Initiation à LaTeX

Stéphane PASQUET

1er juillet 2017

Sommaire

1 Un bon départ

1.1 Pourquoi se mettre à LaTeX ?

Beaucoup de personnes ont l'habitude d'utiliser les logiciels de bureautique dits *WYSIWYG* : ils sont très pratiques car on voit ce que l'on tape. Mais ces mêmes personnes peuvent très vite se rendre compte que le rendu des documents est assez classique et, quand on veut créer des documents d'une prestance autre, cela devient très vite compliqué (avec ces mêmes logiciels).

LaTeX est une solution qui permet de mettre en forme vos documents comme aucun logiciel WYSIWYG. On peut ainsi mélanger le texte au graphisme pour de meilleurs résultats.

De plus, avec un document type LaTeX, il suffit de quelques lignes pour changer le style de tout le document, alors qu'avec un traitement de textes ordinaire, si l'on veut changer la forme du document, il faut changer toutes les pages unes à unes, ce qui peut être relativement fastidieux.

La plupart du temps, ce qui empêche ces personnes de passer à LaTeX, c'est la méconnaissance de cet outil. Mais ces personnes ont-elles pensé que les traitements de textes ont énormément de possibilités et qu'elles n'en savaient qu'un faible pourcentage en général ?

Comme tout outil, il faut du temps pour arriver à confectionner des documents qui nous plaisent. Je me propose ici, dans cet ouvrage, d'exposer les bases et plus encore afin de satisfaire la curiosité des personnes qui veulent franchir le pas ...

1.2 La philosophie de LaTeX

Avant de se lancer dans LaTeX, il vous faut comprendre qu'une communauté grandissante s'intéresse de près à ce langage de programmation . Cette communauté est prête à vous aider à tout moment *via* des forums où vous pourrez exposer vos difficultés et où l'on vous répondra toujours avec plaisir en essayant de coller au mieux à vos souhaits.

C'est aussi un milieu où la gratuité prime (au même titre qu'Open Office). Certes, certaines choses sont payantes en LaTeX, mais c'est réellement un très faible pourcentage par rapport à ce qui est disponible.

Un document se créé à l'aide de commandes. Ces commandes sont déjà programmées mais comme il y en a une liste non exhaustive, il vous sera nécessaire de dire quelles commandes vous voulez. Ceci se traduit par l'appel d'`extensions`, communément appelées `packages`. Ce sont des fichiers qui contiennent la définition de certaines commandes. Vous verrez plus

tard qu'une commande existe pour faire appel simplement à ces fichiers, et je vous dirai le nom des fichiers les plus importants pour commencer.

Ainsi, vous voyez que la philosophie LaTeX consiste à séparer le fond de la forme, le « fond » étant ce que vous écrivez et la « forme » le style que vous souhaitez donner à vos documents.

1.3 Installation

1.3.1 Sous Windows

Les *distributions* qui vont permettre de compiler des documents en TeX, et donc en LaTeX, par exemple, s'appellent `MikTeX` et `TexLive`. Vous pouvez installer l'une ou l'autre de ces distributions, mais pas les deux.

Je vous conseille d'installer `MikTeX` car plus pratique à mon goût (ce qui est relatif, je vous l'accorde ...).
Pour se faire, téléchargez la dernière distribution sur la page `http://miktex.org/2.9/setup`, en cliquant sur le bouton « Download » correspondant à « Basic MiKTeX 2.9 Installer ». Le fichier d'installation se téléchargera alors et vous n'aurez plus qu'à double-cliquer dessus pour que la distribution s'installe.

Par défaut, tout sera installé dans le répertoire :

```
C:\Program Files\Miktex 2.9
```

Une fois MikTex installé, je vous conseille de mettre un raccourci vers le `Manager` sur votre bureau ou ailleurs. Le chemin vers le manager est le suivant :

```
C:\Program Files\MiKTeX 2.9\miktex\bin\mo.exe
```

Le manager servira à chaque fois que l'on installera une extension (un `package`) et que l'on voudra rafraîchir la base de données des extensions.

Ceci étant fait, il nous faut maintenant un `éditeur`, ce qui est plus pratique que d'utiliser le bloc note ...
Là, vous avez le choix. Il y a :

- `LyX`, téléchargeable sur le site `http://www.lyx.org/`, qui ressemble à un traitement de textes classique, donc intéressant pour ceux qui sont encore frileux de passer à LaTeX,
- `TeXnicCenter`, sur le site `http://www.texniccenter.org/`,
- `Texmaker`, sur `http://www.xm1math.net/texmaker/index_fr.html`).

Je n'ai nul conseil à vous donner quant à l'éditeur à utiliser car l'appréciation est au grès de chacun.

1.3.2 Sous une distribution Linux ou Unix

L'installation est, dans la quasi totalité des distributions, déjà faite, mais si tel n'est pas le cas, il vous suffit de taper `texlive` dans l'installateur de paquets, et tout se fera de façon auto-

matique. On peut installer MikTex, mais d'après mes derniers essais, cette distribution n'est pas la plus adéquate aux distributions Unix ou Linux.
Il en est de même pour les éditeurs. En plus de ceux cités précédemment, vous pouvez ajouter ici :

- Kile (téléchargeable sur le site http://kile.sourceforge.net/).

Cette liste n'est pas exhaustive.

1.3.3 Installation sous Mac OS

La distribution s'appelle ici MacTeX, téléchargeable ici :

http://mirror.switch.ch/ftp/mirror/tex/systems/mac/mactex/MacTeX.mpkg.zip

Tout se fait de façon automatique ; il suffit juste de cliquer sur « Continuer », « Continue » et « Agree » à chaque fois que ce bouton s'affiche.

1.4 Complément d'installation

1.4.1 Création d'un chemin type

Nous allons le voir ultérieurement, créer un chemin type s'avèrera très utile pour y déposer tous les fichiers que l'on va installer manuellement.

A la racine d'un dique, créez le répertoire :

```
<L>:\texmf\tex\latex\
```

où <L> est la lettre correspondant à l'un de vos disques (par précaution, je vous conseille d'utiliser un disque autre que celui où est installé votre système d'exploitation, au cas où il serait endommagé plus tard ...

Vous le verrez plus tard, il y a pas mal d'extensions, de packages, qui ne peuvent être installés que manuellement, comme les miens par exemple. Dans ce cas, c'est dans ce répertoire qu'il faudra les enregistrer. En effet, cela évite, lorsque l'on met à jour la distribution (MikTeX ou TexLive) d'écraser tous les packages installés manuellement.

Par exemple, si vous souhaitez installer le package pas-cv, qui permet de créer des CV en mode graphique, il vous faudra télécharger le fichier compressé sur mon site (http://www.mathweb.fr), puis décompresser le tout dans le répertoire suivant :

```
<L>:\texmf\tex\latex\pas-cv\
```

Pour respecter l'arborescence LaTeX, vous pouvez aussi stocker les documentations ici :

```
<L>:\texmf\doc\pas-cv\
```

Mais ce n'est pas une obligation (les documentations peuvent être mises où bon vous semble, mais il est toujours bon de savoir où elles se trouvent toutes n'est-ce pas ? Autant les réunir au même endroit ...)

1.4.2 Déclaration du chemin type

Pour les utilisateurs de MikTeX, il faudra mentionner à cette distribution le fait que l'on a créé ce chemin type. Pour se faire, lancer le manager (dont on a créé un raccourci après l'installation), puis cliquez sur l'onglet `Roots`, puis sur le bouton `Add...`. Là, tapez :

```
<L>:\texmf\
```

Cliquez maintenant sur l'onglet `General`, puis sur le bouton `Refresh FNDB`. Cela aura pour effet de rafraîchir la base de données. N'oubliez pas cette dernière étape car elle sera obligatoire à chaque fois que vous aurez quelque chose de nouveau (donc à chaque fois que vous installerez un nouveau package).

Tant que nous y sommes, regardons la partie `Package installation` : je vous conseille de mettre à « Yes » le bouton-choix en face de « Install missing packages on-the-fly » ; cela aura pour effet d'installer automatiquement un package si celui si est manquant (à condition qu'il soit déposé sur un miroir CTAN).

En cliquant sur l'onglet « Packages », on peut changer le répertoire d'où seront téléchargés les packages (bouton `Change...`).

1.5 Les compilations

Il existe plusieurs types de compilations, et vous le verrez en paramétrant votre éditeur. Tout va dépendre des outils que vous utiliserez pour produire vos documents.
Si vous n'utilisez pas de solutions graphiques, donc si vous n'avez que du texte, une compilation avec `PdfLatex` suffira pour créer un document au format PDF. Mais certains packages nécessitent une compilation autre, comme par exemple `LaTeX+dvips+ps2pdf` pour produire là encore un document au format PDF, mais en ayant au préalable créé un document au format postscript (PS).

Par exemple, si vous utilisez la solution graphique TiKZ, une compilation avec Pdflatex suffira.
Si vous utilisez PSTricks, il vous faudra compiler avec la seconde méthode.

Ces deux méthodes ne sont pas les seules car, en plus de ces deux solutions graphiques, il y a `Asymptote`. Quoi qu'il en soit, votre éditeur pourra toujours se paramétrer pour que vous lui ordonniez la compilation que vous souhaitez.

En ce qui concerne ce document, il a été compilé avec PdfLatex car je n'utilise ni PSTrick ni Asymptote, mais TiKZ pour mes dessins.

1.6 Structure d'un document

Nous allons voir ici la structure de base d'un document. Ultérieurement, nous verrons que cette structure peut évoluer.

Un document est essentiellement constitué de trois parties. En effet, pour produire un document LaTeX, il nous faut écrire des lignes qui seront ensuite interprétées à la compilation pour créer le document visuel. Ecrire un document LaTeX, c'est comme programmer.

1.6.1 La classe du document

Une *classe* est une sorte de modèle de référence au document que nous voulons créer. Ce sont en fait des fichiers dont l'extension est `.cls`. Ils sont déposés dans le répertoire :

```
...\tex\latex\
```

A l'origine, les classes les plus utilisées sont les suivantes :

- article.cls
- book.cls
- report.cls
- minimal.cls
- letter.cls

Mais il en existe beaucoup d'autres. Nous verrons, quand nous serons plus expérimentés, comment créer sa propre classe.

Ce qu'il est nécessaire de comprendre, c'est que chaque classe va définir des *commandes* qui pourront être utilisées dans les documents.

Ainsi, tout document doit commencer par :

```
1 \documentclass[<options>]{NomDeLaClasse}
```

Pour ce qui est des options, qui sont facultatives, il est nécessaire de regarder la documentation de la classe que vous souhaitez utiliser.

Pour créer un document avec la classe `article`, on peut par exemple taper :

```
1 \documentclass[a4paper,12pt]{article}
```

La documentation fournira aussi la liste complète des commandes définies par la classe.

1.6.2 Les extensions (packages)

Après avoir renseigné la classe du document, il faut indiquer au compilateur les extensions que l'on souhaite utiliser.
Les packages sont des fichiers ont sont définies d'autres commandes. Je vous conseille de charger au moins les extensions suivantes :

```
\usepackage[frenchb]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
```

Ces extensions permettent un encodage et une mise en forme générale sans que l'on ait de soucis majeur. A noter qu'ici que l'encodage est informé par la deuxième ligne (ici, on encore le document en « latin1 », mais on aurait pu l'encoder en « utf8 »).
Le package « babel » permet de *franciser* le document.
Le package « fontenc » quant à lui, n'est pas obligatoire mais permet tout de même de spécifier à LaTeX l'utilisation du codage de caractères T1, nouvelle norme LaTeX non utilisée par défaut pour des raisons de compatibilité avec les anciens documents LaTeX.

1.6.3 Le corps du document

Après avoir chargé la classe et les extensions, nous voici prêts à commencer le document à proprement parlé. Pour ce faire, le contenu devra être mis entre les balises :

```
\begin{document}

\end{document}
```

1.6.4 Synthèse

Un document aura donc la structure suivante :

```
\documentclass{article}
\usepackage[frenchb]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\begin{document}
Contenu du document
\end{document}
```

Comme je l'ai signalé précédemment, nous verrons plus tard que nous pouvons ajouter une quatrième partie pour les définitions de commandes personnelles.

2 L'essentiel

Nous allons maintenant voir les notions essentielles à la construction de documents LaTeX.

2.1 Le format du document

Nous l'avons vu précédemment, quand on appelle une classe, on peut spécifier le format du document en option (entre crochets).
Pour les classes `article`, `book` et `report`, nous avons le choix entre :

- a4paper (297mm×210mm)
- a5paper (210mm×148mm)
- b5paper (250mm×176mm)
- landscape (210mm×297mm)
- letterpaper (11in×8.5in)
- legalpaper (14in×8.5in)
- executivepaper (10.5in×7.25in)

L'unité « in » est en *inches*.

Si l'on souhaite que notre document soit dans un autre format, on peut utiliser l'extension `geometry`.

```
1 \usepackage[paperwidth=10cm,paperheight=10cm]{geometry}
```

Ici, en appelant cette extension avec les options `paperwidth` (largeur de la page) et `paperheight` (hauteur de la page), on définit la largeur et la hauteur de la page à 10 cm. On peut aussi mettre les dimensions en millimètres (mm) ou en inches (in) ou en points (pt).

2.2 Les marges du document

Comme vous l'aurez peut-être constaté, par défaut, les marges sont assez grandes. Pour les changer, nous allons ici aussi utiliser l'extension `geometry`. Il y a de multiples options donc je vous conseille de jeter un coup d'œil à sa documentation. Dans un cas général, on peut utiliser les options suivantes :

```
1 \usepackage[lmargin=1cm,rmargin=1.5cm,tmargin=2cm,bmargin=2.5cm]
2 {geometry}
```

Ici, « lmargin » désigne la marge de gauche, « rmargin » la marge de droite, « tmargin » celle du haut et « bmargin » celle du bas.

2.3 Les objets LaTeX

Il existe deux principaux types d'objets en LaTeX : les `commandes` et les `environnements`.

Les commandes sont de la forme :

```
\nomdelacommande[<options>]{Contenu1}{Contenu2}...{ContenuN}
```

Les commandes peuvent comporter plusieurs paires d'accolades, mais ne comportent pas toujours d'options.

Les environnements sont de la forme :

```
\begin{nomdelenvironnement}[<options>]{options 2}
Texte de l'environnement.
\end{nomdelenvironnement}
```

Ici encore, les environnements ne comportent pas toujours d'options et il n'est pas obligatoire qu'un environnement ait une paire d'accolades pour des options 2.

Vous vous familiariserez avec ces deux types d'objets en lisant la suite.

2.4 Un document rédigé sur deux colonnes

Si vous souhaitez que tout votre document soit écrit sur deux colonnes, vous pouvez utiliser l'option suivante :

```
\documentclass[twocolumn]{article}
```

Si vous ne souhaitez mettre qu'une partie de votre document sur plusieurs colonnes, vous pouvez faire appel à l'extension `multicol`.

```
\usepackage{multicol}
```

Ainsi, pour mettre sur plusieurs colonnes, on utilisera la syntaxe suivante :

```
\begin{multicols}{3}
\lipsum[1]
\end{multicols}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Pour cet exemple, j'ai fait appel à la commande `\lipsum`, du package du même nom. Remarquez que j'ai mis en option le nombre « 1 » pour ne demander d'afficher qu'un seul paragraphe.
Si on avait voulu mettre deux paragraphes, on aurait tapé :

```
\lipsum[1-2]
```

Pour en revenir aux colonnes, je vous conseille de regarder la documentation de cette extension afin d'explorer les multiples possibilités qui s'offrent à nous.

2.5 Aucune page vide en classe « book »

Si vous avez la curiosité d'utiliser cette classe, vous vous apercevrez que des pages vides s'insèrent entre chaque chapitre si le nombre de pages du chapitre en question est impair.
Cette particularité a été penser dans un but d'impression. Cependant, si on ne destine qu'à l'affichage écran notre document, cela peut quelques fois s'avérer inutile. Pour y remédier, on peut utiliser l'option suivante lors de l'appel de l'extension :

```
\documentclass[openany]{book}
```

2.6 La taille des caractères

Nous l'avons vu précédemment, quand on appelle une classe, on peut spécifier la taille des caractères que l'on souhaite.
Pour les classes `article`, `book` et `report`, nous n'avons le choix qu'entre trois tailles de caractères : 10pt, 11pt et 12pt.

Si l'on veut utiliser des caractères plus grands ou plus petits, une solution consiste à faire appel à l'extension `extsizes`.

```
\usepackage[Xpt]{extsizes}
```

Ici, on met en option (entre crochets) la taille que l'on veut. On a le choix entre 8pt, 9pt, 10pt, 11pt, 12pt, 14pt, 17pt et 20pt.

Cette extension permet donc de fixer la taille de tous les caractères dans le document. Cependant, il est utile de pouvoir changer **ponctuellement** la taille de certains caractères. Le noyau LaTeX contient des commandes permettant de faire cela :

```
\begin{tiny}Ceci est écrit en tiny.\end{tiny}
\begin{scriptsize}Ceci est en scriptsize.\end{scriptsize}
\begin{footnotesize}Ceci est en footnotesize.\end{footnotesize}
\begin{small}Ceci est en small.\end{small}
\begin{normalsize}Ceci est en normalsize.\end{normalsize}
\begin{large}Ceci est en large.\end{large}
\begin{Large}Ceci est en Large.\end{Large}
\begin{LARGE}Ceci est en LARGE.\end{LARGE}
\begin{huge}Ceci est en huge.\end{huge}
\begin{Huge}Ceci est en Huge.\end{Huge}
```

Qui donne :

Ceci est écrit en tiny.
Ceci est en scriptsize.
Ceci est en footnotesize.
Ceci est en small.
Ceci est en normalsize.
Ceci est en large.
Ceci est en Large.
Ceci est en LARGE.
Ceci est en huge.
Ceci est en Huge.

Remarquez ici que le texte que l'on veut mettre en une certaine taille est mis entre deux *balises* (les balises sont de la forme `\begin{...} ... \end{...}`. On peut aussi utiliser la syntaxe suivante :

```
{\Large Mon texte en Large} et maintenant en taille normale.
```

Qui donne :

Mon texte en Large et maintenant en taille normale.

Mettre entre accolades signifie que l'on créé un *groupe*, c'est-à-dire quelque chose qui va se différencier du reste.
On peut donc utiliser les commandes suivantes :

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

On peut donc réaliser l'exemple précédent en tapant :

```
\Large Mon texte en Large \normalsize et maintenant en taille normale.
```

Mon texte en Large et maintenant en taille normale.

Si vous souhaitez écrire plus gros, vous pouvez définir une commande comme ce qui suit. Ceci est à mettre avant la balise `\begin{document}`.

```
\makeatletter
\newcommand\@cclxvipt{50}
\newcommand\HUGE{\@setfontsize\HUGE\@cclxvipt{50}}
\makeatother
```

Ici, on peut changer le nombre « 50 » pour le nombre que vous voulez.
Nous venons de définir la commande \HUGE, ce qui donne :

```
\HUGE Ceci est en HUGE.
```

Ceci est en HUGE.

2.7 Les commentaires et saut de page

Dans un code, il est important d'annoter certaines parties. En LaTeX, les commentaires se font à l'aide du symbole « % » :

```
% Ceci est un commentaire
```

En général, dans tous les éditeurs, les commentaires paraissent en gris.

La commande `\newpage` permet de créer, quant à elle, une nouvelle page.

```
Blablabla.

\newpage

Texte sur une nouvelle page.
```

2.8 Constructions dynamiques

2.8.1 Le sommaire (table des matières)

Quand on rédige un document avec l'une des classes `book` ou `report`, une commande permet de créer automatiquement le sommaire (que l'on peut mettre où l'on veut). Il s'agit de la commande suivante :

```
\tableofcontents
```

commande à insérer où vous voulez construire votre sommaire. Il est nécessaire de compiler deux fois le document pour que la table des matières apparaisse. En effet, la première compilation créera un fichier (dont l'extension est « toc ») dans lequel seront mises les diverses informations à insérer dans le sommaire) et la seconde compilation lira le contenu de ce fichier.
Nous verrons plus tard, quand nous serons grands, une façon de modifier l'apparence par défaut de ce sommaire.

2.8.2 Index

De la même façon que le sommaire, nous pouvons construire un index, c'est-à-dire une liste de mots référencés dans le document.
Si l'on veut un index, il faut y penser avant de commencer à rédiger le document car chaque mot que l'on souhaite insérer dans l'index devra être « balisé » par la commande `\index` :

```
\index{mot ou groupe de mots}
```

Par exemple, on peut taper ceci :

```
Théorème de Pythagore :\index{Théorème de Pythagore} Dans un
triangle rectangle, le carré de l'hypoténuse est égal à la somme
des carrés des deux autres cotés.
```

Théorème de Pythagore : Dans un triangle rectangle, le carré de l'hypoténuse est égal à la somme des carrés des deux autres cotés.

Vous voyez que la variable de la commande `\index` n'apparaît pas directement dans le texte.

Pour construire l'index, il faut avant tout faire appel à l'extension `makeidx`, puis demandé de créer l'index dans le préambule du document (c'est-à-dire avant la balise `\begin{document}` à l'aide de la commande `makeindex` et enfin insérer la commande `printindex` où on veut afficher l'index :

```
\usepackage{makeidx}
\makeindex
```

```
\begin{document}
Blablabla
\printindex
\end{document}
```

Ici aussi, deux compilations sont nécessaires, mais la création de l'index ne se fait malheureusement pas automatiquement. Avec la plupart des éditeurs, une fois que votre document est terminé, vous devrez regarder dans le menu afin de voir une option « makeindex ». Si celle-ci ne fonctionne pas ou est absente, vous devrez aller en mode console de votre système d'exploitation et taper la ligne :

```
makeindex <nomdufichier>.idx
```

Une autre compilation sera nécessaire pour voir apparaître l'index.

2.9 Titre de document, auteur et date

Que ce soit la classe `\article`, `\book` ou `\report`, elles définissent toutes les commandes suivantes :

```
\title{Titre du document}
\author{Auteur(s) du document}
\date{Date du document}
```

Ces commandes sont à mettre en préambule.
A noter que l'extension `babel` définit la commande `\today` qui donne la date de compilation.

Pour affichez toutes ces informations, on utilise la commande `\maketitle` dans le document (et non dans le préambule) :

```
\maketitle
```

En général, on la met en début de document, juste après la balise `\begin{document}`.

