

# Initiation à L<sup>A</sup>T<sub>E</sub>X

*Pour débutants ou  
jeunes utilisateurs*

Par Adrien BOUZIGUES  
dit Indignation 13 CL215

13 juillet 2016

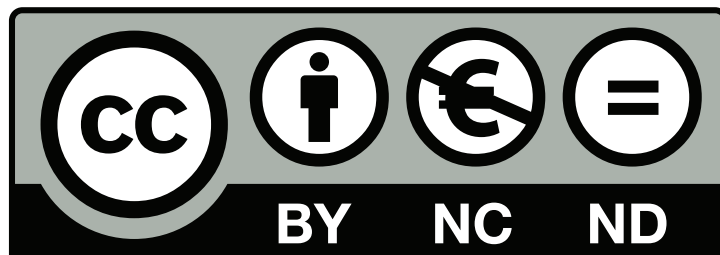
**Version à jour du 26 décembre 2017**



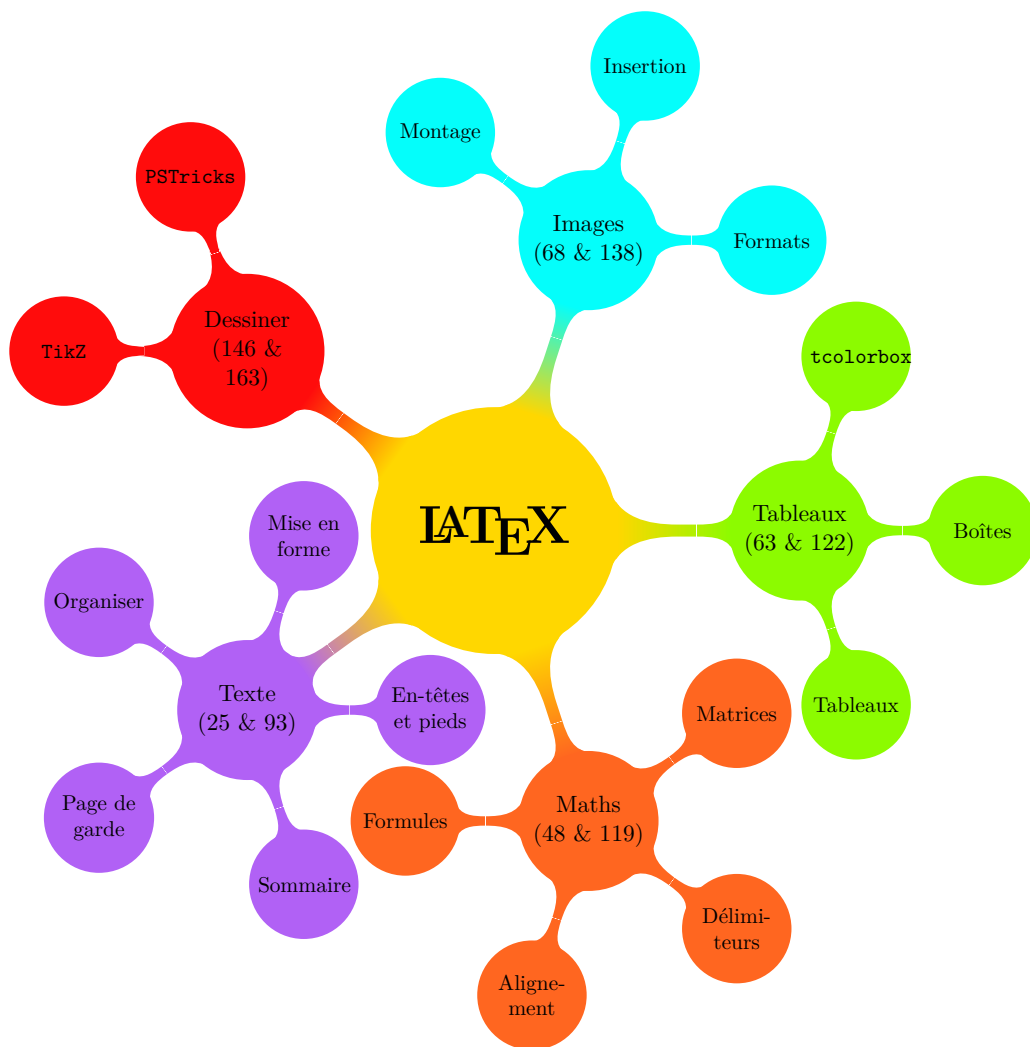
Cette œuvre, création, site ou texte est sous licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Pour accéder à une copie de cette licence, merci de vous rendre à l'adresse suivante <http://creativecommons.org/licenses/by-nc-nd/4.0/> ou d'envoyer un courrier à :

Creative Commons  
444 Castro Street, Suite 900  
Mountain View, California, 94041  
USA



**Toute version de ce guide est soumise à cette licence Creative Commons, y compris les plus anciennes qui peuvent circuler et qui n'y font pas explicitement mention.**



# Sommaire

|  |           |
|--|-----------|
| <b>Préambule</b>   | <b>7</b>  |
| <b>I Prise en main de Texmaker</b>                                     | <b>9</b>  |
| <b>1 Installation de L<sup>A</sup>T<sub>E</sub>X</b>                   | <b>10</b> |
| 1.1 Pourquoi utiliser L <sup>A</sup> T <sub>E</sub> X ? . . . . .      | 10        |
| 1.2 Installation de MiKTeX . . . . .                                   | 12        |
| 1.3 Installation de Texmaker . . . . .                                 | 13        |
| 1.4 Vérification finale . . . . .                                      | 14        |
| <b>2 Démarrage avec Texmaker</b>                                       | <b>15</b> |
| <b>II Démarrer sous L<sup>A</sup>T<sub>E</sub>X</b>                    | <b>17</b> |
| <b>3 Les règles de base</b>  | <b>18</b> |
| 3.1 Les règles pour faire du L <sup>A</sup> T <sub>E</sub> X . . . . . | 18        |
| 3.2 Les 3 règles d'or en L <sup>A</sup> T <sub>E</sub> X . . . . .     | 19        |
| 3.3 La base d'un document L <sup>A</sup> T <sub>E</sub> X . . . . .    | 19        |
| 3.4 Les packages . . . . .   | 22        |
| <b>4 Gestion du texte et mise en forme</b>                             | <b>25</b> |
| 4.1 Notre premier texte . . . . .                                      | 25        |
| 4.2 Un peu de mise en forme . . . . .                                  | 28        |
| 4.3 Organiser son document . . . . .                                   | 30        |
| 4.4 Gestion du sommaire . . . . .                                      | 34        |
| 4.5 La page de garde . . . . .   | 36        |
| 4.6 Création de commandes . . . . .                                    | 39        |
| 4.7 Les listes à puces . . . . .                                       | 42        |
| 4.8 Une petite touche de couleur ? . . . . .                           | 45        |

|            |  |            |
|------------|--|------------|
| <b>5</b>   | <b>Les mathématiques sous <math>\LaTeX</math></b>            | <b>48</b>  |
| 5.1        | Le mode mathématiques . . . . .                              | 48         |
| 5.2        | Les espaces insécables . . . . .                             | 50         |
| 5.3        | Des exemples de formules . . . . .                           | 51         |
| 5.4        | L’affichage et les délimiteurs . . . . .                     | 54         |
| 5.5        | Les matrices . . . . .                                       | 57         |
| 5.6        | Aligner des équations . . . . .                              | 59         |
| <b>6</b>   | <b>Les tableaux et boîtes sous <math>\LaTeX</math></b>       | <b>63</b>  |
| 6.1        | Les tableaux . . . . .                                       | 63         |
| 6.2        | Les boîtes . . . . .   | 65         |
| <b>7</b>   | <b>Insérer des images</b>                                    | <b>68</b>  |
| 7.1        | Les formats d’images . . . . .                               | 68         |
| 7.2        | Les longueurs . . . . .                                      | 70         |
| 7.3        | Insérer une image . . . . .                                  | 71         |
| 7.4        | Les références . . . . .                                     | 74         |
| 7.5        | Un peu de montage . . . . .                                  | 77         |
| <b>8</b>   | <b>Traitement des erreurs</b>                                | <b>80</b>  |
| <b>III</b> | <b>Aller plus loin avec <math>\LaTeX</math></b>              | <b>83</b>  |
| <b>9</b>   | <b>Les modes de compilation sous <math>\LaTeX</math></b>     | <b>86</b>  |
| 9.1        | Présentation des différents modes . . . . .                  | 86         |
| 9.2        | Utilisation des différents modes de compilation . . . . .    | 88         |
| 9.3        | Bilan . . . . .  | 90         |
| <b>10</b>  | <b>Texte et mise en forme</b>                                | <b>93</b>  |
| 10.1       | Changer la police d’écriture . . . . .                       | 93         |
| 10.2       | Inclure des fichiers PDF . . . . .                           | 98         |
| 10.3       | En-têtes et pieds de page . . . . .                          | 101        |
| 10.4       | Centrer verticalement du texte . . . . .                     | 105        |
| 10.5       | Générer une bibliographie . . . . .                          | 106        |
| 10.6       | Générer un index . . . . .                                   | 113        |
| <b>11</b>  | <b>Mathématiques</b>   | <b>119</b> |
| <b>12</b>  | <b>Tableaux &amp; boîtes</b>                                 | <b>122</b> |
| 12.1       | Un autre format de cellules . . . . .                        | 122        |
| 12.2       | Cellules centrées verticalement et horizontalement . . . . . | 123        |

|           |   |            |
|-----------|---|------------|
| 12.3      | Fusion et coloriage de cellules . . . . .                           | 125        |
| 12.4      | <code>longtable</code> & <code>booktabs</code> . . . . .            | 127        |
| 12.5      | Créer sa propre boîte . . . . .                                     | 129        |
| 12.6      | Afficher du code <code>L<sup>A</sup>T<sub>E</sub>X</code> . . . . . | 131        |
| <b>13</b> | <b>Images</b>   | <b>138</b> |
| 13.1      | Une référence toute prête . . . . .                                 | 138        |
| 13.2      | L’environnement <code>subfigure</code> . . . . .                    | 139        |
| 13.3      | Insérer un grand nombre de fichiers . . . . .                       | 141        |
| 13.4      | Insérer un fichier <code>.svg</code> . . . . .                      | 144        |
| <b>14</b> | <b>Dessiner avec <code>PSTricks</code></b>                          | <b>146</b> |
| 14.1      | Fonctionnement général . . . . .                                    | 146        |
| 14.2      | Dessiner des circuits électriques . . . . .                         | 147        |
| 14.3      | Dessiner tout court . . . . .                                       | 150        |
| 14.4      | Éclair . . . . .  | 153        |
| 14.5      | Tête . . . . .  | 154        |
| 14.6      | Utiliser des coordonnées . . . . .                                  | 155        |
| 14.7      | Des boîtes pour le texte . . . . .                                  | 159        |
| 14.8      | Réaliser des intersections . . . . .                                | 159        |
| 14.9      | Extraction du contour d’une image . . . . .                         | 160        |
| <b>15</b> | <b>Dessiner avec <code>TikZ</code></b>                              | <b>163</b> |
| 15.1      | Démarrer sous <code>TikZ</code> . . . . .                           | 163        |
| 15.2      | Un polygone régulier . . . . .                                      | 167        |
| 15.3      | Automatiser les dessins . . . . .                                   | 169        |
| 15.4      | Dessiner des figures mathématiques . . . . .                        | 175        |
| 15.5      | Gestion des styles . . . . .  | 175        |
| 15.6      | Insérer du texte . . . . .  | 178        |
| 15.7      | Création d’un organigramme . . . . .                                | 182        |
| 15.8      | Les possibilités offertes par <code>TikZ</code> . . . . .           | 188        |
| 15.9      | Le fond d’écran <code>L<sup>A</sup>T<sub>E</sub>X</code> . . . . .  | 191        |
| <b>16</b> | <b>Faire des présentations avec <code>Beamer</code></b>             | <b>193</b> |

# Préambule

Ce guide a tout d'abord été construit pour mon usage personnel afin de regrouper toutes mes connaissances en  $\LaTeX$ . Il sert aussi à mes camarades de promotion qui désirent se mettre à  $\LaTeX$  (et prennent donc le chemin de la vérité absolue).

