB. C.20 – Symboles du package textcomp.

\textacutedbl	″	\textascendercompwordmark	
\textasciiacute	´	\textasciibreve	˘
\textasciicaron	ˇ	\textasciidieresis	¨
\textasciigrave	`	\textasciimacron	¯
\textasterisksymcentered		\textbaht	฿
\textbardbl	‖	\textbigcircle	◯
\textblank	ƀ	\textborn	⋆
\textbrokenbar	¦	\textbullet	•
\textcapitalcompwordmark		\textcelsius	℃
\textcent	¢	\textcentoldstyle	¢
\textcircledP	Ⓟ	\textcolonmonetary	₡
\textcopyleft	ↄ	\textcopyright	©

TAB. C.21 – Symboles du package textcomp (suite).

\textcurrency	¤	\textdagger	†
\textdaggerdbl	‡	\textdblhyphen	‐
\textdblhyphenchar	-	\textdegree	°
\textdied	†	\textdiscount	⅋
\textdiv	÷	\textdivorced	⚯
\textdollar	$	\textdollaroldstyle	$
\textdong	₫	\textdownarrow	↓
\texteightoldstyle	8	\textestimated	℮
\texteuro	€	\textfiveoldstyle	5
\textflorin	f	\textfouroldstyle	4
\textfractionsolidus	/	\textgravedbl	‶
\textguarani	₲		
\textinterrobang	‽	\textinterrobangdown	⸘
\textlangle	⟨	\textlbrackdbl	⟦
\textleaf	🍃	\textleftarrow	←
\textlira	₤	\textlnot	¬
\textlquill	⁅	\textmarried	⚭
\textmho	℧	\textminus	−
\textmu	µ	\textmusicalnote	♪
\textnaira	₦	\textnineoldstyle	9
\textnumero	№	\textohm	Ω
\textonehalf	½	\textoneoldstyle	1
\textonequarter	¼	\textonesuperior	¹
\textopenbullet	○	\textordfeminine	ª
\textordmasculine	º	\textparagraph	¶
\textperiodsymcentered	·	\textpertenthousand	‱
\textperthousand	‰	\textpeso	₱
\textpilcrow	¶	\textpm	±
\textquotesingle	'	\textquotestraightbase	‚
\textquotestraightdblbase	„	\textrangle	⟩
\textrbrackdbl	⟧	\textrecipe	℞
\textreferencemark	※	\textregistered	®
\textrightarrow	→	\textrquill	⁆
\textsection	§	\textservicemark	℠
\textsevenoldstyle	7	\textsixoldstyle	6
\textsterling	£	\textsurd	√
\textthreeoldstyle	3	\textthreequarters	¾
\textthreequartersemdash	—	\textthreesuperior	³
\texttildelow	~	\texttimes	×
\texttrademark	™	\texttwelveudash	―
\texttwooldstyle	2	\texttwosuperior	²
\textuparrow	↑	\textwon	₩
\textyen	¥	\textzerooldstyle	0