```
\documentclass[a4paper]{article}
\title{Reproduction des drosophiles en milieu marécageux}
\author{Albert MOUCHE}
\date{\today}
\begin{document}
\maketitle

Blablabla
\end{document}
```

Cette dernière commande étant définie de façon différente selon la classe utilisée, le rendu ne sera pas le même selon que l'on emploi la classe `book` ou la classe `article`.
De plus, la définition est assez basique. Nous verrons plus tard comment redéfinir cette commande pour l'adapter à vos envies.

2.10 Les styles de caractères

Plutôt que de longs discours, je préfère les exemples :

```
\textbf{Du texte en gras}.\\
\textit{Du texte en italique}.\\
\emph{Du texte italique}.\\
\underline{Du texte souligné}.
```

Du texte en gras.
Du texte en italique.
Du texte italique.
Du texte souligné.

Il y a un inconvénient majeur aux commandes `\textit` et `\underline`. En effet, si la variable est plus longue que la ligne, le texte ne va pas revenir à la ligne ... Ce qui est fort désagréable. C'est la raison pour laquelle la commande `\emph` est mieux pour écrire en italique.

La plupart des éditeurs proposent des raccourcis vers les commandes les plus utilisées. Je ne vais donc pas m'attarder plus longtemps sur celles-ci.

Attention : si vous faites appel au package `ulem` (qui permet d'avoir des soulignements doubles, en pointillés, etc. et qui permet une mise en forme correcte des soulignements, notamment une cassure correcte en fin de ligne, indiquez l'option [normalem] lors de l'appel afin qu'il ne redéfinisse pas la commande `\emph`).

2.11 Notes de bas de page

```
Il semblerait que Pythagore
\footnote{Mathématicien grec
qui, à l'âge de 18 ans, a
participé aux jeux olympiques.}
fut un étrange personnage.
```

Il semblerait que Pythagore [a] fut un étrange personnage.

a. Mathématicien grec qui, à l'âge de 18 ans, a participé aux jeux olympiques.

2.12 Note de marge

```
Théorème de Fermat : Il n'existe pas de nombre entier non nuls
$x$, $y$ et $z$ tels que : \marginpar{\emph{Trop long à démontrer}}
\[ x^n+y^n=z^n\]
dès que $n$ est un entier strictement supérieur à 2.
```

Trop long à démontrer

Théorème de Fermat : Il n'existe pas de nombre entier non nuls x, y et z tels que :

$$x^n + y^n = z^n$$

dès que n est un entier strictement supérieur à 2.

Pour mettre les notes dans l'autre marge, on utilisera dans le préambule la commande `reversemarginpar` :

```
\documentclass[a4paper]{book}
\reversemarginpar{}
\begin{document}
Blablabla.
\end{document}
```

N.B. Si vous avez changé les marges avec le package `geometry`, n'oubliez pas de changer aussi la dimension des marges :

```
\geometry{%
marginparwidth=2cm,% largeur souhaitée
marginparsep=1mm % espace entre la marge et le texte
}
```

2.13 Les caractères spéciaux

Comme nous l'avons vu précédemment, les commentaires sont précédés du symbole « % ». Ainsi, pour écrire se symbole, il faudra utiliser une syntaxe indirecte, comme pour d'autres symboles dont voici une liste (non exhaustive) :

```
\textbackslash % pour afficher le caractère : \
\verb+\+ % pour afficher aussi le caractère : \
\% % pour afficher le caractère : %
\& % pour afficher le caractère : &
\$ % pour afficher le caractère : $
\# % pour afficher le caractère : #
\{ % pour afficher le caractère : {
\} % pour afficher le caractère : }
\_ % pour afficher le caractère : _
\verb+~+ % pour afficher le tilde : ~
\verb+^+ % pour afficher l'accent circonflexe : ^
\verb+@+ % pour afficher l'arobase : @
\oe % un "o" et "e" entrelacés
\ae % un "a" et "e" entrelacés
% etc.
```

2.14 Les références internes

Dans les documents assez volumineux, il est quelques fois utile d'insérer des références internes. Par exemple, vous parlez d'un mathématicien à la page 23 et vous le citez à la page 209. Pour éviter au lecteur de chercher à quelle page il a vu ce nom, vous pouvez insérer une référence à l'aide des commandes :

```
\label{Gauss} % pour définir l'endroit de la référence
\ref{Gauss} % pour afficher la référence
```

Par exemple, j'ai créé une référence permettant de revenir au paragraphe concernant l'installation de MikTex sous Windows (1.3.1). La commande `\ref` affiche les références sous forme de suite de nombres (ici, d'abord le numéro du chapitre, après le numéro de la section et enfin le numéro de la sous-section). On peut cliquer dessus pour aller directement à cette référence.
Comme tout ce qui est dynamique, il faut deux compilations pour que cela s'affiche correctement.

2.15 Liens Internet

Les documents produits sont de plus en plus interactifs. On peut ainsi mettre un lien cliquable vers un site Internet. Pour cela, on fera appel à l'extension `hyperref`, dont je vous encourage à lire la documentation pour connaître toutes les options d'appel. On peut par exemple l'appeler ainsi :

```
\usepackage[colorlinks=true,urlcolor=blue]{hyperref}
```

On pourra alors utiliser les commandes `\url` et `\href` comme sur l'exemple suivant :

```
\url{http://www.mathweb.fr}\\
\href{http://www.mathweb.fr}{Mon site}
```

http://www.mathweb.fr
Mon site

2.16 Les différentes sections

Nous l'avons vu précédemment, on peut construire une table des matières. Mais comment est-ce possible ?
Les entres du sommaire sont tout simplement des *sections* du document. On peut dire que ce sont des paragraphes, sous-paragraphes, etc. Il existe différents niveaux de sections.

2.16.1 Les parties

En utilisant les classes `book` ou `report`, on peut définir des titres de parties :

```
\part{Titre de la partie}
\part*{Titre de la partie}
```

La version étoilée consiste à demander à ce que ce titre ne paraisse pas dans le sommaire.

2.16.2 Les chapitres

En utilisant les classes `book` ou `report`, on peut définir des titres de chapitres :

```
\chapter{Titre du chapitre}
\chapter*{Titre du chapitre non visible dans le sommaire}
```

2.16.3 Les sections

En utilisant les classes `article`, `book` ou `report`, on peut définir des titres de sections :

```
\section{Titre de la section}
\section*{Titre de la section non visible dans le sommaire}
```

2.16.4 Les sous-sections

En utilisant les classes `article`, `book` ou `report`, on peut définir des titres de sous-sections :

```
\subsection{Titre de la sous-section}
\subsection*{Titre de la sous-section non visible dans le sommaire}
```

2.16.5 Les sous-sous-sections

En utilisant les classes `article`, `book` ou `report`, on peut définir des titres de sous-sous-sections :

```
\subsubsection{Titre de la sous-sous-section}
\subsubsection*{Titre de la sous-sous-section non visible
dans le sommaire}
```

2.16.6 Les paragraphes

En utilisant les classes `article`, `book` ou `report`, on peut définir des titres de paragraphes :

```
\paragraph{Titre du paragraphe}
\paragraph*{titre du paragraphe non visible dans le sommaire}
```

2.16.7 Les sous-paragraphes

En utilisant les classes `article`, `book` ou `report`, on peut définir des titres de sous-paragraphes :

```
\subparagraph{Titre du sous-paragraphe}
\subparagraph*{titre du sous-paragraphe non visible dans le sommaire}
```

Les styles par défaut sont très classiques mais il est possible de les redéfinir, nous le verrons plus tard.
A noter que toutes les commandes mettent le texte qui les suit à la ligne sauf `\paragraph` et `\subparagraph` (en version normale et étoilée) où là, le texte est mis à la suite sans saut de ligne.

3 Les mises en forme

3.1 Les commandes de base

3.1.1 Les espacements

```
\hspace{<longueur>} % espacement horizontal
\hspace*{<longueur>} % espace horizontal incompressible
\vspace{<longueur>} % espacement vertical
\vspace*{<longueur>} % espacement vertical incompressible
\skip <longueur> % par exemple : \skip 3mm pour un espace vertical de 3 mm
\smallskip % équivalent à \vskip{\smallskipamount}
\medskip  % équivalent à \vskip{\medskipamount}
\bigskip  % équivalent à \vskip{\bigskipamount}
\\ % le double antislash permet de faire un saut de ligne normal
```

3.1.2 Indentation

L'indentation est l'espace horizontal entre la marge de gauche et le premier mot de chaque paragraphe. Si l'on veut que cet espace soit annulé ponctuellement, on utilisera la commande `\noindent`.

```
Texte avec indentation.

\noindent Texte sans indentation.
```

Si l'on souhaite définir ponctuellement une indentation différente, on pourra utiliser la commande `\parindent` :

```
\parindent 5mm % ici, on fixe à 5mm les indentations suivantes
```

Si, comme dans cet ouvrage, on souhaite définir une indentation nulle, on pourra mettre en préambule ceci :

```
\setlength{\parindent}{0pt}
```

3.2 Les polices de caractères

3.2.1 Les commandes par défaut en mode normal

```
\texttt{Type Writter}\\
\textsc{Small Caps}\\
\textsl{Slanted}\\
\textsf{Du texte}
```

Type Writter
SMALL CAPS
Slanted
Du texte

3.2.2 Les commandes par défaut en mode mathématique

```
$\mathbb{R}$, $\mathbb{Z}$,
$\mathbb{N}$, $\mathbb{C}$\\
$\mathbf{a}$\\
$\mathcal{C}$\\
$\mathfrak{F}$\\
$\textrm{Du texte avec des espaces}$
```

$\mathbb{R}$, $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{C}$
$\mathbf{a}$
$\mathcal{C}$
$\mathfrak{F}$
Du texte avec des espaces

A noter que la fonte `mathfrak` est disponible en faisant appel à l'extension `amsfonts`.

A noter aussi que le résultat obtenu ici avec la commande `\mathcal` n'est pas toujours le même. En effet, pour cet ouvrage, j'utilise l'extension `fourier` qui modifie le résultat. Avec la police par défaut (Computer Modern), on obtient ceci : $\mathcal{C}$

Si l'on utilise la police par défaut et que l'on souhaite faire des majuscules manuscrites, on peut utiliser l'extension `mathrsfs` :

```
...
\usepackage{mathrsfs}
...
\begin{document}
$\mathscr{C}$
\end{document}
```

3.2.3 Sélection d'une fonte

Il existe deux façons de choisir une fonte : soit on la sélectionne pour TOUT le document, et dans ce cas, on « appelle » la fonte en chargeant l'extension (comme, par exemple, `fourier`, `kpfonts`, ou encore `lmodern`), soit on la sélectionne à un endroit bien précis du document. C'est ce que je vais détailler ici.

3.2.3.1 Sélection de la famille

```
\fontfamily{cmdh}\selectfont
Ceci est du texte avec la famille
Dunhill.\\
\fontfamily{familydefault}\selectfont
Là, on revient à la famille par défaut.
```

Ceci est du texte avec la famille Dunhill.
Là, on revient à la famille par défaut.

3.2.3.2 Sélection de la graisse

La commande `\fontseries` permet de fixer la graisse de la fonte à « léger », « médium », « gras » ou « très gras ».

```
\fontseries{l}\selectfont Texte en léger.\\
\fontseries{m}\selectfont Texte en médium.\\
\fontseries{b}\selectfont Texte en gras.\\
\fontseries{bx}\selectfont Texte en très gras.
```

Texte en léger.
Texte en médium.
Texte en gras.
Texte en très gras.

3.2.3.3 Sélection de la forme

La commande `\fontshape` permet de fixer la forme de la fonte à « italique », « normale », « penché » ou « petite majuscule ».

```
\fontshape{it}\selectfont Texte en italique.\\
\fontshape{n}\selectfont Texte en normal.\\
\fontshape{sl}\selectfont Texte penché.\\
\fontseries{sc}\selectfont Texte en petites
 majuscules.
```

Texte en italique.
Texte en normal.
Texte penché.
Texte en petites majuscules.

3.2.3.4 Sélection de l'encodage

La commande `\fontencoding` permet de fixer l'encodage entre « T1 », « U » et « C00 ».

```
\fontencoding{T1}\selectfont Texte en T1.\\
\fontencoding{U}\selectfont Texte en U.
```

Texte en T1.

3.2.3.5 Synthèse

On peut mettre toutes ces dernières commandes à la suite. Il existe deux façons de changer de police :

```
\fontfamily{fonte}\fontseries{graisse}\fontshape{forme}
\fontencoding{encodage}\selectfont
% ou
\usefont{encodage}{fonte}{graisse}{forme}
```

Voici quelques exemples :

```
\usefont{T1}{cmr}{m}{n} Mon texte.
\usefont{T1}{phv}{m}{sc} Mon texte.
\usefont{T1}{ptm}{b}{it} Mon texte.
\usefont{U}{pzd}{m}{n} Ceci est un texte.
\usefont{T1}{anttlc}{bx}{sc} Mon texte.
\usefont{U}{yinit}{m}{n} LA
```

Mon texte.
MON TEXTE.
Mon texte.
✣❋❋❉ ❄▲▼ ◆■ ▼❄▼❄✎
MON TEXTE.

3.2.3.6 Revenir aux valeurs par défaut

```
\fontencoding{\encodingdefault}
\fontfamily{\familydefault}
\fontseries{\seriesdefault}
\fontshape{\shapedefault}
\selectfont
```

3.2.3.7 Installer une nouvelle fonte

Pour regarder les différentes fontes LaTeX disponibles, vous pouvez vous rendre sur le site http://www.tug.dk/FontCatalogue.

Si une police de caractères vous plait, cliquez dessus et cliquez sur le lien qui lui correspond. Par exemple, pour la police « french script », il y a un lien qui point sur la page http://www.ctan.org/tex-archive/fonts/frcursive. Pour installer la fonte, vous devez lire le fichier README qui accompagne chacune.

Pour finir sur les fontes, vous pouvez consulter le site http://zoonek.free.fr/LaTeX/Fontes/fontes.html où une masse d'informations non négligeable est stockée.

3.3 Les lettrines

Pour créer des lettrines, on doit utiliser l'extension `lettrine`.

```
\usepackage{lettrine}
```

On pourra alors obtenir les exemples suivants.

```
\lettrine{C}{eci est la première
phrase}. La première phrase est
en majuscules avec cette commande
et le reste devient automatiquement
en taille normale.
```

CECI EST LA PREMIÈRE PHRASE. La première phrase est en majuscules avec cette commande et le reste devient automatiquement en taille normale.

```
\lettrine[lines=3]{\usefont{U}
{yinit}{m}{n}\small C}{eci est la
première phrase}. La première phrase
est  en majuscules avec cette commande
et le reste devient automatiquement
en taille normale.
```

ECI EST LA PREMIÈRE PHRASE. La première phrase est en majuscules avec cette commande et le reste devient automatiquement en taille normale.

En utilisant l'extension `oldgerm`, la lettrine peut-être en gothique :

```
\lettrine{\textgoth{C}}{eci est la
première phrase}. La première phrase
est  en majuscules avec cette commande
et le reste devient automatiquement
en taille normale.
```

ECI EST LA PREMIÈRE PHRASE. La première phrase est en majuscules avec cette commande et le reste devient automatiquement en taille normale.

3.4 Les boîtes

Avant tout, il faut savoir qu'il existe beaucoup d'extensions qui concernent les boîtes. Les commandes suivantes sont les plus basiques, mais si vous souhaitez aller plus loin, n'hésitez pas à chercher sur CTAN des packages concernant les boîtes.

3.4.1 Boîte simple

C'est la commande `\makebox` qui va permettre de créer des boîtes simples :

```
\makebox[longueur][position du contenu]{contenu}
```

Voici ici un exemple d'utilisation :

```
\makebox[.3\linewidth][l]{Gauche}
\makebox[.3\linewidth][c]{Centre}
\makebox[.3\linewidth][r]{droit}
```

Gauche Centre droit

J'ai ici construit 3 boîtes à la suite, de longueur égale au tiers de la ligne de texte. Dans la première, le texte est aligné à gauche, dans la seconde, au centre et dans la troisième à droite.

On peut aussi utiliser la commande \parbox :

```
\parbox[<t|c|b> externe][<hauteur>][<t|c|b> interne]{<largeur>}{Contenu}
```

3.4.2 Boîte encadrée

C'est ici la commande \fbox (abréviation de *framed box*) qui va permettre d'encadrer du contenu.

```
Ceci est une \fbox{phrase} où le mot
\og phrase \fg\ est encadré.
```

Ceci est une phrase où le mot « phrase » est encadré.

3.4.3 Boîte surlignée

Ici, c'est la commande \colorbox qui va nous aider :

```
Ceci est une \colorbox{yellow}{phrase}
où le mot \og phrase \fg\ est surligné.
```

Ceci est une phrase où le mot « phrase » est surligné.

3.4.4 Boîte encadrée et surlignée

Un mélange des deux commandes précédentes fournira la commande \fcolorbox :

```
Ceci est une
\fcolorbox{red}{yellow}{phrase}
où le mot \og phrase \fg\ est encadré
et surligné.
```

Ceci est une phrase où le mot « phrase » est surligné.

La commande \fboxrule stocke l'épaisseur de la bordure. Quant à la séparation du cadre et du contenu, c'est stocké dans la commande \fboxsep.

```
\setlength{\fboxrule}{3pt}
\setlength{\fboxsep}{2pt}
Ceci est une
\fcolorbox{red}{yellow}{phrase}
où le mot \og phrase \fg\ est encadré
et surligné.
```

Ceci est une phrase où le mot « phrase » est surligné.

3.4.5 Boîte décalée

Par défaut, les boîtes sont alignées par rapport à la ligne de base, mais quelques fois, il est utile de pouvoir les décaler verticalement. Cela peut se faire avec la commande `\raisebox` :

```
\raisebox{0pt}[0pt][0pt]
{\Large\bfseries Aaaa
\raisebox{-.3ex}{a}%
\raisebox{-.7ex}{aa}
\raisebox{-1.2ex}{r}%
\raisebox{-2.2ex}{g}
\raisebox{-4.5ex}{h}}
cria-t-il, mais la ligne
suivante ne remarqua pas qu'une
chose horrible lui était arrivée.
```

Aaaa **a**a**a** **r** cria-t-il, mais la ligne s**u**ivante ne remarqua pas qu'une **g**hose horrible **h** lui était arrivée.

3.4.6 Texte trop long dans une boîte

Il existe un inconvénient majeur aux boîtes : si le texte est trop long, la césure ne se fait pas. Regardons cet exemple :

```
\fcolorbox{red}{blue!10}{Ceci est un si long texte qu'il ne rentre
réellement pas dans une boîte si on ne fait rien pour y remédier}
```

Ceci est un si long texte qu'il ne rentre réellement pas dans une boîte si on ne fait rien pour y remédier

Ainsi, il nous faut y remédier et c'est la commande `\parbox` qui va nous aider :

```
\fcolorbox{red}{blue!10}{
\parbox{\textwidth}{
Nous voici à l'aube d'une période
où la richesse n'est plus qu'un
lointain souvenir.
Le roi a décidé d'être plus riche
et d'ôter à ses sujets le peu
d'argent qu'ils possédaient.
}}
```

Nous voici à l'aube d'une période où la richesse n'est plus qu'un lointain souvenir. Le roi a décidé d'être plus riche et d'ôter à ses sujets le peu d'argent qu'ils possédaient.

3.4.7 Boîtes plus sophistiquées

A l'aide de l'extension `fancybox`, nous avons d'autres commandes :

```
\shadowbox{Une ombre}
\doublebox{Un cadre dédoublé}
\ovalbox{Un cadre ovale}
\Ovalbox{Un cadre ovale en gras}
```

Une ombre

Un cadre dédoublé

Un cadre ovale

Un cadre ovale en gras

Je vous encourage à regarder la documentation de ce package pour connaître tous les paramétrages possibles.

Nous avons vu ici différentes méthodes pour encadrer. Ces solutions ne nécessitent pas de solution graphique. Cela sous-entend donc de ma part qu'avec une solution graphique (metapost, PSTricks, TiKZ, Asymptote, ...), on peut définir des cadres bien plus évolués (comme par exemple ceux que vous pouvez voir à chaque fois que je donne un exemple : imitation d'une feuille déchirée, obtenue avec TiKZ).
Mais je ne développerai pas les solutions graphiques ici car il existe énormément de possibilités, propres à chaque solution graphique. Ainsi, il vous faudra vous pencher vous-même sur la solution graphique que vous voulez utiliser pour en connaître tous les rouages.

3.4.8 Sauvegarder une boîte

Dans certains cas, il sera avantageux de sauvegarder une boîte. Pour se faire, nous avons à notre disposition les commandes `newsavebox`, `savebox` et `usebox`.

```
\newsavebox{<nom de la boîte>}
\savebox{<nom de la boîte>}{Contenu de la boîte}
\usebox{<nom de la boîte>}
```

Voyons un exemple :

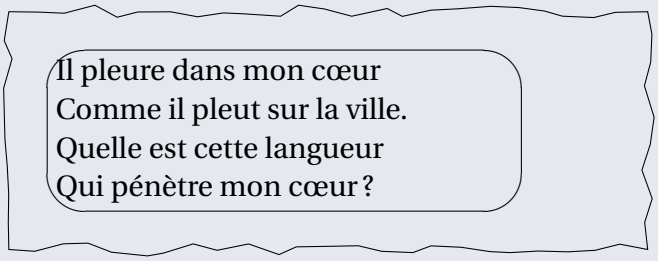
```
\newsavebox{\strophe}
\savebox{\strophe}{%
\shadowbox{%
\begin{minipage}{0.8\linewidth}
Il pleure dans mon c\oe ur\\
Comme il pleut sur la ville.\\
Quelle est cette langueur\\
Qui pénètre mon c\oe ur ?
\end{minipage}
}}
\usebox{\strophe}
```

Il pleure dans mon cœur
Comme il pleut sur la ville.
Quelle est cette langueur
Qui pénètre mon cœur ?

Imaginons que l'on doivent répéter cette mise en forme un certain nombre de fois dans le document ... Alors, il est plus facile de taper `\usebox{\strophe}` que tout le code non ?

Une autre façon de faire est d'utiliser l'environnement `lrbox` :

```
\newsavebox{\maboite}
\begin{lrbox}{\maboite}
\ovalbox{%
\begin{minipage}{0.8\linewidth}
Il pleure dans mon c\oe ur\\
Comme il pleut sur la ville.\\
Quelle est cette langueur\\
Qui pénètre mon c\oe ur ?
\end{minipage}
}
\end{lrbox}
\usebox{\maboite}
```



N.B. Nous avons dû ici utiliser un environnement (`minipage`) pour que le texte ne déborde pas du cadre. Nous allons aborder cet environnement dans le paragraphe suivant.

3.5 Les minipages

Il existe un environnement `minipage` qui permet de fractionner une page en plusieurs parties.

```
\begin{minipage}[<alignement externe : t|m|b>]
                [<hauteur>]
                [alignement interne : t|m|b>]
                {<largeur>}
Contenu de la minipage.
\end{minipage}
```

Pour l'alignement externe et interne, les options sont les initiales de *top*, *middle* et *bottom*. Elles servent à positionner verticalement le minipage par rapport au niveau de la ligne. Voyons de suite un exemple pour visualiser :

```
Ligne de référence.
\fbox{\begin{minipage}[t]{.25\linewidth}
Ceci est un exemple de minipage avec un alignement vertical relatif
à la ligne fixé en haut (t = top).
\end{minipage}}
\hspace{0.5cm} % pour espacer les minipages
\fbox{\begin{minipage}[m]{.25\linewidth}
Ceci est un exemple de minipage avec un alignement vertical relatif
à la ligne fixé au milieu (m = middle).
\end{minipage}}
\hspace{0.5cm} % pour espacer les minipages
\fbox{\begin{minipage}[b]{.25\linewidth}
Ceci est un exemple de minipage avec un alignement vertical relatif
à la ligne fixé en bas (b = bottom).
\end{minipage}}
```

Ligne de référence. Ceci est un exemple de minipage avec un alignement vertical relatif à la ligne fixé en haut (t = top).

Ceci est un exemple de minipage avec un alignement vertical relatif à la ligne fixé au milieu (m = middle).

Ceci est un exemple de minipage avec un alignement vertical relatif à la ligne fixé en bas (b = bottom).

Voyons un deuxième exemple :

```
\begin{minipage}[t][][t]{.45\linewidth}
\lipsum[1]
\end{minipage}
\hspace*{0.1\textwidth}
\begin{minipage}[t][][t]{.45\linewidth}
\lipsum[2]
\end{minipage}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3.6 Les couleurs

Pour les couleurs, il y a une extension incontournable : `xcolor`. Cette extension peut être appelée avec des options (je vous conseille de regarder la documentation). Mais deux options vont nous intéresser essentiellement : `table` (qui va permettre de mettre de la couleur dans les tableaux, nous y viendrons bientôt) et le nom de la table de couleurs à charger. Il y a le choix entre `dvipsnames`, `dvipsnames*`, `svgnames`, `svgnames*`, `x11names` et `x11names*`. Si aucune option n'est indiquée lors de l'appel de cette extension, il y a tout de même des couleurs de base. Comprenez bien que les dernières options chargent des *noms de couleurs*.