Accessoirement, dans l'éventualité où un parfait inconnu viendrait à lire ce guide, j'espère qu'il pourra l'aider à son tour dans son initiation à  $\LaTeX$ . Et si jamais, parfait inconnu, tu trouves que mon point de vue sur  $\LaTeX$  est quelque peu extrémiste - mais tu as déjà dû t'en rendre compte quelques lignes au-dessus, n'est-ce pas? -, je te serai reconnaissant de ne pas me juger.

Effectivement, quand tu te rendras compte, après quelques pages, des possibilités offertes par  $\LaTeX$ , tu ne trouveras pas mon point de vue si obtus et tu en viendras, toi aussi, à détester Word ...

Enfin, tu risques d'apercevoir par la suite que mes connaissances  $\LaTeX$  restent malgré tout limitées. N'hésite donc pas à aller te documenter ailleurs si un point ne te semble pas clair ou si tu cherches d'autres informations. Je n'ai pas la science infuse. Je propose juste des solutions qui fonctionnent.

Je recommande au passage l'excellent  *$\LaTeX$  ... pour un prof' de maths !* d'Arnaud GAZAGNES. Et si jamais tu désires retrouver ce guide à jour ainsi que les fichiers d'aide que j'ai récoltés, ils sont disponibles à l'adresse suivante :

[http://drive.google.com/drive/folders/  
0BzU2BdcGjfU5Tk1XaXhxbk5JcEE?usp=sharing](http://drive.google.com/drive/folders/0BzU2BdcGjfU5Tk1XaXhxbk5JcEE?usp=sharing)

N'hésite pas à laisser des commentaires ou à signaler des fautes. Sur ce, bonne lecture et bon courage!

Adrien BOUZIGUES  
I13 CL215



Lien de mon Drive  $\text{\LaTeX}$  ... au format QR Code!





**Première partie**  
**Prise en main de Texmaker**

# Chapitre 1

## Installation de L<sup>A</sup>T<sub>E</sub>X

Avant de commencer, je tiens à te rappeler que la version à jour de ce guide est normalement disponible à l'adresse suivante :

[http://drive.google.com/drive/folders/  
0BzU2BdcGjfU5Tk1XaXhxbk5JcEE?usp=sharing](http://drive.google.com/drive/folders/0BzU2BdcGjfU5Tk1XaXhxbk5JcEE?usp=sharing)

Tu y trouveras non seulement divers guides sur L<sup>A</sup>T<sub>E</sub>X ou des packages importants que j'ai récoltés à droite à gauche, mais aussi mon fichier type `Rapport_type.tex` qui peut se révéler utile.

De plus, je suppose que tu utilises un système d'exploitation **Windows**. Dans le cas contraire, un utilisateur **Linux** devrait savoir se débrouiller pour tout installer.

Quant à un utilisateur d'**Apple**, ~~je considère déjà ta cause perdue~~ tu trouveras des équivalents grâce à Google ... enfin, c'est ce que je disais initialement. Désormais, les programmes que je présente par la suite possèdent aussi une version **Mac**.

Mais avant de démarrer, je tiens à partager une expérience **LinkedIn** que je trouve pertinente et qui peut éventuellement convaincre les récalcitrants à utiliser L<sup>A</sup>T<sub>E</sub>X.

### 1.1 Pourquoi utiliser L<sup>A</sup>T<sub>E</sub>X ?

Durant l'été 2017, j'ai posé la question suivante sur le groupe « TeX / LaTeX User Group » de **LinkedIn** :



### La question posée

## LaTeX professional experience

Hello everybody,

I'm actually an engineering student and one of my main hobbies is writing stuffs in LaTeX (scientific reports, lessons' synthesis, letter, ..., even a LaTeX manual user for beginners (in French)!). I was wondering if LaTeX is really helpful, in daily life, at work.

So, if anyone would like to share his opinion/experience, about how he uses LaTeX at work (or not), feel free to answer my message.

Thanks a lot and have a good summer,

J'espère pour toi que l'anglais n'est pas une contrainte car c'est loin d'être fini. Si toutes les réponses sont intéressantes, je trouve mon guide un peu terni par ces 6 pages de réponses ... Je vais donc faire un petit résumé :

- certains pensent qu'utiliser L<sup>A</sup>T<sub>E</sub>X est pertinent uniquement dans un milieu académique ou scientifique (recherche, surtout pour les mathématiques),
- beaucoup travaillent avec des gens qui fonctionnent exclusivement sous Word. Toutefois, pour la diffusion de notes internes, l'utilisation de L<sup>A</sup>T<sub>E</sub>X est appréciée (clarté du message, mise en page propre, simplicité, ...),
- beaucoup reconnaissent que L<sup>A</sup>T<sub>E</sub>X possède une forte courbe d'apprentissage, surtout au début<sup>1</sup>. Toutefois, ils utilisent aussi L<sup>A</sup>T<sub>E</sub>X dans leur quotidien (lettres, CV, rendus, ...) car ils préfèrent sa facilité d'utilisation par rapport à Word une fois que le premier est maîtrisé,
- quasiment tous considèrent qu'apprendre à utiliser L<sup>A</sup>T<sub>E</sub>X n'est pas une perte de temps et peut se révéler utile.

Si tu n'es pas convaincu ou si tu crains que j'ai truqué les réponses, laisse-moi au moins en partager deux, que tu puisses te faire une idée :

1. Mais je te rassure, ce guide est justement conçu pour t'aider à passer ce cap difficile



### Les 2 réponses les plus pertinentes à mon sens

- ❖ **Ed Blackburne** : I use LaTeX everyday at work. My responsibilities include the production of Model Validation on reports per SR11-7. These are (generally) very technical and must be compliant with our Enterprise standards as well as regulatory guidance. Although many of my colleagues use MS Word, my team enjoys increased productivity from LaTeX.

Additionally, for the econometric models, my team utilizes R/knitR/LaTeX to create dynamic reports (using methods borrowed from reproducible research techniques).

I have created company-specific memo templates that I use on a daily basis, as well.

If you write technical documents and/or need references (that work) I highly encourage investing the minimal effort to become a competent LaTeX user.

- ❖ **Brian Dunn** : While LaTeX has a learning curve to use it well, so does MS Word or LibreOffice Writer, many people never use a word processor's formatting "styles", for example, and instead manually format everything.

In talking with people at industrial trade shows, I occasionally come across a company which uses LaTeX for their documentation. Usually they are small engineering operations, and often European. Most places use poorly-formatted MS-Word generated documentation, or else InDesign when they want a professional image. I also found that companies which are suffering are not interested in improving their documentation, sales literature, or websites, even though their competitors which are doing well have very nice public-facing literature.

Convaincu cette fois ? Pas vraiment ? Tu hésites encore ? Dans ce cas, continuons sur notre lancée et installons L<sup>A</sup>T<sub>E</sub>X sur notre ordinateur. Tu ne peux pas savoir avant d'essayer, n'est-ce pas ?

## 1.2 Installation de MiKTeX

MiKTeX est une distribution L<sup>A</sup>T<sub>E</sub>X. Si tu te demandes ce qu'est une distribution et si tu es intéressé par la définition, je te laisse aller chercher sur



Google (ou cliquer [ici](#)<sup>2</sup>).

Le seul élément à retenir est le suivant : MiKTeX permet de transformer tes futures lignes de code L<sup>A</sup>T<sub>E</sub>X en un PDF propre et lisible par tous.

Pour commencer, il faut aller sur : <http://miktex.org/download> pour télécharger l'exécutable. Il faut ensuite simplement suivre les instructions d'installation. **Je recommande de laisser les options par défaut** : en tant que débutant, elles conviennent parfaitement et permettent d'éviter tout problème par la suite (fâcheuse expérience personnelle qui m'a conduit à tout désinstaller pour ensuite tout réinstaller ...).

### *Nota Bene*

Je tiens à préciser que je n'ai aucun revenu financier grâce à MiKTeX. Je conseille cette distribution car c'est celle que j'utilise et qui fonctionne parfaitement pour ma part.

Il a aussi l'avantage de proposer un gestionnaire de packages (nous verrons en temps et en heure ce dont il s'agit) : MiKTeX Package Manager.

En revanche, tu es libre de choisir la distribution de ton choix et d'en prendre une autre. À toi d'en trouver une sur Internet : il y a du choix.

## 1.3 Installation de Texmaker

Techniquement, cette étape n'est pas nécessaire car tu pourrais écrire ton fichier L<sup>A</sup>T<sub>E</sub>X dans un bloc-note si le cœur t'en dit. Cependant, le code sera plus compliqué à relire, il faut taper toutes les commandes à la main et il faut indiquer à Windows de transformer ton code en PDF grâce à MiKTeX.

Avec Texmaker, tous ces tracasseries sont épargnés : tu as à disposition un éditeur de fichiers L<sup>A</sup>T<sub>E</sub>X performant, un système d'auto-complétion des formules fort pratique et agréable et toutes les commandes pour utiliser MiKTeX sont intégrées et faciles à utiliser.

Pour cela, il faut aller sur le site de Texmaker : <http://www.xm1math.net/texmaker/download.html>. Je pense que tu sauras télécharger l'exécutable et installer le tout sans trop de difficulté. Comme pour MiKTeX, je te

---

2. À l'avenir, des liens hypertextes pourront être introduits dans ce guide, sous ce format. Naturellement, ils n'ont d'intérêt qu'avec le format numérique de ce guide.



recommande de laisser les options par défaut.

### *Nota Bene*

Même remarque pour **Texmaker** que pour **MiKTeX** : tu peux choisir un autre éditeur L<sup>A</sup>T<sub>E</sub>X, même si celui-ci est vraiment très pratique selon moi.

## 1.4 Vérification finale

Pour s'assurer que tout fonctionne, un fichier `Rapport_type.tex` est normalement disponible avec ce guide (dans mon `Drive LATEX`). Place le fichier dans un dossier (je t'expliquerai pourquoi par la suite) et ouvre-le. **Texmaker** devrait alors se lancer.

Tu verras alors sur l'écran plein de lignes de code qui te semblent obscures. Rassure-toi, tu n'as pas à les comprendre de suite. J'en parle dans tout le reste du guide. L'intérêt ici est juste de vérifier si tout est bien installé. Normalement, tu as juste à appuyer sur la touche `[F6]`. En théorie, la compilation devrait se lancer.

