Introduction à LaTeX

Nicolas Markey
markey@lsv.ens-cachan.fr

16 juillet 2002

Cette doc de 61 pages avait pour but, initialement, il y a quatre ans, de présenter les bases de LaTeX. Rien de bien sorcier au départ. Et puis au fur et à mesure, elle a grossi. C'est une bonne chose, je pense, et je t'encourage donc à ne pas hésiter à m'envoyer tes commentaires, remarques, questions, suggestions, conseils, petit bonjour à l'adresse électronique qui figure ci-dessus.

1 Introduction

1.1 LaTeX, c'est quoi ?

LaTeX (ça se prononce `[latek]`) est un traitement de texte écrit dans les années 1980 par Leslie Lamport. Il est basé sur TeX (prononcer `[tek]`), développé par Donald E. Knuth. La version actuelle de LaTeX est LaTeX 2_ε, c'est celle-là que je décris ici ; elle a succédé à LaTeX2.09 en 1993. La version actuelle de TeX est $3,14159$; les polices de caractère sont créées avec METAFONT, également écrit par Knuth, et dont la version actuelle est $2,7182$.

Contrairement aux traîtements de texte classiques, LaTeX n'est pas un traîtement de texte wysiwyg[1]. Cela signifie que le document final n'est pas celui qu'on voit avec l'éditeur de texte. Le document est en réalité un programme, qui doit être compilé avant de pouvoir être visualisé. C'est cela qui fait la force de LaTeX : si on ajoute quelques ligne au début du document, il n'est pas nécessaire de changer toute la mise en page du reste du document : LaTeX s'en charge. Par ailleurs, il se charge aussi de la numérotation des pages, des chapitres, des figures, il sait également faire des références croisées, la table des matières, l'index et la bibliographie. *Last, but not least*, LaTeX est un traitement de texte à prédominance mathématique ; si on le lui demande gentiment, il est capable de faire ce qu'aucun autre logiciel ne peut faire.

1.2 Comment ça marche ?

Pour écrire un texte en LaTeX, il faut procéder en plusieurs étapes :

1. comme je l'ai dit précédemment, un document LaTeX est un programme. Il doit donc être saisi à l'aide d'un éditeur de texte, comme par exemple `emacs`, `vi` ou autres. Ce programme devra être sauvegardé sous un nom se terminant de préférence par l'extension `.tex`. Appelons-le `doc.tex`,
2. il faudra ensuite compiler le document saisi. Cela se fait au moyen de la commande : `latex doc.tex`. Cette commande peut s'abréger en `latex doc` si le fichier a effectivement l'extension `.tex`. Lors de la compilation, plusieurs fichiers sont créés :
   - un fichier d'extension `.log` contenant les informations sur la compilation, la mémoire utilisée, les packages chargés, et aussi les avertissements et les messages d'erreur éventuels ;

[1]What You See Is What You Get : on voit le document au fur et à mesure qu'on l'écrit. C'est un avantage dans ce sens, mais inversement, cela nécessite une mise en page et un ajustement permanents, donc c'est soit très lent, soit de mauvaise qualité, soit, en général, les deux...

- un fichier `.aux` qui permet à LaTeX de gérer la numérotation, les références, la bibliographie, *etc.* . ;
- un fichier `.dvi` contenant le résultat de la compilation. Il est au format DVI[2].

Bien entendu, il peut y avoir d'autres fichiers auxiliaires créés, comme celui contenant la table des matières, celui contenant l'index, ...

3. pour visualiser le résultat, il faut ensuite taper `xdvi doc.dvi` , ou plus simplement `xdvi doc`. Ce fichier `doc.dvi` est le résultat de la compilation du fichier `doc.tex` par LaTeX (si la compilation ne s'est pas trop mal passé).
4. pour passer au format PostScript, on utilise la commande `dvips`. Par exemple, pour convertir `doc.dvi` au format PostScript, on tapera :

```
dvips -o doc.ps doc.dvi
```

Pour le visualiser, on utilise par exemple ghostview : `ghostview doc.ps`, ou gv : `gv doc.ps`. On peut aussi convertir le document au format PDF[3] à parti du fichier PostScript, avec la commande ps2pdf. Les fichiers PDF se lisent avec gv, xpdf ou Acrobat Reader (acroread). Cette conversion est néanmoins néfaste pour la qualité du document (notamment pour la visualisation à l'écran). Il est préférable d'utiliser PDFLaTeX pour créer du PDF, mais cela nécessite quelques modifications que je ne détaillerai pas ici.

5. enfin, pour imprimer le document, il suffit de taper la commande `dvips doc.dvi`, ou plus simplement `dvips doc`. Pour imprimer un fichier au format `.ps`, on utilise la commande lpr : `lpr doc.ps`.
6. il est possible de sélectionner des pages d'un fichier PostScript ou DVI : pour cela, il existe des petits programmes bien utiles :
   - dviconcat qui permet de concatener plusieurs fichiers DVI ;
   - psselect et dviselect qui permettent de choisir certaines pages, et de supprimer les autres ;
   - psnup qui permet de mettre deux pages sur une ;
   - pstops permet de redimensionner, tourner les pages ;
   - mpage permet de concaténer plusieurs fichiers PostScript, d'extraire une partie du document, de mettre plusieurs pages sur une seule, ...

1.3 Les distributions de LaTeX

Tout comme il existe plusieurs types de navigateurs pour visualiser des pages web, ou plusieurs distributions de Linux, il existe plusieurs distributions de LaTeX, qui diffèrent par leur implémentation, les outils de configuration dont elles disposent, les packages qu'elles fournissent et surtout par les systèmes d'exploitation pour lesquels elles sont prévues.

TeX et LaTeX sont des logiciels « libres », sous licence LPPL. Cela signifie tout d'abord que chacun peut l'utiliser gratuitement[4]. Mais aussi que tout le monde peut en consulter et en modifier[5] le code source. C'est ce qui a permis de les porter sur diverses architectures. LaTeX existe maintenant pour Unix, Macintosh, DOS, Windows, ... Pour chaque système d'exploitation, il existe même plusieurs distributions : teTeX[6], MikTeX, fpTeX[7], AzTeX, ...

Sous Unix, LaTeX est généralement installé dans un répertoire unique, contenant une arborescence `bin/`, une arborescence `man/`, ... et une arborescence `texmf/`, qui contient les packages, les docs, les fichiers de configuration, ...

1.4 Pour en savoir plus...

Plusieurs méthodes sont possibles. Dans l'ordre, je conseillerais de :

[2]DeVice Independant
[3]Portable Document Format
[4]Le package `french` et quelques autres sont cependant distribué en *shareware*.
[5]Sous certaines conditions, assez peu restrictives (changer le nom du fichier dans lequel on est allé trifouiller...).
[6]celle que je conseille pour Un*x.
[7]celle que je conseille sous Windows : elle est basée sur teTeX.

- essayer... C'est la meilleure façon d'apprendre !
- jeter un œil à la FAQ (`http://www.grappa.univ-lille3.fr/FAQ-LaTeX/`). De nombreuses questions classiques sont posées, il y a un moteur de recherche, bref, c'est la doc de référence pour les problèmes pas trop évolués.
- demander autour de soi : LaTeX est un logiciel très utilisé dans le monde scientifique,
- LaTeX est très documenté. Dans le répertoire `texmf/doc` se trouvent de nombreuses documentations sur les innombrables possibilités de LaTeX, Si tu ne trouves pas ce que tu veux, tu peux chercher sur le web, tu trouveras encore plus d'informations.
- lire les news : il existe un groupe de news francophone, `fr.comp.text.tex`, et un groupe international, `comp.text.tex`, qui peuvent apporter des réponses à toutes les questions que tu te poses. Il existe également des listes de discussion, par exemple `gut@ens.fr`.
- sur le web : le réseau CTAN[8] est un ensemble de sites qui recense tout ce qui a rapport avec LaTeX. Le site français de CTAN est situé à l'URL `http://www.loria.fr/services/tex`. Il existe par ailleurs un Groupe francophone d'Utilisateurs de TeX, nommé `GUTenberg`, dont le site web se trouve à l'adresse `http://www.gutenberg.eu.org/`.
- lire des livres : une bibliographie se trouve à la page 54. Elle est bien fournie, et n'est pas exhaustive.

2 Le langage LaTeX

2.1 Le format des commandes

En LaTeX, toutes[9] les commandes doivent commencer par le caractère \, et ne contiennent soit que des lettres (par exemple, `\LaTeX`), soit qu'un caractère (par exemple `\&`, qui permet d'écrire un &.).

Lors de la compilation, LaTeX cherche, dans le fichier, toutes les commandes, et les interprète. D'autres caractères sont également interprétés, comme par exemple :

$	entrée et sortie du mode mathématique ;
%	symbole de commentaire[10] : tout ce qui suit, jusqu'à la fin de la ligne, ne sera pas lu par LaTeX ;
@	symbole réservé par LaTeX, pour des raisons qu'il est trop compliqué de développer ici ;
{ et }	symboles de début et de fin de groupe.

La notion de groupe, qui désigne toute suite de commandes placées entre deux accolades qui se correspondent, est très importante pour LaTeX : un groupe permet *grosso modo* de limiter la portée de certaines commandes, et donc de modifier la définition d'une commande *localement*. J'en reparlerai à la page 15.

2.2 L'inclusion de sous-programmes

Comme tout bon langage de programmation, LaTeX permet d'inclure des « sous-programmes » afin de structurer et de clarifier le document source, et de permettre de réutiliser facilement certaines définitions. Ces sous-programmes peuvent être de deux types :

- un « sous-programme » qui définit des commandes ou des environnements. Par exemple, il est possible, si on s'en sert souvent, de coder l'écriture de $\mathbb{Z}/p\mathbb{Z}$ par la commande `\Z p`, où `p` peut être n'importe quel chiffre ou lettre. Un autre intérêt est que, si on souhaite ensuite écrire $\mathbb{Z}/p$ à la place de $\mathbb{Z}/p\mathbb{Z}$ dans tout le document, il suffit de changer la définition de `\Z` ;
- un « sous-programme » qui contient une partie du document. Par exemple, ce document est formé de 5 sous-programmes, chacun correspondant à une section. De cette manière, on peut aisément trouver le début de la section 3 pour y apporter d'éventuelles modifications.

Dans le premier cas, on appellera ça un *package*, on lui donnera l'extension `.sty`, et on l'inclura dans le document par la commande `\usepackage`. Il importe de noter qu'un package ne peut pas produire de texte, juste définir des commandes. À l'inverse, dans le deuxième cas, un fichier contenant une partie du document peut contenir des définitions de commandes. L'extension restera `.tex`, et on inclura le fichier à

[8]Comprehensive TeX Archive Network
[9]ou presque...

l'aide de la commande `\input`, ou de la commande `\include`. Il y a une assez grande différence entre ces deux dernières commandes : la commande `\input` va « bêtement » inclure les sous-programmes, comme s'ils étaient effectivement dans le document source. La commande `\include` est plus subtile. On inclut tous les fichiers, et pour selectionner ceux que l'on veut voir compiler, on utilise la commande `\includeonly` avant les `\include`. Ainsi, LaTeX lira tous les fichiers inclus, mais ne compilera que ceux qui lui sont indiqués par `\includeonly`, en gardant les numéros de page, la table des matières, ... L'intérêt d'une telle manœuvre apparaît quand on commence à faire des longs textes, avec beaucoup de graphismes : compiler tout le document à chaque fois fait perdre beaucoup de temps.

Voyons un exemple : pour inclure le fichier `chap3.tex`, qui contient le chapitre 3 du document, on tape

```
\input{chap3}
```

Pour inclure le chapitre 3 en conservant le « contexte » des chapitres précédents, on tapera

```
\includeonly{chap3}
\include{chap1}
\include{chap2}
\include{chap3}
```

Cependant, si on utilise `\include`, il faut savoir que LaTeX change de page à chaque nouveau fichier. Donc, il faut bien découper le document en chapitres. Ce n'est pas le cas pour `\input`[11].

2.3 La forme générale d'un fichier

Un fichier `.tex` se décompose en deux parties : tout d'abord, le préambule, qui définit toutes les caractéristiques générales du document, et le document proprement dit. Le préambule commence nécessairement par définir la *classe* du document, avec la commande `\documentclass`. La fin du préambule, et le début du document effectif, sont indiqués par `\begin{document}`. Enfin, `\end{document}` signale la fin du document et donc du fichier. Ces trois commandes sont indispensables dans tout fichier LaTeX.

La section 3 décrira le contenu du préambule, la section 4, celui du corps du document. La section 5 parlera de mathématiques, et la section 6, des graphismes sous LaTeX.

3 Le préambule

C'est cette partie qui définit les caractéristiques générales du document : elle définira par exemple la taille des caractères à utiliser, la langue[12], ainsi que certains « sous-programmes » éventuels à charger.

3.1 La classe du document

La classe du document est définie par la commande `\documentclass`. Cette commande doit être suivie de la classedu document, entre accolades. Les trois classes les plus utilisées sont `article`, `report` et `book`. Leurs caractéristiques sont définies dans le tableau 1. Ce ne sont pas les seules caractéristiques de ces classes, mais ce sont les principales.

La commande `\documentclass` peut aussi prendre des arguments optionnels, qui seront placés entre crochets avant la classe. Ces arguments permettent de définir, par exemple, la taille de la police de caractères, si on veut un document recto-verso, en deux colonnes... Les options possibles sont données dans le tableau 2.

La première ligne d'un document sera donc typiquement[13]

```
\documentclass[a4paper,notitlepage]{report}
```

[11] En déduire que c'est `\input` qui a été utilisé ici...
[12] Les règles typographiques ne sont pas les mêmes en anglais et en français...
[13] En fait, la première ligne doit plutôt être une ligne de commentaire, qui décrit le document...

`article` : pour les petits documents : le titre n'est pas mis sur une page à part, le document se découpe en parties, sections, sous-sections, sous-sous-sections, paragraphes et sous-paragraphes. C'est la classe utilisée pour écrire cette documentation ;

`report` : pour les documents un peu plus gros. Le titre est mis sur une page à part (la première !). Cette classe permet de découper le document en chapitres, en plus des découpages de la classe article. Chaque nouveau chapitre commence sur une nouvelle page ;

`book` : pour les gros documents. Cette classe permet les mêmes découpages que la classe report. Cependant, elle laisse de nombreuses pages blanches, et il ne faut donc l'utiliser que pour les documents d'au moins 100 pages.

Tab. 1 – Les classes de documents

`10pt`, `11pt`, `12pt` : définit la taille de la police de caractères. Ce document est en `10pt`, qui est la valeur par défaut ;

`a4paper`, `a5paper`, `legalpaper`, `letterpaper`... : définit la taille du papier utilisé, et donc la taille du texte qui sera imprimé. L'option `a4paper` correspond à une feuille de dimensions 21 cm par 29, 7 cm, mais l'option par défaut est `letter`, qui correspond *grosso modo* à 21.6 cm par 28 cm ;

`oneside`, `twoside` : permet d'indiquer que le document est destiné à être imprimé en recto simple, ou recto-verso. Par défaut, c'est `oneside` ;

`titlepage`, `notitlepage` : indique si le titre doit être placé sur une page séparée ou en haut de la première page. Par défaut, c'est `titlepage` pour les classes `report` et `book`, et `notitlepage` pour le classe `article` ;

`onecolumn`, `twocolumn` : comme leur nom l'indique... l'option par défaut étant `onecolumn` ;

`openright`, `openany` : permet d'indiquer à LaTeX que l'on veut que chaque nouveau chapitre commence sur une page de droite, ou sur n'importe quelle page. Cette option ne concerne pas la classe `article`. C'est `openright` par défaut pour `book`, et `openany` pour `report` ;

Tab. 2 – Principales options pour la classe d'un document

3.2 Les packages

Un *package*[14] est un fichier qui contient des définitions de sous-programmes, appelés *macros*. La notion de *package* correspond à la notion de module, dans un langage de programmation. Chaque package a un objectif, comme par exemple permettre d'écrire en deux colonnes, ou inclure des figures... Pour inclure un package dans un fichier LaTeX, il faut utiliser la commande `\usepackage`, suivie des éventuelles options, entre crochets, et du nom du package, entre accolades. Par exemple, pour inclure le package `truc.sty` avec l'option `machin`, on tapera :

```
\usepackage[machin]{truc}
```

Le nombre de package étant en constante évolution, il n'existe pas de liste exhaustive des packages existant. La meilleure référence est [GMS98], qui dresse une liste assez complète des principaux packages. Pour ma part, je vais juste citer les principaux, ainsi qu'une petite description de leurs fonctions[15].

`babel` **:** comme son nom l'indique, c'est un package qui permet de « parler » plusieurs langues. Par exemple, pour du français, on mettra :

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

En incluant cette ligne, la date, la typographie, les sections, tout est francisé. Même la date : aujourd'hui (enfin, le jour où j'ai compilé cette doc...), on est le 16 juillet 2002Bien sûr, on peut aussi parler `english`, `german`, `breton`, `catalan`, `czech`, et j'arrête là mais il y en a plein d'autres.

`inputenc` **:** permet de définir le codage des caractères, en lui passant l'option adéquate. Sous UNIX, on utilise généralement le codage `ISO-Latin1`. En conséquence, on inclura la ligne suivante :

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

`fontenc` **:** permet de définir le codage des fontes. Le codage `T1` est le codage de fontes le plus utilisé. On mettra donc toujours (enfin, chacun fait ce qu'il veut, mais bon...) :

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

`multicol` **:** permet d'écrire sur deux (ou plus) colonnes.

Comme ici, par exemple. C'est assez pratique lorsqu'on a beaucoup diminué les marges, comme dans ce document. On pour avoir un style qui fait un peu « journal ». Par contre, ce package ne permet pas de faire, par exemple, un texte en deux langues, en utilisant une colonne pour chaque langue. Le problème qui se pose est qu'il faut s'arranger, dans ce genre de texte, que les paragraphes dans chaque langue se correspondent bien. On utilisera plutôt le package `parallel`.

`amsmath` **:** ce package, comme son nom l'indique, permet de taper des documents scientifiques en général, et mathématiques en particulier. Il a été écrit par l'$\mathcal{AMS}$[16]. Le chapitre 5 en parlera en détails...

`theorem` **:** permet de définir un style particulier pour saisir les définitions, propositions ou théorèmes. J'en parlerai dans le chapitre sur les mathématiques également.

`graphicx` **:** c'est lui qui permet d'inclure des figures au format PostScript encapsulé, d'extension `.eps`, ou PostScript, d'extension `.ps`. Il faut préciser le convertisseur PostScript utilisé, ici ddvips, en option. L'inclusion de ce package se fera donc par la commande

[14] L'extension du nom de fichier d'un package est `.sty`, comme par exemple `truc.sty`. On omettra en général cette extension, en parlant simplement du package `truc`.

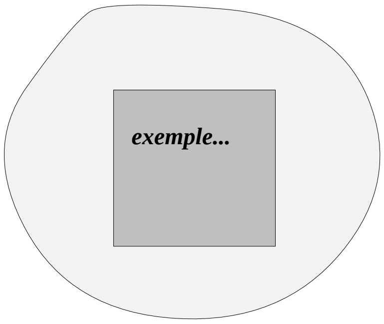
[15] L'annexe E contient une liste un peu plus fournie... et l'index des packages, à la page 59, liste tous les packages cités dans ce document.