Voici le noms des couleurs disponibles en fonction de l'option informée.

3.6.1 xcolor sans option

black	darkgray	lime	pink	violet
blue	gray	magenta	purple	white
brown	green	olive	red	yellow
cyan	lightgray	orange	teal	

3.6.2 xcolor avec option

Vous trouverez à la page 38 de la documentation de cette extension (http://mirrors.ircam.fr/pub/CTAN/macros/latex/contrib/xcolor/xcolor.pdf) tous les noms.

3.6.3 Utilisation des couleurs

On change de couleur avec la commande `color` ou `textcolor`.

```
\textcolor{Indigo}{Texte en Indigo.}
\color{Coral}Tout ce qui suit sera en Coral.
```

Texte en Indigo. Tout ce qui suit sera en Coral.

3.6.4 Définir une couleur

Si vous souhaitez créer votre propre couleur, voici plusieurs commandes :

```
\definecolor{mycolorA}{rgb}{0.2,0.8,0.4}
\definecolor{mycolorB}{cmyk}{0.2,.3,.4,0.5}
\definecolor{mygray}{named}{mycolorB}
\color{mycolorA}Première couleur\\
\color{mycolorB}Seconde couleur\\
\color{mygray}Mygray
```

Première couleur
Seconde couleur
Mygray

Regardez la documentation pour les fonctions plus élaborées.

3.6.5 Couleur de page

On a la possibilité de mettre toutes les pages d'un document en une certaine couleur à l'aide de la commande `\pagecolor`.

```
\pagecolor{gray}
% A partir d'ici, toutes les pages seront grises jusqu'à ...
\pagecolor{white}
```

3.6.6 Souligner en couleur

Pour souligner en une autre couleur que la couleur courante, on va utiliser l'extension `ulem`.

```
\documentclass{article}
...
\usepackage{ulem}

\newcommand{\colorUL}[1][black]
{%
\bgroup%
\ifdim\ULdepth=\maxdimen
  \settodepth\ULdepth{(j}%
  \advance\ULdepth.4pt
\fi
\markoverwith{\kern0em\vtop{%
\kern\ULdepth {%
\color{#1}\hrule width .4em}}%
\kern0em}\ULon}

\newcommand{\colorULdotted}[1][black]
{%
\bgroup%
\ifdim\ULdepth=\maxdimen
  \settodepth\ULdepth{(j}%
  \advance\ULdepth.4pt
\fi
\markoverwith{\kern0em\vtop{%
\kern\ULdepth {%
\color{#1}\hrule width .4em}}
\kern0em}\ULon}
\begin{document}
\colorUL[red]{Blablabla}
\vskip 5mm
\colorULdotted[blue]
{Blablablabla blablablabla blabla}
\end{document}
```

Blablabla

Blablablabla blablablabla blabla

3.7 Écrire sur la même ligne à droite et au centre

En mathématiques, certaines démonstrations peuvent se terminer par un symbole (la plupart du temps, c'est un petit carré plein noir). Ce symbole est en général placé sur la dernière ligne, à droite. Pour se faire, on peut utiliser cette méthode :

```
Voici une phrase.\hfill $\blacksquare$
```

Voici une phrase. ■

Pour insérer quelque chose au centre, on utilisera :

```
A gauche.\hfil Je suis centré
```

A gauche. Je suis centré

3.8 Les tableaux

Les tableaux se construisent à l'aide de l'environnement `tabular`. Cet environnement comporte quelques options.

3.8.1 Un tableau simple sans bordure

```
\begin{tabular}{lcrlcr}
1 & 2 & 3 & l & c & r\\
4 & 5 & 6 & L & C & R\\
7 & 8 & 9 &   &   &
\end{tabular}
```

1	2	3	l	c	r
4	5	6	L	C	R
7	8	9			

Les options mises entre accolades (« lcrlcr ») informent le nombre de colonnes du tableau (ici, 6 lettres dont 6 colonnes) et la position du texte dans chaque colonne : dans la première et quatrième colonne, le texte est aligné à gauche (« l » pour « left »), dans la seconde et cinquième colonne, le texte sera aligné au centre et dans les autres, il sera aligné à droite (« r » pour « right »).
Il existe aussi les options :

- « p{<largeur>} » (la colonne sera de largeur fixée et le texte sera positionné en haut de la cellule),
- « m{<largeur>} » (la colonne sera de largeur fixée et le texte sera centré verticalement dans la cellule),
- « b{<largeur>} » (la colonne sera de largeur fixée et le texte sera positionné en bas de la cellule).

L'inconvénient de ces options est que l'on perd l'alignement horizontal, mais il y a un moyen de le retrouver (voir paragraphe 1.7.3).

Pour passer d'une case à l'autre, on insère le caractère « & » et pour passer d'une ligne à une autre, on insère le double antislash.

Si l'alignement est identique pour chaque colonne, on peut utiliser une écriture simplifier pour construire le tableau :

```
\begin{tabular}{*6{c}}
1 & 2 & 3 & c & c & c\\
4 & 5 & 6 & C & C & C\\
7 & 8 & 9 &   &   &
\end{tabular}
```

1	2	3	l	c	r
4	5	6	C	C	C
7	8	9			

Ici, « *6 » signifie qu'il y a 6 colonnes et ce qui suit entre accolades désigne le formatage des cellules (ici, tous les alignements seront centrés).

Si l'on veut fixer une largeur pour une ou plusieurs colonnes, on utilisera la syntaxe suivante :

```
\begin{tabular}{*6{p{7mm}}}
1 & 2 & 3 & 4 & 5 & 6\\
4 & 5 & 6 & 7 & 8 & 9\\
7 & 8 & 9 & 10 & 11  & 12
\end{tabular}
```

1	2	3	4	5	6
4	5	6	7	8	9
7	8	9	10	11	12

Ici, nous avons fixé la largeur de chaque colonne à 7 millimètres.

Maintenant, penchons-nous sur le "prescripteur" `@` : il ne sert pas beaucoup sauf quand on veut aligner correctement les uns en dessous des autres des nombres décimaux par exemple, de sorte à ce que la virgule soit au même niveau horizontal.

```
\begin{tabular}{r@{,}l}
12 & 0121\\
1 & 18\\
159 & 0012
\end{tabular}
```

12	0121
1	18
159	0012

Comme vous pourrez le constater, quand on construit un tableau avec l'environnement `tabular`, sa largeur s'adapte à son contenu. Si l'on veut que le tableau ait une largeur précise, on préfèrera utiliser l'environnement `tabular*`. Cependant, il y a quelques contraintes pour que le tableau occupe réellement la largeur que l'on veut. En effet, regardons le code suivant :

```
\begin{tabular*}{\textwidth}{*3{c}}
Case 1.1 & Case 1.2 & Case 1.3\\
Case 2.1 & Case 2.2 & Case 2.3
\end{tabular*}
```

Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3

Nous voyons que l'on doit avant tout informer la largeur souhaitée (ici, « \textwidth » signifie que l'on veut un tableau dont la largeur sera égale à celle du texte) mais que le résultat n'est pas vraiment ce que l'on attendait.
Pour y remédier, on utilisera ce code :

```
\begin{tabular*}{\textwidth}
{@{\extracolsep{\fill}}*3{c}}
Case 1.1 & Case 1.2 & Case 1.3\\
Case 2.1 & Case 2.2 & Case 2.3
\end{tabular*}
```

Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3

L'ajout de l'écriture « `@{\extracolsep{\fill}}` » permet de calculer automatiquement la largeur des colonnes. Le problème est que le formatage n'est plus respecté (il n'y a plus d'alignement centré).

L'extension `tabularx` met à notre disposition l'environnement `tabularx` qui calcule automatiquement la largeur des colonnes en fonction de la largeur totale du tableau que l'on veut et respecte le formatage que l'on veut.

```
\begin{tabularx}{\textwidth}
{*3{>{\centering\arraybackslash}X}}
Case 1.1 & Case 1.2 & Case 1.3\\
Case 2.1 & Case 2.2 & Case 2.3
\end{tabularx}
```

Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3

Notez ici que nous avons eu recours à une écriture que nous n'avions jamais croisé : `>{\centering\arraybackslash}X`. Le « X » désigne la largeur inconnue de chaque colonne (calculée automatiquement en fonction de la largeur voulue du tableau) et ce qui précède stipule que l'on veut un alignement centré.

Pour plus de détails, reportez-vous à la documentation de cette extension.

A noter que certains utilisateurs recommandent de charger l'extension `array` afin d'avoir une typographie des tableaux correcte.

3.8.2 Un tableau avec des lignes

En général, quand on veut construire un tableau, c'est pour mettre des séparations.
Les traits verticaux seront dessinés entre deux colonnes s'il y a le caractère « | » dans la déclaration du formatage. Quant à la séparation horizontale entre deux lignes, elle est dessinée avec la commande `\hline`. Sachez que l'on peut mettre plusieurs caractères à la suite pour dessiner plusieurs traits.

```
\begin{tabular}{|*3{|c}||}
\hline
Case 1.1 & Case 1.2 & Case 1.3\\
\hline
Case 2.1 & Case 2.2 & Case 2.3\\
\hline
\end{tabular}
```

Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3

Si l'on avait voulu ne tirer un trait horizontal que sur la largeur des colonnes 1 et 2 au milieu de notre tableau, on aurait utilisé la commande `\cline`.

```
\begin{tabular}{|*3{|c}||}
\hline
Case 1.1 & Case 1.2 & Case 1.3\\
\cline{1-2}
Case 2.1 & Case 2.2 & Case 2.3\\
\hline
\end{tabular}
```

Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3

3.8.3 Définir un formatage de colonne

Comme nous l'avons vu précédemment, en mettant une option « p{<largeur>} », on perd l'information d'alignement horizontale des cellules. On peut alors utiliser la commande `\newcolumntype` :

```
\newcolumntype{M}[1]{>{\raggedright}m{#1}}
\begin{tabular}{|p{1cm}|M{2.5cm}|p{1cm}}
\hline
Case 1.1 & Case 1.2 & Case 1.3\\
\cline{1-2}
Case 2.1 & Case 2.2 & Case 2.3\\
\hline
\end{tabular}
```

Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3

N.B. La commande `\newcolumntype` est définie dans l'extension `array` ; il faudra donc charger cette extension, ou `tabularx` (qui charge `array`) pour l'utiliser.

3.8.4 Fusionner des cellules

La commande `\multicolumn` sert à fusionner des cellules. Voici un exemple d'utilisation.

```
\begin{tabular}{*4{|c}|}
\hline
\multicolumn{4}{|c|}{\scshape Titre}
\\ \hline
\multicolumn{2}{|c|}
{Cases 1.1 \& 1.2} & Case 1.3 &
Case 1.4\\ \hline
Case 2.1 &
\multicolumn{3}{c|}{Cases 2.2, 2.3
\& 2.4}\\ \hline
\end{tabular}
```

TITRE		
Cases 1.1 & 1.2	Case 1.3	Case 1.4
Case 2.1	Cases 2.2, 2.3 & 2.4	

La syntaxe est : `\multicolumn{nombre de cellules à fusionner}{formatage}{contenu}`.

Si l'on veut fusionner des lignes, il faut charger l'extension `\multirow`.

```
\begin{tabular}{||*5{c|}|}
\hline
\multicolumn{5}{||c||}{TITRE} \\
\hline
\hline
\multicolumn{3}{||c|}{1, 2 et 3} &
4 & 5 \\ \hline
\multirow{2}{*}{Fusion!} & 7 & 8 &
9 & 10 \\
\cline{2-5}
& 6 & 11 & 12 & 13 \\
\hline
\end{tabular}
```

TITRE				
1, 2 et 3			4	5
Fusion!	7	8	9	10
	6	11	12	13

3.8.5 Un tableau avec des couleurs

Pour insérer des couleurs dans un tableau, il faut appeler l'extension `xcolor` avec l'option `table` :

```
\usepackage[table]{xcolor}
```

Cela permet d'avoir les commandes `\arrayrulecolor`, `\cellcolor`,`\columncolor` et `\rowcolor`.

```
\arrayrulecolor{red}
\begin{tabular}
{|*4{c|}>{\columncolor{cyan!20!white}}c|}
\hline
\multicolumn{5}{|c|}
{\cellcolor{brown}\color{white}TITRE} \\
\hline\hline
\multicolumn{3}{|c|}{1, 2 et 3}
& 4 & 5 \\
\hline
\multirow{2}{*}{Fusion!}
& 7 & 8 & 9 & 10 \\
\cline{2-5}
& 6 & 11 & 12 & 13 \\
\hline
\rowcolor{green!20!white}
14 & 15 & 16 & 17 & 19 \\
\hline
\end{tabular}
```

TITRE				
1, 2 et 3			4	5
Fusion!	7	8	9	10
	6	11	12	13
14	15	16	17	19

N.B. Il se peut que vous ne voyiez pas directement les traits lorsque vous mettez des cellules en couleurs. Cependant, ils existent bel et bien ; pour les voir, il suffit de faire un zoom sur le tableau.

3.8.6 De longs tableaux

Dans certains documents, les tableaux peuvent commencer en bas de page et il se peut que vous ayez envie qu'ils continuent sur la page suivante. Pour cela, on utilisera l'environnement `longtable`. La syntaxe est la même que l'environnement `tabular` :

```
\begin{longtable}{|*3{|c}|}
\hline \rowcolor{gray!20}
Titre 1 & Titre 2 & Titre 3\\ \hline\endhead
Case 1.1 & Case 1.2 & Case 1.3\\ \hline
Case 2.1 & Case 2.2 & Case 2.3\\ \hline
Case 3.1 & Case 3.2 & Case 3.3\\ \hline
Case 4.1 & Case 4.2 & Case 4.3\\ \hline
Case 5.1 & Case 5.2 & Case 5.3\\ \hline
Case 6.1 & Case 6.2 & Case 6.3\\ \hline
```

```
Case 7.1 & Case 7.2 & Case 7.3\\ \hline
\end{longtable}
```

Titre 1	Titre 2	Titre 3
Case 1.1	Case 1.2	Case 1.3
Case 2.1	Case 2.2	Case 2.3
Case 3.1	Case 3.2	Case 3.3
Case 4.1	Case 4.2	Case 4.3
Case 5.1	Case 5.2	Case 5.3
Case 6.1	Case 6.2	Case 6.3
Case 7.1	Case 7.2	Case 7.3

N.B. Notez que l'environnement `longtable` centre automatiquement le tableau.

Pour plus de détails, consultez la documentation.

3.8.7 Définir la hauteur de chaque ligne

La hauteur des lignes d'un tableau se change à l'aide de la commande `\arraystretch` (en fait, il s'agit ici d'une variable dont on va changer la valeur).

```
\begin{tabular}{|c|}
\hline Case 1.1\\ \hline
\end{tabular}
\renewcommand{\arraystretch}{2}
\begin{tabular}{|c|}
\hline Case 1.1\\ \hline
\end{tabular}
```

Case 1.1

Case 1.1

N.B. Il n'est pas possible, par cette commande, de changer la hauteur d'une ligne en plein milieu d'un tableau (par exemple, on ne peut pas avoir une première ligne d'une certaine hauteur et la suivante d'une autre). Pour cela, on utilisera `\parbox`.

```
\begin{tabular}{|c|}
\hline Case 1.1\\ \hline
\parbox[c][2cm][c]{15mm}{\centering Case 2.1} \\
\hline
\end{tabular}
```

Case 1.1

Case 2.1

3.8.8 Un tableau avec des colonnes séparées

Pour faire cela, on va charger l'extension `hhline` puis on va utiliser la commande `\hhline`.

```
\begin{tabular}{|*3{m{.25\textwidth}|}}
\hhline{|-|~|-|}
\cellcolor{gray!20}\centering\bfseries\sffamily
Cellule 1 & &
\cellcolor{gray!20}\centering\bfseries\sffamily
Cellule 3
\tabularnewline
\hhline{|-|~|-|}
\centering\bfseries\sffamily Cellule 4
& &
\centering\bfseries\sffamily Cellule 6
\tabularnewline
\hhline{|-|~|-|}
\multicolumn{1}{c}{}
& &
\centering Cellule 9
\tabularnewline
\hhline{~~|-|}
\end{tabular}
```

Cellule 1		Cellule 3
Cellule 4		Cellule 6
		Cellule 9

3.9 Les images

Pour insérer des images, on va charger l'extension `graphicx`.

3.9.1 Insérer une image

On va ici utiliser la commande `\includegraphics` :

```
\includegraphics[scale=<echelle>]{<image>}
```

Par exemple :

```
\includegraphics[scale=0.5]{gauss.jpg}
```

3.9.2 Mettre une légende

On va pouvoir faire cela avec la commande `\caption` en mettant l'image dans l'environnement `figure`, qui admet une option de position :

- `h` (= *here*), pour mettre le contenu à l'endroit exact où est insérer l'environnement.
- `b` (= *bottom*), pour mettre le contenu en bas de la page courante.
- `t` (= *top*), pour mettre le contenu en haut de la page courante.

```
\includegraphics[scale=0.5]{gauss.jpg}
\caption{K.F. gauss}
\end{figure}
```

FIGURE 3.1 – K.F. gauss

En mettant les images, et même tout ce que l'on veut d'ailleurs, on pourra construire une liste des *figures*, à l'aide de la commande `\listoffigures` (deux compilations seront alors nécessaires).

3.9.3 Les transformations

Afin de pouvoir appliquer des transformations aux images, il faut charger l'extension `geometry`.

3.9.3.1 Rotation

`\rotatebox`

```
\rotatebox{30}
{%
\includegraphics[scale=0.5]{gauss.jpg}
}
```

3.9.3.2 Symétrie axiale

`\reflectbox`

```
\includegraphics[scale=0.5]{gauss.jpg}
\reflectbox
{%
\includegraphics[scale=0.5]{gauss.jpg}
}
```

3.9.3.3 Homothétie

On peut utiliser la commande `\scalebox`, mais ici, elle ne sert à rien vu que l'échelle peut-être mise directement avec l'option `scale`.

3.10 Insertion de documents

Nous venons de voir qu'il était possible d'insérer des images dans un document. Nous allons voir maintenant comment insérer ou faire appel à d'autres types de documents.

3.10.1 Insérer un document PDF

On va pouvoir insérer des documents PDF à l'aide de l'extension `pdfpages` et de sa commande `\includepdf`. Pour cela, on écrira, par exemple :

```
\documentclass{article}
\usepackage{pdfpages}
\begin{document}
\includepdf{mondocument.pdf}
\end{document}
```

Dans ce cas, le document *mondocument.pdf* sera inséré après la page courante.

Je vous encourage à regarder la documentation de cette extension pour connaître toutes les options ; par exemple, si l'on veut insérer uniquement quelques pages d'un document externe, on pourra écrire :

```
% insérer uniquement les pages 1, 2, 5 et 7
\includepdf[pages={1,2,5,7}]{mondocument.pdf}
% insère la page 1 et toutes celles compris entre 5 et 10 comprises
\includepdf[pages={1,{5-10}}]{mondocument.pdf}
% Insére la page 1 puis une page vide puis la page 8
\includepdf[pages={1,{},8}]{mondocument.pdf}
% Insère la page 1 du document avec une échelle de 1/2
\includepdf[pages=1,nup=2x1]{mondocument.pdf}
```

Cette extension n'est faite que pour insérer des PDF entre les pages, et en aucun cas pour insérer un document en plein milieu d'un texte.
Par exemple, si nous disposons d'un document PDF de dimensions 5×5cm représentant un dessin quelconque, pour l'insérer, il faudra utiliser la commande `\includegraphics`.

3.10.2 Faire appel à un autre fichier TEX

Lorsque l'on écrit un document très long, composé par exemple de multiples chapitres, il est fortement recommandé de ne pas tout écrire dans un seul document. En l'occurrence, il est plus intéressant de faire un document par chapitre et de faire appel à tous ces documents dans un document principal, appelé aussi *document maître*. On utilise pour cela la commande `\input`. On peut même regrouper le chargement des extension dans un seul fichier.

```
\documentclass[a4paper,12pt]{book}
% appelle le fichier "extensions.tex" où tous les pacjages sont chargés
```

```
\input{extensions}
% le fichier commandes.tex peut regrouper toutes les commandes personnelles
\input{commandes}
% le fichier styles.tex peut contenir toutes les redéfinissions de styles
% et couleurs
\input{styles}
\begin{document}
\input{chapitre1} % appelle le document "chapitre1.tex"
\input{chapitre2} % appelle le document "chapitre2.tex"
...
\input{chapitre85}% appelle le document "chapitre85.tex"
\end{document}
```

Cette démarche est assez importante car elle permet de dissocier le fond de la forme en séparant les fichiers qui définissent les styles (par exemple, la façon dont on écrit les titres, les couleurs personnelles, les en-têtes et pieds de page, ...), les commandes, qui chargent les packages, etc.
C'est aussi très pratique quand on veut justement changer un paramètre (par exemple, la couleur du titre des sections).

On peut aussi utiliser la commande `\include`. Elle fait la même chose mais elle offre la possibilité de n'insérer que quelques fichiers si l'on veut (avec la commande `\includeonly`), comme le montre l'exemple suivant :

```
\documentclass[a4paper,12pt,openany]{book}
\input{packages}
\input{parametres}
\input{commandes}
\includeonly{02,03} % seuls les fichiers 02.tex et 03.tex seront inclus
\begin{document}
\include{01}
\include{02}
\include{03}
\end{document}
```

3.11 Les listes

3.11.1 Listes verticales

Il existe plusieurs environnements pour créer de telles listes.

3.11.1.1 L'environnement itemize

```
\begin{itemize}
\item Item 1
\item Item 2
\end{itemize}
```

- Item 1
- Item 2

Dans cet environnement, les items sont précédés d'un tiret, et les listes commencent en début de ligne (sans indentation).

On peut insérer des listes dans des listes de la façon suivante :

```
\begin{itemize}
\item Item 1
\begin{itemize}
\item Item 1.1
\begin{itemize}
\item Item 1.1.1
\end{itemize}
\end{itemize}
\end{itemize}
```

- Item 1
  - → Item 1.1
    - — Item 1.1.1

On a la possibilité de changer le symbole des items :

```
\renewcommand{\labelitemi}
{$\hookrightarrow$}
\renewcommand{\labelitemii}
{$\mapsto$}
\renewcommand{\labelitemiii}
{$\Rightarrow$}
\begin{itemize}
\item Item 1
\begin{itemize}
\item Item 1.1
\begin{itemize}
\item Item 1.1.1
\end{itemize}
\end{itemize}
\end{itemize}
```

- Item 1
  - ↪ Item 1.1
    - ⇒ Item 1.1.1

Cependant, les utilisateurs de l'extension `babel` devront utiliser un autre moyen car ce dernier package devient actif après la balise `\begin{document}`. Il faudra donc mettre en préambule :

```
\frenchbsetup[LabelItemi=<symbole>]
```

Ceci aura pour effet de remplacer tous les tirets par défaut par le symbole choisi.