Cependant, il est possible qu'une fenêtre apparaisse. Tu verras des mots comme « package » et « install » écrits par endroits. Là encore, pas d'inquiétude, j'en parle après. Dis-toi qu'il faut tout installer pour ne pas avoir à le faire par la suite.

Une fois les (nombreuses, potentiellement longues et dernières) installations réalisées, une fenêtre **Texmaker** est censée s'ouvrir. Tu verras un aperçu du document L<sup>A</sup>T<sub>E</sub>X créé. Si tu es arrivé jusqu'ici, alors tout est parfaitement installé sur ton ordinateur. Tu peux poursuivre sereinement la suite.

**Dans le cas contraire** (absence du fichier ou problème de fonctionnement), ne perds pas ton temps et passe à la suite. Nous allons rapidement aborder le fonctionnement de **Texmaker**. **Si des problèmes persistent par la suite**, je ne peux que te conseiller de tout désinstaller et de recommencer.

# Chapitre 2

## Démarrage avec Texmaker

Texmaker est un excellent logiciel pour gérer ses fichiers L<sup>A</sup>T<sub>E</sub>X. Et je sais de quoi je parle car, avant de m’y mettre, j’utilisais un autre logiciel, tellement exécrable que j’ai fini par oublier son nom. Aujourd’hui, je ne fais rien sans Texmaker. Voyons un aperçu de ce dernier :

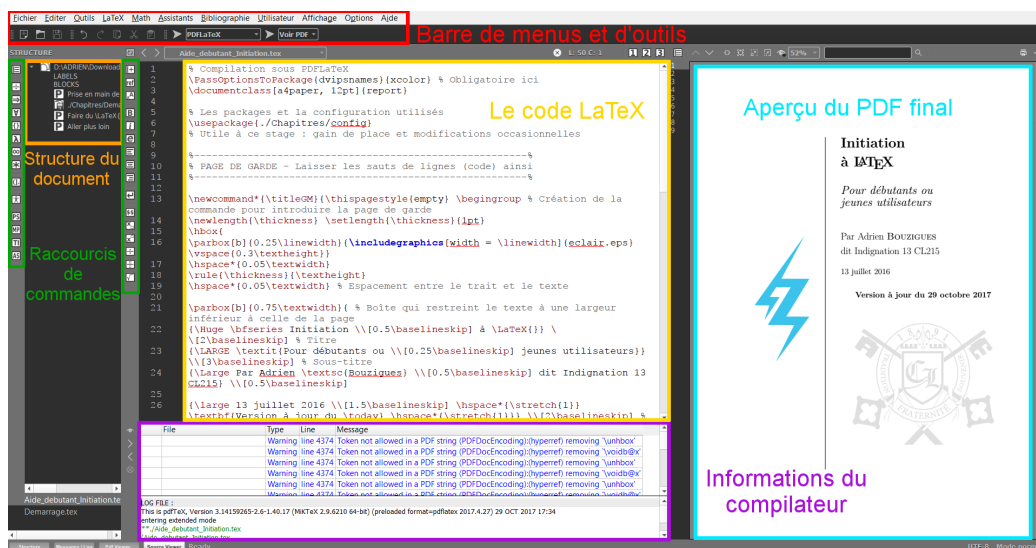


FIGURE 2.1 – La fenêtre Texmaker

Revenons sur chaque point :

- **barre de menus et d’outils** : plein de commandes L<sup>A</sup>T<sub>E</sub>X préremplies. Personnellement, je l’utilise très rarement (y compris le bouton de sauvegarde car je suis un adepte du **ctrl**+**S**),



- **structure du document** : très pratique pour naviguer dans le document en cours,
- **raccourcis de commandes** : encore des commandes. Il peut être intéressant d’y jeter un coup d’œil une fois ce guide bien avancé. Il y a principalement des commandes pour les formules mathématiques et quelques unes pour la mise en forme du texte,
- **code L<sup>A</sup>T<sub>E</sub>X** : c’est ici que tu tapes ton document et les commandes L<sup>A</sup>T<sub>E</sub>X nécessaires pour le mettre en forme,
- **informations du compilateur** : le résultat lors de la génération du PDF. Très utile s’il y a des erreurs pour pouvoir se corriger,
- **aperçu du PDF** : une fenêtre avec l’aperçu du fichier PDF généré.

La dernière option, extrêmement pratique, est disponible ainsi : aller dans “Options” → “Configurer Texmaker” → “Afficheur Pdf” → “Afficheur Pdf interne” + “Intégré à la fenêtre”.

Un bouton “Pdf Viewer” est alors disponible en bas à gauche et te permet d’activer ou non cette fenêtre d’aperçu.

### L’aide en ligne

Si jamais tu as d’autres questions sur Texmaker, son site officiel (<http://www.xm1math.net/texmaker/index.html>) est le meilleur endroit pour avoir des réponses et fournit aussi un tutoriel pour débiter à L<sup>A</sup>T<sub>E</sub>X.

Si certains points de ce guide te semblent obscures, tu peux donc t’y rendre, ainsi que sur <http://fr.wikibooks.org/wiki/LaTeX>.

N’hésite pas quand tu débutes. Les deux sites sont en français et répondent à beaucoup de questions assez facilement (*i.e.* avec un code simple).

Attaquons désormais ce pourquoi tu suis ce guide : faire du L<sup>A</sup>T<sub>E</sub>X.



Deuxième partie

Démarrer sous L<sup>A</sup>T<sub>E</sub>X

# Chapitre 3

## Les règles de base

### 3.1 Les règles pour faire du $\text{\LaTeX}$

Certains pourraient dire que j'ai pété un câble mais les règles générales pour faire du  $\text{\LaTeX}$  restent pragmatiques :

#### Les règles pragmatiques

**Règle n°1** : tout est possible en  $\text{\LaTeX}$ , y compris écrire des partitions de musique ([http://fr.wikibooks.org/wiki/LaTeX/%C3%89crire\\_de\\_la\\_musique](http://fr.wikibooks.org/wiki/LaTeX/%C3%89crire_de_la_musique)).

**Règle n°2** : la règle n°1 est toujours vraie.

**Règle n°3** :  $\text{\LaTeX}$  implique d'écrire des commandes soit des lignes de code. Aérer et ordonner son code en facilite la relecture.

**Règle n°4** : la voie de la perfection en  $\text{\LaTeX}$  passe par une recherche continue sur Internet.

**Règle n°5** : si tu rencontres des difficultés, il ne faut pas hésiter à demander des conseils.



## 3.2 Les 3 règles d'or en L<sup>A</sup>T<sub>E</sub>X

Pour écrire du code L<sup>A</sup>T<sub>E</sub>X, il existe 3 règles, suffisamment importantes à mon sens pour être en or :

### Les 3 règles d'or

**Règle n°1** : toute commande L<sup>A</sup>T<sub>E</sub>X débute par un *backslash* “\” (sous Windows, faire **Alt Gr** + **8**).

**Règle n°2** : tout texte concerné par une commande L<sup>A</sup>T<sub>E</sub>X est délimité par des accolades “{” et “}” (sous Windows, faire **Alt Gr** + **4** et **Alt Gr** + **=**).

**Règle n°3** : toute commande L<sup>A</sup>T<sub>E</sub>X qui comprend un **begin** finit par un **end**. Ce genre de structure s'appelle un **environnement**.

Il s'agit donc, selon moi, de la base en L<sup>A</sup>T<sub>E</sub>X pour écrire du code L<sup>A</sup>T<sub>E</sub>X. Respecter ces règles permet d'éviter un bon nombre d'erreurs, nombreuses quand tu débutes.

Ces 3 règles prendront leur sens sous peu, quand nous allons mettre en forme notre document et commencer à faire du L<sup>A</sup>T<sub>E</sub>X (cf. 4.2 Un peu de mise en forme, p. 28).

## 3.3 La base d'un document L<sup>A</sup>T<sub>E</sub>X

Pour commencer, démarrons un fichier L<sup>A</sup>T<sub>E</sub>X : ouvrir **Texmaker**, créer un nouveau fichier et l'enregistrer au format **.tex**. Pour information, un fichier L<sup>A</sup>T<sub>E</sub>X possède toujours l'extension **.tex**.

**À partir de maintenant, des exemples de code L<sup>A</sup>T<sub>E</sub>X seront fournis dans des encadrés verts et sont normalement fonctionnels.** Toutefois, la copie depuis ce guide au format PDF semble encore présenter quelques lacunes (saut de ligne lors d'une coupure (ligne de code trop longue), apostrophe différente de celle présente sous **Texmaker**, ...). Des erreurs lors de la génération du document PDF peuvent alors survenir ...

À toi de voir si tu préfères recopier chaque ligne de code, ce qui facilite la mémorisation selon moi, ou si tu préfères copier-coller et habilement utiliser



la fonction « Remplacer » de **Texmaker**.

La base d'un document **L<sup>A</sup>T<sub>E</sub>X** est la suivante :

#### La base d'un document **L<sup>A</sup>T<sub>E</sub>X**

```
\documentclass[options]{nom_du_support}

% Préambule

\begin{document}
% Ici s'écrit notre texte
% Notons que le symbole "%" permet de mettre un commentaire
\end{document}

Tout ce que j'écris après \end{document} n'a aucun intérêt et
ne sera pas compilé par LaTeX.
```

Plusieurs points sont à retenir :

- `\documentclass` permet de définir le type de document sur lequel tu vas travailler (j'y reviens ci-après),
- la zone entre `\documentclass` et `\begin{document}` s'appelle le préambule. Je décris cette partie en 3.4 Les packages, p. 22,
- un premier exemple d'illustration de la règle d'or n°3 : un `begin` implique un `end`. Toutes les lignes de code (hormis les commentaires) écrites entre `\begin{document}` et `\end{document}` sont interprétées par **L<sup>A</sup>T<sub>E</sub>X** lors de sa création du fichier PDF final,
- tout ce qui peut être écrit après `\end{document}` n'est pas pris en compte par **L<sup>A</sup>T<sub>E</sub>X**.

J'ai utilisé le terme « compilé » dans ce premier exemple. Il s'agit juste d'une manière de dire que **L<sup>A</sup>T<sub>E</sub>X** transforme les lignes de code en un fichier PDF aussi agréable à lire que ce guide.

Que mettre maintenant dans le `\documentclass`? La partie `options` contient la taille du papier (A4 : `a4paper`, A5 : `a5paper`) ainsi que la taille



de police de base (10, 11 ou 12pt).

Quant à `nom_du_support`, il s'agit du type de document que tu veux rédiger, appelé *classe* sous  $\text{\LaTeX}$  : `report` pour taper des rapports ; `article` pour des articles scientifiques ; `book` pour des livres et `letter`, tu as compris, pour des lettres.

**Personnellement**, je recommande de commencer un nouveau document par `\documentclass[a4paper, 12pt]{report}`. Cela convient pour 90% des cas et le risque de problème est moindre.

Néanmoins, il faut garder à l'esprit que d'autres options de présentation de document existe (`book`, `article` ou `letter`). Ces derniers peuvent toujours servir.

#### Pour aller plus loin

Des explications plus précises et poussées (toutes les options de `report`, `article`, etc.) sont disponibles à [http://fr.wikibooks.org/wiki/LaTeX/Les\\_classes](http://fr.wikibooks.org/wiki/LaTeX/Les_classes).