[16] American Mathematical Society

```
\usepackage[dvips]{graphicx}
```

Un petit exemple rapide :

```
\includegraphics[scale=.5]{ex2.eps}
```



J'en reparlerai plus en détail dans le chapitre consacré aux graphismes sous LaTeX, à la page 29,

`xypic` : il permet de dessiner toutes sortes de diagrammes et de petits dessins dans tous les sens. Il fait également partie des packages dont je parlerai au chapitre sur les graphismes,

`pstricks` : il permet également de créer de nombreux diagrammes, et même beaucoup plus. Pour s'en convaincre, il suffit de regarder l'exemple ci-dessous et de lire le chapitre des graphismes :

```
\pstextpath(0,0){\pscurve[linestyle=none]
(0,0)(2,-1)(4,0)(6,-1)(8,0)(10,-1)}
{Voila un bel exemple, mais \c{c}a sert
pas tous les jours...}
```

Voila un bel exemple, mais ça sert pas tous les jours...

Il est important de n'inclure que les packages nécessaires à la compilation. En effet, ça fait gagner beaucoup de temps, ca évite les problèmes de mémoire, mais en plus, ça évite les conflits entre packages. Il est donc prudent d'éviter d'avoir un préambule par défaut qui inclurait tous les packages que tu as utilisés dans ta vie...

3.3 Définitions diverses

Une fois que tous les packages nécessaires sont inclus, et avant de commencer le document proprement dit, on peut définir des « macro-commandes ». Je ne m'étendrai pas sur le sujet, quelques exemples suffiront... Les commandes LaTeX utilisées ici seront expliquées plus tard, notamment à la section 5.

– `\newcommand` permet de définir une commande qui n'existait pas auparavant. `\renewcommand` permet de (re)définir une commande qui existait déjà :

```
\newcommand\machin{bonjour}
\machin, \machin, \machin, \machin, \machin
\renewcommand\machin{au revoir}
\machin, \machin, \machin, \machin, \machin
```

bonjour, bonjour, bonjour, bonjour, bonjour
au revoir, au revoir, au revoir, au revoir, au revoir

– Les espaces qui suivent l'appel à une macro sont mangés par la macro, il faut donc penser à les inclure dans la définition si c'est possible, et penser à les mettre à la main sinon. Le package `xspace` peut être utile, il ajoute l'espace lui-même quand c'est nécessaire :

```
\newcommand{\ssi}{si, et seulement si,}
$A$ est vrai \ssi   $B$ est vrai
\renewcommand{\ssi}{si, et seulement si,\ }
$A$ est vrai \ssi $B$ est vrai
```

A est vrai si, et seulement si,B est vrai
A est vrai si, et seulement si, B est vrai

– Une commande peut prendre un ou plusieurs arguments. Lors de la définition, il faut préciser le nombre d'arguments (c'est le nombre qui est indiqué entre crochets apres le nom de la commande, dans la définition). Ensuite, dans la définition, on rappelle le premier argument par `#1`, le deuxième par `#2`, ... La commande `\ensuremath` permet de passer en mode mathématique *si nécessaire*, *i.e.* si on n'y était pas déjà :

```
\newcommand\Zsur[1]{\mathbb{Z}/#1\mathbb{Z}}
$\Zsur3$, $\Zsur p$, $\Zsur234$, $\Zsur{234}$
\newcommand\Zs[1]{{\ensuremath{\Zsur{#1}}}}
$\Zs3$ mais aussi \Zs{pq}...
```

$\mathbb{Z}/3\mathbb{Z}$, $\mathbb{Z}/p\mathbb{Z}$, $\mathbb{Z}/2\mathbb{Z}34$, $\mathbb{Z}/234\mathbb{Z}$
$\mathbb{Z}/3\mathbb{Z}$ mais aussi $\mathbb{Z}/pq\mathbb{Z}$...

– Un environnement se définit par la commande `\newenvironment`. Il existe aussi, bien sûr, une commande `\renewenvironment`. Noter également que le `%` en fin de ligne permet d'éviter d'introduire des espaces indésirables, et que les espaces en début de ligne ne comptent pas :

```
\newenvironment{exemple}{\textbf{début de l'envi%
   ronnement }}{\textbf{ fin de l'environnement}}

\begin{exemple}
le texte de l'exemple...
\end{exemple}
```

début de l'environnement
le texte de l'exemple...
fin de l'environnement

Notons tout de même dès maintenant la différence entre une commande et un environnement. Une commande n'a pas de début ni de fin. Elle peut avoir des arguments, et s'applique soit à ses arguments, soit à tout ce qui la suit. Un environnement, ça commence par `\begin`, et ça se termine par `\end`. Par exemple, `\begin{exemple}`, c'est le début de l'environnement `exemple`, qui devra obligatoirement se terminer par `\end{exemple}`.Toutes ces définitions peuvent être placées dans un fichier `macros.tex`, par exemple, et être incluses par la commande `\input`, ou dans un package[17] `macros.sty` et incluses dans le préambule avec la commande `\usepackage`.

Signalons enfin la méthode qui permet de créer un titre : à la fin du préambule, on définit le nom de l'auteur, le titre du document et la date de création avec les commandes `\author`, `\title` et `\date`. Noter que `\today` permet d'avoir la date de la compilation. Ces commandes ont uniquement pour effet de créer des variables contenant le nom de l'auteur, le titre et la date. Pour que LaTeX génère effectivement le titre, il faut utiliser la commande `\maketitle`, à l'intérieur du document (*i.e.* après le `\begin{document}`), à l'endroit où on veut placer le titre (en général, au début !)

Le fichier principal d'un document tapé en LaTeX aura alors la forme du fichier de la figure 1.

4 Le corps du document

Nous entrons ici dans le vif du sujet. Rappelons que le corps du document doit commencer par la ligne `\begin{document}` et se terminer par `\end{document}`. Leur absence posera de graves problèmes au compilateur, et ne produira rien en sortie... Par ailleurs, il est possible de mettre du texte après le `\end{document}`, mais LaTeX ne le prendra pas en compte, puisque le document est fini.

Cette section se découpera en plusieurs parties ; la première partie sera consacrée au découpage du texte. La seconde parlera de la mise en page. La troisième partie présentera les changements de fontes, la quatrième montrera comment obtenir des caractères « particuliers ». Enfin, quelques « effets spéciaux »[18] seront présentés dans la dernière partie.

[17]Enfin... Un package, ça fait plus sérieux, ça a souvent une certaine unité, un but... C'est pas deux lignes qui se battent en duel pour définir une pauvre commande... :-)

[18] Je mets ici une référence qui servira par la suite. Elle contient la commande `\label{pied}`.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% fichier exemple.tex                                  %%%
%%% -> un exemple de fichier LaTeX                       %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\usepackage[francais]{babel}
\usepackage[dvips]{graphicx}

\title{Un exemple de fichier \LaTeX}
\author{moi}
\date\today

\begin{document}

%% Le titre
\maketitle

%% Introduction
\include{intro}
%% Partie 1 :
\include{part1}
%% Partie 2 :
\include{part2}
%% Partie 3 :
\include{part3}
%% Conclusion
\include{conclusion}

\end{document}
```

FIG. 1 – Un exemple de fichier principal

4.1 Le découpage du texte

J'en parlais déjà à la page 5 : un document peut se découper en parties, sections, sous-sections, ... L'ordre et l'effet de ces découpages sont donnés au tableau 3.

`\part`	le plus gros découpage : dans les classes `report` et `book`, cette commande entraîne un changement de page. Dans la classe `article`, ce n'est pas le cas. Le titre de la partie est précédé du mot « partie » et du numéro de la partie.
`\chapter`	ce découpage commence un nouveau chapitre. Il n'existe pas dans la classe `article`, il entraîne un changement de page dans la classe `book`, mais pas dans la classe `report`. Là encore, LaTeX écrit « chapitre » et le numéro du chapitre, avant d'en écrire le titre.
`\section`	ce découpage, et tous les suivants, existent dans les trois classes principales. Ici, seul le numéro de la section, précédé du numéro de chapitre, est écrit avant le titre de la section.
`\subsection`	encore un peu plus petit...
`\subsubsection`	je crois que c'est clair... Un petit changement cependant : les sous-sous-sections ne sont pas numérotées par défaut. Ça peut être gênant pour y faire référence...
`\paragraph`	Encore plus petit. À tel point que ce n'est plus numéroté, et ça n'est plus mentionné dans la table des matières. Le texte commence sur la même ligne que le titre du paragraphe, ce qui n'était pas le cas des découpages précédents.
`\subparagraph`	le petit dernier... C'est comme un paragraphe, mais c'est, en plus, indenté.

TAB. 3 – Les différents découpages d'un texte

Pour commencer une nouvelle partie, on tapera alors

```
\part{\c{C}a, c'est une partie}
```

Le mieux est de faire un exemple : pour la classe article, et en français[19], on obtient quelquechose qui ressemble au tableau 4.

Ceux qui suivent auront remarqué que les découpages que j'ai mis ici ne sont pas répertoriés dans la table des matières de la page 51. C'est normal pour les trois derniers, mais pas pour les trois premiers. Tout simplement parce que j'ai utilisé la version « étoilée » des commandes de découpage : si on utilise `\part*` à la place

[19]En anglais, « Première partie » aurait été « First part »

`\part{\c{C}a, c'est une partie}`	**Première partie** **Ça, c'est une partie**
`\section{Ici, une section}`	**4 Ici, une section**
`\subsection{Une sous-section}`	**4.1 Une sous-section**
`\subsubsection{Une sous-sous-section...}` `Le texte est plac\'e ici...`	**Une sous-sous-section...** Le texte est placé ici...
`\paragraph{Un paragraphe}` `Le texte ne passe plus \`a la ligne...`	**Un paragraphe** Le texte ne passe plus à la ligne...
`\subparagraph{Et enfin, un sous-paragraphe}` `Ici, c'est indent\'e.`	**Et enfin, un sous-paragraphe** Ici, c'est indenté.

TAB. 4 – Un exemple de découpage du document

de `\part`, la nouvelle partie ne sera pas mise dans la table des matières, et ne sera pas numérotée[20]. C'est assez pratique pour les introductions et conclusions. vphantom`\chapter*\section*\subsection*\subsubsection*`

Signalons enfin la commande `\appendix`. Elle permet, à la fin d'un document, d'insérer des annexes. Elle a pour effet de redéfinir la commande `\section`, qui écrira alors Annexe au début de chaque annexe.

4.2 La mise en page

Ça, c'est un point très important, et il convient de ne pas faire n'importe quoi ! Théoriquement, LaTeX fait la mise en page tout seul. Mais bien entendu, il faut lui indiquer les changements de paragraphes, les sauts de lignes, *etc.* Ce dont s'occupe LaTeX, c'est de faire des lignes de la bonne longueur, de couper les mots en fin de ligne si nécessaire, d'ajouter des espaces là où il en faut, d'en supprimer là où il y en a trop[21], ... Une liste de commandes utiles pour la mise en page est donnée dans le tableau 5.

Par ailleurs, il est possible de modifier le « style » de chaque page, comme à la page 16. Il peut être intéressant de modifier un peu l'emplacement des numéros de page, ou d'ajouter le titre du document sur chaque page. Ces changements sont assez simples à effectuer en utilisant le package `fancyhdr`. Le changement se fait en définissant ce que l'on veut mettre en tête et en pied de page, selon qu'on se trouve sur une page de gauche ou une page de droite. Les commandes correspondantes sont : `\lhead`, `\rhead`, `\chead`, pour respectivement le paramètre de gauche, de droite et du centre, pour la tête de page, et `\lfoot`, `\rfoot` et `\cfoot` pour le pied de page. Enfin, la commande

```
\pagestyle{fancy}
```

permet de changer globalement le style de page, et

```
\thispagestyle{fancy}
```

[20]C'est là qu'il y a une bidouille, parce que j'ai quand même mis des numéros... Mais bon, j'avoue, je les ai mis à la main.

[21]Il est donc inutile d'appuyer 389487 fois sur la barre d'espace, une seule fois suffit. Néanmoins, il peut être crucial de bien indenter le fichier source afin de s'y retrouver facilement, notamment dans les formules mathématiques...

[22]Sauf dans les tableaux, où elle est indispensable. J'en reparlerai à ce moment-là.

[23]Le `\null` est une commande qui n'écrit rien, mais met quelquechose quand même. En effet, le ressort a besoin d'un « appui » pour être comprimé !

`\par`	cette commande permet de changer de paragraphe. Elle provoque un saut de ligne, un petit espace vertical et une indentation du nouveau paragraphe. Il est équivalent et souvent plus clair de passer deux lignes dans le fichier LaTeX.
`\\`	cette commande permet de passer à la ligne, mais sans indenter la ligne suivante. Cette commande doit être évitée autant que possible[22]. Elle peut prendre un argument optionnel (donc entre crochets...) lui indiquant l'espace qu'elle doit laisser entre la ligne précédente et la nouvelle ligne.
`\␣`, `\;`, `\:`, `\,`, `\!`	ce sont des commandes d'espacement. Elle peuvent servir par moments, mais sont à utiliser avec parcimonie... La première force LaTeX à laisser un espace (il y a un espace après le \) comme entre deux mots, la seconde laisse un espace un peu plus petit, la troisieme un peu plus petit, et la quatrième encore plus petit. La dernière crée un espace négatif. Un exemple est montré dans le tableau de l'exemple de la page 14.
`\hspace`, `\vspace`	ce sont des commandes dangereuses aussi : elle permettent de mettre un espace de taille donnée où on veut. Évidemment, ça peut être très laid. La distance se spécifie entre accolades, toujours avec les unités de la page 38. Bien entendu, le `\hspace` donne un espace horizontal, et le `\vspace` un espace vertical.
`\smallskip`, `\medskip`, `\bigskip`	ce sont des espaces verticaux prédéfinis. Ils ne peuvent être utilisés qu'aux changements de paragraphes. Bien entendu, le premier est le plus petit, le dernier le plus grand.
`\hfil`, `\vfil`, `\hfill`, `\vfill`	ce sont ce qu'on appelle des ressorts. Pourquoi ? Parce qu'elles agissent comme des ressorts ! Au lieu de mettre un espace de longueur donnée, celui-là s'adapte en fonction du reste du texte. La version avec deux *l* est infiniment plus forte et plus longue que la version avec un seul *l*. L'intérêt des ressorts, c'est qu'on peut centrer du texte, et même « écarter » du texte. Un exemple[23] :

```
\null\hfill un exemple centré \hfill\null \par
\null\hfil un autre... \hfil\null \par
\null\hfil le \string\hfil\ est faible devant
le \string\hfill \hfill\null \par
du \hfill texte \hfill coupé \hfill
en \hfill morceaux \par
```

un exemple centré

un autre...

le \hfil est faible devant le \hfill

du texte coupé en morceaux

`\indent`, `\noindent`	ce sont les commandes qui permettent d'indenter et de ne pas indenter une ligne. Bien entendu, elles sont faites pour être en début de ligne... Elles ne doivent pas servir trop souvent, car en théorie, le package `babel` s'en occupe.
`~`	c'est un espace que l'on qualifie d'« insécable », *i.e.* LaTeX ne peut pas changer de ligne à cet endroit. À quoi sert-ce ? Simplement à ne pas passer à la ligne n'importe quand. Par exemple, quand on fait référence à la page 12, il faut que le 12 soit sur la même ligne que le mot « page », sinon ça va pas être beau. On tape donc `page~$12$`, en attendant de savoir faire des références « croisées ».

Tab. 5 – Quelques commandes utiles pour régler l'espacement permet de ne le changer que pour la page courante. Il peut être utile de savoir enfin que la commande `\thepage` permet d'obtenir le numéro de page, la commande `\thechapter`, le numéro de chapitre[24], `\thesection` pour les sections, et ainsi de suite. Par ailleurs, trois styles sont prédéfinis : le style `plain`, le style `headings` et le style `empty`. Le style `plain` est le style par défaut : il affiche simplement le numéro de page en bas. Le style `headings` écrit le numéro de page en haut, en face du nom du chapitre ou de la section. Enfin, le style `empty` permet de laisser l'en-tête et le pied de page vide.

Toujours pour la mise en page, LaTeX fournit quelques environnements pratiques pour les énumérations, les tableaux... En voici quelques-uns :

Les environnements de listes Ils permettent d'énumérer une liste. On peut choisir le type d'« entrée » pour la liste : soit chaque entrée est précédée par un tiret, soit par un numéro, soit par un mot. Chaque entrée doit être précédée de la commande `\item`. Voici un exemple, qui devrait être assez clair.

```
\begin{description}
\item[itemize] : l'environnement
  \texttt{itemize}
  permet de faire des listes simples, comme
  \begin{itemize}
  \item première entrée,
  \item deuxième entrée, ...
  \end{itemize}
\item[enumerate] : l'environnement
  \texttt{enumerate}
  permet de numéroter les entrées, par exemple~:
  \begin{enumerate}
  \item première entrée,
  \item deuxième entrée, ...
  \end{enumerate}
\item[description] : l'environnement
  \texttt{description} permet
  de mettre un mot devant
  chaque entrée. Par exemple~:
  \begin{description}
  \item[la première] : première entrée,
  \item[la deuxième] : deuxième entrée, ...
  \end{description}
\end{description}
```

itemize : l'environnement `itemize` permet de faire des listes simples, comme
- première entrée,
- deuxième entrée, ...

enumerate : l'environnement `enumerate` permet de numéroter les entrées, par exemple :
1. première entrée,
2. deuxième entrée, ...

description : l'environnement `description` permet de mettre un mot devant chaque entrée. Par exemple :
la première : première entrée,
la deuxième : deuxième entrée, ...

Les environnements de tableaux Le principal environnement permettant de faire des tableaux est l'environnement . Là encore, un exemple devrait suffire. La seule chose qui demande explication est l'argument de l'environnement : il y a en fait deux arguments, mais le premier est optionnel. Il indique l'alignement vertical du tableau par rapport au texte qui se trouve à droite et à gauche. Ça peut être *t* pour *top*, *c* pour *center* ou encore *b* pour *bottom*. Cet argument se met entre crochets. Le second argument, qui se met entre accolades, est là pour indiquer le nombre de colonnes, ainsi que la position du texte dans ces colonnes (par les lettres *c* pour *center*, *l* pour *left* et *r* pour *right*, ou encore, pour donner à une colonne une largeur définie, *p*, suivi de la largeur de la colonne).

Il est également possible d'insérer du texte de façon systématique entre deux colonnes. Pour cela, on utilise les `@-expressions` : il suffit de mettre `@{le texte qu'on veut}` pour que ce texte soit mis à la position demandée sur chaque ligne. Ces indicateurs de colonnes peuvent être séparées par un | qui signifie alors la présence d'un trait vertical de séparation entre deux colonnes. Enfin, si on veut cinquante fois la même

[24]Sauf bien sûr dans le cas d'un article

colonne, on utilisera une `*-expression`, comme par exemple `|*{50}{l|}`, qui permet d'avoir un tableau avec 50 colonnes séparées par un trait vertical. Le package `array` offre encore plus de possibilités, consulter l'annexe E pour plus de détails.

On peut, bien entendu, rassembler plusieurs colonnes en une quand on le veut. Cela se fait avec la commande `\multicolumn`. Il faut préciser, en argument, le nombre de colonnes, la position du texte dans cette « colonne multiple », avec *l*, *c* ou *r*, et si on veut des lignes de séparation sur les côtés. Pour les lignes horizontales, c'est la commande `\hline`. Là encore, il peut être utile de savoir ne faire qu'une ligne partielle, ce que ne permet pas la commande `\hline`. La commande à utiliser est alors `\cline`, en lui passant comme argument la colonne de départ et la colonne d'arrivée[25]. Enfin, pour séparer le contenu de deux colonnes, on utilise le caractères &, et pour séparer deux lignes, le retour à la ligne \\ .