On peut aussi mettre différents symboles pour chaque item (mais ça ne sert pas à grand chose ...) :

```
\begin{itemize}
\item[$\blacktriangleright$] Item 1
\item[$\hookrightarrow$] Item 2
\item[$\Rightarrow$] Item 3
\end{itemize}
```

- ▶ Item 1
- ↪ Item 2
- ⇒ Item 3

3.11.1.2 L'environnement enumerate

Cet environnement permet de numéroter automatiquement les items.

```
\begin{enumerate}
\item Item 1
\begin{enumerate}
\item Item 1.1
\begin{enumerate}
\item Item 1.1.1
\item Item 1.1.2
\end{enumerate}
\item Item 1.2
\end{enumerate}
\item Item 2
\item Item 3
\end{enumerate}
```

1. Item 1
   a. Item 1.1
      i. Item 1.1.1
      ii. Item 1.1.2
   b. Item 1.2
2. Item 2
3. Item 3

On peut, avec cet environnement, décider de commencer la numérotation après « 1 ». Pour cela, il faudra charger l'extension `enumitem` :

```
\begin{enumerate}[start=5]
\item Item 1
\item Item 2
\item Item 3
\end{enumerate}
```

5. Item 1
6. Item 2
7. Item 3

Cette dernière extension offre beaucoup d'options comme, par exemple :

```
\begin{enumerate}
[label=\protect\fcolorbox{blue}
{yellow}{\emph{\alph*}}]
\item Item 1
\item Item 2
\item Item 3
\end{enumerate}
```

a Item 1
b Item 2
c Item 3

3.11.1.3 L'environnement description

Par défaut, cet environnement ne met aucun symbole devant les items.

```
\begin{description}
\item Item 1
\begin{description}
\item Item 1.1
\begin{description}
\item Item 1.1.1
\item Item 1.1.2
\end{description}
\item Item 1.2
\end{description}
\item Item 2
\item Item 3
\end{description}
```

- Item 1
  - Item 1.1
    - Item 1.1.1
    - Item 1.1.2
  - Item 1.2
- Item 2
- Item 3

Bien entendu, on peut tout de même en mettre :

```
\begin{description}
\item[\starredbullet] Item 1
\begin{description}
\item[\starredbullet] Item 1.1
\begin{description}
\item[\starredbullet] Item 1.1.1
\item[\starredbullet] Item 1.1.2
\end{description}
\item[\starredbullet] Item 1.2
\end{description}
\item[\starredbullet] Item 2
\item[\starredbullet] Item 3
\end{description}
```

- ✦ Item 1
  - ✦ Item 1.1
    - ✦ Item 1.1.1
    - ✦ Item 1.1.2
  - ✦ Item 1.2
- ✦ Item 2
- ✦ Item 3

N.B. Le symbole `\starredbullet` (✦) est propre à l'extension `fourier`.

3.11.1.4 L'environnement list

```
\begin{list}{<symbole ou groupe de mots>}{<options>}
Contenu de la liste
\end{list}
```

Les options sont les suivantes :

Options	Descriptions
\topsep	espace interne vertical haut de la liste.
\partopsep	espace externe vertical haut de la liste si l'environnement est précédé d'une ligne vide.
\itemsep	espace vertical entre chaque item.
\parsep	espace vertical entre les paragraphes dans un item.
\leftmargin	espace horizontal (non nul) entre les marges gauches de l'environnement et la liste.
\rightmargin	espace horizontal (non nul) entre les marges droites de l'environnement et la liste.
\listparindent	espace externe pour l'indentation du paragraphe après le premier dans un item.
\itemindent	indentation de la première ligne dans un item. Peut être négative.
\labelsep	séparation entre la fin de la boîte contenant le label et le texte de la première ligne de chaque item.
\labelwidth	largeur de la boîte contenant le label.
\makelabel{label}	génère le label affiché par la commande \item.

Regardons cela sur un exemple :

```
\begin{list}{Vu l'article 1}
{\itemsep=3mm\itemindent=1cm
\labelsep=1cm}
\item Le président se doit d'avoir
un comportement en \og bon père de
famille \fg\ pour la structure
commerciale dont il est à la tête ;
\item Les salariés seront rémunérés
à la hauteur de leur travail
moyennant un salaire horaire brut
fixé par le président.
\end{list}
```

Vu l'article 1 Le président se doit d'avoir un comportement en « bon père de famille » pour la structure commerciale dont il est à la tête ;

Vu l'article 1 Les salariés seront rémunérés à la hauteur de leur travail moyennant un salaire horaire brut fixé par le président.

Ce dernier environnement n'a pas beaucoup d'intérêt à mes yeux., mais peut-être ne vois-je pas certaines choses ...

3.11.1.5 Comparaison des différentes listes

```
\begin{itemize}
\item[\starredbullet] Je déclare
avoir pris connaissance de mes droits
et les accepte avec humilité.
\item[\starredbullet] J'ordonne à mon
stagiaire de corriger mes copies.
\end{itemize}

\begin{enumerate}
\item Je déclare avoir pris
connaissance de mes droits et les
accepte avec humilité.
\item J'ordonne à mon stagiaire de
corriger mes copies.
\end{enumerate}

\begin{description}
\item[\starredbullet] Je déclare
avoir pris connaissance de mes droits
et les accepte avec humilité.
\item[\starredbullet] J'ordonne à mon
stagiaire de corriger mes copies.
\end{description}
```

- ✦ Je déclare avoir pris connaissance de mes droits et les accepte avec humilité.
- ✦ J'ordonne à mon stagiaire de corriger mes copies.

1. Je déclare avoir pris connaissance de mes droits et les accepte avec humilité.
2. J'ordonne à mon stagiaire de corriger mes copies.

- ✦ Je déclare avoir pris connaissance de mes droits et les accepte avec humilité.
- ✦ J'ordonne à mon stagiaire de corriger mes copies.

3.11.2 Listes horizontales

Il existe de multiples extensions pour créer des listes horizontales. Parmi elles, l'extension `shortlst`.

```
\begin{shortitemize}
\item Item 1 \item 2
\item Ceci est un long item
\item Item 4
\end{shortitemize}
\vskip 1cm
\begin{shortenumerate}
\item Item 1 \item Item 2 \item Item 3
\item Ceci est un long item
\item Item 4
\end{shortenumerate}
```

— Item 1 — Item 2
— Item 3
— Ceci est un long item
— Item 4

1. Item 1 2. Item 2
3. Item 3
4. Ceci est un long item
5. Item 4

3.12 Rédiger une lettre

Il existe une classe adaptée à la France pour cela : il s'agit de la classe `lettre`.

Il existe une autre classe anglo-saxonne nommée « letter ». Elle permet elle aussi de rédiger une lettre, mais les commandes ne sont pas adaptées à nos besoins.

Voici un modèle de ce que cette classe peut faire :

```
\documentclass[12pt]{lettre}
\usepackage[latin1]{inputenc}
\usepackage[frenchb]{babel}
\usepackage[T1]{fontenc}
\begin{document}
\begin{letter}{Prénom nom\\Adresse\\Code Postal VILLE} % destinataire
\date{} % Facultatif (permet de changer la date de compilation
\name{Jean DUPONT} % Identité de l'expéditeur
\adress{1, rue de la cigogne\\75000 PARIS} % Adresse de l'expéditeur
\lieu{Paris} % Ville d'expédition
\telephone{01.50.32.03.10.}
% ou \notelephone si l'on ne veut pas faire apparaître cette ligne
\nofax % ou \fax{...}
\email{jdupont@gmail.com}
\signature{Votre ami dévoué} % facultatif
\conc{Pot de vin} % Objet de la lettre
\Nref{Réf. 0123A} % Facultatif -> Votre référence
\Vref{EXP02156} % Facultatif -> Réf. du point de vue destinataire
\opening{Monsieur,} % Phrase d'introduction
Corps de la lettre
\ps{PS : }{Je suis ton père ...} % Facultatif
\cc{Ta mère} % Autre destinataire
\encl{Photo de ta mère} % Pièce(s) jointe(s)
\closing{En vous souhaitant bonne réception,} % Phrase de conclusion
\end{letter}
\end{document}
```

Par défaut, il y aura un trait à gauche au niveau où l'on doit plier la lettre. Si l'on ne souhaite pas le voir, il suffira d'écrire en préambule :

```
\makeatletter
\newcommand*{\norule}{\renewcommand*{\rule@length}{0}}
\makeatother
```

Ensuite, il suffit de mettre entre les balises `lettre` la commande `\norule`.

Il existe d'autres commandes. Pour en prendre connaissance, je vous conseille de regarder la documentation téléchargeable sur http://www.ctan.org/pkg/lettre.

4 Faire des mathématiques

Je n'ai pas la prétention de tout mettre ici. Je peux vous conseiller de regarder le document sur la page ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf qui pourra être source d'informations complémentaires.

4.1 Les trois extensions de base

Les principales extensions à charger sont amsmath, amsfonts et amssymb. Elles permettent d'écrire des mathématiques.

4.1.1 L'extension amsmath

```
1 \usepackage[<options>]{amsmath}
```

On oublie en général trop souvent que cette extension peut s'appeler avec des options. Les options sont les suivantes :

Options	Descriptions
ttags	« *top tags* ». Pour une équation qui tient sur plusieurs lignes, la référence de l'équation sera placée sur la première ligne.
btags	« *bottom tags* ». Pour une équation qui tient sur plusieurs lignes, la référence de l'équation sera placée sur la dernière ligne.
centertags	Valeur par défaut. Pour une équation qui tient sur plusieurs lignes, la référence de l'équation sera placée de façon centrée verticalement.
sumlimits	Valeur par défaut. Lorsque vous écrivez une sommation (par exemple), les indices seront mis au-dessus et au-dessous du symbole (en mode *display* - voir plus loin).
nosumlimits	Les indices seront mis à côté du symbole, même en mode *display*.
intlimits	Valeur par défaut. Fonctionne comme sumlimits avec les intégrales.
nointlimits	Fonctionne comme nosumlimits avec les intégrales.
namelimits	Valeur par défaut. Fonctionne comme sumlimits avec les autres opérateurs (comme det, inf, lim, max, min.
nonamelimits	Fonctionne comme nosumlimits pour ces derniers opérateurs.
leqno	Place la référence des équations à gauche.
reqno	Valeur par défaut. Place la référence des équations à droite.

`fleqn`	Place l'équation à une indentation fixe à gauche plutôt qu'au centre.

4.1.2 L'extension amsfonts

```
\usepackage{amsfonts}
```

Cette extension permet d'écrire avec certaines fontes mathématiques. La commande `\mathbb` permet d'écrire en mode mathématique : $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{H}$, $\mathbb{C}$,.... La commande `\mathfrak` permet d'écrire en mode mathématique : $\mathfrak{F}$, $\mathfrak{C}$, $\mathfrak{R}$,....

4.1.3 L'extension amssymb

```
\usepackage{amssymb}
```

Cette extension permet d'écrire les divers symboles utilisés en mathématiques. Par exemple : $\Rightarrow$, $\mapsto$, $\cap$, $\cup$, $\pm$, $\perp$, ...
En général, les éditeurs possèdent des icônes pour raccourcis.

Il existe d'autres extensions qui possèdent des commandes, ou qui remplacent celles des extensions AMS.
Par exemple, il y a l'extension `euler`, dont la documentation se trouve sur http://www.cs.brown.edu/system/software/latex/doc/euler.pdf

4.2 Les différents modes mathématiques ?

4.2.1 Des mathématiques dans une phrase (« en ligne »)

Si l'on veut qu'une phrase contienne des éléments mathématiques, on va placer le contenu mathématique entre deux symboles « $ ».

```
Or, nous savons que $\sqrt{9}=3$.\\
Ainsi, nous pouvons conclure que ...
```

Or, nous savons que $\sqrt{9}=3$.
Ainsi, nous pouvons conclure que ...

4.2.2 Des mathématiques centrées

On parle ici de mathématiques en mode *displaymath*. Il y a trois façon d'écrire ce genre de mathématiques :

```
\textbf{Méthode 1.}
$$ x\sqrt{3}+y\sqrt{2}=5$$
\textbf{Méthode 2.}
\[ x\sqrt{3}+y\sqrt{2}=5\]
\textbf{Méthode 3.}
\begin{displaymath}
x\sqrt{3}+y\sqrt{2}=5
\end{displaymath}
```

Méthode 1.

$$x\sqrt{3}+y\sqrt{2}=5$$

Méthode 2.

$$x\sqrt{3}+y\sqrt{2}=5$$

Méthode 3.

$$x\sqrt{3}+y\sqrt{2}=5$$

4.2.3 Numéroter les équations

On numérote les équations avec l'environnement `equation`.

```
\begin{equation}
x\sqrt{2}+y\sqrt{3}=1
\end{equation}
```

$$x\sqrt{2}+y\sqrt{3}=1 \tag{4.1}$$

Pour écrire plusieurs équations numérotées, on pourra utiliser l'environnement `eqnarray`, qui se comporte comme une sorte de tableau (ça a l'avantage de pouvoir aligner les signes « = »). Ici, chacune des lignes sera numérotée.

```
\begin{eqnarray}
3x+2y & = & 5\\
x-2y & = & 3
\end{eqnarray}
```

$$\begin{aligned} 3x+2y &= 5 && (4.2)\\ x-2y &= 3 && (4.3) \end{aligned}$$

4.3 Systèmes d'équations

On peut, bien entendu, créer des systèmes d'équations à l'aide de tableaux, mais il existe l'extension `systeme` qui permet de faciliter la syntaxe.

```
\systeme{2x-3y=1,x+102y=6}
\systeme[][:]
{0,2x-1,85y=0,3:1,8x-5,4y=9,3}
```

$$\begin{cases} 2x - 3y = 1 \\ x + 102y = 6 \end{cases} \qquad \begin{cases} 0{,}2x - 1{,}85y = 0{,}3 \\ 1{,}8x - 5{,}4y = 9{,}3 \end{cases}$$

4.4 Quelques objets mathématiques essentiels

4.4.1 Les fractions

Trois commandes sont à notre disposition : `\frac`, `\dfrac` (le « d » est ici pour « *display style* ») et `\tfrac` (le « t » est ici pour « *text style* »). Ces commandes acceptent deux arguments : le premier est le numérateur et le second, le dénominateur.

```
\[
\frac{x}{x+1} \quad ; \quad
\dfrac{x}{x+1} \quad ; \quad
\tfrac{x}{x+1}
\]
```

$$\frac{x}{x+1} \quad ; \quad \dfrac{x}{x+1} \quad ; \quad \tfrac{x}{x+1}$$

Nous voyons ici qu'en mode *displaymath*, les commandes `\frac` et `\dfrac` produise le même objet. Quant à `\tfrac`, elle produit l'objet qu'on aurait obtenu *en ligne* avec la commande `\frac` :

```
$\frac{x}{x+1} \quad ; \quad
\dfrac{x}{x+1} \quad ; \quad
\tfrac{x}{x+1}$
```

$\frac{x}{x+1} \quad ; \quad \dfrac{x}{x+1} \quad ; \quad \tfrac{x}{x+1}$

Pour résumer :

- En mode mathématique *en ligne*, `\frac` = `\tfrac`.
- En mode *displaymath*, `\frac` = `\dfrac`.

La commande`\tfrac` est utile en mode *displaymath* quand on veut écrire des fractions « à étages ».

```
\[
\frac{\tfrac{1}{2}+\tfrac{1}{3}}
{\tfrac{1}{2}-\tfrac{1}{3}}
\]
```

$$\frac{\tfrac{1}{2}+\tfrac{1}{3}}{\tfrac{1}{2}-\tfrac{1}{3}}$$

4.4.2 Indices & exposants

Le caractère « `underscore` » est lié à la notion d'*indice* alors que l'`accent circonflexe` est lié à la notion d'exposant.

```
\[
x^6 \quad ; \quad
y_8 \quad ; \quad
x^{10^{15}} \quad ; \quad
u_{v_p}
\]
```

$$x^6 \quad ; \quad y_8 \quad ; \quad x^{10^{15}} \quad ; \quad u_{v_p}$$

Notons que si aucune accolade n'est mise, seul le caractère qui suit l'underscore ou l'accent circonflexe sera mis en indice ou en exposant.

4.4.3 Les limites

Nous allons voir qu'en ligne, les limites ne s'écrivent pas tout à fait de la même façon qu'en *displaymath*.

```
\textbf{En mode display :}
\[
\lim_{x \to +\infty} f(x)=-\infty
\]
\textbf{En ligne :}\\
Nous savons que
$\lim_{x \to +\infty} f(x)=-\infty$.
\\
Pour une écriture correcte, on fera :
\\
$\lim\limits_{x\to +\infty} f(x)=
-\infty$.
```

En mode display :

$$\lim_{x\to+\infty} f(x) = -\infty$$

En ligne :
Nous savons que $\lim_{x\to+\infty} f(x) = -\infty$.
Pour une écriture correcte, on fera :
$\lim\limits_{x\to+\infty} f(x) = -\infty$.

Pour une limite à gauche ou à droite d'un nombre, on écrira :

```
\[
\lim_{\substack{x \to 1\\x<1}}
f(x)=-\infty
\]
```

$$\lim_{\substack{x\to 1\\ x<1}} f(x) = -\infty$$

4.4.4 Matrices & déterminants

```
\[
A=\begin{pmatrix}
1 & 0\\ -2 & \tfrac{1}{2}
\end{pmatrix} \quad
B=\begin{bmatrix}
1 & -1\\ -\tfrac{1}{3} & 2
\end{bmatrix} \quad
\det A = \begin{vmatrix}
1 & 0\\ -2 & \tfrac{1}{2}
\end{vmatrix}
\]
```

$$A = \begin{pmatrix} 1 & 0 \\ -2 & \frac{1}{2} \end{pmatrix} \quad B = \begin{bmatrix} 1 & -1 \\ -\frac{1}{3} & 2 \end{bmatrix}$$

$$\det A = \begin{vmatrix} 1 & 0 \\ -2 & \frac{1}{2} \end{vmatrix}$$

4.4.5 Les coefficients binomiaux

Comme pour les fractions, il y a 3 commandes : `\binom`, `\dbinom` et `\tbinom`.

```
\[ \binom{n}{p} \quad ; \quad
\tbinom{n}{p} \]
$\binom{8}{2} \quad ; \quad
\dbinom{8}{2}$
```

$$\binom{n}{p} \quad ; \quad \tbinom{n}{p}$$

$$\binom{8}{2} \quad ; \quad \dbinom{8}{2}$$

4.5 Les délimiteurs

Les délimiteurs sont toujours de la forme `\left <délimiteur> ... \right <délimiteur>`.

4.5.1 Les parenthèses

```
\[
\left(
\frac{\tfrac{1}{2}-1}
{1-\tfrac{1}{3}}
\right)
\]
```

$$\left(\frac{\tfrac{1}{2}-1}{1-\tfrac{1}{3}}\right)$$

S'il n'y a qu'un délimiteur, il faut tout de même mettre `\left` et `\right`.

```
\[
\left(
\begin{array}{l}
x+1=-1\\
\textrm{ou}\\
2x-8=7
\end{array}
\right.
\]
```

$$\left(\begin{array}{l} x+1=-1 \\ \text{ou} \\ 2x-8=7 \end{array}\right.$$

On met un point là où on ne veut pas de délimiteur.

4.5.2 Les accolades

```
\[
S=\left\lbrace\frac{1}{2};2\right\rbrace
\]
\[
\left\lbrace
\begin{array}{l}
2x+y=-9\\
x+2y=8
\end{array}
\right.
\]
\[
\underbrace{1+2}_{\textrm{addition}}
\qquad
\overbrace{4-2}^{\textrm{soustraction}}
\]
```

$$S=\left\lbrace\frac{1}{2};2\right\rbrace$$

$$\left\{\begin{array}{l} 2x+y=-9 \\ x+2y=8 \end{array}\right.$$

$$\underbrace{1+2}_{\text{addition}} \qquad \overbrace{4-2}^{\text{soustraction}}$$

4.5.3 Les crochets

```
\[
\left[\frac{3}{4}\right] \quad ;
\quad
\left\llbracket 3;\infty
\right\rrbracket
\]
```

$$\left[\frac{3}{4}\right] \quad ; \quad \left\llbracket 3;\infty \right\rrbracket$$

Les commandes `\llbracket` et `\rrbracket` sont disponibles *via* l'extension `fourier`, mais d'autres extensions les proposent (pas les paquets `ams`).

4.5.4 Les parties entières

```
\[
\left\lfloor\frac{2}{7}\right\rfloor
\quad ; \quad
\left\lceil\frac{2}{7}\right\rceil
\]
```

$$\left\lfloor\frac{2}{7}\right\rfloor \quad ; \quad \left\lceil\frac{2}{7}\right\rceil$$

4.5.5 Les valeurs absolues

```
\[
\left\vert\frac{1}{7}\right\vert
\]
```

$$\left|\frac{1}{7}\right|$$

4.5.6 Norme de vecteur

```
\[
\left\|\vv{AB}\right\|
\]
```

$$\left\|\overrightarrow{AB}\right\|$$

4.6 Un caractère sur ou sous un autre

```
\[
\overset{+\infty}{X} \quad ;
\quad \underset{-\infty}{X}
\]
```

$$\overset{+\infty}{X} \quad ; \quad \underset{-\infty}{X}$$

4.7 Écritures & symboles mathématiques

4.7.1 Multiplications & divisions

```
\[
3 \times 4 \quad ; \quad 5\div 2
\]
```

$$3 \times 4 \quad ; \quad 5 \div 2$$

4.7.2 Expressions conjuguées

```
\[
\overline{z+z'}=\overline{z}
+\overline{z'}
\]
```

$$\overline{z+z'}=\overline{z}+\overline{z'}$$

4.7.3 Les vecteurs

```
\[
\overrightarrow{AB} \quad ;
\quad \overleftarrow{AB}
\]
```

$$\overrightarrow{AB} \quad ; \quad \overleftarrow{AB}$$

Pour des vecteurs un peu plus jolis, je vous propose l'extension `esvect`.

```
\[ \vv{AB} \]
```

$$\vec{AB}$$

4.7.4 Les sommations

```
\[ \sum_{n=1}^{+\infty}\frac{1}{n^2}
=\frac{\pi^2}{6} \]
```

$$\sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

4.7.5 Les produits

```
\[
\prod_{n=0}^{+\infty}\frac{n+1}{n+2}
=0 \]
```

$$\prod_{n=0}^{+\infty} \frac{n+1}{n+2} = 0$$

4.7.6 Les intégrales

```
\[
\int_a^b f(x)\textrm{d}x \quad ;
\quad
\iint f(x,y)\textrm{d}x\textrm{d}y
\]
\[
\iiint f(x,y,z)\textrm{d}x\textrm{d}y
\textrm{d}z
\]
```

$$\int_a^b f(x)\mathrm{d}x \quad ; \quad \iint f(x,y)\mathrm{d}x\mathrm{d}y$$

$$\iiint f(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$$

4.7.7 Les unions & intersections

```
\[
\bigcup_{n=1}^{+\infty}A_n
\quad ; \quad
\bigcap_{n=1}^{+\infty}B_n
\]
```

$$\bigcup_{n=1}^{+\infty} A_n \quad ; \quad \bigcap_{n=1}^{+\infty} B_n$$

4.7.8 Les angles

```
\[
\widehat{ABC}=30^{\circ}
\quad ; \quad
\widecheck{ABC}=330^{\circ}
\]
```

$$\widehat{ABC} = 30^\circ \quad ; \quad \widecheck{ABC} = 330^\circ$$

La commande `\widecheck` est disponible avec l'extension `mathabx`. Malheureusement, cette extension définit beaucoup de commandes qui peuvent déjà être définies par d'autres extensions et donc rencontrer des problèmes de compatibilité.
Par exemple, cette extension a provoqué un message d'erreur lors de la création de cet ouvrage : la commande `\widering` est déjà définie. Pour éliminer ce soucis, j'ai ouvert le fichier `mathabx.dcl` et j'ai commenté la ligne où est définie cette commande :

```
%\def\widering#1{\ring{\wideparen{#1}}}%
```

Comme cela, je rends inopérant cette commande et le soucis s'envole !
Mais quelques fois, les erreurs sont bien trop nombreuses pour faire de même. Dans ce cas, la commande `\widecheck` pourra être définie manuellement en tapant dans le préambule du document :

```
\documentclass[12pt]{article}
\newcommand{\widecheckperso}[1]{%
\overset{\rotatebox{180}{$\widehat{\hphantom{#1}}$}}{#1}}
\begin{document}
$\widecheckperso{ABC}$
\end{document}
```

4.7.9 Les arcs de cercle

```
\[
\wideparen{AB}\quad ; \quad
\widearc{AB}\quad ; \quad
\wideOarc{AB}\quad ; \quad
\widering{AB}
\]
```

$$\wideparen{AB} \quad ; \quad \widearc{AB} \quad ; \quad \wideOarc{AB} \quad ; \quad \widering{AB}$$

Ces commandes sont disponibles avec l'extension `fourier`, mais aussi avec d'autres, comme par exemple `yhmath`.