Ce qu'il faut bien comprendre, c'est que **la forme du document est intrinsèquement liée à sa classe**.

#### Une question ?

« Et si je veux changer la police en 14pt, comment faire ? »

Ah, je vois que le fond de la classe suit. J'aborde ce point en 4.2 Un peu de mise en forme, p. 28.

Enfin, dernier point important : **comment dire à  $\text{\LaTeX}$  de compiler ?** Concrètement, comment dire à `Texmaker` de lancer la génération du fichier PDF ? Dans la partie « Barre de menus et d'outils » de la fenêtre `Texmaker` (cf. FIGURE 2.1 p. 15), deux flèches sont visibles.

Celle de droite doit être réglée sur `Voir PDF` et celle de gauche sur `PDFLaTeX`. **Il s'agit de la compilation qui fonctionne 90% du temps et qui reste suffisante à notre niveau**. Ce mode de compilation est normalement paramétré par défaut avec un appui sur la touche `F1` ; sinon, un appui sur `F6` (lancer la compilation `PDFLaTeX`) suivi de `F7` (afficher le résultat) fonctionne aussi (pour les utilisateurs de `Texmaker`).



### Une question ?

« Dans ce cas, à quoi servent les autres modes ? »

Encore une bonne question ! Pour la faire simple, il peut arriver que des options sous L<sup>A</sup>T<sub>E</sub>X (changer de police d'écriture par exemple) ne passent pas avec ce mode (PDFLaTeX). Tu peux aller te documenter à ce sujet une fois que tu auras pris les rennes de L<sup>A</sup>T<sub>E</sub>X.

Toutefois, avec le recul, je me rends compte qu'il y a généralement moyen de trouver une solution qui fonctionne sous PDFLaTeX. Mais ne précipitons pas tout : nous aurons le temps de revenir sur les modes de compilation plus tard.

### Le conseil personnel

Peu importe le mode de compilation, tu peux remarquer qu'un fichier `.tex` entraîne la génération d'autres fichiers. C'est pourquoi je recommande toujours de travailler avec le fichier `.tex` placé dans un dossier pour éviter de noyer le bureau.

Enfin, ces fichiers sont à conserver le temps de travailler sur un document L<sup>A</sup>T<sub>E</sub>X. **Le seul fichier qui compte est celui avec l'extension `.tex`.** C'est lui qui contient tout le code nécessaire à la compilation et à l'obtention du PDF.

C'est bon ? Toujours là ? Tu verras, avec de la pratique, les bases vont rentrer. Justement, nous allons en faire.

## 3.4 Les packages

Si l'auditoire curieux ne s'est pas encore empressé de faire des essais, je lui recommande d'essayer le code suivant :

### Un premier essai

```
\documentclass[a4paper, 12pt]{report}
```



```

\begin{document}

J'aime écrire en \LaTeX{} !

\end{document}

% N.B. : Les sauts de ligne, c'est important pour la lisibilit
é (règle de base 3)

```

Normalement, tu devrais obtenir : « J'aime écrire en L<sup>A</sup>T<sub>E</sub>X ! ». Analysons le résultat.

La règle d'or n°1 commence à prendre du sens : une commande L<sup>A</sup>T<sub>E</sub>X commence par un backslash (ou contre-oblique pour les puristes) “\”. Cette commande me permet d'écrire le mot « LaTeX » d'une manière plus “raffinée”.

Par contre, aucune trace du “é”. C'est bizarre : moi j'arrive à l'écrire sans souci ! C'est parce que tu n'as pas dit à L<sup>A</sup>T<sub>E</sub>X d'écrire en utf-8 (codage de caractères informatiques, qui tolère les accents : <http://fr.wikipedia.org/wiki/UTF-8>).

Pour ce faire, il faut dire à L<sup>A</sup>T<sub>E</sub>X de charger des options supplémentaires. Dans le jargon L<sup>A</sup>T<sub>E</sub>X, ces options sont appelées des packages. Si tu es un utilisateur de Python, ces derniers sont très similaires aux bibliothèques (ou librairies).

Mais je m'égare. **Les packages sont toujours renseignés dans le préambule**, soit entre `\documentclass` et `\begin{document}`. Pour charger un package, il faut utiliser la commande :

```
\usepackage[options_du_package]{nom_du_package}
```

Dans le cas de la langue française, je recommande de remplir le préambule de la manière suivante. Pour des explications plus poussées, je te renvoie à Internet. Sache juste que le tout fonctionne très bien !

### Un exemple qui fonctionne bien

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

```



```
\usepackage[french]{babel} % Pour la langue
\usepackage{lmodern} % Pour changer le pack de police

\begin{document}

J'aime écrire en \LaTeX{} !

\end{document}
```

### Une question ?

« Pourquoi dire à  $\text{\LaTeX}$  d'aller chercher des options alors que rien n'a été précisé pour la commande  $\text{\LaTeX}\{\}$  ? »

Tout simplement parce qu'il s'agit d'une commande présente de base avec  $\text{\LaTeX}$ . Il n'y a donc pas besoin de charger quoi que ce soit au préalable.

Sache aussi que les packages sont construits par les utilisateurs  $\text{\LaTeX}$ . C'est pourquoi tout est possible avec  $\text{\LaTeX}$  : tout est modifiable ou n'attend qu'à être créé.

C'est bon ? Toujours de la partie ? Dis-toi que, désormais, tu vas enfin pouvoir taper des paragraphes sans avoir à te soucier de quoi que ce soit !



# Chapitre 4

## Gestion du texte et mise en forme

### 4.1 Notre premier texte

Prêt à enfin écrire un roman ? Bon, nous allons poursuivre en douceur. Et, petit à petit, tu auras suffisamment d'outils pour construire quelque chose. Poursuivons avec le code suivant :

#### Un nouvel essai

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\begin{document}

J'aime écrire en \LaTeX{} ! Vraiment ! % saut de ligne

Et toi ?      Qu'en est-il ? \\
% Beaucoup d'espace et un nouveau symbole

Pardon ? Tu débutes ?      Tu vas voir, c'est facile.
% Double saut de ligne
```



```
\end{document}
```

Tu devrais obtenir :

### Le résultat

J'aime écrire en L<sup>A</sup>T<sub>E</sub>X! Vraiment!  
Et toi? Qu'en est-il?

Pardon? Tu débutes? Tu vas voir, c'est facile.

Nous pouvons relever plusieurs points :

- un saut de ligne à l'écran est interprété comme un retour à la ligne,
- la commande `\` permet un véritable saut de ligne,
- les espaces et saut de ligne intempestifs ne sont pas pris en compte,
- les alinéas sont automatiques (pas besoin de faire de tabulations).

Et oui, L<sup>A</sup>T<sub>E</sub>X possède une façon un peu curieuse d'aller à la ligne. Mais c'est aussi sa force : ce qui est écrit à l'écran ne reflète pas la réalité du résultat après compilation.

Ce point peut paraître confus à première vue mais tu te rendras vite compte qu'un fichier `.tex` est compliqué à lire à cause des commandes utilisées mais ce sont ces dernières qui font tout l'intérêt de L<sup>A</sup>T<sub>E</sub>X.

Si les espaces sont naturellement gérés par L<sup>A</sup>T<sub>E</sub>X (inutile donc de mettre du blanc pour compléter la ligne, L<sup>A</sup>T<sub>E</sub>X s'occupe de tout), il peut arriver à l'utilisateur de vouloir prendre la main.

Si le saut de ligne est disponible grâce à la commande `\` (cumulable), l'utilisateur peut jouer sur l'espacement vertical grâce à la commande :

```
\vspace{distance}
```

(v comme pour vertical, space pour espace). De même pour un espacement horizontal avec `\hspace{distance}`.

La distance est totalement libre, à condition de renseigner correctement l'unité. Par exemple :



### Gérer l'espacement

```

\begin{document}

J'aime toujours écrire en \LaTeX{}.

\vspace{2 cm}

Surtout quand je laisse \hspace{3cm} beaucoup de blanc !

% L'espace entre le nombre et l'unité est optionnel
% 3cm est parfaitement interprété par LaTeX

% Le nec plus ultra : \baselineskip = espace correspondant à
%   un saut de ligne
% Donc \\ = \vspace{\baselineskip}

Il est possible de sauter plusieurs \\ \\

lignes \\ \\ \\

ainsi.

\vspace{4\baselineskip} % Quadruple saut de ligne

Cette solution est aussi possible, tout comme \\[3\
  baselineskip]

celle-ci ! Bref, beaucoup de façons de sauter des lignes, de
  manière plutôt concise.

\end{document}

```

### Pour aller plus loin - Les unités de distance

Je te renvoie à <http://en.wikibooks.org/wiki/LaTeX/Lengths#Units> si tu veux plus d'informations à ce sujet. Il n'existe pas que `\baselineskip` comme distance pré-programmée sous  $\text{\LaTeX}$  qui est intéressante.



En revanche, sache qu'il est possible de rentrer des valeurs négatives. C'est surtout pratique pour remonter du texte lors de montages, voire des images si besoin.

### Une question ?

« Que se passe-t-il si je vais juste à la ligne dans mon code  $\LaTeX$  ? »

Rien. Seul un saut de ligne à l'écran compte comme un retour à la ligne.

Cette fois, je crois avoir expliqué suffisamment les bases. Passons à de la mise en forme.

## 4.2 Un peu de mise en forme

Bon, cette fois, nous y sommes. Tu vas commencer à prendre un peu les choses en main. Tu vas aussi comprendre un peu mieux le principe de  $\LaTeX$  : rédiger le fond, avec plein de commandes, tandis que  $\LaTeX$  s'occupe de gérer la forme.

### Gras, italique, accentuation, guillemets, taille de police et remarque

```
% Même préambule pour l'instant
\begin{document}

\textbf{texte en gras} \\

\textit{texte en italique} \\

\textsc{Texte en Petites Majuscles} \\

\'E, \'E, \^E, {\oe}il, c{\oe}ur \\
% accent sur les majuscules et o pris dans e

\log texte entre guillemets \fg{} \\
```



```

“guillemets anglais” \\

% Différentes tailles de texte
{\tiny tiny} {\scriptsize scriptsize} {\footnotesize
  footnotesize} {\small small} {\normalsize normalsize} {\
  large large} {\Large Large} {\LARGE LARGE} {\huge huge} {\
  Huge Huge}

% Pour des zones de texte plus importantes
\begin{Large}
Beaucoup de texte avec une grande taille de police d’écriture.

L’environnement est à utiliser pour des paragraphes (blocs de
  texte avec un ou des sauts de ligne).
\end{Large} \\

Je peux faire une remarque en bas de
page\footnote{La remarque en question.}.

\end{document}

```

Notons que les trois règles d’or de  $\text{\LaTeX}$  sont bien présentes : “\” pour les commandes, accolades pour délimiter le texte concerné par la commande, `begin` implique `end`.

$\text{\LaTeX}$  propose une commande qui peut se révéler pratique : `\emph{texte}`, `emph` pour *emphasis*. Son principe ? Faire ressortir du texte de la meilleure façon possible.

Essayons avec un exemple classique de faux texte : le *lorem ipsum*<sup>1</sup> (exemple d’utilisation très fréquent sur Internet).