```
Le tableau d'exemples pour les espaces~: \\
\begin{tabular}[t]{|p{3cm}||c||r|}
\cline{1-2}
\multicolumn{2}{|c||}{exemple}
             &\multicolumn{1}{c}{} \\
\hline
\texttt{\string\,} & petit & aaa\,aaa \\
\hline
\texttt{\string\:} & un peu plus grand
                                 & aaa\:aaa \\
\hline
\texttt{\string\;} & encore plus & aaa\ aaa\\
\hline
\texttt{\string\ } & espace normal & aaa\ aaa\\
\hline
\texttt{\string\!} & espace négatif & aaa\!aaa\\
\hline
\end{tabular}
```

Le tableau d'exemples pour les espaces :

exemple		
`\,`	petit	aaa aaa
`\:`	un peu plus grand	aaa aaa
`\;`	encore plus	aaa aaa
`\`	espace normal	aaa aaa
`\!`	espace négatif	aaaaaa

Mais cela peut poser des problèmes : en effet, LaTeX ne voudra jamais couper un tableau, et donc laissera un blanc, afin de mettre le tableau à la page suivante. Il existe alors deux solutions : soit définir le tableau comme un flottant[26], soit définir un tableau long. Mais cette deuxième solution ne doit être appliquée que dans le cas où le tableau est effectivement long. Pour ce faire, il faut inclure le package `longtable`. Mais comme le tableau est coupé, il faut pouvoir définir une « en-tête » pour le tableau, ainsi qu'une ligne en bas de page pour indiquer la coupure du tableau. Je pense que le mieux à faire est de donner un exemple :

```
\begin{longtable}[t]{|p{2cm}|p{3cm}|p{2cm}|}
\hline premi\`ere & deuxi\`eme & troisi\`eme          \endfirsthead
\hline premi\`ere & deuxi\`eme & troisi\`eme \\
  \hfill(suite) & \hfill(suite) & \hfill(suite) \endhead
\multicolumn{3}{|c|}{La suite \`a la page suivante...} \\
\hline                                          \endfoot
                                                \endlastfoot
\hline
la & premi\`ere & ligne \\
\hline
la & deuxi\`eme & ligne \\
\hline
[...]
```

[25]Voir également le package `hhline` pour des lignes de séparation un peu plus évoluées.
[26]Les flottants, j'en parle à la page 34

```
la & vingti\`eme & ligne \\
\hline
\end{longtable}
```

permet d'obtenir[27] :

première	deuxième	troisième
la	première	ligne
la	deuxième	ligne
la	troisième	ligne
la	quatrième	ligne
la	cinquième	ligne
la	sixième	ligne
la	septième	ligne
la	huitième	ligne
la	neuvième	ligne
la	dixième	ligne
la	onzième	ligne
la	douzième	ligne
la	treizième	ligne
la	quatorzième	ligne
la	quinzième	ligne
la	seizième	ligne
la	dix-septième	ligne
la	dix-huitième	ligne
la	dix-neuvième	ligne
la	vingtième	ligne

Il est possible d'interdire la coupure du tableau à certains endroits, en utilisant `\\*` au lieu de `\\`. Cependant, un tableau peut toujours être coupé sur un `\hline`.

4.3 Les changements de polices de caractères

Cette partie est assez importante. Les changements de polices de caractères permettent de mettre certaines parties du texte en évidence ou en retrait[28].

Commençons par les changements de taille. Ces commandes ne prennent pas d'argument, et modifient tout ce qui se trouve après. Le tableau 7 donne ces commandes, avec des exemples.

C'est là que surgit l'intérêt des groupes. Comme je le disais à la page 3, un groupe est une partie d'un fichier LaTeX entre deux accolades qui se correspondent[30]. Et à l'intérieur d'un groupe, toutes les commandes sont locales. C'est à dire qu'à la fin du groupe, leur effet est arrêté. Voici quelques exemples qui devraient suffire :

```
\LARGE bonjour \par
au revoir
```

bonjour

au revoir

[27] Je ne sais pas, pour le moment, si le tableau sera coupé ou pas... Cela dépend de ce que je vais ajouter avant ce tableau. Mais quoi que je change avant, je n'ai pas à modifier le tableau, LaTeX le coupera au bon endroit si nécessaire... Et puis si tu veux voir un `longtable` coupé, regarde l'annexe E.

[28] Note importante : on ne souligne pas en LaTeX. Souligner, c'est ce qu'on faisait avec les machines à écrire pour mettre du texte en évidence[29]. LaTeX permet de faire bien mieux que ça... Mais bon, pour ceux qui insistent, allez voir le package `ulem`.

[29] De la même façon, ce sont les machines à écrire qui ont nécessité l'absence d'accentuation sur les majuscules. Cette limitation n'existant plus, il *faut* maintenant accentuer les majuscules.

[30] *i.e.* un truc du genre {{{bonjour} n'est pas un groupe, mais {au {{ revoir}}} est un groupe.

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

TAB. 7 – Les commandes de changement de taille

```
{\footnotesize bonjour\par}
au revoir
```

bonjour
au revoir

```
\Large{\Huge bonjour}\par
au revoir\bigskip

\normalsize
bonj\large o\Large o\LARGE o\huge
o\Large o\large o\normalsize our
```

bonjour
au revoir

bonjoooOoooour

Les commandes de changement de police de caractères sont plus appropriées pour mettre du texte en valeur. Elles sont de deux types : celles qui prennent un argument, et ne modifient que celui-ci, et celles qui, comme les commandes de changement de taille, agissent sur tout le reste du document. Elles sont expliquées dans le tableau 8. Les commandes de la colonne de gauche s'appliquent uniquement à leur argument, celles de la colonne du centre s'appliquent à tout le texte qui suit.

`\textnormal`	`\normalfont`	Texte
`\textrm`	`\rmfamily`	Texte
`\textsf`	`\sffamily`	Texte
`\texttt`	`\ttfamily`	`Texte`
`\textup`	`\upshape`	Texte
`\textsl`	`\slshape`	*Texte*
`\textit`	`\itshape`	*Texte*
`\textsc`	`\scshape`	TEXTE
`\textmd`	`\mdseries`	Texte
`\textbf`	`\bfseries`	**Texte**
`\emph`	`\em`	*Texte*

TAB. 8 – Les commandes de changement de police

Noter le comportement particulier de la commande `\emph` : si le texte est initialement droit, elle passe en mode italique, et vice versa. Ce qui en fait une commande très utile pour mettre du texte en valeur : on n'a pas besoin de se préocupper de ce qui se passe avant. Par exemple :

```
on passe du droit \emph{au penché} et \\
\itshape du penché \emph{au droit} facilement.
```

on passe du droit *au penché* et
du penché au droit *facilement.*

Enfin, LaTeX permet d'utiliser de nombreuses fontes différentes. La gestion des fontes par TeXétait assez mal pensée, elle a été complètement refaite pour LaTeX. Le système utilisé maintenant s'appelle NFSS, pour *New Font Selection Scheme*, la version actuelle étant la 2. Il est détaillé par exemple dans [GMS98, chapitre 7] ou [Zoo99].

Je ne vais pas entrer dans les détails, mais une fonte peut être définie par plusieurs caract´eristiques plus ou moins indépendantes :

- le codage qu'elle utilise. Par défaut, LaTeX utilise le codage de fontes `OT1`, mais on l'a déjà vu, il est préférable d'utiliser le codage `T1`, afin de pouvoir écrire certains caractères particuliers ;
- la famille à laquelle elle appartient (comme *Times*, *Garamond*, ...) ;
- la « série », c'est à dire l'épaisseur (`bold`, `medium`, ...) ;
- la « forme », par exemple italique ou en petites capitales ;
- la taille de la fonte, et la taille de l'interligne.

A priori, ces différents critères sont indépendants, on peut donc les spécifier indépendamment. En pratique, il arrivera surement que telle ou telle fonte est manquante.

Pour sélectionner une nouvelle fonte, on écrira donc, par exemple :

```
\fontencoding{T1}
\fontfamily{cmr}
\fontseries{b}
\fontshape{sc}
\fontsize{20.74pt}{24.88pt}
\selectfont
```

pour passer dans une fonte en « small caps », en gras, issue de la famille « Computer Modern », et à une taille de 20.74 points, avec un interligne de 24.88 points[31]. Voila le résultat :

Ça ne donne pas des « small caps » parce que la fonte que j'ai demandée n'existe pas. LaTeX a donc modifié la « forme » de la fonte, et a remplacé « small caps » par « normal ».

Voyons quand même un exemple qui marche, pour montrer que ça sert quand même à quelquechose. Par la même occasion, j'introduis la commande `\usefont`, qui est un condensé des précédentes, mais qui ne permet pas de définir la taille de la nouvelle fonte :

```
\usefont{T1}{ptm}{b}{it}
```

qui donne ça :

J'utilise ici la fonte Times d'Adobe (son petit nom, c'est* `ptm`*), en gras italique. Ça donne ça, pas forcément très beau, mais bon...

[31]Ce sont des valeurs « usuelles », correspondant à la taille obtenue par la commande `\huge`. Certaines fontes peuvent être « agrandies » exactement à la taille voulue, d'autres n'existent que dans certaines tailles.

Bien sûr, les commandes de changement de taille fonctionnent encore dans ce cas, et on peut mettre un `\Large` après le `\usefont` pour augmenter le taille de la fonte choisie.

Il existe quelques packages permettant de changer « automatiquement » de fonte dans tout le document. Par exemple, `palatino`, `avant` ou `times` permettent de changer, de manière propre, la fonte dans tout le document.

Il ne faut pas s'étonner des éventuels messages affichés par xdvi et dvips lorsqu'on utilise des fontes « inhabituelles » : en effet, ces fontes doivent être « compilées » avant de pouvoir être utilisées. C'est le programme METAFONT qui s'en charge[32]. Par ailleurs, la « liste » des fontes disponibles se trouve dans le fichier `fonts` du répertoire d'installation de LaTeX.

Bon, je ne détaille pas plus, parce que c'est à peu près tout ce que je sais et que je risquerais de dire des bêtises. Si vraiment tu veux en savoir plus, faudra aller lire les docs idoines.

4.4 Les caractères particuliers

Voyons maintenant comment on obtient les caractères particuliers, commes les caractères accentués, ou une cédille. Deux solutions sont possibles : soit on utilise les commandes d'accentuation de LaTeX, soit on tape directement l'accent. Toutefois, l'accent tapé directement, ça va pas marcher comme il faut si tu passes d'une machine unix à une machine Mac ou Windows. Il vaut donc mieux utiliser les commandes d'accentuation de LaTeX, qui sont décrites dans le tableau 9.

`\'`	á, é, í, ó, ú, ý
`` \` ``	à, è, ì, ò, ù, ỳ
`\^`	â, ê, î, ĵ, ȷ̂[33], ô, û, ŷ
`\~`	ñ, m̃
`\.`	ȧ, ė, Ċ...
`\=`	ā, ē, C̄...
`\d`	ạ, ẹ, C̣...
`\c`	ç, Ç, et même P̧ !!
`\k`	c̨, C̨, et toujours P̨...

TAB. 9 – Les commandes d'accentuation

N'oublions pas les œ et autres æ, qui ont leurs équivalents Œ et Æ en majuscule, et qui s'obtiennent simplement par les commandes `\oe`, `\ae`, `\OE` et `\AE`.

En français, il existe différents tirets. Le tiret simple, entre deux mots, qu'on appelle souvent « trait d'union », s'obtient avec un tiret simple, `-`. Mais un tiret peut aussi signaler le changement de locuteur dans un dialogue. Il est alors un peu plus long, et s'obtient avec `--`. Enfin, le tiret utilisé comme une paranthèse — comme ici —, est encore plus long, et s'obtient logiquement par la commande `---`. Pour éviter un saut de ligne juste après un — , on l'entoure par des `~`. On mettra donc `---~par exemple~---`.

Par ailleurs, le package `babel`, avec l'option `francais`, permet d'obtenir des guillemets « à la française ». En français, ce sont les seuls reconnus. Les " et ' sont à exclure... Pour obtenir le « et le », il suffit de taper `<<` et `>>`, sans espace, respectivement... Il existe également des commandes pour faire ça, qui s'appellenet `\og` et `\fg`.

Il reste enfin les caractères protégés, comme %, qui est le symbole reservé aux commentaires. Pour obtenir ces caractères, une règle générale est d'utiliser la commande `\string`. Cependant, on peut aussi plus simplement taper `\%`, par exemple, pour le caractère %. Les cas où `\string` ne marche pas, c'est pour obtenir un backslash, un `%` ou un `#` : en effet, `\string` écrit le caractère qui la suit sans l'interpréter, sauf si ce qui la suit, c'est un nom de commande, auquel cas `\string` écrit le nom de cette commande précédé du backslash. Ainsi, `\string\` produit un backslash suivi d'une espace. Et on ne peut pas définir backslash de la façon suivante :

[32] METAFONT a été écrit par KNUTH également...

```
\newcommand\bs{\string\}
```

parce que, dans ce cas, l'argument de `\string` est la commande \}. Dans le cas de `%` et de `#`, c'est parce qu'ils sont interprétés avant la commande `\string`.

4.5 Quelques commandes spéciales

Voyons maintenant quelques commandes particulièrement puissantes de LaTeX.

- Les « footnotes », *i.e.* les notes de bas de page, tout d'abord. J'en ai déjà utilisé pas mal ici : ce sont les petits numéros qui renvoient à une remarque dans le bas de la page[34]. Pour obtenir une note de pied de page, il suffit de taper, à l'endroit où on veut la marque (le numéro), la commande `\footnote`, avec en argument le texte de la note. Cette commande ne fonctionne pas à l'intérieur d'un tableau, ni dans la définition d'une partie, d'un chapitre, d'une section[35]...
- les références croisées sont aussi une grande force de LaTeX. Lorsque je fais référence au tableau 1 de la page 5, je ne mets pas le numéro du tableau, puisque *a priori* , je ne le connais pas, et il peut changer si j'insère un autre tableau avant. LaTeX fournit pour cela trois commandes très intéressantes : `\label`, qui prend en argument un nom quelconque, qui fera toujours référence à ce label, `\ref`, qui prend en argument le nom d'un label, et renvoie le numéro « courant » : ça peut être le numéro du chapitre, du tableau ou de la figure suivant le contexte. Enfin, `\pageref` renvoie le numéro de la page correspondante à son argument. Bien entendu, comme on peut faire référence à un truc qui se trouve dix pages plus loin, et comme la compilation est linéaire, il est nécessaire de compiler deux fois.
  Un petit exemple :

```
La note de pied de page numéro \ref{pied} se
trouve à la page \pageref{pied}.
```

La note de pied de page numéro 18 se trouve à la page 8.

  LaTeX se charge également de la table des matières, de la liste des figures ou de la liste des tableaux. Il suffit de le lui demander en anglais : `\tableofcontents`, `\listoffigures` et `\listoftables`. Il est nécessaire de compiler trois fois pour que la table des matières soit à jour[36] ;
- parlons également de l'index : tout le long du document, on indique les mots que l'on désire retrouver dans l'index, grace à la commande `\index`. Il faut ensuite indiquer, dans le préambule, que l'on veut un index, par la commande `\makeindex`, et indiquer l'endroit où on veut écrire l'index, par la commande `\printindex`. Lorsqu'il rencontre la commande `\makeindex`, LaTeX collecte tous les mots de l'index dans un fichier d'extension `.idx`. Cependant, ces mots ne sont pas ordonnés. Pour les ordonner, il faut compiler l'index, avec la commande[37]

```
makeindex toto
```

  l est donc nécessaire de compiler plusieurs fois le fichier LaTeX pour obtenir l'index complet. Pour ceux que ça intéresse, l'annexe B.1 est en partie consacrée à l'index ;
- enfin, LaTeX permet de faire une bibliographie. Il existe deux solutions : l'une, simple mais peu puissante, est de la faire dans le fichier LaTeX. L'autre, beaucoup plus compliquée mais plus puissante, consiste à faire un fichier de bibliographie à part, le compiler avec BibTeX, et l'inclure dans le fichier LaTeX. Pour un peu plus de détails sur cette deuxième solution, c'est dans l'annexe B.2. Pour la première solution, c'est un peu comme les références croisées, on associe un label à une entrée de la bibliographie, à l'aide de la commande `\bibitem`. Comme c'est une liste de références bibliographiques, il faut utiliser un environnement de listes, un peu adapté, qui s'appelle `thebibliography`. Pour faire référence à une entrée de la bibliographie, on utilise la commande `\cite`. Un exemple suffira :

---

[34] comme ici...
[35] Voir l'annexe « Trucs et Astuces » à ce sujet.
[36] La première fois pour que LaTeX mémorise toutes les entrées, la deuxième pour qu'il « mesure » la taille de la table des matières et qu'il lui prévoie une place, et la troisième pour qu'il la mette effectivement...
[37] Attention, ce n'est pas une commande LaTeX, mais un programme... On peut lui passer l'option `-s` suivi du nom d'un fichier de style d'index (d'extension `.ist`) pour modifier la mise en page de l'index.