4.8 Poser une opération

Les enseignants de primaire et collège le savent bien : pouvoir poser simplement une opération (multiplication, division, addition, soustraction) est très apprécié. Ceci est possible avec l'extension `xlop`.
Voici quelques exemples :

```
\opdiv[decimalsepsymbol={,}]{47}{5}
```

$$\begin{array}{rr|l} 4\,7 & & 5 \\ 2\,0 & & 9{,}4 \\ 0 & & \end{array}$$

```
\opdiv[decimalsepsymbol={,},period,
style=text]{47}{5}
```

$47 \div 7 = 6{,}\underline{714285}\ldots$

```
\newcommand\hole[1]{$\bullet$}
\opadd[operandstyle.1.1=\hole,
operandstyle.1.-2=\hole,
operandstyle.2.3=\hole,
resultstyle.2=\hole]
{45.89}{127.5}
```

$$\begin{array}{rr} & {\scriptstyle 1\ 1} \\ & 4\bullet.8\bullet \\ + & \bullet\,2\,7.5 \\ \hline & 1\bullet.3\,9 \end{array}$$

4.9 Simplification de fractions

L'extension `cancel` permet de barrer du texte, des nombres.

```
\[ \frac{2 \times \cancel{3} \times
7 \times \cancel{13}}{5 \times
\cancel{3} \times \cancel{13} \times
21} \]
\[ \frac{\cancel{25}^{\textcolor{red}
{5}}\times \cancel{27}^{\textcolor
{blue}{9}}}{\cancel{12}^{\textcolor
{blue}{4}}\times\cancel{35}^{
\textcolor{red}{7}}}\]
```

$$\frac{2 \times \cancel{3} \times 7 \times \cancel{13}}{5 \times \cancel{3} \times \cancel{13} \times 21}$$

$$\frac{\cancel{25}^{5} \times \cancel{27}^{9}}{\cancel{12}^{4} \times \cancel{35}^{7}}$$

4.10 Formater un nombre

Les nombres à plus de 3 chiffres doivent faire apparaître les classes (groupes de 3 chiffres). Pour ne pas m=passer trop de temps sur le formatage des nombres, l'extension `numprint` offre la commande éponyme :

```
\usepackage{numprint}
\numprint{152531,12032}
```

152 531,120 32

4.11 Faire de la géométrie

Pour dessiner en LaTeX, il existe plusieurs *solutions graphiques* : PSTricks, TikZ, Asymptote, Metapost sont les 4 solutions graphiques prépondérantes.

Pour chacune de ses solutions, il vous faudra apprendre leur propre syntaxe.
Heureusement, pour les plus récalcitrants d'entre vous, le logiciel geogebra, qui permet de créer des graphiques et dessins de façon très simple, peut exporter vos dessins en TikZ ou PSTricks. L'inconvénient de cette technique est que le code généré n'est pas très digeste. Donc, pour le modifier, ce n'est pas toujours très simple.

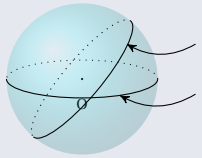
En ce qui me concerne, je n'utilise que TikZ (cet ouvrage comporte des morceaux de TikZ, comme par exemple les titres, les numéros de pages, les cadres qui imitent une feuille déchirée pour vous montrer le résultat des codes donnés en encadré bleu, etc.). Je ne peux donc vous renseigner que sur les outils TiKZ.

Il existe les extensions d'Alain Matthes pour créer facilement des objets en géométrie euclidienne, des graphes, etc. Vous pourrez trouver cela sur le site http://www.altermundus.fr, comme les packages :

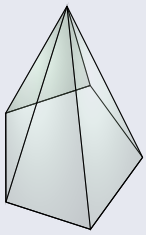
- tkz-euclide, très complet pour faire des cercles, droites, points, etc. Très bien fait !
- tkz-graph pour faire des graphes
- tkz-tab pour faire des tableaux de signes et de variations.
- tkz-fct pour tracer des courbes et tout ce qui concerne les fonctions (intégrales, somme de Rieman, ...)

J'ai moi-même fait une extension (très modeste par rapport aux précédentes) qui permet de faciliter le travail des enseignants de collège : cette extension, pas-cours, téléchargeable sur mon site http://www.mathweb.fr, permet de dessiner des cubes, des sphères, etc.

```
\begin{tikzpicture}
\boule[incolor=cyan,scale=0.5,
grandcercle,coefopaq=0.2,legende,
name]
\end{tikzpicture}
```

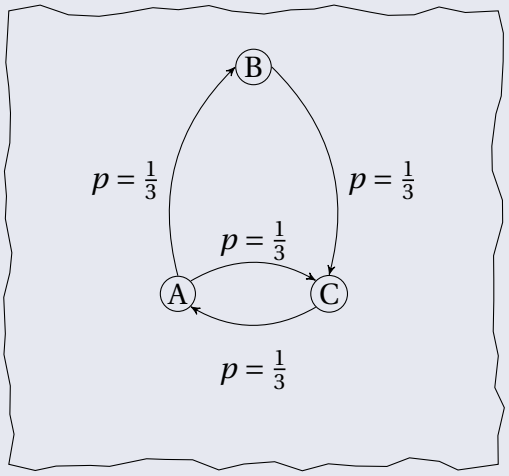



```
\begin{tikzpicture}
\pyramreg[n=5,hauteur=2,
incolor=green!20,coefopaq=0.2,
rayon=1]
\end{tikzpicture}
```



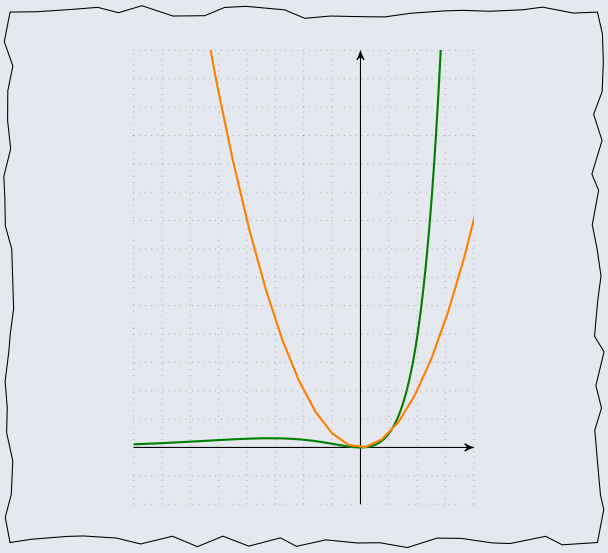
4.11.1 Un exemple de graphe

```
\begin{tikzpicture}
\node[circle,draw] (A) at (-1,0) {A};
\node[circle,draw] (B) at (0,3) {B};
\node[circle,draw] (C) at (1,0) {C};
\draw[->,>=stealth'] (A.north) to
[bend left=30] (B.west);
\node at(-1.7,1.5) {$p=\frac{1}{3}$};
\draw[->,>=stealth'] (B.east) to
[bend left=30] (C.north);
\node at (0,.7) {$p=\frac{1}{3}$};
\draw[->,>=stealth'] (C.south west) to
[bend left=30] (A.south east);
\node at (1.7,1.5) {$p=\frac{1}{3}$};
\draw[->,>=stealth'] (A.north east) to
[bend left=30] (C.north west);
\node at (0,-1) {$p=\frac{1}{3}$};
\end{tikzpicture}
```



4.11.2 Tracer une courbe

```
\begin{tikzpicture}
[scale=.75,domain=-4:3]
\draw[step=.5cm,gray,very thin,
dotted](-4,-1) grid (2,7);
\draw[->,>=stealth'] (-4,0)--(2,0);
\draw[->,>=stealth'] (0,-1)--(0,7);
\clip (-4,-1) rectangle (2,7);
\draw[samples=500,thick,
green!50!black] plot
(\x,{exp(2*\x)-(\x+1)*exp(\x)}) ;
\draw[thick,orange] plot (\x,{\x*\x});
\end{tikzpicture}
```



Il se peut que vous ayez le message d'erreur suivant :

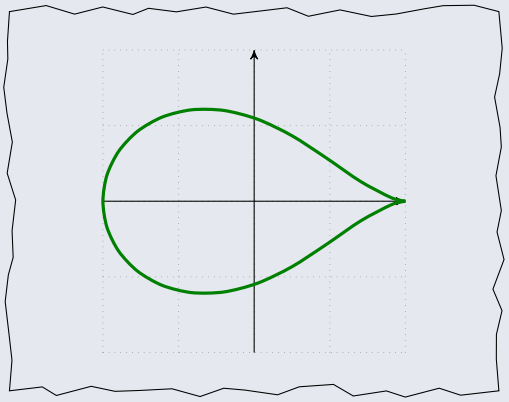
```
! Paragraph ended before \tikz@plot@samples@recalc was complete.
```

Dans ce cas, il vous faudra ajouter les commandes :

```
\shorthandoff{:}
\begin{tikzpicture}
...
\end{tikzpicture}
\shorthandon{:}
```

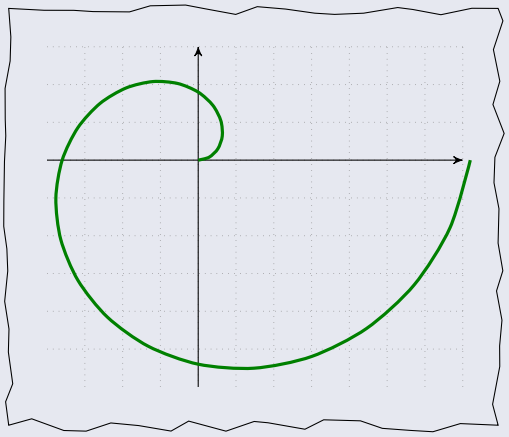
Pour les courbes paramétriques, on pourra écrire les choses ainsi :

```
\begin{tikzpicture}[scale=2]
\draw[step=.5cm,gray,very thin,dotted]
(-1,-1) grid (1,1);
\draw[->,>=stealth'] (-1,0)--(1,0);
\draw[->,>=stealth'] (0,-1)--(0,1);
\draw[smooth,domain=0:360,very thick,
green!50!black] plot ({(5*cos(\x)-3*cos(\x)
*cos(\x)-1)/(5-4*cos(\x))},
{(3*sin(\x)*(1-cos(\x)))/(5-4*cos(\x))});
\end{tikzpicture}
```



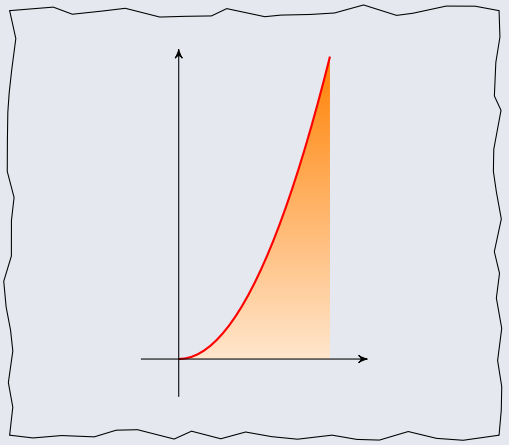
Pour les courbes polaires, on fera :

```
\begin{tikzpicture}[scale=0.01]
\draw[step=50cm,gray,very thin,dotted]
(-200,-300) grid (350,150);
\draw[->,>=stealth'] (-200,0)--(350,0);
\draw[->,>=stealth'] (0,-300)--(0,150);
\draw[smooth,domain=0:360,very thick,
green!50!black] plot ({\x}:\x);
\end{tikzpicture}
```



Pour colorier sous une courbe, on fera :

```
\begin{tikzpicture}[domain=0:2]
\shade[top color=orange,bottom color=
orange!20] (0,0) plot
(\x,{\x*\x}) |- (0,0);
\draw[thick,red] plot (\x,{\x*\x});
\draw[->,>=stealth'] (-0.5,0) -- (2.5,0);
\draw[->,>=stealth'] (0,-0.5) -- (0,4.1);
\end{tikzpicture}
```



4.11.3 Tracer une surface

TikZ a ses limites. En effet, les surfaces 3D ne peuvent pas se représenter (directement) avec cette solution graphique. Dans ce cas, on pourra utiliser l'une des solutions suivantes :

Avec GNUPlot. `GNUPlot` est un logiciel qui permet, entre autre, de tracer des courbes et des surfaces.
Pour les utilisateurs de Windows, il faut veiller à ce que le PATH contienne le chemin qui

pointe vers `wgnuplot.exe`.
Pour l'ajouter, suivez les étapes (testées sous Windows Vista) :

- Cliquez sur le menu « Démarrer »
- Cliquez avec le bouton droit de la souris sur « Ordinateur »
- Sélectionnez « Propriétés »
- Cliquez sur « Paramètres système avancé »
- Cliquez sur le bouton « Continuer »
- Cliquez sur le bouton « Variables d'environnement ... »
- Dans la seconde fenêtre, cliquez sur la ligne « Path », puis sur le bouton « Modifier ... »
- Si vous avez installé GNUPlot à la racine d'un lecteur dans le répertoire `gnuplot`, en fin de ligne, tapez le chemin suivant :

```
<lecteur>:\gnuplot\binary\;
```

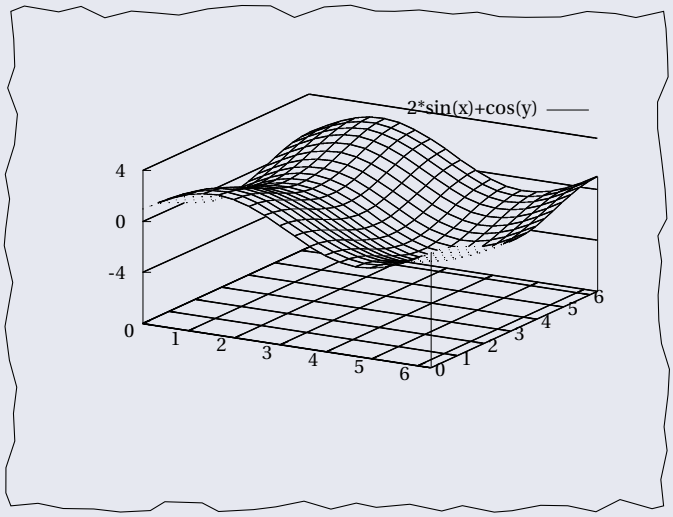
Sinon, adaptez le chemin en fonction de l'endroit où est installé GNUPlot.

Ensuite, il faudra paramétrer votre éditeur TEX de sort à ce qu'il compile avec l'option :

```
latex --shell-escape-enable-write18
pdflatex  --shell-escape-enable-write18
```

Maintenant, l'extension `gnuplottex` va nous permettre d'insérer tout graphique que l'on peut faire avec GNUPlot dans notre document :

```
\begin{gnuplot}[scale=.75]
set pm3d
set surface
set hidden3d
set isosamples 20
set xrange[0:6.283]
set yrange[0:6.283]
set ztics 4
set xtics 1
set ytics 1
set grid xtics ytics ztics
splot 2*sin(x)+cos(y)
\end{gnuplot}
```



Pour les utilisateurs de MikTex, il faudra passer l'option `miktex` lors de l'appel de l'extension :

```
\usepackage[miktex]{gnuplottex}
```

Avec TeXgraph. `TeXgraph` est un traceur assez performant. Il peut tracer, entre autre, des surfaces 3D. Avec l'extension `texgraph`, on peut inclure directement du code TeXgraph pour inclure directement la surface souhaitée.
Attention toutefois, il vous faudra faire quelques réglages avant de compiler votre premier document :

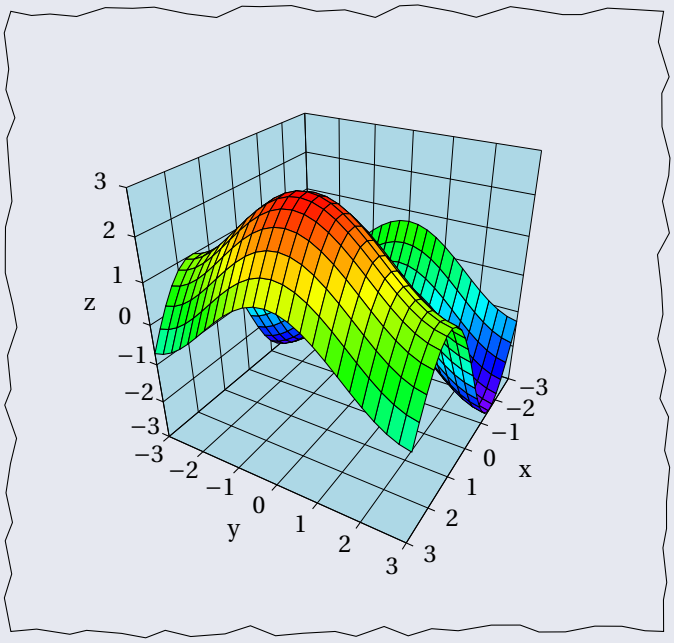
- Avant tout, télécharger le fichier zip sur http://texgraph.tuxfamily.org/Telecharger.html
- Décompressez-le à la racine d'un lecteur. Ceci créera le répertoire `<lecteur>:\TeXgraph1.97` (la version change selon la date à laquelle vous téléchargerez le fichier)
- Dans ce répertoire, il y a le fichier `texgraph.sty` ; copiez-le dans votre arborescence TEX et rafraichissez la base de données (faites un *texhash* en ligne de commande par exemple)
- Maintenant, pour les utilisateurs de Windows :
  - Ajoutez au PATH le chemin :

    ```
    <lecteur>:\TeXgraph\TeXgraph\
    ```

    (Menu démarrer > clic droit sur « Ordinateur »> Propriétés > Paramètres systèmes avancés > Variables d'environnements > Clic sur « Path » > Bouton « Modifier ... »)
  - Créez au même endroit la variable d'environnement « TeXgraphDir », et entrez le même chemin.
  - Redémarrez le système

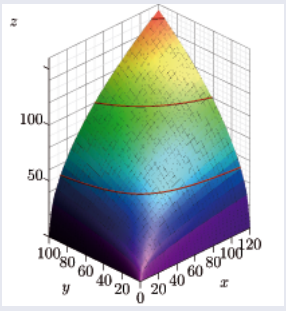
Vous voilà fin prêt(e) à tester le code suivant :

```
\begin{texgraph}[export=pgf]
view(-6.5,6,-6.5,5.5),
Marges(0,0,0,0),size(7.5),
view3D(-3,3,-3,3,-3,3),
ModelView(central),
S:=GetSurface([u+i*v,2*sin(u)+
cos(v)],
-3+3*i,-3+3*i),
stock:=for facette in S By jump do
z:=Zde(isobar3d(facette)),
facette,
ColorJump
(Hsb(270*(Zsup-z)/(Zsup-Zinf),1,1))
od,
FillStyle:=full,
LabelSize:=footnotesize,
BoxAxes3D(grid:=1,
FillColor:=lightblue),
Ligne3D(SortFacet(stock),1)
\end{texgraph}
```



Avec Asymptote. Sur le site http://asymptote.sourceforge.net/, vous pourrez télécharger `Asymptote`. Dans le fichier ainsi installé, il y a l'extension `asymptote` qu'il faudra mettre dans l'arborescence LATEX. Mais une documentation est présente dans le répertoire `Asymptote` créé et je vous recommande de vous y reporter pour l'installation.
Voici un exemple de surface plane obtenue :

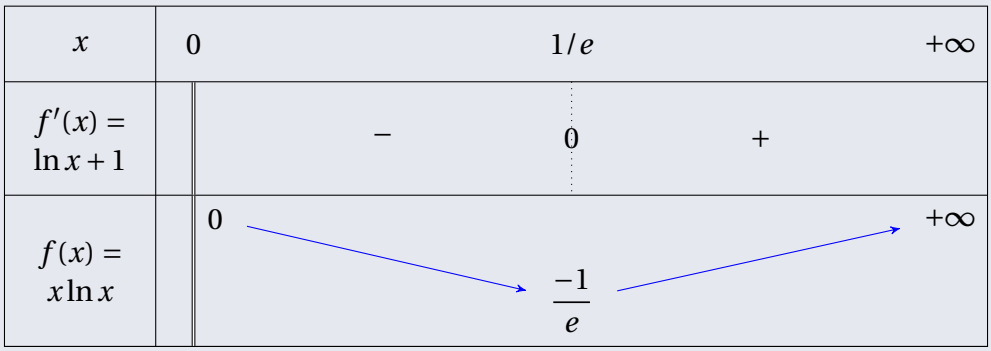
```
\begin{asy}
import graph3;
import contour;
import grid3;
import palette;

size(8cm);
real[] lignesniveaux={50,100,150};

currentprojection=orthographic(-10,-10,8);
limits((0,0,0),(120,100,160));

real f(pair z) {return sqrt(2*z.x*z.y);}

surface s=surface(f,(0,0),(120,100),30,Spline);

draw(s,mean(palette(s.map(zpart),Rainbow())),black);

grid3(gridroutine=XYXgrid(),Step=20);
grid3(gridroutine=XZXgrid(pos=Relative(1)),Step=20,step=10);
grid3(gridroutine=YZYgrid(pos=Relative(1)),Step=20,step=5);

xaxis3(Label("$x$",position=MidPoint,align=SE),
           Bounds(Min,Min),
           OutTicks(Step=20));
yaxis3(Label("$y$",position=MidPoint,align=SW),
           Bounds(Min,Min),
           OutTicks(Step=20));
zaxis3(Label("$z$",position=EndPoint,align=N+W),
           XYEquals(0,100),
           InTicks(beginlabel=false,endlabel=false,Label(align=Y))
           );
draw(lift(f,contour(f,(0,0),(120,100),lignesniveaux)),1bp+red);
\end{asy}
```



4.12 Tableaux de variations

Il existe l'extension `variations` mais aussi, comme cité précédemment, `tkz-tab`.
Voici un exemple avec cette dernière :

```
\begin{tikzpicture}
\tikzset{arrow style/.style={blue,->,>=stealth',shorten>=6pt,
shorten<=6pt}}
\tkzTabInit[espcl=5]{$x$/1,$f'(x) = \ln x +1$/1.5,$f(x) = x \ln x$/2}
{$0$,$1/e$,$+\infty$} \tkzTabLine{d,-,z,+,}
\tkzTabVar{ D+/ / $0$,-/$\dfrac{-1}{e}$/,+/$+\infty$/}
\end{tikzpicture}
```



4.13 PdfAdd

Cet outil, téléchargeable sur la page http://www.xm1math.net/pdfadd/shots.html, créé par la même personne que Texmaker, à savoir Pascal Brachet, permet de faire avec une aisance non négligeable :

- Des tableaux de variations ;
- Des courbes ;
- Des diagrammes en boîtes ;
- Des illustrations de convergence pour les suites récurrentes ;
- Des arbres de probabilités ;
- Des histogrammes ;
- Des graphes ;
- Des répartitions suivant une loi binomiale ;
- Un cercle trigonométrique complété ;
- Des figures géométriques classiques ;
- De la géométrie dans l'espace.