### L’emphase ou *emphasis*

```

\begin{document}

\emph{Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Nunc est leo, facilisis non nisi eget, auctor eleifend

```

1. Pour plus de renseignements, aller sur <http://fr.lipsum.com/> ou sur <http://fr.wikipedia.org/wiki/Faux-texte>.



```

metus. Sed nec condimentum erat. Aliquam pulvinar feugiat
enim et porttitor.} \\

\textbf{Vestibulum porttitor, ligula vitae suscipit bibendum,
\emph{lorem ligula vestibulum ipsum, sed ultricies tellus
dolor sit amet odio.} Sed sodales eros vitae pulvinar
venenatis.} \\

\textit{Aliquam in massa eu tortor pellentesque posuere. \emph
{Nam sit amet tellus eleifend, ornare augue ut, congue ex
.} Morbi neque dolor, tincidunt sed est id, imperdiet
tempus ipsum. Ut sed dignissim tellus, id efficitur dui.}

\end{document}

```

Alors? As-tu remarqué la différence? Cette commande est très pratique dans certaines circonstances.

### Une question ?

« La note en bas de page possède un numéro alors que je ne l'ai pas noté. Comment est-ce possible ? »

C'est le côté pratique de  $\LaTeX$ . Il gère lui-même ce genre de détail. Il s'agit encore d'un parfait exemple où tu rédiges le fond tandis que  $\LaTeX$  gère la forme.

Bien, passons à un peu d'organisation.

## 4.3 Organiser son document

Taper du texte brut, c'est bien. Mais mettre des titres, c'est mieux. Là encore, rien à gérer.  $\LaTeX$  met en forme le titre selon son importance et s'occupe de la numérotation. Par ordre d'importance, nous pouvons avoir :

- |                                    |                                       |
|------------------------------------|---------------------------------------|
| → <code>\part{titre},</code>       | → <code>\subsubsection{titre},</code> |
| → <code>\chapter{titre},</code>    | → <code>\paragraph{titre},</code>     |
| → <code>\section{titre},</code>    | → <code>\subparagraph{titre}.</code>  |
| → <code>\subsection{titre},</code> |                                       |



### *Nota Bene*

La commande `\part` n'est disponible que pour un document de classe `report` (cf. 3.3 La base d'un document L<sup>A</sup>T<sub>E</sub>X p. 19).

De même, la commande `\chapter` n'est valable que pour les classes `book` et `report`.

Si tu veux sauter une page, la commande `\newpage` est là. Un exemple d'organisation serait donc :

### Exemple d'organisation

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\begin{document}

% Penser à enlever les % ici

%\part{Partie 1}

%\section{Section 1}

Comment est-ce numéroté ?

%\section{Section 2}

D'une manière bizarre !

%\part{Partie 2}

%\chapter{Chapitre 1}

%\section{Section 1}
```



```

Lorem ipsum \dots{}

%\section{Section 2}

Curabitur interdum \dots{}

%\chapter{Chapitre 2}

%\section{Section 1}

Sed eget \dots{}

%\section{Section 2}

Mauris at \dots{}

\newpage

Bis repetita \dots{}

%\part{Partie 3}

%\section{Section 1}

Tiens, encore une erreur de numérotation !

%\section{Section 2}

J'ai l'impression \dots{}

\end{document}

```

Comme tu as pu le constater, il y a quelques problèmes de numérotation. Si les compteurs tournent normalement, il faut juste donner un coup de pouce à  $\text{\LaTeX}$  pour faire correctement les choses. Retente le même code avec ceci dans le préambule :





### La numérotation des titres

```
% Dans le préambule

\makeatletter\@addtoreset{section}{chapter}\makeatother
% RAZ des numéros de section après un chapitre
\renewcommand{\thepart}{\Roman{part}}
% Pour mettre des I, II, etc. aux parties
\renewcommand{\thechapter}{\arabic{chapter}}
% Pour mettre des 1, 2, etc. aux chapitres
\renewcommand{\thesection}{\thechapter.\arabic{section}}
% Idem pour les sections et imposer le numéro de chapitre
  devant
```

### Aide en ligne

D'autres exemples sont disponibles en ligne sur <http://www.xmlmath.net/doculatem/structure.html>.

Naturellement, selon le document en cours de rédaction, toutes les commandes précédentes ne sont pas obligatoires et sont à adapter selon le rendu désiré.

### Les marges

Les gens me demandent souvent comment modifier les marges en  $\LaTeX$ . Tout d'abord, si les marges sont plus grandes que celles d'un document Word, c'est parce que  $\LaTeX$  a initialement été inventé par des Américains et que la reliure de rapports scientifiques épais demande une marge plus importante.

Autrement, je te laisse aller consulter la page suivante pour répondre à un éventuel besoin : [http://fr.wikibooks.org/wiki/LaTeX/Mise\\_en\\_page#Modification\\_des\\_marges](http://fr.wikibooks.org/wiki/LaTeX/Mise_en_page#Modification_des_marges).

Bon, c'est gentil d'avoir des titres mais comment obtenir un sommaire ?



## 4.4 Gestion du sommaire

Pour une fois, nouvelle option, pas de nouveau package. Pour le sommaire, tout est déjà inclus dans  $\LaTeX$ . Pour l'afficher, il faut juste renseigner dans le code la commande `\tableofcontents`.

### Un problème ?

« J'ai lancé la compilation du sommaire mais rien ne s'affiche hormis **Table des matières**. Est-ce normal ? »

Oui. Dès lors que tu génères un sommaire, il faut toujours compiler deux fois pour avoir le résultat final.

À la première compilation,  $\LaTeX$  crée un nouveau fichier, d'extension `.toc`, où il stocke le sommaire. À la seconde, il regarde si un tel fichier existe et, dans ce cas, récupère les informations pour afficher le sommaire.

### Une question ?

« Je ne veux pas lire **Table des matières** mais **Sommaire**. Est-ce possible ? »

Oui, tout à fait. C'est possible avec la commande suivante (dans le corps du texte, avant `\tableofcontents` par exemple) :

```
\renewcommand{\contentsname}{Sommaire}
```

Pour aller rapidement d'un point à un autre du document grâce à un clic sur le sommaire, là encore, un package le permet. Il s'agit d'`hyperref`. Cependant, le résultat n'est pas très esthétique et peut entraîner une crise d'épilepsie aux plus sensibles, c'est pourquoi je recommande d'utiliser l'option `hidelinks`.

Ce même package permet d'indiquer des adresses internet grâce à la commande `\url{adresse_internet}`. En revanche, si certains liens sont trop longs et finissent en fin de ligne, ils sortent de la page. Ajouter `breaklinks` dans les options du package `hyperref` permet de résoudre le problème.



### Le sommaire - Bilan

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[hidelinks, brealinks, linktoc = all]{hyperref}
% hidelinks : pour faire un renvoi du sommaire
% breaklinks : pour couper les liens trop longs
% linktoc = all : pour faire un renvoi du sommaire avec les
    numéros de page aussi (bonus)

\begin{document}

\renewcommand{\contentsname}{Sommaire}
\tableofcontents % Pour afficher le sommaire

% Parties, chapitres, texte, etc.

\url{http://www.google.com}

\end{document}

```

Enfin, dans le cas où tu veux juste écrire un titre sans numéro, il suffit d'ajouter une `*` à la commande : `\part*{titre}`, etc. Cependant, le titre en question n'apparaît pas dans le sommaire ... Il existe un moyen de l'ajouter manuellement :

### Ajout d'un titre sans numéro

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

```



```

\usepackage[hidelinks, breaklinks, lintoc = all]{hyperref}

\begin{document}

% Ajout d'un titre sans numéro

\phantomsection % Renvoi correct dans le sommaire
\addcontentsline{toc}{section}{Introduction} % Ajout dans le
    sommaire

% Penser à enlever le % la ligne en dessous
%\section*{Introduction} % Les titres doivent correspondre

Blablabla

\end{document}

```

Si la ligne avec la commande `\addcontentsline...` sert à implémenter dans le fichier `.toc` (fichier qui gère le sommaire) le titre `Introduction` en tant que `section` (`part` si partie, `chapter` si chapitre, etc.), ce n'est, bien sûr, qu'une illusion. Le numéro de page correspond par contre à l'endroit où est tapée la commande.

Il faut donc faire bien attention avec cette situation pour assurer la cohérence du document (titres identiques). C'est pourquoi je l'utilise le moins possible personnellement.

Bon, maintenant que nous avons un magnifique sommaire, est-il possible d'ajouter une page de garde ?

## 4.5 La page de garde

Comme pour le sommaire, il faut d'abord le créer puis indiquer à  $\LaTeX$  de l'afficher. Pour ce faire, il faut remplir les informations suivantes :



### La page de garde - Définition

```
% A mettre dans le préambule
% ou après \begin{document}

\title{\textbf{Initiation à \LaTeX} \ \ --- \ \Pour débutants
      ou jeunes utilisateurs}}
% Volonté personnelle de mettre le titre en gras - Ressort
      mieux selon moi

\author{Prénom \textsc{Nom} \ \ Profession} % L'auteur

\date{\today} % La date du jour - Sinon \date{date_souhaitée}

\begin{abstract}
Ceci est un résumé du document.
\end{abstract}
```

Naturellement, tout n'a pas besoin d'être renseigné. Si tu veux uniquement le titre, tu laisses juste cette commande.

Pour afficher la page de garde, il faut ensuite renseigner dans le corps du document la commande `\maketitle`.

### La petite astuce

S'il y a plusieurs auteurs dans ton document, tu peux tous les indiquer. Il faut juste les séparer par un `\and` :

```
\author{Nom1 \and Nom2 \and Nom3}
```

$\text{\LaTeX}$  s'occupe de toute la mise en forme de tous ces noms. Pratique, n'est-ce pas ?

Mais, tu devrais te rendre compte, après compilation, que, si tu demandes à ton lecteur de fichier PDF d'aller à la page  $N$ , tu te retrouves en page  $N + 1$  (même si je viens de me rendre compte qu'Adobe a corrigé ce détail ... mais je ne trouve pas le résultat esthétique).

Comme je trouve très pénible que le sommaire soit la page 1 et non la page 2, il suffit d'ajouter après `\maketitle` la commande :

```
\setcounter{page}{2}
```



### Une question ?

« Ce n'est pas pratique ta page de garde. C'est sobre, impossible de mettre une image ! »

... Tu as parfaitement raison ! Heureusement, il y a une solution. Pour insérer l'image, je te laisse attendre la partie 7.3 Insérer une image p. 71.

### Une solution personnalisable

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\begin{document}

% Environnement titlepage : permet de créer une page de garde
% et de la personnaliser à volonté
\begin{titlepage}

% Insérer l'image ici (par exemple)

\vspace*{\stretch{1}} % Pour centrer verticalement le texte
% compris entre ces deux commandes

\begin{center}
% Pour centrer horizontalement le texte
\begin{Huge}
\textbf{Initiation à \LaTeX{} \linebreak \linebreak ---
\linebreak \linebreak Pour débutants ou jeunes utilisateurs}
\end{Huge}

\vspace{1.5cm}

{\Large Indignation 13 C1215 \linebreak \linebreak dit \
\linebreak \linebreak Adrien \textsc{Bouzigues}}

```



```

\vspace{2cm}

{\Large \today\par}
\end{center}

\vspace*{\stretch{1}}

\end{titlepage}

\setcounter{page}{2}

Document à rédiger

\end{document}

```

Les dimensions sont un peu plus grandes mais le tout est modulable par tes soins sans souci. Je crois que tout est bon.