```
Pour des renseignements sur \textsc{Bib}\TeX,
je te conseille de lire \cite{bib}.
Si vraiment tu veux plus de d\'etails, tu
peux jeter un \oe il \`a~\cite{bib2},

\begin{thebibliography}{9}
\bibitem{bib} Leslie \textsc{Lamport}, \textit
        {\LaTeX, a Document Preparation System},
        annexe~B, p.~155--164.
\bibitem{bib2} Nicolas \textsc{Markey}, \textit
        {Tame the BeaST}, \textsc{Bib}\TeX{} de B
        \`a X, 2002.
\end{thebibliography}
```

Pour des renseignements sur BIBTEX, je te conseille de lire [1]. Si vraiment tu veux plus de détails, tu peux jeter un œil à [2],

Références

[1] Leslie LAMPORT, *LATEX, a Document Preparation System*, annexe B, p. 155–164.

[2] Nicolas MARKEY, *Tame the BeaST*, BIBTEX de B à X, 2002.

5 Les mathématiques

C'est le gros morceau. C'est ça qui fait la force de LATEX. Mais la qualité du résultat a une contrepartie : c'est pas évident à manipuler...

5.1 Les environnements

Il existe deux environnements : l'un permet de placer les formules mathématiques au milieu d'une ligne de texte, comme par exemple $A \subset B$, l'autre permet de le mettre en évidence, comme par exemple

$$\forall \varepsilon > 0, \exists \eta > 0 : \forall x \in \mathbb{R}, |x| < \eta \Rightarrow |f(x)| < \varepsilon.$$

Tout objet mathématique doit se trouver à l'intérieur d'un environnement mathématique : en effet, il y a une différence entre un m en mode texte et un m en mode mathématique. Pour de nombreux symboles mathématiques, LATEX va exiger d'être en mode mathématique, mais pour les lettre, par exemple, il ne peut pas faire la distinction.

Le premier environnement sera plutôt réservé pour les petites formules mathématiques, le deuxième pour les grosses formules.

Pour les obtenir : le premier environnement se nomme `math`, et bien entendu, on l'obtient avec les commandes `\begin{math}` et `\end{math}`. Le deuxième environnement, c'est `displaymath`, on y entre avec `\begin{displaymath}` et on en sort avec `\end{displaymath}`. Mais comme ces environnements servent souvent, il existe des raccourcis : pour l'environnement `math`, les commandes `\(` et `\)` permettent d'y entrer et d'en sortir respectivement, et pour `displaymath`, c'est `\[` et `\]`. Il existe enfin un troisième moyen d'appeler le mode mathématique : l'environnement `math` peut être appelé par la « commande » `$`[38], et ceci que ce soit pour entrer ou sortir de l'environnement. Il vaut mieux n'utiliser `$` que pour des *petites* expressions mathématiques isolées, parce que sinon, on finit par ne plus s'y retrouver.

5.2 Les packages

Les environnements mathématiques sont accessibles sans l'intermédiaire de packages. Cependant, dès qu'on veut écrire une formule mathématique légèrement compliquée, il est nécessaire d'inclure certains packages. Le tableau 10 donne les principaux packages, avec leurs caractéristiques principales. La sous-section 5.3 donne une liste des principaux symboles mathématiques, ainsi que les packages qu'il faut inclure pour les obtenir.

[38] Et `displaymath` par `$$`, mais ce n'est pas recommandé. Il vaut mieux utiliser `\[` et `\]`.

`amsmath`	le package le plus général. Il est souvent suffisant, car il inclus automatiquement plusieurs packages, à savoir `amstext`, `amsbsy` et `amsopn` ;
`amstext`	permet, grâce à la commande `\text` d'inclure du texte dans une formule mathématique[39] ;
`amsbsy`	permet d'écrire en gras des caracteres mathématiques ;
`amsopn`	permet de définir des opérateurs ;
`amscd`	pour les diagrammes commutatifs ;
`amsfonts`	définit quelques fontes supplémentaires pour le mode mathématique ;
`amssymb`	tous les symboles mathématiques définis par l'$\mathcal{AMS}$;
`latexsym`, `stmaryrd`	quelques autres symboles.

Tab. 10 – Les packages mathématiques

5.3 Les symboles mathématiques

Les tableaux 11 à 30 donnent une liste non-exhaustive des différents symboles mathématiques fournis par les différents packages cités précédemment, ainsi que la façon de les obtenir.

α	`\alpha`	θ	`\theta`	o	`o`	τ	`\tau`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	υ	`\upsilon`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	ϕ	`\phi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	φ	`\varphi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	χ	`\chi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ψ	`\psi`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`	ω	`\omega`
η	`\eta`	ξ	`\xi`				
Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

Tab. 11 – Les lettres grecques

$\pm$	`\pm`	$\cap$	`\cap`	$\diamond$	`\diamond`	$\oplus$	`\oplus`
$\mp$	`\mp`	$\cup$	`\cup`	$\bigtriangleup$	`\bigtriangleup`	$\ominus$	`\ominus`
$\times$	`\times`	$\uplus$	`\uplus`	$\bigtriangledown$	`\bigtriangledown`	$\otimes$	`\otimes`
$\div$	`\div`	$\sqcap$	`\sqcap`	$\triangleleft$	`\triangleleft`	$\oslash$	`\oslash`
$\ast$	`\ast`	$\sqcup$	`\sqcup`	$\triangleright$	`\triangleright`	$\odot$	`\odot`
$\star$	`\star`	$\vee$	`\vee`	$\lhd$	`\lhd`[40]	$\bigcirc$	`\bigcirc`
$\circ$	`\circ`	$\wedge$	`\wedge`	$\rhd$	`\rhd`[40]	$\dagger$	`\dagger`
$\bullet$	`\bullet`	$\setminus$	`\setminus`	$\unlhd$	`\unlhd`[40]	$\ddagger$	`\ddagger`
$\cdot$	`\cdot`	$\wr$	`\wr`	$\unrhd$	`\unrhd`[40]	$\amalg$	`\amalg`
$+$	`+`	$-$	`-`				

Tab. 12 – Les opérateurs binaires

$\leq$	`\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`	$\Join$	`\Join`[40]
$\sqsubset$	`\sqsubset`[40]	$\sqsupset$	`\sqsupset`[40]	$\neq$	`\neq`	$\smile$	`\smile`
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\doteq$	`\doteq`	$\frown$	`\frown`
$\in$	`\in`	$\ni$	`\ni`	$\propto$	`\propto`	$=$	`=`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$<$	`<`	$>$	`>`

Tab. 13 – Les symboles de relation

$,$	`,`	$;$	`;`	$\colon$	`\colon`	$\ldotp$	`\ldotp`	$\cdotp$	`\cdotp`

Tab. 14 – Les symboles de ponctuation

$\leftarrow$	`\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`
$\rightleftharpoons$	`\rightleftharpoons`	$\leadsto$	`\leadsto`[40]		

Tab. 15 – Les flèches

$\ldots$	`\ldots`	$\cdots$	`\cdots`	$\vdots$	`\vdots`	$\ddots$	`\ddots`
$\aleph$	`\aleph`	$\prime$	`\prime`	$\forall$	`\forall`	∞	`\infty`
$\hbar$	`\hbar`	$\emptyset$	`\emptyset`	$\exists$	`\exists`	$\Box$	`\Box`[40]
$\imath$	`\imath`	∇	`\nabla`	$\neg$	`\neg`	$\Diamond$	`\Diamond`[40]
$\jmath$	`\jmath`	$\surd$	`\surd`	$\flat$	`\flat`	$\triangle$	`\triangle`
ℓ	`\ell`	$\top$	`\top`	$\natural$	`\natural`	$\clubsuit$	`\clubsuit`
$\wp$	`\wp`	$\bot$	`\bot`	$\sharp$	`\sharp`	$\diamondsuit$	`\diamondsuit`
$\Re$	`\Re`	$\Vert$	`|`	$\backslash$	`\backslash`	$\heartsuit$	`\heartsuit`
$\Im$	`\Im`	$\angle$	`\angle`	∂	`\partial`	$\spadesuit$	`\spadesuit`
$\mho$	`\mho`[40]	$.$	`.`	$\vert$	`|`		

Tab. 16 – Quelques symboles supplémentaires

[40] Nécessitent le package `amssymb` ou `latexsym`

$\sum$	`\sum`	$\bigcap$	`\bigcap`	$\bigodot$	`\bigodot`
$\prod$	`\prod`	$\bigcup$	`\bigcup`	$\bigotimes$	`\bigotimes`
$\coprod$	`\coprod`	$\bigsqcup$	`\bigsqcup`	$\bigoplus$	`\bigoplus`
$\int$	`\int`	$\bigvee$	`\bigvee`	$\biguplus$	`\biguplus`
$\oint$	`\oint`	$\bigwedge$	`\bigwedge`		

Tab. 17 – Les grands opérateurs

`\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. 18 – Les symboles de fonctions

<table>
<tr><td>$($</td><td><code>(</code></td><td>$)$</td><td><code>)</code></td><td>$\uparrow$</td><td><code>\uparrow</code></td><td>$\Uparrow$</td><td><code>\Uparrow</code></td></tr>
<tr><td>$[$</td><td><code>[</code></td><td>$]$</td><td><code>]</code></td><td>$\downarrow$</td><td><code>\downarrow</code></td><td>$\Downarrow$</td><td><code>\Downarrow</code></td></tr>
<tr><td>$\{$</td><td><code>\{</code></td><td>$\}$</td><td><code>\}</code></td><td>$\updownarrow$</td><td><code>\updownarrow</code></td><td>$\Updownarrow$</td><td><code>\Updownarrow</code></td></tr>
<tr><td>$\lfloor$</td><td><code>\lfloor</code></td><td>$\rfloor$</td><td><code>\rfloor</code></td><td>$\lceil$</td><td><code>\lceil</code></td><td>$\rceil$</td><td><code>\rceil</code></td></tr>
<tr><td>$\langle$</td><td><code>\langle</code></td><td>$\rangle$</td><td><code>\rangle</code></td><td>$/$</td><td><code>/</code></td><td>$\backslash$</td><td><code>\backslash</code></td></tr>
<tr><td>$|$</td><td><code>|</code></td><td>$\|$</td><td><code>\|</code></td><td></td><td></td><td></td><td></td></tr>
<tr><td>$\rmoustache$</td><td><code>\rmoustache</code></td><td>$\lmoustache$</td><td><code>\lmoustache</code></td><td>$\rgroup$</td><td><code>\rgroup</code></td><td>$\lgroup$</td><td><code>\lgroup</code></td></tr>
<tr><td>$\arrowvert$</td><td><code>\arrowvert</code></td><td>$\Arrowvert$</td><td><code>\Arrowvert</code></td><td>$\bracevert$</td><td><code>\bracevert</code></td><td></td><td></td></tr>
</table>

Tab. 19 – Les délimiteurs

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

Tab. 20 – Les caractères accentués

$\dashrightarrow$	`\dashrightarrow`	$\dashleftarrow$	`\dashleftarrow`	$\rightsquigarrow$	`\rightsquigarrow`	$\leftrightsquigarrow$	`\leftrightsquigarrow`
$\twoheadrightarrow$	`\twoheadrightarrow`	$\twoheadleftarrow$	`\twoheadleftarrow`	$\leftarrowtail$	`\leftarrowtail`	$\rightarrowtail$	`\rightarrowtail`
$\leftrightharpoons$	`\leftrightharpoons`	$\rightleftharpoons$	`\rightleftharpoons`	$\circlearrowleft$	`\circlearrowleft`	$\circlearrowright$	`\circlearrowright`
$\upuparrows$	`\upuparrows`	$\downdownarrows$	`\downdownarrows`	$\curvearrowleft$	`\curvearrowleft`	$\curvearrowright$	`\curvearrowright`
$\leftrightarrows$	`\leftrightarrows`	$\rightleftarrows$	`\rightleftarrows`	$\leftleftarrows$	`\leftleftarrows`	$\rightrightarrows$	`\rightrightarrows`
$\multimap$	`\multimap`	$\Lleftarrow$	`\Lleftarrow`	$\looparrowleft$	`\looparrowleft`	$\looparrowright$	`\looparrowright`
$\downharpoonleft$	`\downharpoonleft`	$\downharpoonright$	`\downharpoonright`	$\Lsh$	`\Lsh`	$\Rsh$	`\Rsh`
$\upharpoonleft$	`\upharpoonleft`	$\upharpoonright$	`\upharpoonright`				

Tab. 21 – Les flèches de `amssymb`

$\hbar$	`\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`				

Tab. 22 – Divers symboles fournis par `amssymb`

$\dotplus$	`\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`	$\centerdot$	`\centerdot`	$\intercal$	`\intercal`		

TAB. 23 – Les opérateurs binaires de amssymb

$\leqq$	`\leqq`	$\leqslant$	`\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`				

TAB. 24 – Les relations binaires de amssymb

$\nless$	`\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`
$\nsucceq$	`\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. 25 – Les relations binaires négatives de amssymb

$\Lbag$	`\Lbag`	$\Rbag$	`\Rbag`	$\lbag$	`\lbag`	$\rbag$	`\rbag`
$\llceil$	`\llceil`	$\rrceil$	`\rrceil`	$\llfloor$	`\llfloor`	$\rrfloor$	`\rrfloor`
$\llbracket$	`\llbracket`	$\rrbracket$	`\rrbracket`				

TAB. 26 – Les délimiteurs de stmaryrd

$\Longmapsfrom$	\Longmapsfrom	$\Longmapsto$	\Longmapsto	$\Mapsfrom$	\Mapsfrom	$\Mapsto$	\Mapsto
$\nnearrow$	\nnearrow	$\nnwarrow$	\nnwarrow	$\ssearrow$	\ssearrow	$\sswarrow$	\sswarrow
$\shortdownarrow$	\shortdownarrow	$\shortuparrow$	\shortuparrow	$\shortleftarrow$	\shortleftarrow	$\shortrightarrow$	\shortrightarrow
$\longmapsfrom$	\longmapsfrom	$\mapsfrom$	\mapsfrom	$\leftarrowtriangle$	\leftarrowtriangle	$\rightarrowtriangle$	\rightarrowtriangle
$\lightning$	\lightning	$\rrparenthesis$	\rrparenthesis	$\leftrightarroweq$	\leftrightarroweq	$\leftrightarrowtriangle$	\leftrightarrowtriangle

TAB. 27 – Les flèches de stmaryrd

$\Ydown$	\Ydown	$\Yleft$	\Yleft	$\Yright$	\Yright	$\Yup$	\Yup
$\baro$	\baro	$\bbslash$	\bbslash	$\binampersand$	\binampersand	$\bindnasrepma$	\bindnasrepma
$\boxast$	\boxast	$\boxbar$	\boxbar	$\boxbox$	\boxbox	$\boxbslash$	\boxbslash
$\boxcircle$	\boxcircle	$\boxdot$	\boxdot	$\boxempty$	\boxempty	$\boxslash$	\boxslash
$\curlyveedownarrow$	\curlyveedownarrow	$\curlyveeuparrow$	\curlyveeuparrow	$\curlywedgedownarrow$	\curlywedgedownarrow	$\curlywedgeuparrow$	\curlywedgeuparrow
$\fatbslash$	\fatbslash	$\fatsemi$	\fatsemi	$\fatslash$	\fatslash	$\interleave$	\interleave
$\leftslice$	\leftslice	$\merge$	\merge	$\minuso$	\minuso	$\moo$	\moo
$\nplus$	\nplus	$\obar$	\obar	$\oblong$	\oblong	$\obslash$	\obslash
$\ogreaterthan$	\ogreaterthan	$\olessthan$	\olessthan	$\ovee$	\ovee	$\owedge$	\owedge
$\rightslice$	\rightslice	$\sslash$	\sslash	$\talloblong$	\talloblong	$\varbigcirc$	\varbigcirc
$\varcurlyvee$	\varcurlyvee	$\varcurlywedge$	\varcurlywedge	$\varoast$	\varoast	$\varobar$	\varobar
$\varobslash$	\varobslash	$\varocircle$	\varocircle	$\varodot$	\varodot	$\varogreaterthan$	\varogreaterthan
$\varolessthan$	\varolessthan	$\varominus$	\varominus	$\varoplus$	\varoplus	$\varoslash$	\varoslash
$\varotimes$	\varotimes	$\varovee$	\varovee	$\varowedge$	\varowedge	$\vartimes$	\vartimes

TAB. 28 – Les opérateurs binaires de stmaryrd

$\bigbox$	\bigbox	$\bigcurlyvee$	\bigcurlyvee	$\bigcurlywedge$	\bigcurlywedge
$\biginterleave$	\biginterleave	$\bignplus$	\bignplus	$\bigparallel$	\bigparallel
$\bigsqcap$	\bigsqcap	$\bigtriangledown$	\bigtriangledown	$\bigtriangleup$	\bigtriangleup

TAB. 29 – Les grands opérateurs de stmaryrd

$\inplus$	\inplus	$\niplus$	\niplus	$\subsetplus$	\subsetplus	$\subsetpluseq$	\subsetpluseq
$\supsetplus$	\supsetplus	$\supsetpluseq$	\supsetpluseq	$\trianglelefteqslant$	\trianglelefteqslant	$\trianglerighteqslant$	\trianglerighteqslant

TAB. 30 – Les relations binaires de stmaryrd

Bien entendu, cette liste de symboles ne suffit pas à faire tout ce que l'on veut. Les sections suivantes apportent quelques commandes supplémentaires.

5.4 Indices et exposants

Les indices et les exposants s'obtiennent avec les deux symboles ^ et _, dont l'argument est mis respectivement en exposant et en indice. La position de l'indice et de l'exposant dépend du mode mathématique choisi ainsi que du symbole précédent. Les exemples qui suivent sont assez explicites à ce sujet :

```
$x^2$, $a_i$, $b_{i,j}$, $b_i,j$, \\
$\sum_{k=0}^{n} k = (n^2+n)/2$,
$$\sum_{k=0}^{n} k^2 = (2n^3+3n^2+n)/6,$$
```

x^2, a_i, $b_{i,j}$, b_i,j,
$\sum_{k=0}^{n} k = (n^2+n)/2$,

$$\sum_{k=0}^{n} k^2 = (2n^3+3n^2+n)/6,$$

```
% Deux petits trucs qui peuvent être utiles :
$\sum\limits_{k=0}^{n} k = (n^2+n)/2$, \\
$\displaystyle{\sum_{k=0}^{n} k = (n^2+n)/2}$,

% Le multi-indices...
$x_{i_{j_{k}}}$, $2^{2^{2^{2}}}$, ...
```

$\sum\limits_{k=0}^{n} k = (n^2+n)/2$,
$\displaystyle{\sum_{k=0}^{n} k = (n^2+n)/2}$,
$x_{i_{j_{k}}}$, $2^{2^{2^{2}}}$, ...

Pour faire des intégrales au lieu des sommes, c'est `\int` au lieu de `\sum`, en mettant toujours en indice et en exposant les bornes de l'intégrale.

5.5 Les tableaux

L'environnement `tabular`, que j'ai présenté à la page 13, n'est pas compatible avec le mode mathématique. Il faut alors utiliser l'environnement `array`.

Le fonctionnement est exactement le même que celui de `tabular`, je ne m'étendrai donc pas là-dessus. Simplement, avant d'utiliser la commande `\begin{array}`, il faut être en mode mathématique, et n'en sortir qu'après le `\end{array}`.

Il peut être utile de savoir dessiner des matrices, et donc en particulier de savoir mettre des parenthèses autour des tableaux. Cela peut être fait simplement avec les commandes `\left` et `\right`. Ces commandes peuvent être utilisées en toute généralité, et leur argument, normalement, un délimiteur, se met à la taille de l'objet qu'il délimite. Un `\left` doit toujours être accompagné d'un `\right`. Si on ne veut qu'un délimiteur d'un côté, on utilise, pour l'autre côté, le délimiteur « blanc », qui est le point. Les formules de l'exemple qui suit devraient permettre à chacun de comprendre.