TAB. C.22 – La police Zapf Dingbats



C

TAB. C.23 – Le police Symbol

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	! 33	∀ 34	# 35	∃ 36	% 37	& 38	∍ 39	(40	) 41	* 42	+ 43	, 44	− 45	. 46	/ 47
0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
≅ 64	Α 65	Β 66	Χ 67	Δ 68	Ε 69	Φ 70	Γ 71	Η 72	Ι 73	ϑ 74	Κ 75	Λ 76	Μ 77	Ν 78	Ο 79
Π 80	Θ 81	Ρ 82	Σ 83	Τ 84	Υ 85	ς 86	Ω 87	Ξ 88	Ψ 89	Ζ 90	[91	∴ 92	] 93	⊥ 94	_ 95
‾ 96	α 97	β 98	χ 99	δ 100	ε 101	φ 102	γ 103	η 104	ι 105	φ 106	κ 107	λ 108	μ 109	ν 110	ο 111
π 112	θ 113	ρ 114	σ 115	τ 116	υ 117	ϖ 118	ω 119	ξ 120	ψ 121	ζ 122	{ 123	| 124	} 125	~ 126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	ϒ 161	′ 162	≤ 163	⁄ 164	∞ 165	ƒ 166	♣ 167	♦ 168	♥ 169	♠ 170	↔ 171	← 172	↑ 173	→ 174	↓ 175
° 176	± 177	″ 178	≥ 179	× 180	∝ 181	∂ 182	• 183	÷ 184	≠ 185	≡ 186	≈ 187	… 188	| 189	— 190	↵ 191
ℵ 192	ℑ 193	ℜ 194	℘ 195	⊗ 196	⊕ 197	∅ 198	∩ 199	∪ 200	⊃ 201	⊇ 202	⊄ 203	⊂ 204	⊆ 205	∈ 206	∉ 207
∠ 208	∇ 209	® 210	© 211	™ 212	∏ 213	√ 214	· 215	¬ 216	∧ 217	∨ 218	⇔ 219	⇐ 220	⇑ 221	⇒ 222	⇓ 223
◊ 224	⟨ 225	® 226	© 227	™ 228	∑ 229	⎛ 230	⎜ 231	⎝ 232	⎡ 233	⎜ 234	⎣ 235	⎛ 236	⎧ 237	⎩ 238	⎪ 239
240	⟩ 241	∫ 242	⌠ 243	⎮ 244	⌡ 245	⎞ 246	⎟ 247	⎠ 248	⎤ 249	⎥ 250	⌋ 251	⎞ 252	⎫ 253	⌋ 254	255

Ↄ

D Notes de production

Sommaire

J'AI RASSEMBLÉ ici des éléments permettant d'exploiter les sources de ce document, comment les compiler, avec quelle distribution de LaTeX, comment les fichiers sont organisés, etc.

D.1 Distribution du moment

Le présent ouvrage a été compilé sur deux systèmes Linux contenant la distribution TeXlive :

1. la distribution TeXlive de la Ubuntu 07-10
2. une distribution TeXlive pour Debian Etch provenant de http://people.debian.org/~preining/TeX/

Toute autre expérience fructueuse sur des systèmes différents est bien entendu la bienvenue... De plus pour la génération du document au format pdf, seule la famille Computer Modern dans sa version « CM-Super » a été testée.

D.2 Les sources du manuel

D.2.1 Structure

Les sources du manuel sont organisées selon le principe suivant :
- sources LaTeX dans le répertoire `corps` avec un fichier par chapitre ;
- les styles (`sty` et `cls`) dans le répertoire `styles` ;
- les images dans un répertoire `pngs` ;
- le répertoire `texs` contient des sources LaTeX à inclure dans le document (modèles de lettre, de fax, code contenant des appels à Psfrag ou Pstricks) ;
- les sources xfig dans le répertoire `figs` ;
- tout ce qui a trait à l'index, à la bibliographie et au glossaire est stocké dans le répertoire `bibidx` ;

Les sources xfig et certains « bouts » de fichiers LaTeX sont traduits au format pdf, postscript encapsulé ou non, par un makefile les stockant :
- dans le répertoire `epss`
- dans le répertoire `pdfs`

– dans le répertoire `pss`

selon le moteur utilisé (`latex` ou `pdflatex`).

D.2.2 Styles

Le fichier `manuel.cls` contient la définition de la classe du manuel. Ce fichier fait appel à une série de packages « du commerce » et une série de packages « maison ». On trouve, pour ces derniers, un fichier source pour :

– chaque « nouveau jouet » : onglets, nota, sommaire, glossaire, boite avec un titre (`titlebox.sty`), exemples, lettrine, renvois (`voir.sty`), citations et épigraphes
– le sommaire
– la géométrie globale du document
– l'allure des en-têtes et pieds de page
– l'allure des sections/chapitres/etc.
– des commandes en vrac utilisées dans le document (`manumac.sty`)

Sauf indication contraire, ces fichiers portent un nom ressemblant étrangement à ce qu'ils contiennent.