### Non, rien ne va plus !

« Dans ton code, tu as utilisé la commande double `\linebreak` au lieu du classique `\\`. Pourquoi ? »

Il peut arriver que `\\` ne fonctionne pas tout le temps. Dans ce cas, la commande `\linebreak` peut servir de remplacement.

Pour plus de renseignements, aller sur [http://en.wikibooks.org/wiki/LaTeX/Paragraph\\_Formatting#Manual\\_breaks](http://en.wikibooks.org/wiki/LaTeX/Paragraph_Formatting#Manual_breaks) ou <http://www.personal.ceu.hu/tex/breaking.htm>.

Allez, faisons une petite pause sur la mise en forme pour étudier un point un peu abstrait mais extrêmement puissant et nécessaire pour poursuivre.

## 4.6 Création de commandes

Il peut arriver que tu aies besoin de cumuler des commandes, et ce, un très grand nombre de fois.  $\text{\LaTeX}$  t'offre pour cela la possibilité de créer de nouvelles commandes.



C'est possible suite au renseignement dans le préambule de la commande suivante :

```
\newcommand{nom_commande}[nombre_arguments]{commande}
```

Étudions son fonctionnement avec un exemple. Disons que je veuille mettre un mot (ou un groupe de mots) en gras et en italique. Je vais donc procéder ainsi :

### Une première commande

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\newcommand{\grasital}[1]{\textbf{\textit{#1}}}
% Le nom de la commande commence par "\"
% La position de l'argument se fait avec un "#" et son numéro

\begin{document}

J'aime le chocolat ! \\

\grasital{J'aime le chocolat !} \\

J'aime le \grasital{chocolat} !

\end{document}
```

Note bien qu'il peut y avoir aucun argument comme plusieurs, avec une limite de 9. Si l'utilisation avec plusieurs arguments sera peut-être plus concrète avec les mathématiques, voici un cas sans argument :

### Un second cas

```
\documentclass[a4paper, 12pt]{report}
```





```

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\newcommand{\SAV}{\textbf{Service Après-Vente}}

\begin{document}

Notre \SAV propose une large gamme de services. \\\

Grâce à la performance de notre \SAV, vous serez comblés.

\end{document}

```

J’ai décidé de mettre le texte en gras, mais rien ne m’empêche en cours de création de changer la donne. L’avantage ? Tu as juste à modifier la commande et, lors de la compilation, la modification se répercute sur tout le document ! Pratique, non ?

### Une question ?

« Pourquoi, dans le second cas, n’y a-t-il pas d’espaces dans le résultat entre `SAV` et `propose` ? »

C’est vrai, c’est un bon point. Suite à une commande,  $\LaTeX$  ignore les espaces. Tu peux en ajouter un à la fin de la commande mais il y en aura alors aussi un après la virgule.

Si tu veux une solution, je te propose soit d’écrire “`\SAV{}`, vous” et “`\SAV{} propose`” (merci à Bertrand MASSON et à sa fiche  $\LaTeX$ , *créer ses commandes* (disponible sur Internet)) soit de te rendre directement à la partie 5.2 Les espaces insécables p. 50.

Avec du recul, la première solution est sûrement la meilleure : plus simple, moins prise de tête et plus cohérente avec la philosophie  $\LaTeX$ .

Tu ne trouves pas cet aspect utile pour l’instant mais tu verras que, quand tu prendras un peu d’expérience, tu finiras par créer toi-même tes commandes



pour plus de simplicité et de rapidité.

Allez, plus que deux parties sur le texte. C'est quoi déjà la suite ? Ah oui, les listes à puces.

## 4.7 Les listes à puces

Les listes sont des outils fort pratiques quand il s'agit d'énumérer des éléments, comme, par exemple, les ingrédients d'une recette de cuisine.

Les listes peuvent être soit non numérotées (tiret, point, etc. ; environnement `itemize`) soit numérotées (numéro ou lettre ; environnement `enumerate`). Pour faire apparaître un tiret ou un numéro, il faut utiliser la commande `\item`.

Voyons plutôt le résultat avec un exemple :

### Les listes à puces

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\begin{document}

Gâteau au chocolat :

\begin{itemize}
\item du sucre, % Le saut de ligne est optionnel, j'aère juste
    mon code
\item de la farine,
\item des {\oe}ufs,
\item et du chocolat ! \\
\end{itemize}
```



Cookies :

```
\begin{enumerate}
\item du chocolat, \\ % Le vrai saut de ligne est licite

\item du sucre, \\

\item de la farine, \\

\item quoi d'autre ? \\
\end{enumerate}

% Si tu n'as pas encore essayé d'imbriquer les commandes, c'
  est le moment de se faire plaisir
Cookies au gâteau au chocolat :

\begin{itemize}
\item du chocolat :

\begin{enumerate}
\item du cacao,

\item du beurre. \\
\end{enumerate}

\item de la farine,

\item etc.
\end{itemize}

\end{document}
```

### Le conseil personnel

Je me suis depuis peu mis aux listes à puces donc je ne suis pas encore un expert en la matière. Cependant, le package `babel` avec l'option `french` est censé remplacer le point • par un tiret -.

Chez moi, j'obtiens un double tiret tout moche. C'est pourquoi j'ai



créé une commande `\tiret` qui me simplifie la vie :

```
\newcommand{\tiret}{\item[-]}
```

Pour plus de renseignements sur les listes à puces, tu peux aller sur <http://www.xmlmath.net/doculatem/lists.html>. Il explique aussi comment changer l'affichage des listes numérotées (lettres ou nombres, minuscules ou majuscules).

Toutefois, la meilleure personnalisation possible des listes se fait avec les packages `\enumitem` (personnalisation des listes) et `pifont` (symboles supplémentaires) de la manière suivante :

#### Des listes avec des symboles

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{enumitem, pifont}

\begin{document}

J'ai envie de dire :

\begin{itemize}[label = \ding{215}] % \ding : commande du
    package pifont - Internet devrait fournir la liste des num
    éros associés à un symbole
\item une chose,

\item et un autre,

\item[\ding{213}] voire avec une puce ponctuelle ! \\
\end{itemize}

Tadaaa ! Pratique, n'est-ce pas ?
```



```
\end{document}
```

Toujours intéressé par L<sup>A</sup>T<sub>E</sub>X ? Tu vas voir, ce n'est que le commencement d'une infinité de possibilité ! Terminons d'ailleurs ce chapitre sur le texte.

## 4.8 Une petite touche de couleur ?

Nous avons vu beaucoup d'éléments de mise en page et de mise en forme du texte mais tu conviendras qu'avoir un gros pavé en noir, cela peut parfois être rebutant à la lecture.

Pour mettre un peu de couleur, il faut d'abord **charger le package xcolor** puis utiliser la **commande** :

```
\textcolor{nom_couleur}{texte}
```

Les couleurs pour `nom_couleur` disponibles sont alors :

|            |           |              |
|------------|-----------|--------------|
| → red,     | → yellow, | → black,     |
| → green,   | → orange, | → darkgray,  |
| → blue,    | → violet, | → gray,      |
| → cyan,    | → purple, | → lightgray, |
| → magenta, | → brown,  | → white.     |

Si jamais tu trouves qu'il n'y a pas assez de couleurs, tu peux utiliser l'option `dvipsnames` dans le package (`\usepackage[dvipsnames]{xcolor}`) puis te référer à la FIGURE 4.1 pour `nom_couleur` :



FIGURE 4.1 – Les couleurs avec dvipsnames

Enfin, si jamais tu trouves que tu n’as toujours pas assez de couleur pour laisser ton talent artistique s’exprimer, sache qu’il est possible d’en créer dans le préambule avec la commande :

```
\definecolor{nom_couleur}{modèle}{définition_couleur}
```

Le modèle correspond à RGB par exemple et `définition_couleur` à 255, 215, 0 (couleur or). Pour plus de renseignements quant à cette commande, tu peux consulter la page suivante : [http://fr.wikibooks.org/wiki/LaTeX/Options\\_de\\_mise\\_en\\_forme\\_avanc%C3%A9es#Mod.C3.A8les\\_de\\_couleur](http://fr.wikibooks.org/wiki/LaTeX/Options_de_mise_en_forme_avanc%C3%A9es#Mod.C3.A8les_de_couleur).

Sache qu’il existe aussi des commandes comme `\colorbox` ou `\pagecolor`. Je te laisse aller te renseigner si tu es intéressé pour te laisser un peu en autonomie.

Cette fois, nous en avons fini avec le texte et sa mise en forme. Tout d’abord, une référence s’impose :



FIGURE 4.2 – Non, je ne suis pas un fan d’Evangelion !

Toujours des nôtres ? Alors, si tu te sens prêt, nous allons aborder une partie également intéressante !

# Chapitre 5

## Les mathématiques sous L<sup>A</sup>T<sub>E</sub>X

Comme tu peux le constater, il y a fort à faire sous L<sup>A</sup>T<sub>E</sub>X. Les combinaisons sont déjà impressionnantes. Je te laisse maintenant découvrir la raison d’être de L<sup>A</sup>T<sub>E</sub>X, ce pourquoi il a été créé : écrire proprement des formules mathématiques !

### 5.1 Le mode mathématiques

Bon, c’est bien beau de vouloir faire écrire des maths à L<sup>A</sup>T<sub>E</sub>X, encore faut-il lui indiquer qu’il s’agit justement de maths ! C’est le principe du “mode mathématiques”.

Celui-ci est défini soit par le symbole \$ (**un ouvrant et un fermant**) soit par des “backslash-crochets” \[ et \] soit par l’environnement `equation` :

#### Le mode mathématiques

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\begin{document}

\'Ecrire  $x = 2 !$  et  $\$x = 2 !\$$  ne donnent pas le même résultat
! \\\
```





De même si j'écris `\[x = 2 !\]` % Ne pas mettre de `\` car il y a déjà un saut de ligne

Par contre, on obtient la même chose que précédemment avec :

```
\begin{equation}
x = 2 !
\end{equation}
```

mais l'équation est numérotée ! `\`

Que donne `$a b c d$` ?

```
\end{document}
```

### Repérer le mode mathématiques sous Texmaker

Il est très aisé de voir si du texte est en mode mathématiques : Texmaker affiche ce texte en vert !

Il existe une petite exception, que nous verrons en 5.6 Aligner des équations p. 59.

### Le conseil personnel

J'utilise très peu l'environnement `equation`, sauf quand j'ai besoin de numéroter des formules.

Si je n'ai pas besoin de numérotation, l'environnement `equation*` donne le même résultat que `\[ et \]`.

De ce que tu as pu observer dans l'exemple, le mode mathématiques met le texte en italique et supprime les espaces. En effet, dans ce mode, L<sup>A</sup>T<sub>E</sub>X considère que tout ce qui est écrit n'est que produit (comme pour l'exemple `$a b c d$`).

L'utilisation de `\[ et \]` permet d'aller à la ligne et de centrer la formule. Cette option est très pratique pour présenter un résultat ou une longue équation.