5 Aller plus loin

5.1 Les commandes

Nous avons vu jusqu'à présent qu'il existait des commandes qui nous simplifient bien la vie. Mais quelques fois, on a envie de plus ... Nous allons donc ici voit comment créer nos propres commandes et environnements.

Par habitude, on préfèrera définir une commande ou un environnement dans le préambule.

5.1.1 Avec la commande \def

La commande `\def` est tout de suite suivie du nom de la commande que vous voulez et de #1, #2, ..., #n, où n est le nombre d'arguments que vous souhaitez donner à votre commande. Ensuite, entre accolades, le code de votre commande.

Regardons sur un exemple : nous allons construire une commande appelée `ombre` qui va afficher un texte avec son ombre, avec 3 arguments : le premier désignera le texte, le second sera la couleur du texte et le troisième, la couleur de l'ombre.

```
\def\ombre[#1][#2]#3{%
\begin{tikzpicture}
\node[text=#2,opacity=0.3] (ombre)
at (0,0) {#3};
\node[text=#1] at (-.05,.05) {#3};
\end{tikzpicture}
}
\ombre[bleu][red]{Un texte ombré}
```

Un texte ombré

Pour les arguments qui peuvent être plus longs, comme des paragraphes, on peut utiliser la commande `\long` :

```
\long\def\parombre[#1][#2]#3{%
\begin{tikzpicture}
\node[text=#2,opacity=0.3] (ombre)
at (0,0) {\begin{minipage}
{0.95\textwidth}#3\end{minipage}};
\node[text=#1] at (-.05,.05) {
\begin{minipage}{0.95\textwidth}#3
\end{minipage}};
\end{tikzpicture}
}
\parombre[blue][red]{%
Ceci est un premier paragraphe.

Ceci est un second paragraphe.
}
```

Ceci est un premier paragraphe.
Ceci est un second paragraphe.

Essayez d'enlever la commande `\long` et vous verrez qu'une erreur apparaîtra : ceci est du au fait qu'il y a des espaces entre chaque phrase du dernier argument.

5.1.2 Avec la commande \newcommand

5.1.2.1 Sans argument

En mathématiques, la constante « e » s'écrit toujours (par convention) en écriture droite (et non en italique). Or, dans un environnement mathématique, les lettres sont toujours en italique. Pour y remédier, on peut définir une commande qui permet d'écrire ce « e » en écriture droite :

```
\newcommand\e{\textrm{e}}
\[ f(\e)=\frac{\e-1}{\e^2+3}\]
}
```

$$f(\mathrm{e}) = \frac{\mathrm{e}-1}{\mathrm{e}^2+3}$$

Il pourra en être de même si l'on veut écrire très souvent l'ensemble $\mathbb{R}$.

```
\newcommand\R{$\mathbb{R}$}
Nous savons que E est un \R-espace
vectoriel. Or, F est aussi un
\R-espace vectoriel.
}
```

Nous savons que E est un $\mathbb{R}$-espace vectoriel. Or, F est aussi un $\mathbb{R}$-espace vectoriel.

Cependant, il y a des fois où le nom que l'on souhaite donner à notre commande existe déjà. Si on souhaite écraser cette définition, il faudra utiliser la commande `\renewcommand`. Cela sera le cas si l'on veut faire la même chose que ce que l'on a fait à « e »pour « i ». En effet, `\i` est déjà définie donc on fera :

```
\renewcommand\i{\textrm{i}}
\[ \overline{3-2\i}=3+2\i\]
}
```

$$\overline{3-2\mathrm{i}} = 3+2\mathrm{i}$$

5.1.2.2 Avec arguments obligatoires

Dans un texte mathématique où l'on utilise très souvent un même type d'écriture, il est très pratique de définir une commande. Par exemple, si vous devez écrire beaucoup d'intégrales, il vaut mieux définir une commande :

```
% "3" est le nombre d'arguments
\newcommand\integrale[3]
{%
$\displaystyle\int_{#1}^{#2}#3
\textrm{d}x$
}
Nous allons calculer
\integrale{0}{1}{e^x},
puis \integrale{1}{3}{\ln x}.
```

Nous allons calculer $\displaystyle\int_{0}^{1} e^{x}\mathrm{d}x$, puis $\displaystyle\int_{1}^{3} \ln x\mathrm{d}x$.

Le nombre d'arguments ne peut pas être supérieur à 9.
Vous voyez ici que l'argument $n°1$ est appelé par #1, que l'argument $n°2$ est appelé par #2, etc.

A ce stade, vous allez me dire : « Mais quelle est la différence entre `\def` et `\newcommand` ? ». Outre la syntaxe, la commande `\def` est beaucoup moins sécurisée. En effet, si vous chargez une extension dans laquelle la commande `\integrale` est déjà définie, en définissant à nouveau cette commande par le biais de `\def`, votre définition écrasera sans rechigner l'ancienne, ce qui peut être embêtant dans certains cas.
Si vous définissez votre commande `\integrale` par le biais de `\newcommand` alors qu'elle a déjà été définie dans une autre extension par `\newcommand` aussi, LaTeX vous le dira par un message d'erreur. Ainsi, vous pourrez changer le nom de votre commande pour qu'il y ait compatibilité entres les extensions que vous avez chargé.

Si vous souhaitez remplacer une commande existante malgré tout, il faudra utiliser la commande `renewcommand`.

5.1.2.3 Avec arguments optionnels

Si, dans la définition d'une commande, un argument est pratiquement toujours le même, on peut se débrouiller pour passer sa valeur par défaut. On fera alors ainsi :

```
\newcommand\suite[2][0]
{%
$\left(u_{#1},\ldots,u_{#2}\right)$
}
La \suite[5]{n} est une suite
extraite  de \suite{n}.
```

La $(u_5,\ldots,u_n)$ est une suite extraite de $(u_0,\ldots,u_n)$.

On ne peut mettre qu'une seule valeur par défaut.
On peut y remédier à l'aide de l'extension `xargs` qui offre la commande `\newcommandx` où, à la place du seul argument, on peut mettre une liste de valeurs par défaut, chaque argument étant représenté par un numéro (non précédé de la dièse).

```
\newcommandx\suisuite[3][1=u,2=0,3=n]
{%
$\left(#1_{#2},\ldots,#1_{#3}\right)$
}
On considère les suites \suisuite,
\suisuite[v] et \suisuite[w][2].
```

On considère les suites $(u_0,\ldots,u_n)$, $(v_0,\ldots,v_n)$ et $(w_2,\ldots,w_n)$.

Chacune des commandes `\newcommand`, `\renewcommand`, `\newcommandx` et `\renewcommandx` ont leur version étoilée.

```
\newcommand*{<nom>}[<arguments>]{<macro>}
\renewcommand*{<nom>}[<arguments>]{<macro>}
\newcommandx*{<nom>}[<liste arguments>]{<macro>}
\renewcommandx*{<nom>}[<liste arguments>]{<macro>}
```

Ces versions sont privilégiées dans un cas général. L'étoile dit à LaTeX que les arguments seront courts (ce qui est pratiquement toujours le cas car si les arguments sont longs, il sera très souvent plus avantageux de définir un *environnement* plutôt qu'une *commande*).

5.2 Les environnements

Pour définir un environnement, on fera :

```
\newenvironment<*>{<nom>}[<nombre d'arguments>]
{<macro à exécuter au début de l'appel>}
{<macro à exécuter à la fin de l'appel>}
```

Voyons un exemple simple :

```
\newenvironment{reflet}[1]
{
\begin{center}
\begin{tikzpicture}[inner sep=3pt]
\node[scale=2,above,yslant=0.05]{#1};
\node[scale=2,above,yslant=0.05,
yscale=-1,scope fading=south,
opacity=0.4]{#1};
\end{tikzpicture}

\medskip
}
{
\medskip

\begin{tikzpicture}[inner sep=3pt]
\node[scale=2,above,yslant=0.05,
rotate=180]{Fin};
\node[scale=2,above,yslant=0.05,
yscale=-0.75,
scope fading=north,opacity=0.4,
rotate=180]{Fin};
\end{tikzpicture}
\end{center}
}
\begin{reflet}{Mon titre}
Ceci est un texte.
\end{reflet}
```

5.3 Versions étoilées d'une commande

On peut définir une version étoilée d'une commande. Dans ce cas, on va utiliser la macro `\@ifstar`. Mais comme vous allez très vite vous en rendre compte, pour LaTeX, la lettre `@` est réservée ; il faudra donc dire à LaTeX de l'accepter. Pour cela, on va encadrer la définition de la commande par les commandes `\makeatletter` et `\makeatother`.

```
\makeatletter
\newcommand\frigo@star{La commande est étoilée.}
\newcommand\frigo@nostar{La commande n'est pas étoilée.}
\newcommand*\frigo{\@ifstar{\frigo@star}{\frigo@nostar}}
\makeatother
```

Il n'existe pas d'équivalent pour les environnements. Si l'on veut définir un environnement `toto` et sa version étoilée `toto*`, il faudra définir deux environnements bien distincts.

5.4 Déclarer une commande robuste

Dans la plupart des cas, utiliser la commande `\newcommand` suffira pour définir une commande qui vous est propre. Cependant, dès lors qu'il s'agit d'une commande que l'on va utiliser un peu partout (donc destinée à un usage très répandu), il serait peut-être préférable de rendre la commande robuste, c'est-à-dire « universelle ». Par exemple, la commande `\section` est une commande robuste.
Pour se faire, on utilisera la commande `\DeclareRobustCommand`. Ainsi, si l'on veut redéfinir les commandes « \[» et « \] », on fera par exemple :

```
\DeclareRobustCommand{\[}{\begin{equation*}}
\DeclareRobustCommand{\]}{\end{equation*}}
```

5.5 Déclarer une commande mathématique

Si vous souhaitez créer une commande qui sera exécutée en modes mathématiques, alors il sera sans doute judicieux d'utiliser la commande `\mathchoice`. Voici un exemple :

```
\newcommand{\intervalle}[4]{%
\mathchoice
% mode \displaystyle
{\left#1#2\mathclose{}\mathpunct{};#3\right#4}
% mode \textstyle
{\mathopen{#1}#2\mathclose{}\mathpunct{};#3\mathclose{#4}}
% mode \scriptstyle
{\mathopen{#1}#2\mathclose{}\mathpunct{};#3\mathclose{#4}}
% mode \scriptscriptstyle
{\mathopen{#1}#2\mathclose{}\mathpunct{};#3\mathclose{#4}}
}
```

$$I = \left]-\frac{2}{3};\frac{7}{4}\right]$$

$$I =]-\tfrac{2}{3};\tfrac{7}{4}]$$

$$\int_{]-\frac{2}{3};\frac{7}{4}]} \mathscr{F}$$

$$J_{]-\frac{2}{3};\frac{7}{4}]}$$

5.6 Les compteurs

Comme vous pouvez vous en douter, les compteurs sont omniprésents en LaTeX ; par exemple, quand vous créez un titre, une section, une sous-section, etc.

5.6.1 Les différents types de compteurs

Ils sont au nombre de 6 :

`\arabic` (chiffres arabes)	`\roman` (chiffres romains minuscules)	`\alph` (lettres minuscules)
`\fnsymbol` (symboles)	`\Roman` (chiffres romains majuscules)	`\Alph` (lettres majuscules)

5.6.2 Déclaration d'un compteur

C'est la commande `\newcounter` qui va déclarer un nouveau compteur :

```
\newcounter{<nom>}[<nom du compteur de dépendance>]
```

La notion de dépendance d'un compteur s'explique en pensant à la dépendance du compteur `subsection` au compteur `section` (à chaque fois que le compteur `section` est incrémenté, le compteur `subsection` est initialisé (à « 0 »).
Si on n'informe pas la dépendance du compteur au moment de sa déclaration, on peut tout de même le faire plus tard avec la commande suivante :

```
\@addtoreset{<nom du compteur>}{<nom du compteur de dépendance>}
```

Au contraire, si on souhaite supprimer la dépendance d'un compteur à un autre, on utilisera :

```
\@removefromreset{<nom du compteur>}{<nom du compteur de dépendance>}
```

5.6.2.1 Opérations sur un compteur

On donnera une valeur à un compteur avec la commande `\setcounter` :

```
% Ici, je donne la valeur "2" au compteur "moncompteur"
\setcounter{moncompteur}{2}
```

On incrémentera un compteur avec la commande `\addtocounter` :

```
% J'incrémente "moncompteur" de 2 unités
\addtocounter{moncompteur}{2}
```

Si on souhaite augmenter de 1 unité la valeur d'un compteur, on pourra utiliser la commande `\stepcounter` ou `\refstepcounter`.

```
\stepcounter{moncompteur}
\refstepcounter{moncompteur}
```

Ces dernières commandes initialisent tous les compteurs dépendant de "moncompteur", contrairement à la commande `\addtocounter`.

5.6.2.2 Valeur d'un compteur

La commande `\value` donne la valeur d'un compteur. On peut s'en servir dans la commande `\addtocounter` :

```
\addtocounter{moncompteur}{\value{moncompteurbis}}
```

5.6.2.3 Exemple

```
\newcounter{moncompteur}
\newcounter{moncompteurbis}
[moncompteur]
\setcounter{moncompteur}{1}
\setcounter{moncompteurbis}{1}
"moncompteur" en chiffre arabe :
\arabic{moncompteur}\\
"moncompteur" en lettre :
\alph{moncompteur}\\
"moncompteurbis" en chiffre arabe :
\arabic{moncompteurbis}\\
\stepcounter{moncompteur}
J'incrémente "moncompteur".\\
"moncompteur" en chiffre arabe :
\arabic{moncompteur}\\
"moncompteurbis" en chiffre arabe :
\arabic{moncompteurbis}
```

"moncompteur" en chiffre arabe : 1
"moncompteur" en lettre : a
"moncompteurbis" en chiffre arabe : 1
J'incrémente "moncompteur".
"moncompteur" en chiffre arabe : 2
"moncompteurbis" en chiffre arabe : 0

5.7 Les longueurs

5.7.1 Les différentes unités

`cm` (centimètre)
`mm` (millimètre)
`in` (inch)
`pt` (point)
`em` (cadrat)
`ex` (hauteur d'œil)
`bp` (point PostScript)

5.7.2 Les correspondances d'unités

1 in = 2,54 cm
1 pt = 0,3515
1 bp = $\frac{1}{72}$ in
1 em = largeur d'un « M » dans la fonte courante
1 ex = hauteur d'un « x » dans la fonte courante

5.7.3 Déclaration et manipulation d'une nouvelle longueur

On utilisera la commande \newlength pour déclarer une nouvelle longueur.
Pour lui attribuer une valeur, on utilisera \setlength.
Pour lui ajouter une valeur, on utilisera \addtolength.

```
\newlength\malongueur\setlength\malongueur{1em plus 1em minus 2em}
\addtolength\malongueur{2em}
```

5.7.4 Quelques longueurs définies

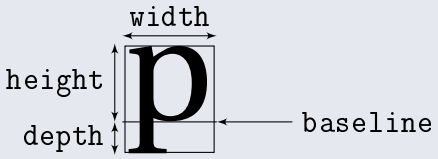
Certaines commandes représentent déjà des longueurs :

\parskip : espace vertical inter-paragraphes
\parindent : longueur des indentations
\baselineskip : espace vertical entre deux lignes de base consécutives
\textwidth : largeur occupée par le texte
\textheight : hauteur occupée par le texte
\marginparwidth : largeur de la marge où l'on peut écrire des notes
\oddsidewidth : espace horizontal entre la marge de gauche et le texte pour les pages impaires
\evensidewidth : espace horizontal entre la marge de gauche et le texte pour les pages paires
\marginparsep : espace horizontal séparant le texte de la marge de droite
\headsep : espace vertical qui sépare l'en-tête du texte

5.7.5 Récupération de longueurs

Quand on parle de longueurs, on pense souvent à récupérer la longueur ou la hauteur d'un objet. On pourra faire cela avec les commandes \settowidth (la longueur), \settoheight (la hauteur à partir de la ligne de base) et \settodepth (la profondeur à partir de la ligne de base).

Pour comprendre tout ceci, regardons un exemple :



On peut se servir de ces commandes pour construire un texte à trous par exemple.

```
\newlength\htrou
\newcommand*\trou[1]{%
\settowidth\htrou{#1}
\hspace*{2\htrou}
\setlength{\baselineskip}
{1.2\baselineskip}
}
D'après le théorème de
\trou{Pythagore}, le \trou{carré}
de l'\trou{hypoténuse} est égal
à la somme des carrés des deux
autres côtés.
```

D'après le théorème de , le de l' est égal à la somme des carrés des deux autres côtés.

5.7.6 Espaces élastiques horizontaux

On dispose d'une panoplie de commandes se terminant par `fill`. Mais avant tout, il faut comprendre ce que fait la commande `\hfill`. Regardons sur un exemple :

```
Ceci conclut le paragraphe.
\hfill$\blacksquare$
```

Ceci conclut le paragraphe. ■

Ce qui suit la commande sera placé à droite sur la même ligne.
Ainsi, toutes les commandes se terminant par `fill` auront la même philosophie.

```
Un mot \hrulefill\\
Un mot \dotfill\\
Un mot \dingfill{52}
```

Un mot _____
Un mot .....
Un mot ✔✔✔✔✔✔✔✔✔✔

La commande `\dingfill` est fournie par l'extension `pifont`.

5.8 Les réglures

La commande `\rule` créée une boîte, mais on s'en sert plus souvent pour créer un trait. L'objet créé est aligné sur la ligne de base. Voyons un exemple :

```
Nous avons ici un paragraphe
que l'on va tout de suite finir.
\par\nointerlineskip\noindent
\rule{\textwidth}{1pt}\par
Et maintenant un second paragraphe
avec une réglure verticale
\rule[-.3\baselineskip]{1pt}
{\baselineskip} voilà !
```

Nous avons ici un paragraphe que l'on va tout de suite finir.
Et maintenant un second paragraphe avec une réglure verticale | voilà !

On peut reprendre la commande `\trou` définie précédemment pour mettre un trait horizontal à la place du trou.

```
\newlength\htroub
\newcommand*\troub[1]{%
\settowidth\htroub{#1}
\rule[-1pt]{2\htroub}{1pt}
\setlength{\baselineskip}
{1.2\baselineskip}
}
D'après le théorème de
\troub{Pythagore}, le \troub{carré}
de l'\troub{hypoténuse} est égal à
la somme des carrés des deux autres
côtés.
```

D'après le théorème de _____ , le _____ de l' _____ est égal à la somme des carrés des deux autres côtés.

Il existe des réglures un peu spéciales. Il s'agit des réglures de largeur nulle mais qui occupent un espace vertical.

La commande \strut fixe la hauteur du plus haut caractère dans la fonte utilisée ainsi que la profondeur du caractère qui a la plus grande profondeur dans la fonte utilisée. Voyons une utilité :

```
\fbox{je} \fbox{dis} ou bien
\fbox{je\strut} \fbox{dis\strut}.
```

je dis ou bien je dis.

Cette dernière commande est très utile pour harmoniser la hauteur de différents objets sur une ligne.

La commande \arraystretch permet de fixer la hauteur des cases d'un tableau. Par défaut, elle vaut 1.

```
\renewcommand\arraystretch{1}
\begin{tabular}{|c|}
\hline
Hauteur normale\\
\hline
\end{tabular}
\renewcommand\arraystretch{1.5}
\begin{tabular}{|c|}
\hline
Hauteur 1.5\\
\hline
\end{tabular}
```

Hauteur normale
Hauteur 1.5

Les commandes \hphantom et \vphantom permettent de fixer respectivement une largeur et une hauteur phantom.

```
\[ \sqrt{x}\quad\textrm{contre}
\quad
\sqrt{\vphantom{\frac{x}{a}}x}\]
\[ \sqrt{x}\quad\textrm{contre}
\quad
\sqrt{\hphantom{-}x\hphantom{-}}\]
```

$$\sqrt{x} \quad \text{contre} \quad \sqrt{x}$$

5.9 Les styles

Le style des titres, des en-têtes, des pieds de page, table des matières, ... sont définis par défaut dans la classe appelée en début de document. Et comme vous pouvez le constater, quand on veut donner plus de relief aux documents, il est très vite nécessaire de les redéfinir.

5.9.1 Les titres

Une extension presque incontournable est `titlesec`. Elle met à notre disposition des commandes qui permettent de redéfinir le titre d'un document, des parties, chapitres, sections, sous-sections, sous sous-sections, paragraphes.

Je ne vous proposerai pas d'exemples ici car il y a pas mal de possibilités et je vous encourage à regarder la documentation de cette extension pour en prendre connaissance.

En ce qui concerne le titre de la table des matières, de la table des figures, de la table des tableaux, de la bibliographie et de l'index, on peut les changer avec ceci :

```
\renewcommand{\contentsname}{Sommaire}
\renewcommand{\listfigurename}{Liste des figures}
\renewcommand{\listtablename}{Liste des tables}
\renewcommand{\bibname}{Références bibliographiques}
\renewcommand{\indexname}{Index des mots clés importants}
```

5.9.2 En-têtes & pieds de page

On peut redéfinir les en-têtes et pieds de page à l'aide de l'extension `fancyhdr`.

Voici un exemple de ce que l'on peut faire :

```
% redéfinit l'épaisseur du trait en en-tête
\renewcommand{\headrulewidth}{0.8pt}
% ajoute à la hauteur de l'en-tête la hauteur d'une interligne
\addtolength{\headheight}{\baselineskip}
% refédinit la couleur des traits en marron
\renewcommand{\headrule}{\color{brown}\hrule}
\renewcommand{\footrule}{\color{brown}\hrule}
% style des en-têtes des pages impaires
\fancyhead[LE,RO]{\textcolor{brown}{\slshape \rightmark}}
% style des en-têtes des pages paires
\fancyhead[LO,RE]{\textcolor{brown}{\slshape \leftmark}}
 % style des pieds de page
\fancyfoot[C]{\vskip 3pt\textcolor{brown}{\thepage}}
```

Bien entendu, on peut se reporter à la documentation pour en savoir plus.