```
$$\begin{array}{crcl}
  f \colon & \{0,1\}^n & \to & \{0,1\} \\
         & (a_1, \ldots, a_n) & \mapsto &
          \prod_{i=1}^n a_i \end{array}$$
```

$$\begin{array}{crcl} f \colon & \{0,1\}^n & \to & \{0,1\} \\ & (a_1, \ldots, a_n) & \mapsto & \prod_{i=1}^n a_i \end{array}$$

```
$$\mathbb{Q} =
\left\{
\left.\frac{p}{q}\right|
p\in\mathbb{Z},\ q\in\mathbb{N}^*
\right\}
$$
```

$$\mathbb{Q} = \left\{ \left.\frac{p}{q}\right| p\in\mathbb{Z},\ q\in\mathbb{N}^* \right\}$$

```
$$\operatorname{sgn}(x) = \left\{
\begin{array}{ll}
 -1 & \text{ si }x\text{ est négatif,} \\
 0  & \text{ si }x\text{ est  nul,} \\
 1 & \text{ si }x\text{ est positif.}
  \end{array}\right. $$
```

$$\operatorname{sgn}(x) = \left\{ \begin{array}{ll} -1 & \text{ si }x\text{ est négatif,} \\ 0 & \text{ si }x\text{ est nul,} \\ 1 & \text{ si }x\text{ est positif.} \end{array}\right.$$

Cependant, ce dernier exemple peut être amélioré grâce à l'environnement `cases`, qui place une accolade à gauche et crée un tableau à deux colonnes. Par exemple :

```
$$\operatorname{sgn}(x) = \begin{cases}
-1 & \text{ si }x\text{ est négatif,} \\
0  & \text{ si }x\text{ est  nul,} \\
1 & \text{ si }x\text{ est positif.}
 \end{cases} $$
```

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{ si }x\text{ est négatif,} \\ 0 & \text{ si }x\text{ est nul,} \\ 1 & \text{ si }x\text{ est positif.} \end{cases}$$

5.6 Les équations

Pour écrire une équation, on peut utiliser simplement l'environnement `displaymath`. Mais plusieurs améliorations peuvent être apportées :

– une équation peut être numérotée, afin d'y faire référence par la suite. On utilisera pour ce faire l'environnement `equation`. On peut bien entendu mettre un `\label` dans une équation pour y faire référence par la suite.

– on peut vouloir mettre plusieurs équations l'une en-dessous de l'autre. On utilisera pour ce faire l'environnement `align`, qui crée un tableau à deux colonnes, la première étant alignée à droite et la deuxième alignée à gauche. Chaque ligne sera numérotée. On obtiendra ça, par exemple :

```
 \begin{align}
   2x + 3y + 5z & =  u \\
   4x + y - 2z & =  v^2. \label{eq}
\end{align}
On peut ensuite faire référence à
l'équation~\eqref{eq}...
```

$$\begin{aligned} 2x + 3y + 5z &= u && (1) \\ 4x + y - 2z &= v^2. && (2) \end{aligned}$$

On peut ensuite faire référence à l'équation (2)...

– dans le cas d'équations un peu longues, par exemple si on écrit le texte sur deux colonnes par page, on pourra vouloir couper une équation pour la reprendre sur la ligne suivante. On pourrait le faire avec `align`, mais c'est pas beau. On utilisera donc l'environnement `multline`, qui est fait pour ça :

```
\begin{multline}
\forall \varepsilon>0, \exists \eta >0 :
\forall x \in \mathbb{R}, \forall y \in
\mathbb{R}  \\
|x - y|<\eta \Rightarrow |f(x) - f(y)|
                              <\varepsilon.
\label{eq2}
\end{multline}
On peut ensuite faire référence à
l'équation~\eqref{eq2}...
```

$$\forall \varepsilon > 0, \exists \eta > 0 : \forall x \in \mathbb{R}, \forall y \in \mathbb{R} \\ |x - y| < \eta \Rightarrow |f(x) - f(y)| < \varepsilon. \quad (3)$$

On peut ensuite faire référence à l'équation (3)...

Au passage, tu remarqueras que c'est le même compteur qui est utilisé pour `align` et `multline`, et en fait par tous les environnements du même genre ;

– enfin, on peut vouloir supprimer ou modifier la numérotation des équations. Pour la supprimer, il suffit d'utiliser la version étoilée des environnements `equation`, `align` et `multline`. Pour ne pas numéroter *une* équation, notamment dans l'environnement `align`, on pourra utiliser la commande `\notag`. Et pour préciser le « nom » de l'équation, on utilisera la commande `\tag`. Un exemple et on n'en parle plus :

```
\begin{align}
2x + 3y + 5z &= u \notag \\
4x + y - 2z &= v^2.\tag{E}\label{eq3}
\end{align}
Bien s\^ur, le label \eqref{eq3} est bien celui
que j'ai mis en argument de \verb+\tag+.
```

$$2x + 3y + 5z = u$$
$$4x + y - 2z = v^2. \tag{E}$$

Bien sûr, le label (E) est bien celui que j'ai mis en argument de `\tag`.

5.7 Les environnements de type `theorem`

Les packages `amsthm` et `theorem` permettent de définir des environnements de type *theorem*, comme par exemple le lemme 5.1. Le fonctionnement de ces packages est très simple : on définit l'environnement à l'aide de la commande `\newtheorem`. Bien sûr, on peut définir un environnement de type theorem pour les définitions, les propositions, ... Voici un exemple : si on définit l'environnement `lemme` par

```
\newtheorem{lemme}{Lemme}[section]
```

chaque lemme sera numéroté automatiquement, et le compteur associé aux lemmes sera remis à zéro à chaque changement de section. On peut bien entendu faire référence à un lemme avec un `\label` placé après le `\begin{lemme}`, et un `\ref`. Par exemple, le lemme ci-dessous est le lemme 5.1.

Lemme 5.1 *Pour commencer ce lemme, j'ai tapé* `\begin{lemme}`*, et pour en sortir, je vais taper* `\end{lemme}`*.*

Si on souhaite définir un environnement de type `theorem` non numéroté, on procèdera de la façon suivante :

```
\newtheorem{monlemme}{Lemme }[section]
\renewcommand\themonlemme{}
```

La commande `\themonlemme` est la commande qui est appelée pour afficher le compteur `monlemme`. De manière générale, quand on a un compteur, on le fait précéder de `\the` pour demander à LaTeX d'en afficher la valeur.

La définition ci-dessus produira ensuite des lemmes ressemblant à ça :

Lemme *Voici un lemme non numéroté. Je reparlerai un peu plus en détails des manipulations de compteurs dans l'annexe A.1.*

Le package `theorem` permet de définir des styles de théorèmes bien plus évolués, je te laisse aller regarder la doc correspondante.

5.8 Quelques commandes diverses

Terminons par quelques commandes utiles :

les fontes spéciales : il existe plusieurs fontes différentes pour écrire en mode mathématique. Elles sont similaires aux fontes définies en mode texte, et s'obtiennent par les commandes `\mathrm`, `\mathbf`, `\mathit`, `\mathsf`, `\mathtt` et `\mathnormal`. Les symboles mathématiques gras s'obtiennent avec la commande `\boldsymbol`. D'autres fontes, plus spéciales, sont également disponibles. En particulier, le fonte `\mathcal`, qui permet d'obtenir des lettres caligraphiques majuscules uniquement, comme $\mathcal{ABCDEF}$, la fonte `\mathbb`, qui permet d'écrire les ensembles $\mathbb{R}$, $\mathbb{Q}$ ou $\mathbb{Z}$, et enfin la fonte `eufrak`, qui permet d'obtenir des lettres comme ça : $\mathfrak{ABCDEF}$ grâce à la commande `\mathfrak`. Enfin, un petit package qui peut être « amusant » : `oldstyle`, qui permet d'écrire les chiffres à l'ancienne. Par exemple, `\mathos{123}` donne 123.

les fractions et les racines : la commande permettant de faire des fractions est `\frac`. Elle prend pour arguments le numérateur d'abord, et le dénominateur ensuite. Par exemple,

```
$\frac{1234}{4321}$
```

donnera $\frac{1234}{4321}$. En mode `math`, c'est écrit en petit afin de ne pas espacer les lignes, mais c'est peu lisible. Il est donc préférable d'utiliser en mode `displaymath`. Cependant, si on emboîte deux fractions, ou si on met une fraction dans un tableau, ça va être écrit petit aussi. Il est donc préférable d'utiliser la commande `\cfrac`, qui agit comme `\frac` mais en évitant les désagréments[41]. Enfin, une commande très générale, `\genfrac`, permet de définir des délimiteurs pour une fraction (ce sont les deux premier arguments), mais aussi de régler l'épaisseur du trait de fraction (le troisième argument), la position du trait (quatrième argument). Il reste bien entendu deux arguments, qui sont le numérateur et le dénominateur... Par exemple, `$\genfrac<>{0pt}{1pt}{n}{p}$` permet d'obtenir $\genfrac{\langle}{\rangle}{0pt}{1}{n}{p}$.
Les racines k-ièmes sont obtenues avec la commande `\sqrt`, qui prend un argument obligatoire, ce qui se trouve sous la racine, et un argument optionnel, le « degré » de la racine. Par exemple,

```
$\sqrt[k]{n}$
```

donnera $\sqrt[k]{n}$.

flèches extensibles : c'est pratique, notamment pour écrire qu'une fonction ou une suite converge. Un petit exemple devrait suffire, je pense :

```
\[
f_n \xrightarrow[n\to\infty]{\text{unif.}} f
\]
```

$$f_n \xrightarrow[n\to\infty]{\text{unif.}} f$$

flèches, barres : pour écrire un vecteur, par exemple, ou un complexe conjugué, on aura besoin de ce genre de choses. Deux solutions :
- les commandes `\vec` et `\bar`, qu'on a déjà vues dans le tableau 20 et qui donnent ça : $\vec{x}$ et $\bar{x}$. L'inconvénient, c'est pour $\vec{AB}$, la flèche est manifestement trop petite ;
- les commandes `\overrightarrow`, `\overleftarrow` et `\overline`, qui sont faites pour ça : on obtient $\overrightarrow{x}$, $\overrightarrow{AB}$, ou $\overline{z}$.
- dans le même ordre d'idées, on peut vouloir mettres des chapeaux, ronds (vive les bretons) ou tildes au dessus de certains symboles. On regardera alors du côté du package `yhmath`.

Reste un problème de taille, qui se pose autant pour les barres ou les vecteurs que pour les racines : quand on écrit $\sqrt{y} = \sqrt{h}$, c'est très laid : ce seait plus beau si les deux symboles « racine carrée » étaient de la même taille, comme $\sqrt{y\vphantom{h}} = \sqrt{h\vphantom{y}}$. Pour obtenir ce résultat, on utilise la commande `\vphantom`, qui permet d'insérer un truc invisible mais qui a une hauteur donnée. Si on met ça en dessous d'une racone, LaTeX va agrandir un peu la racine pour que ce truc invisible soit bien en dessous. Pour obtenir la deuxième version, j'ai donc tapé :

```
$\sqrt{y\vphantom{h}}= \sqrt{h\vphantom{y}}$
```

L'argument de `\vphantom` peut être ce qu'on veut, et le résultat sera un machin invisible qui a exactement la même hauteur et la même profondeur que l'argument. Il existe l'équivalent horizontal, qui s'appelle `\hphantom`.

6 Les graphismes sous LaTeX

Quand on écrit un article, on a très souvent besoin de faire des figures. LaTeX possède quelques commandes permettant de le faire, mais elles sont très limitées. Par contre, il existe deux packages très intéressants : `xy` et `pstricks`.

[41] `\cfrac` veut dire *continuous fraction*, donc c'est particulièrement adapté à l'emboîtement de plusieurs fractions.

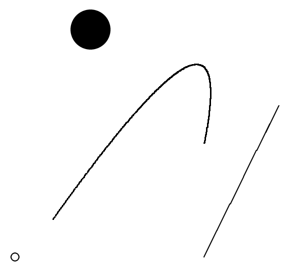
6.1 Les graphismes simples

Comme nous l'avons vu précédemment, LaTeX permet d'inclure des figures au format « PostScript encapsulé » grâce au package `graphicx` et à la commande `\includegraphics`. Ces figures peuvent être des objets flottants, *i.e.* ils peuvent être déplacés par LaTeX, lorsqu'il n'y a pas assez de place pour les mettre à l'endroit où ils sont définis. Tout cela est développé dans l'annexe 6.4. XFIG permet de créer facilement des dessins exportés en PostScript encapsulé.

Cependant, LaTeX permet de faire des graphismes — très, voire trop, simples — directement, en utilisant l'environnement `picture`. Cet environnement permet principalement de tracer des lignes, des cercles, et des courbes de Bézier. Il doit prendre deux arguments, qui sont tous les deux placés entre parenthèses. Le premier indique les dimensions du dessin, le second indique les coordonnées du point inférieur gauche. Ces dimensions sont indiquées avec des nombres décimaux, mais il faut cependant préciser l'unité de longueur que LaTeX doit utiliser pour tracer le dessin. Cela se fait avec la longueur `\unitlength`. Pour modifier cette longueur, on peut utiliser les commandes `\setlength` ou `\addtolength`, qui sont expliquées plus en détail dans l'annexe A.2.

Pour tracer un dessin, c'est enfantin : si on veut tracer une ligne, c'est la commande `\line`, qui prend comme argument le vecteur directeur et la longueur de la ligne à tracer. Le vecteur s'indique entre paranthèses, la longueur entre accolades. Un cercle s'obtient par la commande `\circle` suivi de son rayon. La commande `\circle*` trace un cercle plein. Pour placer ces deux types de tracés à un endroit précis, on utilise la commande `\put`, suivie d'un point entre parenthèses qui indique la position où il faut placer le dessin passé en argument. Enfin, une courbe de Bézier de degré 3 s'obtient avec la commande `\qbezier`, suivi d'une liste de points entre parenthèses bien sûr. Voici un petit exemple :

```
\setlength\unitlength{1mm}
\begin{picture}(40,40)(-10,-1)
\put(0,0){\circle{1}}
\put(25,0){\line(1,2){10}}
\put(10,30){\circle*{15}}
\qbezier(5,5)(30,40)(25,15)
\end{picture}
```



Ça ne représente rien, je sais, je dessine mal.

6.2 XY-pic

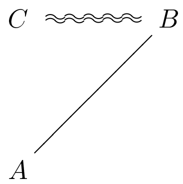
Le package `xy` est un package qui se divise en plusieurs parties. Pour charger ces parties, on le signale dans l'argument optionnel de `\usepackage`.

Je ne détaillerai pas tous les sous-packages de XY-pic ici. Le manuel de référence, écrit par l'auteur du package, Kristoffer ROSE, est très bien fait pour les personnes qui voudront y consacrer un peu de temps.

Tout objet utilisant XY-pic doit être placé dans l'environnement `xy` définit par le package. Pour faire un diagramme, il suffit de placer les sommets, de la façon suivante : `(x,y)*+{label}`, où x et y sont des entiers, et `label` est le « nom » du sommet. La définition de deux sommets doit toujours être séparée par un point-virgule.Pour relier des sommets, juste après les avoir définis, on utilise la commande suivante : `**\dir{-}`

pour tracer une ligne. Si on remplace le - par un ~, la ligne sera ondulée. Un > permet d'indiquer qu'on veut une flèche à la fin de la ligne. Voici un exemple[42] :

```
\catcode`\;=12\catcode`\:=12
\begin{xy}
(0,0)*+{A}; (20,20)*+{B} **\dir{-};
(0,20)*+{C} **\dir2{~}
\end{xy}
```



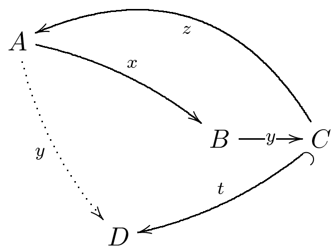
On s'aperçoit bien vite que ce n'est pas la meilleure façon d'écrire un diagramme : l'écriture est compacte, mais complètement hermétique pour quelqu'un qui ne connaît pas XY-pic.

C'est pourquoi nous allons plutôt utiliser la commande `\xymatrix`. Il faut pour cela charger tous les packages de XY-pic, *i.e.* passer l'option `all` lors du chargement du package `xy`.

La commande `\xymatrix` est basée sur le modèle des tableaux classique que l'on fait avec LaTeX : on entre les éléments du tableau, séparés par des & et des \\ en fin de ligne, et on les relie par des flèches, sur lesquelles ont peut mettre un « label », et indiquer la « courbure » à suivre. Les flèches sont indiquées juste après la définition de leur point de départ, par la commande `\ar`, suivie du style de flèche à utiliser – `@{-}` pour un trait classique, `@{.}` pour un trait pointillé... – du sens de courbure de la flèche –indiqué par `@/_/` pour une flèche courbée vers le bas et `/^/` pour une courbure vers le haut –, de la case cible, désignée entre crochets avec les lettres *d*, *u*, *r* et *l* pour le bas, le haut, la droite et la gauche respectivement. Chaque lettre déplace la cible d'une case du tableau. Pour aller trois cases à gauche et deux cases en haut, on tapera donc *llluu*. Enfin, on entre le label de la transition, précédé de `_`, `^` ou `|-` suivant que le label doit être en dessous de, au dessus de ou sur la flèche.

Voici un exemple, qui permet de bien comprendre[43] :

```
\xymatrix{
A \ar@/_/@{.>}[ddr]_y \ar@/^/[drr]^x & & & \\
& & B \ar@{->}[r]|-{y} & C \ar@/_1cm/[ulll]^z
\ar@{^{(}->}@/^/[dll]_{t} \\
& D & &
}
```



6.3 PSTricks

Le package `pstricks` est plutôt un package consacré aux dessins purs et durs, et non pas simplement aux diagrammes. Les diagrammes créés par `pstricks` sont beaucoup plus simples que ceux que permet de faire XY-pic. Par ailleurs, comme l'indique bien son nom, `pstricks` ne crée que du code PostScript. Il faut donc, pour visualiser le résultat, convertir le document en PostScript, avec la commande `dvips`. Le fichier `.dvi` laissera un espace blanc. Enfin, comme XY-pic, le package est « découpé » en plusieurs sous-packages.

Le mieux pour apprendre à s'en servir est bien entendu de lire le manuel de référence. Il a été écrit par l'auteur du package PSTricks, Timothy VAN ZANDT.

Les principaux packages de PSTricks s'appellent

[42]La première ligne, c'est juste pour dire à LaTeX de ne pas interpréter les symboles de ponctuation ; ils ne sont pas interprétés par défaut, mais `babel` les interprète pour ajuster l'espace devant de derrière. Pour comprendre mieux ce que fait la commande `\catcode`, je te renvoie à l'annexe D.5.

[43]C'est ironique... Il est déconseillé d'utiliser XY-pic sans suivi médical...

– `pstricks`, qui est le package de base,
– `pst-text`, qui permet de manipuler du texte,
– `pst-grad`, qui permet de faire des dégradés,
– `pst-plot`, qui permet de tracer des courbes,
– `pst-node`, qui permet de construire des diagrammes, des automates.

La première chose que permet `pstricks`, c'est le changement de couleurs. De nombreuses couleurs sont définies[44], et on peut écrire de toutes les couleurs. On peut même écrire [45]. Il existe par ailleurs plusieurs nuances de gris. On peut redéfinir des couleurs que l'on veut, grâce à la commande `\newrgbcolor`, à laquelle on passe le nom de la couleur qu'on définit. On peut définir des nuances de gris avec `\newgray`, qui demande le nom de la nuance et le degré de clarté du gris. Un exemple, et on n'en parle plus :

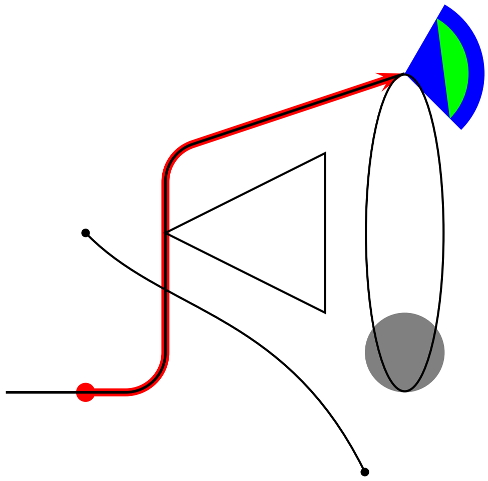
```
{\red Les commandes de changement
\blue de couleurs ne prennent pas
\yellow d'argument. Il faut donc
\green les mettre dans un groupe.}

\newrgbcolor{pasbelle}{.3 .7 .4}
{\pasbelle Cette nouvelle couleur
n'est pas belle}.

Il existe aussi des {\gray nuances
de gris}.