D.3 Compilation

D.3.1 Makefile

L'arborescence racine des sources contient un fichier Makefile pour le manuel, qu'il faut copier :

```
cp Makefile.manuel Makefile
```

D.3.2 Figures

Les figures peuvent êtres compilées grâce aux commandes :

```
#1. produire les figures
make figs
#2. générer les figures « récursives » de l'introduction
command latex --interaction nonstopmode guide-local
command latex --interaction nonstopmode guide-local
# 2 fois pour avoir la table des matières
make recursefigs
```

Il faut disposer du logiciel transfig connexe à xfig pour traduire les sources en pdf et eps. Sur le site http://cours.enise.fr/info/latex est disponible une archive contenant les figures déjà traduites...

D.3.3 Dvi et postscript

```
latex guide-local
make  bibindex
latex guide-local
latex guide-local
dvips guide-local -o guide-local
```

D.3.4 Pdf

Rien de particulier

```
pdflatex guide-local
make  bibindex
pdflatex guide-local
pdflatex guide-local
```

D.3.5 Versions imprimables sur papier A4

Les différentes versions imprimables dont il est question au paragraphe page x sont générées à partir d'un fichier postscript à jour :

```
dvips guide-local -o guide-local
```

et des utilitaires psutils. Les versions « reliure longue », « reliure courte 1 exemplaire » et « reliure courte 2 exemplaires » sont respectivement génrérées par les commandes suivantes :

```
make print-0
make print-1
make print-2
```

D.3.6 Nettoyage de printemps

En bref :

```
make cleanfigs      ← efface tous les eps/pdfs/...
make cleantex       ← efface tous les fichiers auxiliaires
make cleandocs      ← efface tous les documents générés
                      (dvi/ps/pdf/...)
```

Bibliographie

[1] Knuth (D. E.), *The Art of Computer Programming*, vol. 1–3. Addison-Wesley, 1997–98.

Trois volumes sur « l'art de programmer ». Un quatrième tome est en préparation. Cet ensemble de livres a été accueilli par la communauté scientifique comme un des ouvrages les plus importants de ce siècle (cf. http://www.amsci.org/amsci/bookshelf/centurylist.html à ce sujet et http://www-cs-staff.stanford.edu/~knuth/taocp.html sur la page web de Knuth pour plus d'info sur « TAOCP »).

[2] Lamport (L.), *LaTeX : A Document Preparation System*. Addison-Wesley, édition 2e, 1994.

Le livre de l'auteur de LaTeX dans sa seconde édition couvrant LaTeX 2ε. Bien évidemment une très bonne introduction, avec en fin d'ouvrage un guide de référence des commandes.

[3] Knuth (D. E.), *The TeXBook*. Addison-Wesley, 1988.

LA bible de TeX. Un livre plein de « virages dangereux » expliquant très précisément le fonctionnement interne de TeX. C'est un ouvrage de référence assez difficile à lire et qui ne constitue pas une introduction à TeX destinée aux débutants — à mon avis.

[4] Goossens (M.), Mittelbach (F.) et Samarin (A.), *The LaTeX companion*. Addison-Wesley, 1994.

LA bible de LaTeX2ε et de ses packages. Ce livre qui est un must pour tout utilisateur qui veut comprendre les fonctions internes de LaTeX contient des informations très précises sur : la manière de personnaliser les mises en pages par défaut, l'utilisation des fontes, moult packages, etc.

[5] Madsen (L.), « Avoid eqnarray ! », *The PracTeX Journal*, no 4, 2006.

Un article recensant les « pourquoi » ne pas utiliser cet environnement. L'article doit être disponible à http://home.imf.au.dk/daleif.

[6] Trettin (M.), « Une liste des péchés des utilisateurs de LaTeX », 2004.

Ce document connu sous le nom l2tabu, traite « des commandes et extensions obsolètes, et quelques autres erreurs ».

[7] Lozano (V.). « Tout ce que vous avez toujours voulu savoir sur unix sans jamais oser le demander », 2006. http://www.enise.fr/cours/info/unix.

[8] *Lexique des règles typographiques en usage à l'Imprimerie nationale*, 1990.

Il s'agit de l'ensemble des règle qui sont appliquées dans les livres produits par l'Imprimerie nationale. Ce lexique est présenté sous la forme de thèmes classés par ordre alphabétique. C'est une source d'informations intéressante puisqu'elle fait référence dans l'imprimerie française.