### Une question ?

« Si le mode mathématiques revient à mettre du texte en italique, pourquoi ne pas écrire du texte et utiliser la commande `\textit` ? »

Outre l'aspect esthétique de la formule, le mode mathématiques est le seul mode qui tolère les commandes que nous verrons par la suite et qui permettent d'afficher des formules mathématiques (fraction, symbole somme, intégrale, dérivée partielle, ...).

S'il est possible de mélanger le mode mathématiques avec du texte grâce au \$, c'est plus délicat avec les autres commandes, tant pour l'écriture en italique que pour l'absence d'espace. Mais il existe une solution ...

## 5.2 Les espaces insécables

Si L<sup>A</sup>T<sub>E</sub>X s'occupe de gérer le fond, ce dernier reste entièrement personnalisable par la main de l'utilisateur à condition d'utiliser les bonnes commandes.

Dans le cas des espaces, il existe des commandes autre que `vspace` et `hspace` bien plus efficace pour le mode mathématiques. Ces commandes portent le nom d'espaces insécables (insécables car L<sup>A</sup>T<sub>E</sub>X ne peut y toucher et se plie à la volonté de l'utilisateur). Nous pouvons relever :

- |  |   |
|--|---|
| → <code>\!</code> : espace très petit,     | → <code>\;</code> : espace large,                 |
| → <code>\,</code> : espace fin,            | → <code>\quad</code> : espace très large,         |
| → <code>\:</code> : espace moyen,          | → <code>\qquad</code> : espace encore plus large. |
| → <code>\~</code> (tilde) : espace normal, |   |

Toutefois, si j'utilisais précédemment à outrance les espaces insécables, ils peuvent vite se révéler pénibles à écrire. Il faut généralement faire confiance à L<sup>A</sup>T<sub>E</sub>X pour la mise en forme et **les utiliser avec parcimonie**.

Personnellement, je les utilise surtout, par exemple, après le symbole  $\forall$  (`\forall`) car l'espacement est très faible. À toi de choisir :

$$\forall x \quad \text{vs} \quad \forall x$$

Enfin, pour pouvoir librement écrire du texte dans le mode mathématiques, la commande `\text{texte_à_écrire}` est très utile et permet d'éviter bien des tracas.



### Espaces insécables et la commande `text`

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\begin{document}

Nous obtenons donc  $x + y = 3$ . Avec  $y = 2$ , cela donne :  $x = 1$  \text{ selon un calcul élémentaire}\

% Notons la présence d'un espace au tout début dans \text pour
% que le texte ne soit pas collé au "1"

% Utiliser \quad est aussi possible : c'est un exemple d'
% utilisation assez fréquent
Nous obtenons alors :  $x = 1$  \quad \text{et} y = 2\

\end{document}

```

Bref, après cette brève initiation aux mathématiques, allons *vraiment* écrire des formules mathématiques.

## 5.3 Des exemples de formules

Avant de se lancer, les mathématiques n'échappent pas à la règle : il faut charger des packages avant de commencer.

Après plusieurs recherches, je recommande `amsmath`, `amsfonts` et `amssymb`. Il semblerait que ces trois packages suffisent pour traiter 95 % des formules mathématiques.

Pour les 5 % restants, je te laisse aller chercher sur Internet selon ta demande. Mais tu vas voir, les exemples qui suivent donnent la majorité des formules utiles :



### Le mode mathématiques - Le retour

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb}

\begin{document}

% Le "hasard" fait que j'utilise la variable i mais tu peux l'
%   appeler x, n, z, bob, gnou, ... : peu importe !
Si je veux écrire un indice, je fais $i_2$. \\

% Si j'ai plus d'un élément, j'encadre avec des {}
$i_{13}$ est différent de $i_{13}$ : le second revient à écrire
  $x_1 3$. \\

Pour un exposant, c'est $i^3$ ou $i^{13}$. \\

Une barre de fraction ? $\frac{x}{y}$ \\

Une racine carrée ? $\sqrt{13}$ \\

Une racine énième ? $\sqrt[n]{13}$ ! \\

Une combinaison de formules ? $\sqrt{\frac{a}{b}}$ \\

Une intégrale ? $\int_0^{13} f(x)dx$ \\
% Attention aux bornes : les {} sont vite oubliées

Une somme ? Comme pour une intégrale ! $\sum_{i=1}^n x^i$
  \\

Une équation ? $x + y - z = 3 \times t + f$ \\
% Symbole +, - et = au clavier ; \times pour un produit

$x < y$, $y \leq z$, $z \geqslant c$, $c > d$ mais $d \neq f$

```



```

alors que $f \simeq g$ ! \\
% D'autres symboles - A toi de voir si tu préfères \leq à \
leqslant (idem pour \geq)

Les lettres grecques ? Facile : $\alpha$, $\beta$, $\mu$, \
dots{} \\

En majuscules ? $\Omega$, $\Delta$, $\Lambda$, \dots{}
% Ne fonctionne pas pour toutes les majuscules : \Alpha entraî
ne une erreur

\end{document}

```

### Utiliser Texmaker

Que ce soit pour les lettres grecques ou plein d'autres éléments mathématiques, **Texmaker** offre des raccourcis sur le côté gauche de la fenêtre.

N'hésite pas à aller jeter un coup d'œil au début. Je trouve que c'est mieux de taper les commandes mais il faut bien les avoir vues une ou deux fois avant pour savoir qu'elles existent.

### Une question ?

« J'ai tenté un `\mathrm` sur une lettre grecque pour enlever son "caractère italique" mais ça n'a pas fonctionné ... »

Ah, j'ai affaire à un petit malin (qui a au moins le mérite d'être allé fouiné une nouvelle commande). Tout d'abord, `\mathrm` est une commande qui ne fonctionne qu'en mode mathématiques et qui permet de redresser le texte. Il faut donc bien écrire :

$$\mathrm{\mu}$$

Mais, `\mathrm` ne fonctionne pas dans le cas des lettres grecques. C'est pourquoi tu peux ajouter le package suivant : `\upgreek`. Il permet d'écrire les lettres grecques droites. La commande `\upmu` est censée fonctionner.



**Attention**, ce package ne concerne pas toutes les lettres grecques ! `\upOmega` ne fonctionne pas car  $\Omega$  est déjà considéré comme droit (ce qui est logique). Cette commande est donc à manier avec prudence et qu'en cas de nécessité absolue (les lettres grecques en italique rendent déjà très bien).

Bon, je crois que nous avons déjà pas mal fait le tour. J'ai bâillonné l'élève curieux qui voulait savoir comment améliorer l'affichage de la fraction, de la somme et de l'intégrale : nous allons traiter ce point immédiatement.

## 5.4 L'affichage et les délimiteurs

Comme je sens que tu es extrêmement attentif derrière ton écran, je ne vais pas perdre plus de temps dans un long monologue et te donner la solution :

### Le mode mathématiques - Toujours là

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb}

\begin{document}

 $\frac{a}{b}$  est différent de  $\cfrac{a}{b}$  mais  $\cfrac{a}{b}$ 
  équivaut à  $\dfrac{a}{b}$ . \\

Cependant, \texttt{displaystyle} est la solution :  $\displaystyle \frac{a}{b} = \cfrac{a}{b}$ 

De même,  $\displaystyle \sum_{i=1}^n x^i$  est plus joli.
\\
```



```

Impossible d'empiler des \texttt{frac},
avec ou sans \texttt{displaystyle} : \[\displaystyle{\frac{\frac{a}{b}}{\frac{c}{d}}} \neq
\cfrac{\cfrac{a}{b}}{\cfrac{c}{d}}\]

\end{document}

```

### Pour résumer :

- `\displaystyle{formule}` force localement l'affichage au grand format,
- utiliser plutôt `\cfrac{numér.}{dénom.}` ou `\dfrac{idem}{idem}` pour les barres de fraction. Faire des cascades de `cfrac` ou `dfrac` est possible,
- il n'y a pas de différences notables entre `cfrac` et `dfrac`, du moins lors de l'affichage.

### La commande ultime

`\everymath{\displaystyle}` **juste après** `\begin{document}` force l'affichage sur tout le document !

Bien, maintenant que les choses sont correctement posées, tu peux avoir le meilleur rendu au monde mais L<sup>A</sup>T<sub>E</sub>X reste toujours extrêmement puissant, à condition de le lui dire.

En effet, écrire  $\left(\frac{a}{b}\right)$  et  $\left(\frac{a}{b}\right)$  sont deux choses totalement différentes. L<sup>A</sup>T<sub>E</sub>X est donc capable d'adapter la taille des parenthèses, crochets, accolades et autres, en mode mathématiques et toujours à condition de le lui signaler. Cette particularité est appelé un *délimiteur*.

### Les règles élémentaires des délimiteurs

- 1) Un délimiteur n'existe qu'en mode mathématiques,
- 2) Un délimiteur entrant implique un délimiteur sortant,



- 3) Un délimiteur entrant est défini par la commande `\left` suivi du nom du délimiteur ; pour le sortant, de même avec `\right`,
- 4) Si tu ne veux pas afficher un délimiteur, il faut utiliser la commande `\left.` ou `\right.` (il y a un point à la fin).

Voyons de suite les noms des délimiteurs et leur fonctionnement avec un exemple :

### Les délimiteurs en action

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb}

\begin{document}

Les parenthèses ?  $\left( \frac{a}{b} \right)$  \\

Des crochets ?  $\left[ \frac{a}{b} \right]$  \\

Un mix ?  $\left( \frac{a}{b} \right)$  \\
% LaTeX ne fait pas la différence entre le type de délimiteur
% Il faut juste respecter la règle 2

Une accolade à gauche ?  $\left\{ \frac{a}{b} \right.$  \\

Une accolade à droite ?  $\left. \frac{a}{b} \right\}$  \\

Une combinaison bizarre ?  $\left\} \frac{a}{b} \right\{$  \\
% Preuve que LaTeX ne fait pas la distinction tant que la règle 2 est respectée

Du bonus ?  $\left\langle \left\{ \frac{a}{b} \right\} \right\rangle$ 

```





```

rbrace \right\rangle$ \\  

% \lbrace ou \rbrace est équivalent à \{ ou \  

% Personnellement, je préfère \  

Les délimiteurs sont parfois inutiles : $(a \times b)$ et $\  

left( a \times b \right)$  

% Les délimiteurs sont pratiques dès lors qu'il y a un "étage"  

  ou si tu trouves que l'affichage rend mal (la taille peut  

  être parfois légèrement modifiée)  

\end{document}

```

C'est bon ? Pas de questions ? Ouah, je dois commencer à bien expliquer les choses pour une fois ! La suite ? Une petite escale dans le monde des matrices ...

## 5.5 Les matrices

Je préfère le répéter au cas où mais il faut naturellement employer le mode mathématiques. En revanche, pas besoin de nouveaux packages. Tout est déjà inclus avec les trois de base (`amsmath`, `amsfonts` et `amssymb`).

Ce n'est pas compliqué mais c'est aussi soumis à quelques règles alors je préfère bien les poser maintenant car nous les retrouverons un peu plus loin dans le guide :

### Les règles de base pour les matrices - Introduction aux tableaux

- 1) Il faut considérer la matrice comme un tableau vide à  $n \times m$  cases,
- 2) Une matrice est générée par l'environnement `pmatrix`,
- 3) Les colonnes sont séparées par une esperluette "&" (1 sous Windows),
- 4) Le passage à une ligne suivante se fait grâce à `\\`.