5.9.3 La table des matières

L'extension `tocloft` permet, a priori, de changer la table des matières ainsi que la table des figures et la liste des tableaux.
Il y a aussi l'extension `minitoc` qui permet de construire des tables de matières, de figures et de tableaux en début de chaque chapitre.

Pour ma part, afin de changer l'aspect de la table des matières, j'utilise la syntaxe suivante dans le préambule :

```
On renomme la table des matières
\renewcommand{\contentsname}{Sommaire}
% On définit la couleur des liens dans la table des matières en fonction
% de leur niveau
\newcommand{\chaptertoccolor}{brown}
\newcommand{\sectiontoccolor}{blue}
\newcommand{\subsectiontoccolor}{green}
\newcommand{\subsubsectiontoccolor}{red}
% Profondeur d'affichage dans la table des matières
\setcounter{tocdepth}{4} % ici, 4 niveaux
\makeatletter
% On redéfinit la façon dont doit être écrit le titre des chapitres
\renewcommand*\l@chapter[2]{
\ifnum \c@tocdepth >\m@ne
\addpenalty{-\@highpenalty}
\vskip 1.0em \@plus\p@
\setlength\@tempdima{1.5em}
\begingroup
\parindent \z@ \rightskip \@pnumwidth
\parfillskip -\@pnumwidth
\leavevmode \bfseries
\advance\leftskip\@tempdima
\hskip -\leftskip
\def\@linkcolor{\chaptertoccolor}
\color{\chaptertoccolor}{\mathversion{bold} #1}\nobreak\
\leaders\hbox{$\m@th
\mkern \@dotsep mu\hbox{.}\mkern \@dotsep
mu$}\hfil\nobreak\hb@xt@\@pnumwidth{\hss
\def\@linkcolor{\chaptertoccolor}
\color{\chaptertoccolor}#2}\par
\penalty\@highpenalty
\endgroup
\fi}
% Définition des couleurs des différents niveaux dans la table des matières
\renewcommand*\l@section{\color{\sectiontoccolor}
\def\@linkcolor{\sectiontoccolor}
\@dottedtocline{1}{1.5em}{2.3em}}
\renewcommand*\l@subsection{\color{\subsectiontoccolor}
\def\@linkcolor{\subsectiontoccolor}
\@dottedtocline{2}{2.5em}{3.3em}}
```

```
\renewcommand*\l@subsubsection{\color{\subsubsectiontoccolor}
\def\@linkcolor{\subsubsectiontoccolor}
\@dottedtocline{3}{3.5em}{4.3em}}
% Rendre les numéros de la table des matières cliquables
\def\contentsline#1#2#3#4{
\ifx\\#4\\%
\csname l@#1\endcsname{#2}{#3}
\else
\csname l@#1\endcsname{\hyper@linkstart{link}{#4}{#2}\hyper@linkend}{
\hyper@linkstart{link}{#4}{#3}\hyper@linkend
}
\fi
}
\makeatother
```

Dans ce dernier code, j'ai voulu construire une table des matières en :

- définissant la profondeur à « 4 » ;
- attribuant des couleurs différentes aux entrées en fonction de leur profondeur ;
- rendant cliquable les numéros de pages tout en les mettant de la même couleur que le texte auquel ils correspondent ;
- mettre des pointillés uniquement au niveau du titre des chapitres ;
- redéfinissant le titre de la table des matières.

5.9.4 Les annexes

Insérer une annexe se fait en mettant la commande `\appendix`.

Si on ne souhaite pas faire paraître les annexes dans la table des matières, on écrira :

```
\appendix
\addtocontentsline{toc}{\setcounter{tocdepth}{0}}
% ou \addtocontentsline{toc}{\protect\setcounter{tocdepth}{0}} avec "calc"
\chapter{Titre de l'annexe}
```

5.9.5 Le titre du document (la couverture)

Les classes `article`, `book` et `report` définissent le titre *via* la commande `\maketitle`. Pour changer le style, on pourra utiliser dans le préambule la syntaxe suivante :

```
\makeatletter
\def\maketitle{
\null
\thispagestyle{empty}
[code]
\clearpage
% ou \cleardoublepage pour laisser une page vide
}
```

```
\makeatother
```

Vous pourrez alors insérer n'importe quel code : un mode graphique pour faire des choses bien sympathiques, ou un mode classique pour plus de sobriété.

5.10 Définir un fond de pages

L'extension `\eso-pic` permet de mettre un fond sur une ou plusieurs pages.

On peut se servir de cette extension pour, par exemple, créer des liserés sur toute la hauteur des pages. On pourra alors le faire ainsi :

```
\usepackage{eso-pic}
...
\AddToShipoutPicture{
  \AtPageLowerLeft{
    \rotatebox{90}{\colorbox{orange}{
      \begin{minipage}{\paperheight}Votre texte\end{minipage}}
  }
}
```

Ce code mettra un liseré orange à gauche de toutes les pages.
Comme l'extension `eso-pic` ne fournit pas l'équivalent à droite, on peut se fabriquer une commande qui fait ce que l'on veut :

```
\makeatletter
\newcommand\At@Page@Upper@Right[1]
{
  \put(\LenToUnit{\paperwidth},\LenToUnit{\paperheight}){#1}
}
\newcommand\AtPageLowerRight[1]
{
   \At@Page@Upper@Right
   {
      \put(0,\LenToUnit{-\paperheight}){\llap{#1}}
   }
}
\makeatother
\AddToShipoutPicture
{
   \AtPageLowerRight
   {
      \rotatebox{90}{\colorbox{orange}
      {
      \begin{minipage}{\paperheight}
         Votre texte
      \end{minipage}
      }
   }
}
```

Si l'on veut que le liseré se mette alternativement à gauche et à droite selon la parité des pages, on pourra écrire :

```
\documentclass[a4paper]{article}
\usepackage{graphicx}
\usepackage{eso-pic}
\usepackage{xcolor}
\usepackage{ifthen}
\makeatletter
\newcommand\At@Page@Upper@Right[1]
{
  \put(\LenToUnit{\paperwidth},\LenToUnit{\paperheight}){#1}
}
\newcommand\AtPageLowerRight[1]
{
  \At@Page@Upper@Right
  {
    \put(0,\LenToUnit{-\paperheight}){\llap{#1}}
  }
}
\makeatother
\AddToShipoutPicture
{
  \ifthenelse
  {\isodd{\value{page}}} % si la page est paire
  {
    \AtPageLowerRight
    {
      \rotatebox{90}
      {
        \colorbox{orange}
        {
          \begin{minipage}{\paperheight}
            Votre texte
          \end{minipage}
        }
      }
    }
  }
  {
    \AtPageLowerLeft
    {
      \rotatebox{90}
      {
        \colorbox{orange}
        {
          \begin{minipage}{\paperheight}
            Votre texte
          \end{minipage}
        }
      }
```

```
    }
  }
\begin{document}
...
\end{document}
```

Cette solution a été proposée sur la page :
http://forum.mathematex.net/latex-f6/liseret-cote-droit-t11233.html

5.11 Définir une subsubsubsection

Comme nous l'avons vu précédemment, les différents niveaux de sections sont :

- part
- chapter
- section
- psubsection
- subsubsection
- paragraph
- subparagraph

Si l'on veut créer un niveau qui s'intercale entre `subsubsection` et `paragraph`, on pourra faire :

```
\documentclass{book}
\usepackage{hyperref}
\setcounter{secnumdepth}{4}
% change le niveau maximum des entrées dans la table des matières
\setcounter{tocdepth}{4}
\makeatletter
% on définit un nouveau compteur pour la section
\newcounter{subsubsubsection}[subsubsection]
\renewcommand\thesubsubsubsection{%
\thesubsubsection .\@alph\c@subsubsubsection}
\newcommand\subsubsubsection{\@startsection{subsubsubsection}{4}{\z@}
{-3.25ex\@plus -1ex \@minus -.2ex}
{1.5ex \@plus .2ex}
{\normalfont\normalsize\bfseries}}
% Après, il faut bien tout décaler pour les "paragraph" et "subparagraph"
\renewcommand\paragraph{\@startsection{paragraph}{5}{\z@}
{3.25ex \@plus1ex \@minus.2ex}
{-1em}
{\normalfont\normalsize\bfseries}}
\renewcommand\subparagraph{\@startsection{subparagraph}{6}{\parindent}
{3.25ex \@plus1ex \@minus .2ex}
{-1em}
{\normalfont\normalsize\bfseries}}
% Ensuite, on s'occupe de l'affichage dans la table des matières
\newcommand*\l@subsubsubsection{\@dottedtocline{4}{10.0em}{4.1em}}
```

```
\renewcommand*\l@paragraph{\@dottedtocline{5}{10em}{5em}}
\renewcommand*\l@subparagraph{\@dottedtocline{6}{12em}{6em}}
\newcommand*{\subsubsubsectionmark}[1]{}
\def\toclevel@subsubsubsection{4}
\def\toclevel@paragraph{5}
\def\toclevel@subparagraph{6}
\makeatother
\begin{document}
\tableofcontents
...
\end{document}
```

Cette solution a été proposée sur la page :
http://forum.mathematex.net/latex-f6/subsubsub-t4031.html

5.12 Ajouter une entrée à la table des matières

```
% ici, je veux que mon entrée soit au même niveau que le titre des
% chapitres. Ici, je veux ajouter mon index au sommaire
\addcontentsline{toc}{chapter}{Index}
% \addstarredchapter{\indexname} si vous utilisez le package minitoc
\phantomsection % si vous utilisez le package hyperref
```

5.13 Insérer les index, bibliographie, etc. au sommaire

L'extension `\tocbibind` permet d'insérer un tas d'entrées qui, par défaut, n'y paraissent pas (comme l'index).

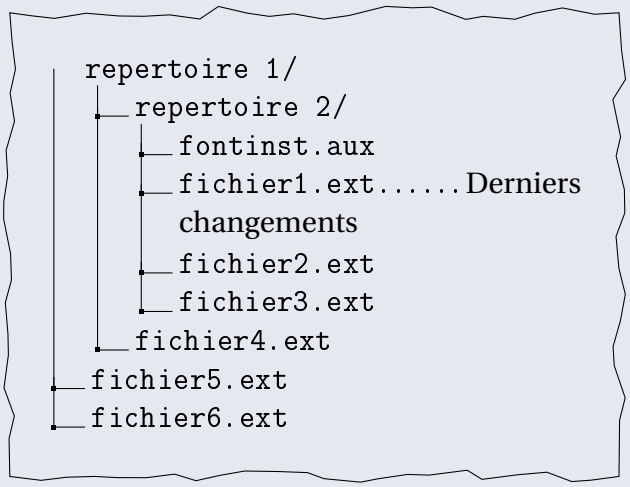
```
\usepackage[nottoc]{tocbibind}
% . . .
\chapter*{Introduction générale}
\phantomsection
\addstarredchapter{Introduction générale}
% . . .
\phantomsection
\listoffigures
\phantomsection
\listoftables
\phantomsection
\bibliography{ma-bibligraphie}
\printindex
```

Source : monbloginformatique.blogspot.com

5.14 Construire une arborescence

L'extension `dirtree` nous permet de faire cela :

```
\documentclass[12pt,a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{dirtree}
\begin{document}
\dirtree{%
.1 repertoire 1/.
.2 repertoire 2/.
.3 fontinst.aux.
.3 fichier1.ext
\DTcomment{Derniers changements}.
.3 fichier2.ext.
.3 fichier3.ext.
.2 fichier4.ext.
.1 fichier5.ext.
.1 fichier6.ext.
}
\end{document}
```



5.15 Un texte encadré en parallèle

On va ici utiliser les extensions `wrapfig` et `thmbox`.

```
\begin{wrapfigure}{r}{6cm}
\begin{thmbox}[L]{\textbf{John Conway}}
\tiny
John Horton Conway (né le 26 décembre 1937 à Liverpool, Angleterre) est
un mathématicien britannique. Extrêmement prolifique, il s'est penché
sur les théories des groupes finis, des n\oe uds, des nombres, des jeux
et du codage.\\
Né en 1937 en Angleterre, Conway s'est intéressé très tôt aux
mathématiques, et avait décidé de devenir mathématicien dès l'âge de 11
ans. Il étudie les mathématiques à Cambridge, au Gonville and Caius
College, et obtient son Bachelor of Arts en 1959. Ses premières
recherches, sous la direction de Harold Davenport, concernent la théorie
des nombres. Il s'intéresse aux ordinaux infinis. Joueur passionné de
backgammon, c'est pendant ces années universitaires qu'il développe son
intérêt pour la théorie des jeux. Il obtient son doctorat en 1964, puis
un poste à l'université de Cambridge.\\
En 1981, il devient membre de la Royal Society.\\
Conway quitte Cambridge en 1986 pour prendre en charge la chaire John
von Neumann de mathématiques à l'université de Princeton. Il vit depuis
à Princeton dans le New Jersey,
\end{thmbox}
\end{wrapfigure}
```

```
23
24 \begin{Large}
25 \textbf{Suite de Conway}
26 \end{Large}
27 \bigskip
28 La suite de Conway est une suite inventée en 1986 par le mathématicien
29 John Horton Conway, initialement sous le nom de "suite audioactive".
30 Elle est également connue sous le nom anglais de Look and Say ("regarder
31 et dire"). Dans cette suite, un terme se détermine en annonçant les
32 chiffres formant le terme précédent.
33 \medskip
34 Le premier terme de la suite de Conway est posé comme égal à 1. Chaque
35 terme de la suite se construit en annonçant le terme précédent, c'est-à-
36 dire en indiquant combien de fois chacun de ses chiffres se répète.\\
37 Concrètement : \\
38 $u_0=1$ ; $u_1=11$ ; $u_2=21$ ; $u_3=1211$ ; $u_4=111221$
```

Suite de Conway

La suite de Conway est une suite inventée en 1986 par le mathématicien John Horton Conway, initialement sous le nom de "suite audioactive". Elle est également connue sous le nom anglais de Look and Say ("regarder et dire"). Dans cette suite, un terme se détermine en annonçant les chiffres formant le terme précédent.

Le premier terme de la suite de Conway est posé comme égal à 1. Chaque terme de la suite se construit en annonçant le terme précédent, c'est-à-dire en indiquant combien de fois chacun de ses chiffres se répète. 45 Concrètement :
$u_0 = 1$; $u_1 = 11$; $u_2 = 21$; $u_3 = 1211$; $u_4 = 111221$

John Conway

John Horton Conway (né le 26 décembre 1937 à Liverpool, Angleterre) est un mathématicien britannique. Extrêmement prolifique, il s'est penché sur les théories des groupes finis, des nœuds, des nombres, des jeux et du codage.
Né en 1937 en Angleterre, Conway s'est intéressé très tôt aux mathématiques, et avait décidé de devenir mathématicien dès l'âge de 11 ans. Il étudie les mathématiques à Cambridge, au Gonville and Caius College, et obtient son Bachelor of Arts en 1959. Ses premières recherches, sous la direction de Harold Davenport, concernent la théorie des nombres. Il s'intéresse aux ordinaux infinis. Joueur passionné de backgammon, c'est pendant ces années universitaires qu'il développe son intérêt pour la théorie des jeux. Il obtient son doctorat en 1964, puis un poste à l'université de Cambridge.
En 1981, il devient membre de la Royal Society.

Conway quitte Cambridge en 1986 pour prendre en charge la chaire John von Neumann de mathématiques à l'université de Princeton. Il vit depuis à Princeton dans le New Jersey,

En fait, c'est l'environnement `wrapfigure` qui permet de mettre en parallèle des objets (textes ou figures).

5.16 Ecrire un listing

L'extension `\listings` permet d'écrire des listings d'une façon remarquable en mettant en valeur les mots du langage utilisé ; comme vous avez pu le voir pour cet ouvrage, les codes LaTeX sont mis en encadré et les commandes sont mises en couleurs.
Pour faire apparaître vos listings, on utilisera l'environnement `lstlisting`. Par défaut, il se peut que le listing ne vous plaise pas ; aussi est-il possible de changer l'apparence avec la commande `\lstset`.

```
\lstset{numbers=left, % met à gauche le numéro des lignes
   numberstyle=\tiny, % style des numéros
   stepnumber=1, % ici, les numéros vont de 1 en 1
   numbersep=3pt, % séparations des numéros
   language=[LaTeX]TeX, % langage du listing -> voir documentation
   backgroundcolor=\color{brown!10}, % couleur de fond
   frame=shadowbox, % style du cadre
   rulecolor=\color{brown}, % couleur du cadre
   framexleftmargin=10pt, % marge gauche
   keywordstyle=\color{green}\bfseries, % style des mots reconnus du
   % langage utilisé
   basicstyle=\ttfamily, % style par défaut des caractères
   keepspaces=true, % on indique que les espaces sont à garder
   commentstyle=\color{Gray}, % couleur du texte mis en commentaire
   morekeywords={mot1,mot2,mot3} % mots supplémentaires à reconnaître dans
   % le langage
}
```

5.17 Insérer des données d'un fichier CSV

L'extension `\datatool` permet, à partir d'un fichier CSV, d'insérer les différentes données.

Le fichier `donnees.csv` est :

```
40,120,40
40,90,60
35,180,20
55,190,40
```

et le code LaTeX est :

```
\DTLloaddb[noheader,%
keys={Temperature,Time,T2G},%
headers={\shortstack{Incubation\\Temperature},%
\shortstack{Incubation\\Time},\shortstack{Time to\\Growth}}%
]{t2g}{donnees.csv}

\begin{table}[htbp]
\caption{Time to Growth Data}
\centering
\DTLdisplaydb{t2g}
\end{table}
```

TABLE 5.1 – Time to Growth Data

Incubation Temperature	Incubation Time	Time to Growth
40	120	40
40	90	60
35	180	20
55	190	40

Je vous encourage à regarder la documentation pour en savoir plus sur cette extension.

6 Programmer avec LaTeX

6.1 Définir une variable

Une variable est un mot (ou une lettre) précédée d'un « \ ».
On lui attribue une valeur, numérique ou littérale, de plusieurs façons.

- En utilisant la commande `\def` si on veut lui attribuer directement la valeur ;
- En utilisant la commande `\newcommand` pour la même chose ;
- En utilisant la commande `\let`
- En utilisant la commande `\edef` pour adjoindre une nouvelle valeur.

Voyons des exemples :

```
% Commande \edef
\edef\test{A}
\edef\test{\test B}
\test\\
% Commande \def
\def\K{K\sc ern}
\K\\
% Commande \newcommand
\newcommand\Rb{$\mathbb{R}$}
\Rb\\
% Commande \let
\let\a\@empty % \a est vide
```

ABC
KERN
ℝ

Quelques fois, il est nécessaire de rendre la variable *globale*, c'est-à-dire « valable » dans tout le document (en effet, quand vous définissez une variable dans une macro, elle est définie de façon locale). Pour cela, on utilisera la commande `\global` :

```
\newcommand\init{%
\global\def\pi{3.14159265358979323846264338327 95}
\global\def\e{2.71828182845904523536028747135 27}
}
% On définit la valeur de Pi et de e
\init
```

6.2 Les tests

La commande \ifx permet de comparer deux chaînes de caractères.

```
\def\chaine{HELLO}
\def\chainebis{BYE}
\ifx\chaine\chainebis
Les deux chaînes sont identiques.
\else
Les deux chaînes sont différentes.
\fi
```

Les deux chaînes sont différentes.

Pour les variables numériques, il y a la commande \ifnum :

```
\def\i{2}
\def\j{9}
\ifnum\i=\j
  Les deux variables sont égales.
\else
  \ifnum\i<\j
    La variable $i$ est inférieure à
    la variable $j$.
  \else
    La variable $i$ est supérieure à
    la variable $j$.
  \fi
\fi
```

La variable i est inférieure à la variable j.

Toutes les commandes commençant par \if doivent impérativement se terminer par \fi.

L'extension ifthen permet de faire des tests avec sa commande \ifthenelse dont la syntaxe est la suivante :

```
\ifthenelse{<condition>}
{<commande si la condition est remplie}
{<commande si la condition n'est pas remplie>}
```

Voyons un exemple :

```
\usepackage{ifthen}
...
\newcommand\jour[1]
{
\ifthenelse{\equal{#1}{0}}{Dimanche}{}
\ifthenelse{\equal{#1}{1}}{Lundi}{}
\ifthenelse{\equal{#1}{2}}{Mardi}{}
\ifthenelse{\equal{#1}{3}}{Mercredi}{}
\ifthenelse{\equal{#1}{4}}{Jeudi}{}
\ifthenelse{\equal{#1}{5}}{Vendredi}{}
\ifthenelse{\equal{#1}{6}}{Samedi}{}
}

\jour{0} - \jour{1} - \jour{2}
```

Dimanche - Lundi - Mardi

6.3 Manipuler les chaines de caractères

L'extension `xstring`, quant à elle, offre la possibilité de :

- Compter le nombre de caractères d'une chaîne ;
- Comparer deux chaînes de caractères ;
- Concaténer deux chaînes de caractères ;
- Ôter une partie des caractères d'une chaîne ;
- Etc.

Je vous encourage à regarder la documentation sur CTAN (en français car l'auteur est français). Voyons quelques exemples tout de même :

```
% Teste si "CDE" est dans "ABCDEF"
\og CDE \fg\ apparaît dans \og ABCDEF
\fg\ :
\IfSubStr{ABCDEF}{CDE}{vrai}{faux}\\
% Teste si "A" est présent 5 fois dans
% "ABRACADABRA"
Le caractère \og A \fg\ apparaît 5 fois
dans \og ABRACADABRA \fg\ :
\IfSubStr[2]{ABRACADABRA}{A}{vrai}{faux}
% Compte le nombre de caractères de la
% chaîne "ABRACADABRA"
Le nombre de caractères de \og
ABRACADABRA \fg\ est :
\StrLen{ABRACADABRA}
% Teste si deux chaînes sont semblables
Ai-je bien écrit le mot ?
\IfStrEq{ACCUEIL}{ACCEUIL}{Pas de faute}
{Faute !}
```

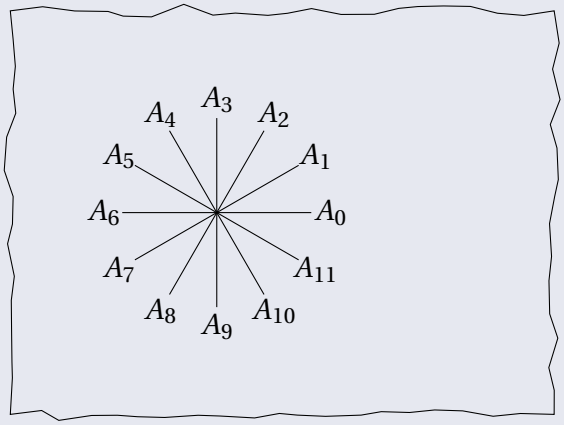
« CDE » apparaît dans « ABCDEF » : vrai
Le caractère « A » apparaît 5 fois dans « ABRACADABRA » : vrai
Le nombre de caractères de « ABRACADABRA » est : 11
Ai-je bien écrit le mot ? Faute !

6.4 Les boucles

L'extension `multido` met à notre disposition, entre autres, les commandes `\multido` et `\Multido`.