\newgray{clair}{.8}
{\clair On peut en définir d'autres...}
```

Les commandes de changement de couleurs ne prennent pas d'argument. Il faut donc les mettre dans un groupe.
Cette nouvelle couleur n'est pas belle.
Il existe aussi des nuances de gris.
On peut en définir d'autres...

Beaucoup d'objets de `pstricks` peuvent être placés directement dans le document, mais il vaut mieux, en général, utiliser l'environnement `pspicture` que fournit `pstricks`. Cet environnement prend un argument obligatoire, qui indique les coordonnées du point supérieur droit de la « picture », et un argument facultatif, qui donne les coordonnées du point inférieur gauche, par défaut $(0, 0)$. Rien n'empêche ensuite de sortir de cette « boîte », mais ces coordonnées seront utilisées pour centrer l'image.

À l'intérieur de cet environnement, on peut faire plein de choses. Tout d'abord, on peut préciser une unité de longueur, comme avec l'environnement `picture`. La variable qui contient cette unité de longueur s'appelle `unit`, et elle vaut par défaut 1 cm. Elle peut être modifiée par la commande `\psset`. Par défaut, les angles s'expriment en degrés. Le package `pstricks` permet alors de faire des dessins de façon très simple, mais plus puissamment que LaTeX. Un exemple suffit pour convaincre :

[44]Évidemment, pour ceux qui lisent la version `dvi`, ce n'est pas très palpitant, puisque `xdvi` est noir et blanc...
[45]Pour ceux qui lisent la version PostScript ou papier : c'est écrit « en blanc ». Pour ceux qui ont la version dvi, c'est écrit en blanc, donc ça laisse un blanc !

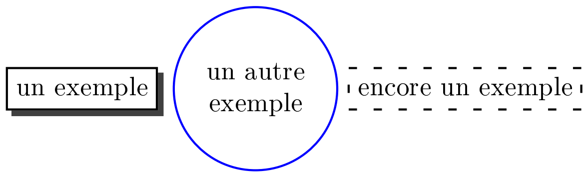


```
\psset{unit=3pt}
\begin{pspicture}(-10,-10)(50,50)
\psline[linewidth=1,linecolor=red,linearc=5]
                {*->}(0,0)(10,0)(10,30)(40,40)
\psline[linewidth=1pt,linearc=5]
                (-10,0)(10,0)(10,30)(40,40)
\pspolygon(30,10)(30,30)(10,20)
\pscircle*[linewidth=5pt,linecolor=gray](40,5){5}
\pswedge*[linecolor=blue](40,40){10}{-45}{60}
\psarc*[linecolor=green](40,40){8}{-45}{60}
\psbezier{*-*}(0,20)(10,10)(25,10)(35,-10)
\psellipse(40,20)(5,20)
\end{pspicture}
```

On s'apreçoit tout de suite que le code est beaucoup plus lisible, et ne demande que très peu d'explications[46]. En voici quelques-unes quand même :

- les deux `\psline` ont pour principale différence que la largeur de la première ligne est donnée sans unité, et est donc multipliée par `unit`. La largeur de la deuxième vaut un point.
- La valeur de `linearc` définit l'« arrondi » des angles.
- La version étoilée de chaque commande produisant une forme fermée remplit alors la forme créée de la couleur du crayon.
- La commande `\psbezier` est une courbe de Bézier d'ordre trois ou quatre, suivant le nombre d'arguments. L'argument `{*-*}` est un type de flèche, comme `{->}`.

`pstricks` permet également d'agir sur du texte. Pour cela, il faut charger le package `pst-text`. On peut alors faire beaucoup de choses :



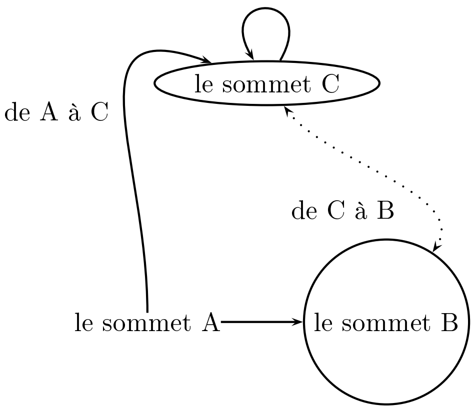

```
\psshadowbox{un exemple}
\pscirclebox[linecolor=blue]{\begin{tabular}{c}
un autre \\ exemple \end{tabular}}
\psframebox[linestyle=dashed, dash=3pt 6pt]
  {encore un exemple}
```

Voyons enfin ce que l'on peut faire pour les diagrammes : c'est le package `pst-node` qui permet de faire ce travail. Il faut tout d'abord définir les sommets de notre diagramme. Les sommets simples se définissent avec `\rnode`, suivi du nom du sommet et du texte qui doit être écrit sur ce sommet. Par exemple, `\rnode{A}{A}`

[46]Essaye par exemple de trouver à quelle objet de la figure correspond chaque ligne du code... C'est pas très compliqué.

définit le sommet A et y écrit A. Si on veut que le sommet soit matérialisé par un cercle autout du texte, on utilisera la commande `\circlenode`, toujours suivie du nom du sommet et du texte à écrire. Il existe même une commande `\ovalnode`. Bien entendu, il faut indiquer à quelle position on veut que ces nœuds soient placés. Pour cela, on utilise la commande `\rput`. Si on veut que notre nœud A soit placé en $(10, 20)$, on écrit `\rput(10,20){\rnode{A}{A}}`.

Une fois les sommets définis, on va tracer les arêtes. La commande `\ncline` permet de faire un trait simple entre les deux sommets qu'on lui passe en argument. On peut également lui demander de mettre une flèche aux extrêmités. La commande `\ncarc` permet de faire des arêtes courbées. Il faut pour cela lui indiquer, par l'argument `arcangleA`, l'angle qu'elle doit faire avec la ligne droite en partant du premier sommet, et par `arcangleB`, l'angle qu'elle doit faire avec la ligne droite en arrivant au deuxième sommet. Enfin, le label que l'on veut placer sur larête s'indique à l'aide des commandes `\aput`, `\bput` ou `\mput` suivant qu'on le veut à gauche, à droite ou sur l'arête lorsqu'on la parcourt. Voici un exemple pour clarifier tout ça :




```
\psset{unit=3pt}
\begin{pspicture}(50,50)
\rput(10,10){\rnode{A}{le sommet A}}
\rput(40,10){\circlenode{B}{le sommet B}}
\rput(25,40){\ovalnode{C}{le sommet C}}

\ncline{->}{A}{B}
\nccurve[angleA=90,angleB=160,ncurv=1.2,
    arrows=->]{A}{C}
\aput{0}{de A à C}
\ncarc[arcangleA=10,arcangleB=60,
       linestyle=dotted]{<->}{C}{B}
\bput{0}{de C à B}
\nccurve[angleA=60,angleB=120,ncurv=6,
         arrows=->]{C}{C}
\end{pspicture}
```

6.4 Les objets flottants

On appelle ainsi les objets qui peuvent être déplacés par LaTeX, généralement par manque de place. Ce sont généralement les grands objets, tels que les figures et les tableaux. Les environnements qui génèrent des objets flottants sont `figure` et `table`. Ces deux environnements sont assez similaires : leur contenu sera « mobile », c'est à dire que LaTeX peut les déplacer s'il n'arrive pas à les placer là où on le lui demande. Comme on le verra plus tard, on peut ajouter une légende dans ces environnements, ce qui permet d'y faire référence. Et c'est là la seule différence entre ces environnements : le « nom » placé devant la légende, afin de différencier le tableau 2, qui est à la page 5, de la figure 2 de la page 35. Pour le reste, ces environnements sont identiques, et LaTeX ne se plaindra pas de trouver un environnement `picture` dans un environnement

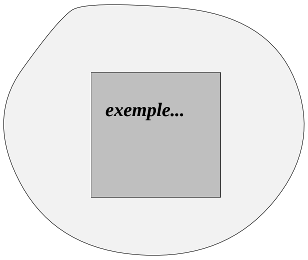


FIG. 2 – Une figure

`table` ou un environnement `tabular` dans une `figure`[47].

Revenons au problème du placement d'un flottant : on peut inciter LaTeX à placer le flottant à un endroit plutôt qu'à un autre. Pour cela, les environnements `figure` et `table` peuvent prendre un argument optionnel, composé de lettres parmi les suivantes : *h* comme *here*, qui demande que le flottant soit placé là où il est déclaré, *t* comme *top*, pour qu'il soit mis en haut d'une page, *b* comme *bottom*, pour que le flottant soit en bas d'une page, et *p* comme *page*, pour qu'il soit sur une page réservée aux flottants. Bien sûr, ce ne sont que des indications, et LaTeX se réserve le droit de ne pas les suivre. Il existe cependant un package, nommé `here`, et qui permet de mettre dans l'argument optionnel la lettre *H*, qui signifie *here*, mais qui est cette fois impérative.

Lorsqu'une figure « flotte », *i.e.* lorsqu'elle est déplacée, il faut quand même savoir où elle est pour pouvoir en parler. On peut alors y faire référence avec les commandes `\label` et `\ref`. Cependant, il faut mettre un numéro pour pouvoir faire référence à l'objet en question. Ce numéro est créé en même temps que la légende de la figure ou du tableau, par la commande `\caption` , qui prend le texte de la légende en argument. Suivant l'environnement utilisé (`table` ou `figure`), LaTeX fera précéder la légende de TAB. ou FIG. [48].

Voici un petit exemple :

```
La figure~\ref{unefig} est un exemple de flottant.
\begin{figure}[t]
\begin{center}
\includegraphics[scale=.4]{ex2.eps}
\caption{Une figure} \label{unefig}
\end{center}
\end{figure}
Pour montrer que ça flotte, je mets du texte après... J'ai demandé
ici que la figure soit placée en haut d'une page.
```

donne :

> La figure 2 est un exemple de flottant. Pour montrer que ça flotte, je mets du texte après... J'ai demandé ici que la figure soit placée en haut d'une page. Elle doit y être, normalement...

Afin de ne pas surcharger une page de flottants, il est possible d'en limiter le nombre ou la proportion par page, pour le haut et le bas de la page. `topnumber` et `bottomnumber` sont des compteurs indiquant le nombre de flottants autorisés en haut et en bas de la page, et `\topfraction` et `\bottomfraction` sont des commandes définissant la proportion d'espace rempli par des flottants. Par exemple, LaTeX définit

[47] Attention, il ne faut pas utiliser une `longtable` dans un environnement flottant : `longtable` est fait pour être positionné à l'endroit demandé et sectionné si besoin. Un tableau créé avec `longtable` à l'intérieur d'une `table` ne serait pas coupé.

[48] Le `\label` doit être situé immédiatement après la commande `\caption`, sous peine de ne pas renvoyer le numéro de la figure mais le numéro du chapitre.

```
\setcounter{topnumber}{2}
\newcommand\topfraction{.7}
\setcounter{bottomnumber}{1}
\newcommand\bottomfraction{.3}
\setcounter{totalnumber}{3}
\newcommand\textfraction{.2}
\newcommand\floatpagefraction{.5}
```

ce qui limite à 50% la proportion de flottants pour une page, et à 2 le nombre de flottants à mettre en haut d'une page.

A Les variables de LaTeX

Comme on l'a vu tout au long de cette documentation, LaTeX est extrêmement paramétrable. On peut régler pratiquement tout[49], c'est ce que va montrer cette section.

A.1 Les compteurs

Un compteur c'est une variable qui contient un entier. Son nom est une suite de lettres, et ne commence pas par un \. Par exemple, les numéros de chapitre sont mémorisés dans le compteur `chapter`, les numéros de page dans `page`, ... Si on veut utiliser un nouveau compteur, il faut le déclarer, avec la commande `\newcounter`, suivie du nom du compteur, entre accolades. Pour assigner une valeur à un compteur, c'est la commande `\setcounter`, suivie du nom du compteur et de la valeur qu'on veut lui donner. Pour augmenter la valeur d'un compteur, c'est `\addtocounter`, avec la même syntaxe que la commande précédente. La commande `\multiply` permet de multiplier la valeur du compteur, qui est son premier argument, par le nombre qui est son deuxième argument. Enfin, pour diviser un compteur — division entière, bien sûr — on utilisera la commande `\divide`, suivie de la valeur d'un compteur, de `by` et du dénominateur. Dans tous les cas, la valeur retournée est placée dans le compteur qui est passé en premier argument.

Pour afficher la valeur d'un compteur dans le document, on utilise la commande `\the` suivie, sans espace, du nom du compteur sans \. La commande `\the` suivie du nom du compteur est une commande à part entière, que l'on peut donc redéfinir. Par exemple, pour que le compteur soit écrit en chiffre romains :

```
\newcounter{exemple}
\setcounter{exemple}{194}
\theexemple

\renewcommand{\theexemple}{Le compteur
      vaut maintenant \Roman{exemple}}

\theexemple
\addtocounter{exemple}{10}
\theexemple
```

194
Le compteur vaut maintenant CXCIV Le compteur vaut maintenant CCIV

La commande `\Roman` permet d'afficher la valeur d'un compteur en chiffres romains majuscules, `\roman` fait la même chose en minuscules, `\arabic` l'écrit en chiffres arabes, `\alph` écrit la i-ème lettre de l'alphabet, i étant la valeur du compteur, et `\Alph` fait la même chose en majuscules.

ela permet en particulier de changer la présentation des environnements `enumerate`, les compteurs utilisés étant `enumi`, `enumii`, ... respectivement pour un `enumerate` de niveau 1, un `enumerate` de niveau 2, ...

```
\begin{enumerate}
\item numero $1$,
\item numero $2$,
% Ca, c'est le enumerate classique
\renewcommand{\theenumi}{\alph{enumi}}
\item numero $3$,
\renewcommand{\labelenumi}{(\theenumi)}
\item numero $4$.
\end{enumerate}
```

1. numero 1,
2. numero 2,
c. numero 3,
(d) numero 4.

Pour obtenir la valeur d'un compteur, et l'utiliser comme argument d'une commande LaTeX, on utilisera la commande `\value`. Pour illustrer tout ça, voici un petit programme LaTeX qui converti un nombre décimal en binaire :

[49] Et qui dit « régler » dit « dérégler ». Il faut donc veiller à ne pas faire n'importe quoi !

```
\def\bin{}
\newcounter{result}
\newcommand{\binaire}[1]{%
\ifthenelse{\value{#1}=0}{$\bin$}{%
  \ifthenelse{\isodd{\value{#1}}}{%
     \edef\bin{1\bin}%
     \addtocounter{#1}{-1}%
     \divide \value{#1} by 2}{%
     \edef\bin{0\bin}%
     \divide \value{#1} by 2}%
     \binaire{#1}}%
}
\setcounter{result}{1234}
\theresult{} s'écrit en binaire :
                        \binaire{result}
```

1234 s'écrit en binaire : 10011010010

Ce programme n'est pas très difficile à comprendre, le plus dur étant de connaître et savoir utiliser les commandes de programmation de LaTeX. Voir l'annexe D.1 pour comprendre l'utilisation de `\edef`, et consulter la documentation du package `ifthen` pour en savoir un peu plus...

A.2 Les longueurs

Tout comme les compteurs, LaTeX sait gérer les longueurs. Ce sont cette fois des nombres décimaux suivis d'une unité de longueur parmi les suivantes :

`sp`		la plus petite unité de longueur
`pt`		1 point = 65536 sp
`mm`		1 millimètre = 2,845 pt
`cm`		1 centimètre = 10 mm
`in`		1 inch = 25,4 mm
`ex`		1 ex = hauteur d'un x dans la fonte courante
`em`		1 em = longueur d'un m dans la fonte courante
`mu`		18 mu = 1 em, en mode mathématique.

TAB. 31 – Les longueurs reconnues par LaTeX

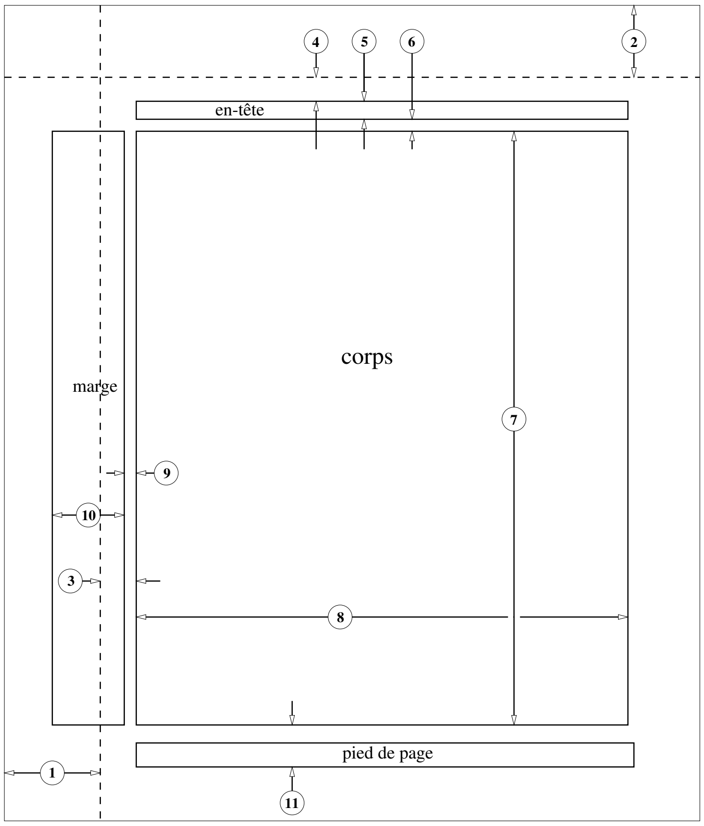
Les longueurs se manipulent de la même façon que les compteurs, avec les commandes `\newlength`, `\setlength` et `\addtolength`. À la différence d'un compteur, le nom d'une longueur doit commencer par un \, ce qui en fait une commande pour LaTeX. Cette commande renvoie la valeur de la longueur comme argument pour une commande LaTeX. Pour afficher une longueur sur le document, on utilisera à nouveau la commande `\the`. Enfin, trois commandes qui peuvent être utiles : `\settowidth`, `\settoheight` et `\settodepth`. Ces commandes prennent en premier argument une longueur, et en deuxième argument une chaîne de caractères. La longueur, hauteur ou profondeur[50] du deuxième argument est mémorisée dans le premier argument.

La figure 3 donne quelques longueurs qui peuvent servir à modifier la disposition des pages. Il convient cependant d'être très prudent dans leur modification. Le package `fullpage` permet d'agrandir les marges sans prendre de risque.

A.3 Les autres variables

LaTeX contient encore d'autres variables, qui sont en fait simplement des macro-commandes. Par exemple,

[50] Le hauteur est la longueur de ce qui est au dessus de la ligne de texte, la profondeur est la longueur de ce qui est en dessous.



1. un pouce + \hoffset
2. un pouce + \voffset
3. \evensidemargin = 70pt
4. \topmargin = 22pt
5. \headheight = 13pt
6. \headsep = 19pt
7. \textheight = 595pt
8. \textwidth = 360pt
9. \marginparsep = 7pt
10. \marginparwidth = 106pt
11. \footskip = 27pt

\hoffset = 0pt
\voffset = 0pt
\paperwidth = 597pt
\paperheight = 845pt

FIG. 3 – Quelques longueurs servant à la mise en page

la variable `\chaptername` contient le mot *Chapitre* en français. La variable `\labelitemi` contient le symbole –, qui est le symbole qui apparaît au début de chaque nouvelle entrée d'une liste[51]. La preuve :