[9] Perrousseaux (Y.), *Manuel de typographie française élémentaire*. Atelier Perrousseaux, 1995.

Un « petit » livre très pédagogique sur la typographie, contenant un historique très intéressant, et une liste de règles en usage dans le monde de la typographie.

[10] André (J.), « Petites leçons de typographie », 1990.

On doit pouvoir trouver ce document à l'url http://jacques-andre.fr Il s'agit d'un article intéressant sur la typographie avec beaucoup d'exemples sur l'emploi des majuscules, la ponctuation, l'usage du souligné et les caractères français.

[11] Goossens (M.), Rahtz (S.) et Mittelbach (F.), *The LaTeX Graphics Companion*. Addison-Wesley, 1997.

Par les auteurs du *LaTeX Companion*, un livre sur l'utilisation des graphiques au sens large du terme, avec notamment une exploration des packages permettant de dessiner avec LaTeX et une présentation de l'utilisation des fontes Postscript.

[12] Bitouzé (D.) et Charpentier (J.-C.), « LaTeX », dans *Collection Synthex*. Pearson Education France, septembre 2006.

[13] Appel (W.), Chevalier (E.), Cornet (E.) *et al.*, « LaTeX pour l'impatient », dans *Technique et pratique*. H & K, 2007.

[14] *the UK List of TeX Frequently Asked Questions on the Web*.

une mine d'informations en anglais, disponible à http://www.tex.ac.uk/cgi-bin/texfaq2html, listant les fameuses questions « fréquemment posées » sur TeX & LaTeX (contrairement à ce que le titre indique).

Glossaire

Compilation

Même si ce terme n'est pas très rigoureux d'un point de vue scientifique, on appelle compilation la phase permettant de traduire le source LaTeX en un fichier au format DVI ou PDF.

Document maître

C'est le document source qui contient le `begin{document}` dans le contexte d'un document divisé en plusieurs fichiers.

Document source

Un document texte contenant le texte et les commandes LaTeX. TeX C'est le document à ne pas perdre car il est à la source de la production papier, écran, etc. au même titre qu'un code source en langage C est la source d'un programme exécutable.

Dvi

Format de fichier *Device Independent* mis au point par Knuth de manière à créer, à partir du document source, un document dont le format est indépendant de la plateforme et du matériel.

Fichiers auxiliaires

Les fichiers produits par LaTeX lors d'une compilation. Ils portent le nom du document source, et ont une extension de trois lettres rappelant leur rôle.

Format

C'est un ensemble de commandes ou macro précompilées et stockées dans un fichier portant généralement l'extension `.fmt`. Les plus connus de ces ensembles sont le format `plain` de TeX et le format LaTeX.

Macro

C'est l'outil permettant de faire faire des choses compliquées à LaTeX en passant en ordre simple. Les macros, appelées aussi commandes, ressemblent un peu aux routines des langages de programmation.

PDF

Pour *portable document format*, format de fichier créé par la société Adobe, dont le but est de pouvoir échanger facilement des documents d'un système à un autre. Le format PDF peut être créé de plusieurs façons à partir d'un source LaTeX(cf. A page 187).

PostScript

Langage défini par la société Adobe pour décrire un document destiné à l'impression. Ce langage composé de primitives de bas niveau peut être interprété par des logiciels pour réaliser des aperçus avant impression ou directement par des circuits électroniques embarqués sur les imprimantes pour générer l'image à imprimer.

Références

Système permettant de manipuler les numéros des paragraphes, équations, chapitres, etc. de manière symbolique, pour s'affranchir de la difficulté de les mettre à jour lorsque l'on change la mise en page.

LaTeX

C'est l'ensemble de macros défini par Leslie Lamport au dessus de TeX. La version utilisée aujourd'hui est LaTeX 2_ε.

TeX

Le moteur de base, LaTeX étant un ensemble de macros formant une surcouche. La version de TeX est stabilisée à la version 3.14159, à chaque nouvelle version Knuth ajoute une décimale.

Index

Symboles

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Table des matières