Voyons un premier exemple :

```
\begin{tikzpicture}
\multido{\i=0+1, \n=0+30}{12}
{
\draw (0,0) -- (\n:1.25cm);
\node at (\n:1.5cm) {$A_{\i}$};
}
\end{tikzpicture}
```



Attention aux noms que vous donnez aux variables.
En effet, il semblerait que l'extension n'aime réellement pas que les variables s'appellent par exemple `\a` ou `\j`.

Voyons maintenant la façon dont on peut construire un tableau de valeurs à l'aide d'une boucle. Nous allons utiliser un alias de `\global\edef` : la commande `\xdef` et `\Multido`.

```
\xdef\ligneX{Valeurs}
\xdef\ligneY{Carrés}
\Multido{\i=1+1}{6}
{
\xdef\ligneX{\ligneX & \i}
\FPmul\carre\i\i
\FPclip\carre\carre
\xdef\ligneY{\ligneY & \carre}
}

\begin{tabular}{||l||*6{c|}}
\hline
\ligneX\\
\hline
\ligneY\\
\hline
\end{tabular}
```

Valeurs	1	2	3	4	5	6
Carrés	1	4	9	16	25	36

L'utilisation de commandes de calculs dans la boucle nécessite celle de `\Multido` à la place de `\multido`.

Une autre façon de faire des boucles est d'utiliser la commande `\foreach` de l'extension `TikZ`. Elle offre des possibilités assez intéressantes, comme vous pourrez vous en rendre compte dans l'exemple suivant :

```
\foreach \i in {0,...,10}
{\i\ ;}

\foreach \i in {0,2,4,...,10}
{\i\ ;}

\foreach \nom/\gain in
{Luc/90,Corentin/50,Jean/10}
{\nom\dotfill\gain\ \eurologo\\ }
```

0 ;1 ;2 ;3 ;4 ;5 ;6 ;7 ;8 ;9 ;10 ;

0 ;2 ;4 ;6 ;8 ;10 ;

Luc.....90 €
Corentin.....50 €
Jean.....10 €

La commande `\eurologo` est disponible avec l'extension `fourier`, fonte que j'utilise pour cet ouvrage.

6.5 Calculer avec LaTeX

La première extension est `calc`. Mais vous allez très vite vous rendre compte que ce n'est pas la meilleure. Regardons un exemple :

```
\newcounter{x}
\setcounter{x}{3*2*\real{2.8}}
x = \thex\ (et non 16.8 ...)
```

x = 16 (et non 16.8 ...)

Très vite en lisant la documentation, on se rend compte des limites de cette extension et on va lui préférer l'extension `\fp`, surtout depuis qu'elle ne rentre plus en conflit avec `multido`. La syntaxe des commandes est assez intuitive mais on regrettera sans doute l'absence d'une vraie documentation au détriment de consignes sur la page http://www.ctan.org/tex-archive/macros/latex/contrib/fp. Voyons quelques exemples :

```
\FPadd\somme{3.8}{2.1}
\FPtrunc\somme\somme{1} $3,8+2,1=$ \somme
\
\FPsub\dif{3.8}{2.1}
\FPtrunc\dif\dif{1} $3,8-2,1=$ \dif\\
\FPmul\produit{3.8}{2.1}
\FPtrunc\produit\produit{2}
$3,8 \times 2,1=$ \produit\\
\FPdiv\quotient{3.8}{2.1}
$\frac{3,8}{2,1}\approx$ \quotient\\
\FPabs\valabs{-2.9}
\FPtrunc\valabs\valabs{1}
$\left| -2,9 \right|=$ \valabs\\
$\pi \approx $ \FPpi\\
\FPdiv\angle\FPpi{3}
\FPsin\sinus\angle
$\sin\left(\frac{\pi}{3}\right)\approx$
\sinus\\
\FPdiv\quint\somme{5}
\FPclip\quint\quint
$\somme\div 5 = \quint$
```

$3,8+2,1=5.9$

$3,8-2,1=1.7$

$3,8\times 2,1=7.98$

$\frac{3,8}{2,1}\approx 1.809523809523809523$

$|-2,9|=2.9$

$\pi\approx 3.141592653589793238$

$\sin\left(\frac{\pi}{3}\right)\approx 0.866025403784438646$

$5.9\div 5=1.18$

Par défaut, les résultats s'affichent avec les zéros inutiles après la virgule. Pour y remédier, on utilise la commande `\FPclip`.

Créer sa propre extension

Créer une extension n'est pas réservé aux experts. En effet, on peut écrire un package avec des commandes simples, que nous avons vu précédemment.
Bien entendu, celles et ceux qui connaissent les commandes du noyau TEX feront des extensions rapides et très performantes mais on peut tout de même utiliser LATEX.

7.1 Structure générale

Une extension pourra être structurée de cette façon :

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{nom du package}[<informations : date, version, but, ...]
<code>
\end{input}
```

La convention veut que l'on mette la date sous le format `AAAA/MM/JJ`.
La première ligne indique le langage utilisé par votre extension, ici, LATEX 2ε. Certaines extensions ne précisent rien à ce niveau et commencent par écrire la seconde ligne directement.

La seconde ligne contient le nom de votre extension ainsi que, facultativement, les informations concernant la date de création, la version et son but, par exemple :

```
\ProvidesPackage{monpackage}
[2011/10/05 v1.00 Extension permettant de faciliter l'écriture des normes]
```

Le nom d'une extension sera de la forme `monpackage.sty`.

7.2 Parties optionnelles

7.2.1 Faire appel à d'autres extensions

Tout comme on peut charger des extensions dans des documents classiques, on peut faire appel à d'autres extensions dans les nôtres avec la commande `\RequirePackage` (l'équivalent de `\usepackage`).

```
\RequirePackage[<options>]{<nom>}
```

Pour l'extension TikZ, on pourra aussi charger les librairies :

```
\RequirePackage{tikz}
\usetikzlibrary{calc,arrows,...}
```

7.2.2 Ne pas charger une extension déjà chargée

Afin de gagner un peu de temps, il est intéressant d'éviter de charger des extensions déjà chargées, et ce avec la commande \@ifpackageloaded.

```
\@ifpackageloaded{nom du package}
{\typeout{Le package <nom> n'a pas été chargé une seconde fois.}}
{\RequirePackage{nom du package}}
```

Remarquez ici la commande \typeout qui permet décrire dans le fichier log compilé.

7.2.3 Tester si un fichier existe

Comme nous l'avons vu pour les documents classiques, une façon s'alléger le code d'une extension est d'inclure des fichiers externes avec la commande \input.
Pour vérifier que le fichier externe existe bien, on pourra utiliser la commande \InputIfFileExists.

```
\InputIfFileExists{nom du fichier}
{\typeout{Le fichier de configuration a été trouvé et est utilisé.}}
{\typeout{Le fichier de configuration n'a pas été trouvé.}}
```

7.2.4 Afficher un message d'erreur

Quand on créé des extensions, il faut éviter à tout prix que nos commandes rentrent en conflit avec d'autres définies dans d'autres packages. On pourra donc créer une macro qui vérifie si la commande est déjà définie et, dans un tel cas, afficher un message d'erreur :

```
\def\PERSO@newmacro#1{%
\ifdefined#1%
\errmessage{Package systeme Error: la macro \string#1\space existe.
Contactez moi.}%
\fi
\def#1%
}
```

7.2.5 Déclaration d'options

Comme nous l'avons vu précédemment, dans un document TEX, on peut appeler des extensions avec des options. Par exemple :

```
\usepackage[table,svgnames]{xcolor}
```

Mais comment déclarer des options dans notre extension ? Et bien, avec les commandes \newif et \DeclareOption et \ProcessOptions.

```
\newif\if@svgnames
\DeclareOption{svgnames}{\@svgnamestrue}
\newif\if@table
\DeclareOption{table}{\@tabletrue}
\ProcessOptions
\if@svgnames
  Code si l'option "svgnames" apparaît
\else
  Code si l'option "svgnames" n'apparaît pas
\fi
```

Dans une extension, les noms peuvent comporter le caractère « @ » contrairement aux documents TEX. Si on veut manipuler des noms de commandes comportant le caractère « @ » dans un document TEX, il faut encadrer ces noms avec les commandes \makeatletter et \makeatother.

7.3 Créer une commande avec options

Nous allons maintenant nous tourner vers l'extension xkeyval car elle permet d'attribuer des options aux commandes que nous allons créer. Voyons cela sur un exemple :

```
\define@cmdkey [EX] {cercle} {r}{} % On déclare le rayon "r"
\define@cmdkey [EX] {cercle} {x}{} % On déclare l'abscisse du centre
\define@cmdkey [EX] {cercle} {y}{} % On déclare l'ordonnée du centre
% On déclare le booléen pour savoir si on écrit le rayon
\define@boolkey[EX] {cercle} {rayon}[true]{}
\presetkeys [EX] {cercle} {
r = 2, % Par défaut, le rayon vaudra 2cm
x = 0, % Par défaut, les coordonnées seront (0,0)
y = 0,
rayon = false % Par défaut, le rayon ne sera pas affiché
}{}
\newcommand*\cercle[2][]{
\setkeys[EX]{cercle}{#1}
\draw (\cmdEX@cercle@x,\cmdEX@cercle@y) circle (\cmdEX@cercle@r cm);
\ifEX@cercle@rayon
\draw (\cmdEX@cercle@x,\cmdEX@cercle@y) --
(\cmdEX@cercle@x+\cmdEX@cercle@r,\cmdEX@cercle@y)
node[midway,above] {\cmdEX@cercle@r cm};
\fi
\fill (\cmdEX@cercle@x,\cmdEX@cercle@y) circle (1pt);
\node[below] at (\cmdEX@cercle@x,\cmdEX@cercle@y) {#2};
}
```

Analysons ce code :

La commande `\define@cmdkey` créée une option (par exemple, l'option « r » pour la commande « cercle »), ce qui est mis entre crochets étant peu important (cela désigne un préfixe que l'on utilisera par la suite).

La commande `\define@boolkey` créée une option booléenne (valant `true` ou `false`) de la même façon que la commande précédente.

La commande `\presetkeys` attribue des valeurs par défaut aux options créées.

La commande `\setkeys` se met à l'intérieur de la commande pour prendre en compte les options créées.

Si on a déclaré une option comme ceci :

```
\define@cmdkey[PREFIXE]{macroname}{optionname}{}
```

Alors, on pourra faire appel à cette commande par son nom :

```
\cmdPREFIXE@macroname@optionname
```

Dans le cas d'une option booléenne, si on l'a déclaré ainsi :

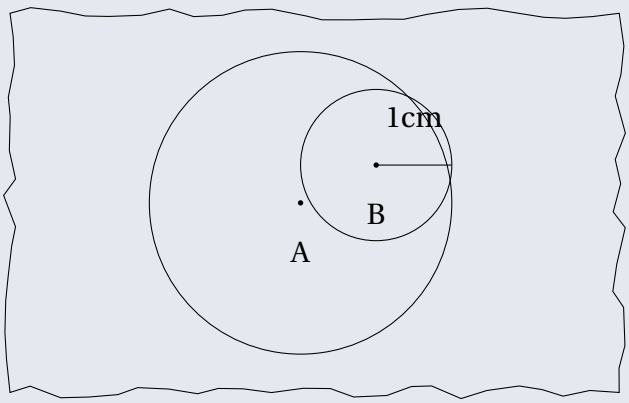
```
\define@boolkey[PREFIXE]{macroname}{optionname}[true]{}
```

On pourra tester sa valeur ainsi :

```
\ifPREFIXE@macroname@optionname
Code si l'option booléenne vaut "true"
\else
Code si l'option booléenne vaut "false"
\fi
```

Regardons maintenant comment on appelle la commande créée par le code précédent :

```
\begin{tikzpicture}
\cercle{A}
\cercle[x=1,y=.5,r=1,rayon]{B}
\end{tikzpicture}
```



7.4 Créer un environnement avec options

Voici un exemple trivial pour comprendre :

```
\define@cmdkey [EX] {environ} {color}{}
\presetkeys [EX] {environ} {color=black}{}

\newenvironment*{environ}[1][]
{
\setkeys[EX]{environ}{#1}
\color{\cmdEX@environ@color}
}
{
\color{black}
}
```

Voyons maintenant son application :

```
\begin{environ}[color=cyan]
Ce texte est dans l'environnement
défini.
\end{environ}

Ce texte est hors de l'environnement
défini.
```

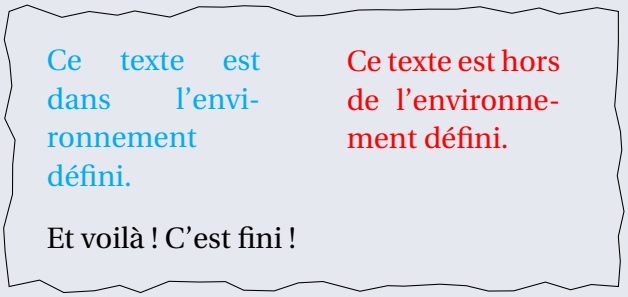
Ce texte est dans l'environnement défini.
Ce texte est hors de l'environnement défini.

Si on souhaite mettre un argument à l'environnement, il suffit de remplacer « 1 » par « 2 » :

```
\define@cmdkey [EX] {environbis} {color}{}
\presetkeys [EX] {environbis} {color=black}{}

\newenvironment*{environbis}[2][]
{
\setkeys[EX]{environbis}{#1}
\begin{minipage}[t][][t]{#2}
\color{\cmdEX@environbis@color}
}
{
\end{minipage}
}
```

Voyons le résultat :

```
\begin{environbis}[color=cyan]
{.4\textwidth}
Ce texte est dans l'environnement
défini.
\end{environbis}
\hspace*{.1\textwidth}
\begin{environbis}[color=red]
{.4\textwidth}
Ce texte est hors de l'environnement
défini.
\end{environbis}
\vskip 5mm
Et voilà ! C'est fini !
```



Foire aux codes

Ce chapitre est dédié à divers codes qui pourraient vous aider, mis sans ordre précis.

8.1 Définir une commande subsubsubsection

```
\documentclass{book}
\usepackage{hyperref}
\setcounter{secnumdepth}{4}
\setcounter{tocdepth}{4} % change niveau max. dans la table des matières
\makeatletter
\newcounter{subsubsubsection}[subsubsection]
\renewcommand\thesubsubsubsection{\thesubsubsection .\@alph
\c@subsubsubsection}
\newcommand\subsubsubsection{\@startsection{subsubsubsection}{4}{\z@}
{-3.25ex\@plus -1ex \@minus -.2ex}
{1.5ex \@plus .2ex}
{\normalfont\normalsize\bfseries}}
\renewcommand\paragraph{\@startsection{paragraph}{5}{\z@}
{3.25ex \@plus1ex \@minus.2ex}
{-1em}
{\normalfont\normalsize\bfseries}}
\renewcommand\subparagraph{\@startsection{subparagraph}{6}{\parindent}
{3.25ex \@plus1ex \@minus .2ex}
{-1em}
{\normalfont\normalsize\bfseries}}
\newcommand*\l@subsubsubsection{\@dottedtocline{4}{10.0em}{4.1em}}
\renewcommand*\l@paragraph{\@dottedtocline{5}{10em}{5em}}
\renewcommand*\l@subparagraph{\@dottedtocline{6}{12em}{6em}}
\newcommand*{\subsubsubsectionmark}[1]{}
\def\toclevel@subsubsubsection{4}
\def\toclevel@paragraph{5}
\def\toclevel@subparagraph{6}
\makeatother
\begin{document}
...
\end{document}
```

Source : http://forum.mathematex.net/latex-f6/subsubsub-t4031.html

8.2 Écrire un texte ombré dégradé avec TikZ

```
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[french]{babel}
\usepackage{tikz}
\usetikzlibrary{fadings}
\begin{document}
\begin{tikzfadingfrompicture}[name=degrade]
\node[text=transparent!20] at (0,0)
{\fontfamily{anttlc}\selectfont\Huge\bfseries
 Exemple};
\end{tikzfadingfrompicture}
\begin{tikzpicture}
\node at (0.03,-0.03)
{\fontfamily{anttlc}\selectfont\Huge\bfseries
Exemple};
\node[inner sep=0cm] (textbox) at (0,0)
{\phantom{\fontfamily{anttlc}\selectfont\Huge
\bfseries Exemple}};
\shadedraw[path fading=degrade,fit fading=false,
bottom color=DarkRed!30,top color=DarkRed,right]
(textbox.south west) rectangle (textbox.north east);
\end{tikzpicture}
\end{document}
```

Exemple

8.3 Barrer d'un trait ou d'une croix un bloc

```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{shapes.misc}
\makeatletter
\newcommand{\tikz@bcancel}[1]
{
\begin{tikzpicture}[baseline=(textbox.base),inner sep=0pt]
\node[strike out,draw] (textbox) {#1};
\useasboundingbox (textbox);
\end{tikzpicture}
}
\newcommand{\strikecancel}[1]
{
\relax\ifmmode
\mathchoice{\tikz@bcancel{$\displaystyle#1$}}
{\tikz@bcancel{$\textstyle#1$}}
{\tikz@bcancel{$\scriptstyle#1$}}
{\tikz@bcancel{$\scriptscriptstyle#1$}}
```

```
\else % "\strut" insère un espace vertical correspondant à 1 ligne
\tikz@bcancel{\strut#1}
\fi
}
\newcommand{\tikz@xcancel}[1]
{
\begin{tikzpicture}[baseline=(textbox.base),inner sep=0pt]
\node[cross out,draw] (textbox) {#1};
\useasboundingbox (textbox);
\end{tikzpicture}
}
\newcommand{\crosscancel}[1]
{
\relax\ifmmode
\mathchoice{\tikz@xcancel{$\displaystyle#1$}}
{\tikz@xcancel{$\textstyle#1$}}
{\tikz@xcancel{$\scriptstyle#1$}}
{\tikz@xcancel{$\scriptscriptstyle#1$}}
\else
\tikz@xcancel{\strut#1}
\fi
}
\makeatother
\begin{document}
\bcancel{Une phrase barrée en diagonale} \\
\xcancel{Une phrase barrée avec une croix} \\
$\bcancel{5+\frac{4^3}{1+7}}$
\[\xcancel{5+\frac{4^3}{1+7}} \]
\end{document}
```

Une phrase barrée en diagonale

$$5+\cancel{\frac{4^3}{1+7}}$$

~~Une phrase barrée avec une croix~~

$$\cancel{5+\frac{4^3}{1+7}}$$

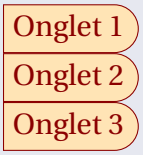
8.4 Du texte avec un reflet avec TikZ

```
\documentclass{article}
\usepackage[french]{babel}
\usepackage{tikz}
\usetikzlibrary{shadows}
\newcommand{\textreflect}[3]
{%
\parbox[c]{5ex}
{
\begin{tikzpicture}[text=#2]
\node[at={(0,0)},yshift=1ex,yslant=#3]{%
\LARGE \bfseries #1};
\node[above,at={(0,0)},yslant=#3,yscale=-1,
scope fading=south,
opacity=0.5]{\LARGE \bfseries #1};
\end{tikzpicture}
}%
}
\begin{document}
\textreflect{1}{red}{0}
\textreflect{2}{black!85}{0.3}
\textreflect{3}{green}{0.5}
\textreflect{C'est beau !}{LightSkyBlue}{0.2}
\end{document}
```

Source :
http://forum.mathematex.net/latex-f6/chiffre-avec-reflet-t11776.html

8.5 Créer des onglets avec TikZ

```
\documentclass{article}
\usepackage[french]{babel}
\usepackage[svgnames]{xcolor}
\usepackage{tikz}
\usetikzlibrary{shapes.misc}
\begin{document}
\begin{tikzpicture}
[every node/.style={draw,rounded rectangle,rounded
rectangle left arc=none,fill=Moccasin,draw=DarkRed,
text=DarkRed}]
\node (o1) {Onglet 1};
\node[below right] (o2) at (o1.south west) {Onglet 2};
\node[below right] (o3) at (o2.south west) {Onglet 3};
\end{tikzpicture}
\end{document}
```



Pour des onglets horizontaux, il va falloir fabriquer une commande spéciale car TikZ n'offre pas d'option `rectangle bottom arc=none`.

```
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[french]{babel}
\usepackage[T1]{fontenc}
\usepackage[svgnames]{xcolor}
\usepackage{tikz}
\newcommand{\tkzonglet}[4]
{%
\node[right] (texte) #2 {#1};
\filldraw[rounded corners,fill=#3,draw=#4]
(texte.south west) -- (texte.north west) --
(texte.north east) -- (texte.south east);
\draw[color=#4] (texte.south east) --
(texte.south west);
\node[right,text=#4] (texte) at (texte.west) {#1};
}
\begin{document}
\begin{tikzpicture}
\tkzonglet{Onglet 1}{at (0,0)}{Moccasin}{DarkRed}
\tkzonglet{Onglet 2}{at (texte.east)}{Moccasin}
{DarkRed}
\tkzonglet{Onglet 3}{at (texte.east)}{Moccasin}
{DarkRed}
\end{tikzpicture}
\end{document}
```

8.6 Résoudre un conflit entre deux packages

Il n'est pas rare d'être confronté à ce genre de problème : on fait appel à deux extensions qui définissent chacune de leur côté une même commande. Dans ce cas, la compilation n'aboutit pas.

Une première solution consiste à regarder si la commande qui est définie dans une des extensions (celle à laquelle on fait appel uniquement pour cette commande) ne peut pas être définie manuellement (on peut regarder le code du fichier `sty`).

Si les deux extensions doivent être impérativement chargées, alors on peut faire appel au package `savesym` comme dans l'exemple suivant :

```
\documentclass[a4paper,oneside,11pt]{article}
\usepackage{savesym}
\usepackage{amsmath} % ce package définit la commande \iint
\savesymbol{iint}    % renomme temporairement \iint en \origiint

\usepackage{wasysym} % ce package définit aussi la commande \iint
\restoresymbol{wasy}{iint}  % renomme le \iint de wasysym en \wasyiint et
% restaure le \iint d'AMS
```

```
9
10 \usepackage[francais]{babel}
11
12 \begin{document}
13 \verb+$\iint$+ : $\iint$ % c'est celui d'AMS
14 \verb+$\wasysymiint$+ : $\wasysymiint$\\
15 \end{document}
```

Origine du code :
http://forum.mathematex.net/latex-f6/conflit-entre-packages-t4062.html

Une seconde façon de faire est :

```
1 \documentclass[a4paper,oneside,11pt]{article}
2
3 % renomme le symbole #2 en #1#2 et invalide le symbole initial #2:
4 \def\renamesymbol#1#2
5 {
6 \expandafter\let\expandafter\newsym\expandafter=\csname#2\endcsname
7 \expandafter\global\expandafter\let\csname#1#2\endcsname=\newsym
8 \expandafter\global\expandafter\let\csname#2\endcsname=\origsym
9 }
10
11 \usepackage{amsmath}
12 \renamesymbol{AMS}{iint}
13 % \iint (de AMS) est devenu \AMSiint
14
15 \usepackage{wasysym}
16 \renamesymbol{wasysym}{iint}
17 % \iint (de wasysym) est devenu \wasysymiint
18
19 \usepackage[francais]{babel}
20 \usepackage[latin1]{inputenc}
21 \usepackage[T1]{fontenc}
22
23 \begin{document}
24 \verb+$\AMSiint$+ : $\AMSiint$\\
25 \verb+$\wasysymiint$+ : $\wasysymiint$\\
26 % \verb+$\iint$+ : $\iint$
27 % la ligne ci-dessus ne compile pas, puisque \iint n'existe plus
28 \end{document}
```

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X