```
\begin{itemize}
\item Le symbole au début de la ligne est
bien un \labelitemi.
\renewcommand{\labelitemi}{$\ast$}
\item Le symbole au début de la ligne est
bien un \labelitemi.
\end{itemize}
```

– Le symbole au début de la ligne est bien un –.
$\ast$ Le symbole au début de la ligne est bien un $\ast$.

La encore, toutes ces variables peuvent être modifiées — avec la commande `\renewcommand` puisqu'elles sont déjà définies auparavant — mais avec prudence et parcimonie.

[51] Le `i` à la fin de `\labelitemi` signifie que c'est le premier niveau de liste. La variable `\labelitemii` contient le symbole `-`, qui est le symbole pour une liste emboîtée dans une autre...

B Constituer un index et une bibliographie

B.1 L'index

L'index peut être généré automatiquement par LaTeX, à condition bien entendu de lui indiquer les mots qui doivent y figurer. Pour ce faire :

– dans le préambule, indiquer que LaTeX doit « retenir » les mots qui iront dans l'index. Pour cela, on tape `\makeindex`. Lorsqu'il rencontre cette commande, LaTeX crée un fichier `toto.idx`, si le fichier de départ s'appelle `toto.tex`, et dans lequel il place tous les mots qui doivent apparaître dans l'index,
– pour lui indiquer les mots qui doivent se trouver dans l'index, tu tapes `\index{nom}`, pour que `nom` soit répertorié. Comme l'index renvoie un numéro de page, il ne faut pas placer ces commandes n'importe où... Il existe quelques variantes à la commande `\index`. Par exemple, si `nom` doit apparaître à la place du mot `place`[52], il faut taper `\index{place@nom}`. Si par exemple, l'index doit comporter `nom commun` et `nom propre`, il est intéressant de faire une sous-entrée au mot `nom`. Pour cela, il faut utiliser la commande `\index{nom!commun}` et `\index{nom!propre}`. Bien entendu, si on veut mettre un ! ou un @ dans l'index, il faut un peu ruser : on le fait précéder d'un `"`. C'est également le cas de |. En effet, on peut faire référence à plusieurs pages, en indiquant une page de début et une page de fin. Pour ce faire, on met `\index{nom|(}` sur la première page, et `\index{nom|)}` sur la dernière.
– il faut indiquer dans le fichier `.tex` l'endroit où tu veux que l'index apparaisse. C'est la commande `\printindex`.
– compiler le fichier `.tex`, l'index puis à nouveau le fichier `.tex`. Il faut en effet dans un premier temps que LaTeX mette les mots de l'index dans le fichier `.idx`, puis qu'il lise le fichier trié, d'extension `.ind`. Ce fichier est créé à la compilation de l'index, par la commande `makeindex toto` où `toto` est le nom du fichier.

Il est possible, comme je l'ai fait dans cette doc, de trier l'index suivant plusieurs catégories. Je te renvoie au package `multind` si tu veux en savoir plus.

B.2 La bibliographie

La bibliographie peut être faite de deux manières différentes. La manière la plus simple est expliquée à la page 19. Je vais présenter ici la manière plus sophistiquée et beaucoup plus puissante de faire une bibliographie : en utilisant BibTeX.

Pour cela, il faut encore procéder en plusieurs étapes :

– créer un fichier de bibliographie. Ce fichier peut être aussi gros qu'on veut. On peut donc n'avoir qu'un fichier de bibliographie, que l'on complète à chaque nouveau document, et dans chaque document, on sélectionne les entrées que l'on veut. Cela évite de taper une bibliographie à chaque fois... L'extension de ce fichier doit être `.bib`, et nous l'appellerons `biblio.bib`. Le contenu de ce fichier sera détaillé plus loin.
– tout au long du document, on cite certains livres de la bibliographie. Cela se fait avec la commande `\cite`, suivie du label du livre que l'on cite. Pour les bouquins qui doivent figurer dans la bibliographie, mais que l'on ne cite pas dans le document, il faut utiliser la commande `\nocite`. Si on veut inclure toute la bibliographie, on tape alors `\nocite{*}`.
– signaler à LaTeX l'endroit où on veut que la bibliographie apparaisse. En général, c'est plutôt vers la fin. Pour cela, il faut choisir un style de bibliographie, par exemple `alpha` ou `plain`, et indiquer le nom du fichier de bibliographie, `biblio`. Le fichier `toto.tex` devra donc contenir les lignes

```
\bibliographystyle{alpha}
```

– enfin, il faut compiler le tout. En premier lieu, le fichier `toto.tex`, pour que LaTeX dresse la liste des références à inclure. Puis, le fichier `biblio.bib`. Pour cela, il faut taper `bibtex toto`. En effet, `bibtex` va lire le fichier `toto.aux` créé à la première compilation, qui va lui même appeler le fichier `biblio.bib`. Enfin, recompiler le fichier `toto.tex`.

Il ne nous reste plus qu'à voir la tête d'une entrée dans le fichier `biblio.bib`. Voici l'entrée correspondant à [GMS98] :

[52] L'index est trié par ordre alphabétique...

```
@book{LatexCompanion,
  author = "Goossens, Michel and Mittelbach, Frank and Samarin,
            Alexander",
  title = "the {{\LaTeX}} {C}ompanion",
  publisher = "Addison-Wesley",
  year = 1994,
  edition = 2,
  OPTaddress = "Je ne la conna{\^\i}s pas..."
}
```

Voici une explication rapide, dans l'ordre :

`@book` **:** indique le type de référence. Ici, c'est un livre, mais ça peut aussi être `@article`, `@conference`, `@manual`, ...

`LatexCompanion` **:** c'est le label correspondant à cette entrée. Ça peut être à peu près n'importe quoi... Pour le citer tout à l'heure, j'ai utilisé la commande `\cite{LatexCompanion}`,

`author = { ... }` **:** le ou les auteurs. On les cite généralement sous la forme `nom, prénom`, et deux auteurs d'un même livre sont séparés par un `and`,

`title = { ... }` **:** le titre... BibTEX met automatiquement le titre, excepté la première lettre, en minuscule. Pour obtenir une majuscule, il faut alors mettre la lettre correspondante entre accolades, comme le `C` de `companion`. Les doubles accolades autour de `\LaTeX`, c'est trop long à expliquer mais c'est comme ça...

`publisher` **:** l'éditeur,

`year` **:** l'année,

`edition` **:** le numéro d'édition,

`address` **:** l'adresse de l'éditeur. Les lettres `OPT` indiquent que ce champ est optionnel. Lorsque `OPT` (ou n'importe quoi d'autre) est présent, la ligne n'est pas prise en compte.

Il existe d'autres champs, comme par exemple `number`, `address`, `journal`, ...

Je signale à toutes fins utiles que j'ai écrit une documentation ([Mar02b]) sur « comment faire une bibliographie et comment modifier un style bibliographique », qui est beaucoup plus détaillée que ce que j'ai dit ci-dessus.

C Comment faire des transparents ?

Ben en fait, il y a plusieurs façons...

- `seminar` : c'est le nom d'un package de Tymothy Van Zandt, basé sur `pstricks`, et qui permet de faire des transparents assez évolués, avec des cadres, des couleurs, ... Ce package est très bien documenté, je te laisse aller regarder `Seminar-FAQ.html` et `sem-user.dvi` pour obtenir les renseignements idoines. Un article de Michel Goossens et Sebastian Rahtz, paru dans les Cahiers GUTenberg, et qui se nomme `seminar-GUT.ps`, peut également être intéressant.
- `prosper` : une extension de `seminar`, qui simplifie encore un peu plus la vie. Là encore, la documentation est bien faite, il suffit d'aller lire par exemple [Gou00] pour s'en convaincre.
- `foiltex` : écrit par Jim Hafner, `foiltex` est un peu moins puissant que `seminar` du point de vue des possibilités, mais il définit des commandes qui le rendent assez agréable à utiliser. La documentation s'appelle `foiltex.ps`.
- `slides` : ce package est peut-être très bien, très puissant, mais son principal défaut est le manque de documentation...

D Trucs et Astuces

On m'a dit : « Une doc sur LaTeX est bien faite si celui qui l'a lue est capable de la réécrire... ». Cette doc ne sera surement jamais bien faite[53], mais je vais essayer, dans cette section, de l'améliorer.

D.1 Définition de commandes

Revenons tout d'abord sur `\newcommand`. C'est bien beau (et surtout propre) de distinguer `\newcommand` et `\renewcommand`, mais s'il fallait tester à chaque fois qu'on définit une commande si elle existait déjà ou pas, on y passerait du temps. On va donc plutot utiliser la commande `\def`, qui est la version « TeX » de `\newcommand`. La commande `\def` ne s'occupe pas de savoir si la commande qu'elle va définir existait ou pas, elle la redéfinit. Par ailleurs, le nombre d'argument n'est plus passé entre crochets, mais explicitement. On écrira donc :

```
\def\commande#1#2{Ce que fait la commande avec les arguments
  #1 et #2...}
```

Il y a plusieurs interets à utiliser cette commande : d'abord, ça va plus vite à taper, ce qui n'est pas négligeable quand on a une tripotée de commandes à définir, mais aussi qu'elle permet de mettre des délimiteurs, et ainsi de définir des commandes avec arguments optionnels. Voici un exemple :

```
\makeatletter
\def\nodot#1{\ifx#1.\else#1\fi}
\def\initiales#1{\espace #1 /\nodot}
\def\espace#1 #2/{\tiret #1- /%
    \ifx#2 \else{} \espace #2 /\fi}
\def\tiret#1-#2/{\initiale #1/%
    \ifx#2 \else-\tiret #2/\fi}
\def\initiale#1#2/{#1.}
\makeatother

La commande \verb+\itnitiales+ \'ecrit
les initiales de son argument, et bouffe
\'eventuellement le point qui suit, comme
dans \initiales{Nicolas Markey}, et
\initiales{Nicolas Markey}.
```

La commande `\itnitiales` écrit les initiales de son argument, et bouffe éventuellement le point qui suit, comme dans N. M., et N. M.

Cela mérite une petite explication : la commande `\nodot` sert à éviter qu'une phrase se termine par deux points consécutifs (celui de la fin de la phrase et celui de l'initiale). Les commandes servant à extraire les initiales utilisent des délimiteurs pour séparer les noms.

Quelques autres remarques sur `\def` : normalement, les arguments ne peuvent pas contenir de sauts de paragraphe, les déclarations sont locales dans le groupe (elles sont perdues lorsqu'on sort du groupe courant), et la définition n'est « développée » que lorsque la commande est utilisée. Revenons en détail sur chacun de ces trois points :

- les arguments ne peuvent pas contenir de saut de paragraphe. En effet, c'est suffisant dans la plupart des cas, et souvent, quand un argument contient un saut de paragraphe, c'est qu'on a oublié de fermer une accolade. Cependant, il arrive qu'on veuille autoriser les sauts de paragraphe dans un argument. On fera alors précéder le `\def` par `\long`. dans le même ordre d'idées, on peut, lorsqu'on définit une commande, préciser qu'elle ne peut pas être utilisée dans l'argument d'une autre commande. C'est, là encore, une précaution pour détecter une accolade fermante qui manque. Pour définir ce genre de commande, on met `\outer` devant `\def`.

[53] Je ne mets pas en doute tes capacités, je veux juste dire qu'il va surement exister un lecteur qui, bien qu'ayant lu cette doc, ne serait pas capable de la refaire.

- les définitions sont locales à un groupe. Lorsqu'on sort d'un groupe, les définitions qui ont été faites dans le groupe sont perdues. Pour changer ce comportement, on fera précéder le `\def` de `\global` ;
- lorsqu'on définit une commande, les commandes utilisées dans la définition ne sont pas développées à la définition mais à l'utilisation de la commande. L'exemple ci-dessous devrait être assez clair :

```
\def\b{b}
\def\a{\b}
La commande \verb+\a+ \'ecrit \a.
\def\b{a}
La commande \verb+\a+ \'ecrit \a.
```

La commande `\a` écrit b.
La commande `\a` écrit a.

On utilisera alors `\edef` si on veut que les commandes qui servent dans la définition soient développées au moment de la définition. L'exemple précédent devient alors :

```
\def\b{b}
\edef\a{\b}
La commande \verb+\a+ \'ecrit \a.
\def\b{a}
La commande \verb+\a+ \'ecrit \a.
```

La commande `\a` écrit b.
La commande `\a` écrit b.

La commande `\let` permet aussi de faire ce genre de choses : elle « duplique » en fait une commande. Cela permet de faire une « sauvegarde » d'une commande, par exemple. Par exemple, si on prend la commande `\LaTeX`, elle a l'inconvénient de manger les espaces qui la suivent. Il faut donc taper `\LaTeX{}` pour éviter les problèmes d'espaces. Il serait mieux de lui ajouter `\xspace`, du package `xspace`, à la fin, afin que l'espace soit automatiquement mis si nécessaire. On pourrait tenter un audacieux

```
\def\LaTeX{\LaTeX\xspace}
```

mais LaTeX n'aime pas ça, ça le fait boucler jusqu'à ce qu'il dépasse la mémoire. Noter que l'erreur n'arrive que lorsqu'on *utilise* la commande, pas quand on la définit, puisque, à la définition, LaTeX ne cherche pas à savoir ce que fait la commande.
On va donc faire une « copie » de la commande `\LaTeX`, puis lui ajouter l'espace voulu :

```
\LaTeX bouffe les espaces apr\`es les commandes.
\let\oldLaTeX=\LaTeX
\def\LaTeX{\oldLaTeX\xspace}
\LaTeX ne bouffe plus les espaces.
```

LaTeXbouffe les espaces après les commandes.
LaTeX ne bouffe plus les espaces.

D.2 Quelques définitions qui peuvent servir

Voyons maintenant quelques macros pratiques :
- les messages du genre

```
Overfull \hbox (9.46796pt too wide) in paragraph at lines 148--160
```

c'est bien, mais quand c'est pour un quart de poil qui dépasse, on s'en passe... Pour demander à LaTeX de n'émettre un warning que si ça dépasse de plus de `5pt`, on mettra, après le `\begin{document}` la commande suivante :

```
\setlength{\hfuzz}{5pt}
```

Bien entendu, on peut mettre beaucoup plus, de façon à éviter tous les messages de ce genre. Mais il est plus propre d'aller corriger à la main les endroits ou le texte dépasse, quand même...

– pour éviter la coupure des formules mathématiques : il est assez désagréable de voir certaines formules mathématiques, courtes, être coupées en fin de ligne, alors qu'en ajoutant un peu d'espace entre les mots, ces formules pourraient apparaître en entier au début de la ligne suivante. Pour cela, on définit :

```
\newcommand\dontcutmath{\relpenalty=10000\relax
  \binoppenalty=10000\relax}
\newcommand\cutmath{\relpenalty=500\relax
  \binoppenalty=500\relax %definitions normales de LaTeX
  }
```

Les explications se trouvent dans la sous-section D.5, je pense que c'est la meilleure place. Lorsqu'on a défini ces deux commandes, il suffit alors de commencer le document par un `\dontcutmath` pour que les formules ne soient pas coupées. Cependant, les formules trop grandes devront soit être coupées, et il faudra pour cela soit exécuter `\cutmath`, dans un groupe par exemple pour que les définitions soient locales, soit les mettre dans un environnement `displaymath`.

– Pour mettre deux objets l'un au dessus de l'autre, TEX fournissait une commande `\atop`[54] qui était bien pratique, mais provoque les foudres du package `amsmath`. On la redéfinit alors avec `\genfrac` :

```
\renewcommand\atop[2]{\ensuremath{\genfrac..{0pt}{1}{#1}{#2}}}
```

– là, il ne faudra pas m'en demander trop pour les explications, mais il est souvent agréable d'avoir un signe de fin de démonstration. Voici :

```
\def\somepl@ce{%
  \parfillskip=0pt%
  \widowpenalty=10000%
  \displaywidowpenalty=10000%
  \finalhyphendemerits=0%
  \unskip\nobreak\null\hfil\penalty50\hskip5em\null\hfill%
}

\def\cqfd{\somepl@ce\ensuremath\square\break}
```

Comme pour `\dontcutmath`, cette macro se sert du principe des pénalités. Elle regarde en fait si, à la fin de la ligne courante, il y a un petit espace pour mettre le symbole. Sinon, elle passe à la ligne. Puis, elle affiche le symbole de fin de démonstration. Par exemple, si je tape `\cqfd` ici, j'ai un carré comme ça au bout de la ligne. $\square$

D.3 Autres petits trucs...

Je ne vais pas ici définir de nouvelles commandes mais en citer quelques-unes, qui existent déjà, qui sont peu connues, mais qui peuvent être assez utiles.

– les « minipages » : `minipage`, c'est le nom d'un environnement de LATEX qui est très pratique. C'est par exemple celui que j'utilise pour les exemples, afin qu'ils soient sur deux colonnes : l'environnement `minipage` permet d'élargir ou de rétrécir localement la longueur du texte. Cependant, une minipage n'est pas numérotée, n'a pas d'en-tête, ... Voici donc (à peu près) ce qui me servait pour faire les exemples :

[54] `\atop` était infixé sous TEX, mais la commande que je définis ici est préfixée.

```
\long\def\exemple#1{\null\hspace{-.1\textwidth}\noindent%
        \begin{minipage}{1.1\textwidth}
          \begin{minipage}{.5\textwidth}\verbatiminput{#1}
          \end{minipage}
          \hspace{.03\textwidth}
          \begin{minipage}{.5\textwidth}
            \null\vspace{-2.6ex}\input{#1}
          \end{minipage}
        \end{minipage}
}
```

Il suffit donc de passer à LaTeX le nom d'un fichier contenant l'exemple, et le tour est joué !

– Les notes de pied de page, comme je l'avais mentionné, ne fonctionnent pas normalement dans des tableaux et dans certaines autres commandes, comme les footnotes elles-mêmes. Je vais ici résoudre le premier problème... La commande `\footnote` et en fait composée de deux commandes : l'une qui met la petite marque (et qui s'appelle `\footnotemark`), et l'autre qui récupère le texte et le met en bas de la page, qui s'appelle `\footnotetext`. Cette deuxième commande n'est pas utilisable dans certains cas, comme par exemple les commandes de sectionnement ou les tableaux. Il est alors nécessaire de dissocier la marque et le texte, et de mettre `\footnotemark` dans le tableau, par exemple, et de mettre `\footnotetext` en dehors du tableau.

D.4 Comportement particulier de certaines commandes

Certaines commandes ne se comportent pas toujours comme on le voudrait. Par exemple, concernant encore la commande `\footnote`, il n'est pas possible de mettre une note de bas de page depuis une commande de sectionnement.

Ces commandes sont dites fragiles. Ce sont *grosso modo* les commandes qui ont un argument optionnel, comme par exemple `\footnote` et apparentées, `\item`, `\\`, `\(`, `\)`, `[`[55], `]`, et toutes les commandes qui ont une forme « étoilée », comme par exemple `\hspace`. Les commandes `\begin` et `\end` sont également fragiles. Bien entendu, dans le cas général, ces commandes ne posent pas de problème. On peut mettre une note de bas de page au milieu d'un texte sans problème. Là où un problème se pose, c'est lorsqu'on utilise une de ces commandes dans un argument « mobile ». Un argument mobile, c'est un argument qui sert plusieurs fois, et qui est donc mis « en mémoire » par LaTeX. Par exemple, toutes les commandes de sectionnement d'un texte, comme `\section`, ont un argument mobile. Les commandes `\addcontentsline`, `\addtocontents`, `\markright` et `\markboth` en font partie aussi. Noter également que les @-expressions des environnements `tabular` et `array` sont mobiles. Maintenant que le problème est décrit, voyons-en la solution : il faut « protéger » les commandes fragiles quand elles sont dans un argument mobile, et ça se fait avec la commande `\protect`. L'exemple suivant produirait une erreur en l'absence de `\protect` :

```
\section{Une section\protect\footnote{avec note de pied de page !}}
```

Cependant, dans le cas de notes de pied de page, cette solution est à proscrire : en effet, l'argument étant mobile, il est interprété à chaque « utilisation », donc en particulier, dans la table des matières, on retrouvera la note de pied de page. De même, utiliser `\footnotemark` et `\footnotetext` ne permettra pas d'enlever la marque de la note, et donc elle sera encore présente, et incrémentée, dans la table des matières. Il faut donc utiliser l'argument optionnel des commandes de sectionnement. On préfèrera donc :

```
\section[Une section]{Une section\protect\footnote{avec note de pied de page !}}
```

D.5 Ce qu'il est inutile de savoir si on ne veut pas en savoir plus...

... mais qui n'est pas bien compliqué à comprendre !

Premièrement, les codes de catégorie[56]. Pour LaTeX, chaque caractère entre dans une catégorie, définie au départ mais modifiable. Par exemple, \ est dans la catégorie des caractères d'échappement, et LaTeX sait, lorsqu'il le rencontre, que ce caractère est le début d'une macro. Voici le tableau des catégories existantes :

[55] Cependant, \$ et \$\$ ne sont pas fragiles

[56] Pour les LaTeX-niciens, on parle de *catcode*

0	caractère d'échappement	\
1	début de groupe	{
2	fin de groupe	}
3	passage au mode mathématiques	$
4	changement de colonne dans un tableau	&
5	fin de ligne	ENTER
6	paramètre d'une macro	#
7	exposant	^
8	indice	_
9	caractère ignoré	
10	espace	␣
11	lettres	a
12	autres caractères	@
13	caractères actifs	~
14	caractère de commentaires	%
15	caractère invalide	

Pour modifier la catégorie d'un caractère, on utilise la commande `\catcode`. Pour spécifier le caractère, on le fait précéder de ``\` ``. Par exemple, ``\catcode \`*=0`` aura pour effet de faire de * un caractère d'échappement, et une commande comme `*LaTeX` sera interprétée, et produira LaTeX. Bien entendu, il est possible par ce moyen de ne plus s'y retrouver du tout et de produire du code incompréhensible, ce qui n'est pas le but premier. On n'utilisera donc cette commande le moins souvent possible, bien qu'elle soit parfois inévitable.

Une autre question que l'on peut se poser est : « Comment LaTeX fait-il la mise en page ? » Voici une version simplifiée de la réponse :

Lors de la compilation, LaTeX peut se trouver dans trois « modes » différents : le mode « paragraphe », le mode « math » et le mode « LR », qui signifie *Left-Right*. LaTeX change de mode uniquement lorsqu'il entre ou quitte un environnement, ou lorsqu'il travaille sur les arguments de certaines commandes, comme par exemple les commandes définies avec `\long`.

Le mode courant est le mode « paragraphe ». C'est le mode utilisé pour lire du texte. Dans ce mode, LaTeX prend les lettres les unes après les autres, cherche dans les fichiers de fontes les dimensions de la lettre, remplace la lettre par une « boîte » aux mêmes dimensions, et met ces boîtes les unes après les autres. Entre chacune de ces boîtes, *i.e.* de ces lettres, LaTeX va placer une « pénalité »[57] : il places des pénalités spéciales aux endroits où il pourrait couper un mot, et en ajoute aux espaces. Lorsqu'il rencontre une commande, il la remplace par sa définition, autant de fois qu'il le faut pour arriver à des commandes de base[58]. Lorsqu'il doit sortir du mode « paragraphe », il va couper les lignes de la meilleure façon possible. LaTeX crée ainsi le fichier `DVI`, qui est en fait une liste *fonte – caractère – position dans la page – numéro de page*. Pour afficher le résultat, `xdvi` lit cette liste, va chercher dans le fichier de fonte le caractère voulu et le place à la position souhaitée.

Lorsque LaTeX entre en mode « math », il procède de la même façon, mais il ne met pas les mêmes pénalités. En mode « LR », il change également les pénalités, et de plus il ne cherche pas à couper les lignes. Le mode « LR » est par exemple utilisé avec les commandes `\mbox`, `\hbox` et `\vbox`. Ces commandes permettent *grosso modo* de remplacer plusieurs petites boîtes – des lettres – par une seule grosse boîte. Ça sert à éviter de séparer deux mots qu'on voudrait garder collés, à éviter une césure, ... Un autre avantage est qu'on peut définir des boîtes, comme on définit une variable, et obtenir sa longueur (avec `\wd`), sa hauteur (avec `\ht`), sa profondeur (avec `\dp`), ... La commande pour définir des boîtes est `\setbox`, les boîtes que l'on peut définir sont numérotées de 1 à 255, mais quelques-unes sont utilisées par LaTeX, et il est prudent d'utiliser les boîtes centrales plutôt que les boîtes latérales. Bon, par exemple, `\setbox222\hbox{bonjour}` met *bonjour* dans la boîte 222, la commande `\box222` écrira donc *bonjour*. Si la boîte 222 dépassait de la ligne, elle ne serait pas coupée. Enfin, par exemple, `\ht222` contient la hauteur de la boîte 222. Toutes ces commandes sont des commandes de base de TeX, et elles sont généralement à éviter.

Bon, pour terminer, parlons des pénalités . Une pénalité, c'est *grosso modo* un nombre entre -10000 et $+10000$. Entre chaque boîte, LaTeX place une pénalité. Certaines commandes ne sont que des pénalités. Par

[57] J'en reparlerai plus tard...

[58] En fait, jusqu'à arriver à des commandes TeX uniquement.

exemple, la commande `\break`, qui force LaTeX à changer de page, est simplement définie par une pénalité de -10000. Il est possible, mais déconseillé[59], d'ajouter à la main des pénalité, avec la commande `\penalty` suivie d'un nombre entre -10000 et 10000. Et ces pénalités sont utilisée par LaTeX pour savoir à quel endroit il doit couper une ligne : lorsqu'il construit sa suite de boîtes, il additionne toutes les pénalités qu'il place entre les boîtes. Puis il choisit le découpage qui minimise la pénalité globale de la ligne. Par exemple, si tu mets `\penalty-10000`, il va couper la ligne juste après cette pénalité, puisque c'est la pénalité minimale. De même pour couper une page, il va additionner les pénalités de chaque ligne, et couper à l'endroit opportun. La encore, le `\penalty-10000` est radical.

Il existe de nombreuses pénalités prédéfinies, qui sont insérées par LaTeX aux endroits nécessaires, comme par exemple les pénalités qui permettaient de couper ou ne pas couper les lignes en mode mathématique (voir à la page 44). Le système de pénalités est aussi à la base de la commande `\cqfd` de la page 44. Et de façon générale, ce système est à la base de LaTeX et de TeX, c'est l'idée géniale qu'a eu Knuth et qui donne toute la puissance de LaTeX.

E Quelques packages intéressants

Pour terminer, je vais lister ici quelques packages, avec leurs principales caractéristiques, ce qu'ils permettent de faire... Je n'entre pas dans les détails, chaque package est — théoriquement — fourni avec une documentation[60], il suffit de la lire pour avoir les informations suffisantes à son utilisation.

nom du package	auteur(s)	documentation
`amsmath`	$\mathcal{AMS}$	`amstex/amsguide.dvi`
	Package mathématiques. Il inclut en fait plusieurs packages, augmentant les possibilités mathématiques de LaTeX, que ce soit en ajoutant des commandes, des symboles, des environnements, ... Je ne détaille pas tout cela ici, j'en ai parlé un peu à la section 5.	
`array`	Franck Mittelbach, David Carlisle	`latex/tools/array.dvi`
	Nouvelles possibilités pour les tableaux. Ce package permet, avec des commandes supplémentaires dans la définition du tableau, d'automatiser certaines commandes dans la tableau. Il corrige également certains bugs dans l'implémentation des tableaux sous LaTeX.	
`babel`	Johannes Braams	`generic/babel/user.dvi`
	Package multilingue. Il permet d'utiliser plusieurs langues dans un même document, avec la commande `\selectlanguage`. Il définit ainsi les caractéristiques d'une trentaine de langues, du breton à l'esperanto. Il fournit également la commande `\iflanguage`qui permet de connaître la langue actuellement utilisée.	

[59]Il est donc déconseillé d'utiliser la commande `\break`. Par ailleurs, il faut savoir être parcimonieux avec la valeur de la pénalité infligée.

[60]Je donne dans ce tableau le nom du fichier contenant la documentation. Ces indications sont valables pour la distribution teTeX. En cas de problème, la commande `locate` sous UNIX peut permettre de trouver la localisation exacte du fichier.

nom du package (suite)	auteur(s) (suite)	documentation (suite)
`changebar`	Michael Fine, Johannes Braams	`latex/changebar/changebar.dvi`
	Ce package définit un environnement permetant de mettre un filet dans la marge. C'est assez pratique pour faire resortir un passage important. Il est bien entendu possible de paramétrer l'épaisseur, l'éloignement et l'emplacement du filet.	
`comment`	Graham Williams	
	Ce petit package assez simple définit l'environnement `comment`. Tout ce qui est placé dans un environnement `comment` est placé en commentaire. Ca permet de faire de longs commentaires, ce qui n'est pas souvent utile, mais aussi de ne pas compiler certaines parties d'un document.	
`fancybox`	Timothy Van Zandt	`fancybox.ps`
	Ce package permet d'encadrer du texte, que ce soit un mot, une phrase, un paragraphe, ou même une page entière. Ça permet de faire resortir certaines parties d'un document, c'est pas toujours une réussite, il faut utiliser avec parcimonie... Attention, ce package peut poser des problèmes avec la table des matières. Lire le début de `fancybox.sty` pour plus d'informations.	
`fancyhdr`	Piet Van Oostrum	`latex/fancyhdr/fancyhdr.dvi`
	Ce package, dont le vrai nom est `fancyheadings` mais qui a été racourci à 8 lettres, permet, comme je l'ai déjà dit à la page 11, de modifier les en-têtes et pied de page. Ça permet d'être original et de faire des choses vraiment affreuses, il convient donc d'être prudent.	
`fontenc`		
	Définit le codage utilisé pour les fontes.	
`french`	Bernard Gaulle	`french/frguide.dvi`
	Le package pour écrire en français. Il se décompose en deux packages : `frenchle`, qui définit les règles de la typographie française, les espaces, indentations, ponctuations. Il traduit également la date, et les mots utilisés par LaTeX. Le deuxième package se nomme `french`, il contient de nombreuses définitions supplémentaires, mais est maintenant shareware et donc moins intéressant que l'option `francais` et `babel`.	
`fullpage`		
	Permet simplement de redéfinir les dimensions afin que les marges soient diminuées, et que le texte s'adapte bien sur une feuille de format A4.	

nom du package (suite)	auteur(s) (suite)	documentation (suite)
`graphicx`	David Carlisle	`latex/graphics/grfguide.ps`
	Pour inclure des « images ». Toute image incluse doit être convertie au préalable au format PostScript Encapsulé. L'inclusion se fait avec la commande `\includegraphics`. Mais ce package fournit aussi des commandes de changement de couleur, de rotation, de redimensionnement, ...	
`hhline`	David Carlisle	`latex/tools/hhline.dvi`
	Ce package permet de faire des doubles lignes dans les tableaux, comme j'en fais ici... Ça ajoute une petite touche d'originalité, et ça permet de bien séparer certaines lignes. La commande pour ce faire est `\hhline`, suivie d'un argument indiquant le motif de la ligne.	
`inputenc`	Alan Jeffrey, Franck Mittelbach	`latex/base/inputenc.dvi`
	Permet de définir le codage à utiliser. On utilisera en général le codage `ISO-8859-1`, c'est à dire `latin1`, que l'on passe en option au package. Ça définit alors le comportement de chaque caractère lu par le compilateur.	
`longtable`	David Carlisle	`latex/tools/longtable.dvi`
	Permet de faire des tableaux s'étendant sur plusieurs pages. Les tableaux sont coupés automatiquement en bas de la page, et recommencent en haut de la page suivante. L'utilisation est similaire à celle de l'environnement `table`.	
`moreverb`	Robin Fairbans & Co.	`latex/styles/moreverb.dvi`
	Le nom signifie « more verbatim », ça permet de faire plein de choses en mode « verbatim », notamment d'include un listing avec l'environnement du même nom.	
`multicol`	Franck Mittelbach	`latex/tools/multicol.dvi`
	Permet d'écrire une portion de texte sur plusieurs colonnes, grace à l'environnement `multicols`. Ce package n'a rien à voir avec la commande `\multicolumn` dont j'ai parlé à la page 14.	
`multirow`	Graham Williams	
	Ce package permet de rassembler plusieurs lignes d'un tableau en une seule, tout comme la commande `\multicolumn`, dont je viens de parler, permet de rassembler les colonnes. Ce package est livré avec deux autres packages, `bigdelim` et `bigstrut`, qui permettent de mettre, par exemple, une grande accolade à côté d'une case créée avec `\multirow`.	

nom du package (suite)	auteur(s) (suite)	documentation (suite)
`pstricks`	Tymothy VAN ZANDT	`generic/pstricks/obsolete/`
	PSTriks est un énorme package permettant de faire tout ce que PostScript peut faire, c'est à dire beaucoup de choses... Des figures géométriques, par exemple, des diagrammes, ...	
`seminar`	Tymothy VAN ZANDT	`latex/seminar/`
	`seminar` est le package (en fait, c'est surtout une classe) permettant de faire des transparents. J'en parle un peu à la page 41. `seminar` est construit sur `pstricks`, il est donc très puissant.	
`stmaryrd`	Jeremy GIBBONS, Alan JEFFREY	`latex/styles/stmaryrd.dvi`
	Ce package définit principalement de nouveaux symboles mathématiques, comme $\square$ (`\square`) ou $\llbracket$ et $\rrbracket$ (`\llbrackets` et `\rrbracket`).	
`xspace`	David CARLISLE	`latex/tools/xspace.dvi`
	Ce package permet, lorsqu'on définit une abréviation, de placer une espace supplémentaire si nécessaire. Par exemple, si on définit la raccourci `\L` pour le pas devoir taper `\LaTeX`, il n'y aura par défaut pas d'espace après. Si on ajoute la commande `\xspace` dans la définition, il y aura une espace si le caractère suivant est une lettre ou un chiffre, et pas d'espace si c'est un symbole de ponctuation.	
`xy`	Kristoffer ROSE	`generic/xypic/xyguide.ps`
	Ce package permet de faire des graphes et des diagrammes très variés, avec de nombreuses possibilités de flèches. Son utilisation nécessite une certaine habitude...	
`yhmath`	Yannis HARALAMBOUS	`yhmath.ps`
	Ce package fournit de nouveaux symboles mathématiques, notamment des symboles de « grande taille » comme `\widehat`, `\widetilde` ou `\wideparen`.	

Table des matières

Références

[Bay95a] Benjamin Bayart. *Joli Manuel Pour LaTeX*, Dec. 1995. — LE manuel. Très complet, mais un peu vieillissant. Une mise à jour est prévue depuis bien longtemps... Cette doc reste néanmoins, parmi les docs gratuites, la meilleure.

[Bay95b] Benjamin Bayart. *Joli Manuel Pour LaTeX*, Dec. 1995. Sections 33–40 : Bibliographie. — Une très succinte présentation sur BibTeX.

[Dal99] Partick W. Daly. *A Master Bibliographic Style File*, Mai 1999. — La documentation de makebst, qui permet de créer, assez simplement, un style bibliographique.

[Des00] Bernard Desgraupes. *LaTeX : Apprentissage, Guide et Référence.* Nov. 2000. — Le livre que je préfère. Aussi bien pour les débutants, qui y trouveront une partie « apprentissage », que pour ceux qui veulent approfondir.

[GMS98] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *the LaTeX Companion.* Addison-Wesley, 1998. — Le must, disent certains. Un peu vieillissant, peut-être. Cependant, c'est quand même celui qui est probablement le plus utilisé.

[Gou00] Frédéric Goualard. Manual for the `prosper` class, 2000. — La doc de la classe `prosper`, pour faire des transparents. `prosper` est sans doute la meilleure alternative pour les transparents.

[GR95] Vincent Granet and Jean-Pierre Regourd. *Objectif LaTeX.* Masson, 1995. — Bon, je vais peut-être être violent, mais je trouve que ce livre est un exemple prouvant qu'on peut faire des trucs très moches avec LaTeX. Sinon, il est assez complet, bien pour commencer, à mon avis...

[GRM97] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *the LaTeX Graphics Companion.* Addison-Wesley, Nov. 1997. — Un petit frère de [GMS98], qui explique tout, ou presque, sur les différentes méthodes pour faire des dessins, des graphismes, ... sous LaTeX.

[Her96] Matthieu Herrb. *Une (courte) introduction à LaTeX 2ε*, Oct. 1996. — Une petite doc basique, assez complète sans pour autant entrer dans les détails. Idéal pour commencer, mais pas pour approfondir.

[Lam97] Leslie Lamport. *LaTeX : a Document Preparation System.* Addison-Wesley, Mars 1997. — La documentation originale de LaTeX, par son auteur. Pas forcément ce qui se fait de mieux, mais y a des gens qui préfèrent celui-là.

[LR95] Thomas Lachand-Robert. *La maîtrise de TeX et LaTeX.* Masson, 1995. — Ce livre est beaucoup plus orienté vers TeX que vers LaTeX. Pour approfondir, cependant, c'est pas mal. En français, en plus...

[Mar02a] Nicolas Markey. *Introduction à LaTeX*, Juil. 2002.

[Mar02b] Nicolas Markey. *Tame the BeaST – BibTeX de B à X*, Juil. 2002. — Tout sur les bibliographies sous LaTeX, de la bibliographie simple à la création ou la modification des styles de BibTeX.

[Pat98a] Oren Patashnik. *BibTeXing*, Fev. 1998. — La doc originale de l'auteur. Hélas trop succinte et plus très à jour...

[Pat98b] Oren Patashnik. *BibTeXing*, Fev. 1998. Chap. 5 : Designing BibTeX Styles. — La doc de l'auteur sur les styles bibliographiques de BibTeX. Courte, mais dense.

[RM97] Kristoffer H. Rose and Ross Moore. *XY-pic Reference Manual*, Mai 1997. — Le manuel de référence. Tout ce qu'on peut faire avec le package `xy`. Ça nécessite cependant une certaine quantité de motivation...

[Rol99] Christian Rolland. *LaTeX par la pratique.* O'Reilly, 1999. — Ça doit être le plus utilisé en français. Très complet, agréable à lire... Un bon bouquin.

[Ros97] Kristoffer H. Rose. *XY-pic User's Guide*, Mai 1997. — Le guide du package `xy`. Moins impressionnant que le manuel de références, mais déjà assez fort.

[Sal95] David Salomon. *The Advanced TeXbook.* Springer, 1995. — Euh... Je sais pas, j'ai pas tout compris. C'est vraiment le bouquin le plus impressionant que j'ai vu sur TeX. Tous les détails sont abordés. Vraiment bien, mais difficile à aborder.

[SL91] Raymond Séroul and Silvio Levy. *A Beginner's Book of TEX*. Springer, 1991. — Un livre « pour débutants » sur TEX. C'est un vieux livre, je n'en vois plus trop l'intérêt, même si certains continuent à faire du TEX pur... Enfin, pourquoi pas.

[vZ93] Timothy van Zandt. *PSTricks User's Guide*, Mars 1993. — Le guide d'utilisation de `pstricks`. Intéressant, mais j'ai toujours eu un peu de mal à m'y retrouver...

[Zoo99] Vincent Zoonekyndt. *Les fontes sous LATEXpour les nuls (et les autres)*, Juil. 1999. — Une excellente doc sur la gestion des fnotes par LATEX. Il faut s'accrocher pour comprendre, et ce n'est sans doute pas essentiel de comprendre tous les détails, puisque certains packages permettent de manipuler les fontes sans y mettre les mains.

Commandes et environnements

– W –

– X –

Classes et packages

Index