De même, ne nous privons pas d'un petit exemple pour comprendre et



digérer le tout :

### Les matrices

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb}

\begin{document}

Une matrice 2 x 2 ?  $\begin{pmatrix}$ 
a & b \\
c & d
\end{pmatrix}
% Rien de compliqué :
% & Pour distinguer les colonnes
% \\ Pour changer de ligne

% L'espace entre les & est optionnel mais recommandé pour
% faciliter la relecture

Une matrice 2 x 4 ?  $\begin{pmatrix}$ 
a & b & c & d \\
e & f & g & h
\end{pmatrix}

Une matrice à points (taille n) ?  $\begin{pmatrix}$ 
1 & 1 & \cdots & 1 \\
1 & 2 & \cdots & 2 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 2 & \cdots & n
\end{pmatrix}
% Les points remplissent une case
% Ne pas hésiter à faire un dessin si des difficultés sont
% rencontrées au début

```



```
\end{document}
```

Il n'existe pas qu'un seul environnement pour écrire des matrices. Nous pouvons relever :

- `matrix` : absence de délimiteurs,
- `pmatrix` : présence de parenthèses,
- `vmatrix` : présence de barres verticales,
- `Vmatrix` : présence de doubles barres verticales,
- `bmatrix` : présence de crochets,
- `Bmatrix` : présence d'accolades.

Un bon exemple de création de commande avec plusieurs arguments intervient ici. J'ai eu un jour à rédiger un corrigé d'exercices de physique. Ce corrigé contient énormément de vecteurs. J'ai donc inventé la commande `vectcol` de la manière suivante<sup>1</sup> :

```
\newcommand{\vectcol}[3]
{\begin{pmatrix} #1 \\ #2 \\ #3 \end{pmatrix}}
```

qui s'appelle de cette façon :  $\$ \backslash \text{vectcol} \{a\} \{b\} \{c\} \$$ . Plus pratique, n'est-ce pas ?

Bon, finissons-en avec les mathématiques sous L<sup>A</sup>T<sub>E</sub>X par la présentation et l'alignement des équations.

## 5.6 Aligner des équations

Comme une image sera plus parlante que des mots, j'aimerais obtenir ce résultat :

« Nous cherchons  $a$  tel que :

$$\begin{aligned} P(\mu \in I) &= 1 - \alpha \\ &= 0,9 \end{aligned}$$

1. Sans retour à la ligne : c'est juste pour que la formule ne déborde pas de la page



car l'énoncé indique que  $1 - \alpha = 0,9$

$$\begin{aligned}
 &= P\left(\frac{\bar{X} - \mu}{S}\sqrt{n-1} \in \left[\frac{-a}{S}\sqrt{n-1}; \frac{a}{S}\sqrt{n-1}\right]\right) \\
 &= 2\mathcal{S}_{n-1}\left(\frac{a}{S}\sqrt{n-1}\right) - 1
 \end{aligned}$$

Nous pouvons donc conclure par<sup>2</sup> :

$$\left\{ \begin{array}{l} I = [74,98 - 0,0428; 74,98 + 0,0428] \\ n = 20 \\ 1 - \alpha = 0,9 \end{array} \right. . \gg$$

Pour ce faire, soit pour aligner des équations, tu peux utiliser l'environnement `align` (ou `align*` pour éviter la numérotation de chaque ligne). L'environnement `eqnarray` se comporte de la même façon et produit le même résultat.

### Remarque personnelle

Après avoir vagabondé sur Internet et essayé différents rendus, je préfère utiliser `align`. Je ne m'avance donc pas sur `eqnarray` comme je l'utilise très peu. À toi de faire des essais et de voir celui qui te convient le mieux.

Pour le second résultat avec des accolades, il faut utiliser les délimiteurs et un tableau avec l'environnement `array`.

Si `array` fonctionne en mode mathématiques, **fais attention** : `align` ou `eqnarray` s'emploie sans ! C'est parti pour un exemple. Reproduisons le cas en page 59 :

### Aligner des équations

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

```

2. Si c'est du chinois pour toi, je te rassure, ce sont des statistiques !



```

\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb}
% Pas de nouveaux packages nécessaires

\begin{document}

Nous cherchons  $a$  tel que : \begin{align*}
P, (\mu \text{ in } I) \& = 1 - \alpha \\
% Présence d'un & : point de repère pour l'alignement (
%   similaire aux matrices)
& = 0,9
\intertext{car l'énoncé indique que  $1 - \alpha = 0,9$ }
% La commande intertext te permet d'ajouter une remarque sous
%   Texmaker, note que le vert du mode mathématiques a disparu
%   ( $1 - \alpha \dots$ ) mais c'est normal
& = P \left( \frac{\bar{X} - \mu}{S} \dots \right) \\
\\
& = 2 \mathcal{S}_{n-1} \left( \frac{a}{S} \dots \right) \\
\right)
% La commande mathcal permet de transformer les caractères
% Tu peux remarquer aussi les imbrications de commandes
\end{align*}

Nous pouvons donc conclure par : \left[ \begin{array}{rcl}
% Fonctionnement similaire aux matrices
% Il faut juste préciser le nombre de colonnes (les lettres)
% 1 lettre = 1 colonne
% l = left ; c = center ; r = right
I \& = & [74,98 \dots] \\
n \& = & 20 \\
1 - \alpha \& = & 0,9
\end{array} \right. \\
% Un array à 2 colonnes est envisageable mais j'ai préféré
%   distinguer les trois symboles

\end{document}

```

C'est dans la boîte? T'as déjà tout compris? C'est que ... je n'ai plus grand chose à t'apprendre dans le domaine des mathématiques sous L<sup>A</sup>T<sub>E</sub>X.



À toi d'essayer, d'être curieux, de te documenter . . . tout dépend de ce dont tu as besoin.

Pardon, tu veux en savoir plus? Que vais-je bien pouvoir t'expliquer désormais? Et surtout, comment vais-je bien pouvoir remplir le bas de cette page avant de passer au prochain chapitre . . .



FIGURE 5.1 – Je ne sais vraiment pas . . .

Tu auras le droit à un gâteau si tu me croises un jour et que tu me donnes l'origine de cette image. Et il y a une référence dans cette référence . . . #The cake is a lie!

# Chapitre 6

## Les tableaux et boîtes sous $\text{\LaTeX}$

### 6.1 Les tableaux

Si le tableau est un outil extrêmement pratique pour résumer l'information, ce n'est pas forcément mon passage préféré. En effet, ces derniers peuvent être un peu retors à gérer, surtout au niveau de la taille ... mais comme rien ne peut être pire que Word, c'est parti pour faire du  $\text{\LaTeX}$  avec le sourire!

Comme toujours, il faut charger un petit package avant de commencer : le package `array`. Je te préviens de suite : il ne faut pas confondre l'environnement `array` qui correspond à un tableau ultra-basique en mode mathématiques exploité pour aligner des équations, avec le package `array` qui permet d'utiliser l'environnement `tabular` pour faire des tableaux bien propres!

Tu as réussi à me suivre? Autrement, hormis l'affichage des traits pour les colonnes et les lignes, le fonctionnement est identique à celui d'`array`. Un petit exemple, comme toujours :

#### Les tableaux

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}
```



```

\usepackage{array}

\begin{document}

\begin{tabular}{cc}
% Toujours les mêmes lettres : l, c ou r
Texte centré & Ici aussi \\
Je crois que c'est bon & Oui \\
\end{tabular} \\ \\

% Affichage des traits verticaux : | (Alt Gr + 6)
\begin{tabular}{|l|c|r}
% Le nombre de trait dépend du nombre de |
OK ? & ok ! & Vraiment ? \\ \\
Je crois \dots{} & Mais je ne sais pas. &  $\alpha = 13$  \\
\end{tabular} \\ \\
% Rien n'empêche de mettre des maths

% Les traits horizontaux : commande \hline
% Toujours sauter une ligne avant d'appeler \hline (sauf au dé
but)
\begin{tabular}{|l|l|}
\hline
Les tableaux & rendent bien \\ \\
sous & \LaTeX{} \\
\end{tabular}

\end{document}

```

Bon, tu vois, rien de bien sorcier ! Si tu préfères gérer toi-même la largeur totale du tableau pour éviter les débordements de page, tu peux utiliser l'environnement `tabular*`, qui prend une commande supplémentaire. Je te laisse aller te renseigner sur <http://fr.wikibooks.org/wiki/LaTeX/Tableaux>.

### Une question ?

« Et si j'ai un grand nombre de colonnes à déclarer, comment faire ? Dois-je tout copier à la main ? »





Oui, tu es soumis à l'ennui de devoir tout taper ! ... non, je plaisante, il y a un petit raccourci. Au lieu d'écrire `|c| ... |c|`, tu peux écrire `*{N}{|c|}`. Ainsi, tu crées  $N$  colonnes centrées avec des traits verticaux partout. Pratique, non ?

### Plein d'autres options

Les tableaux ne se limitent pas qu'à ce point. Tu peux naturellement fusionner les lignes ou les colonnes, colorer des cases, ...

Encore une fois, je te renvoie au même lien : c'est assez complet. Autrement, en annexe de ce guide, à la fin, je détaille des commandes personnelles. Les fusions et le coloriage en font partie, ainsi que la création de colonnes personnalisables.

Enfin, dans le cas de tableaux tellement longs qu'ils dépassent une page, il faut utiliser le package `longtable`. Le package `booktabs` constitue aussi une autre solution pour créer des tableaux.

Je viens de me rendre compte que je ne sais même pas comment changer les couleurs des traits du tableau ... Pas grave, la partie suivante est là pour ça !

## 6.2 Les boîtes

Si tu peux tout à fait utiliser un tableau pour encadrer des formules, des images ou du texte, il existe d'autres solutions plutôt complètes et personnalisables sous L<sup>A</sup>T<sub>E</sub>X.

Tout d'abord, les `framebox` constituent la base en L<sup>A</sup>T<sub>E</sub>X. La commande est assez simple : `\framebox[taille_boîte]{texte}`. Si `texte` fait référence à l'objet à encadrer (texte, image, formule, etc.), `taille_boîte` fait référence à la largeur de la boîte.

Tu peux renseigner une unité de distance comme nous l'avons vu avec les commandes `hspace` et `vspace` (cf. la boîte bleue sur les « Unités de distance » p. 27). Naturellement, si tu renseignes `1cm` alors que le texte en fait 2, cela va être gênant ...

Heureusement, il existe une solution, avec la commande `\textwidth`. Comme son nom l'indique, la boîte prendra comme largeur celle du texte. Et



si tu veux directement avoir une boîte qui s'étend d'une marge à une autre, la commande `\linewidth` est faite pour toi ! Souviens-t'en : elle va ressortir très prochainement.

Mais la `framebox` n'est rien comparée au Saint-Graal que j'ai découvert lors de la rédaction de la première version de ce guide<sup>1</sup>. Il s'agit du package `tcolorbox`. Si tu fouines un peu sur Internet, tu devrais trouver la documentation associée (<http://lmgty.com/?q=tcolorbox+help><sup>2</sup>) : plus de 466 pages de bonheur pour faire des boîtes aussi jolies que celles présentes dans ce guide et bien plus ! Beaucoup plus ! Un exemple, ici et maintenant, histoire de te convaincre :

### Le package `tcolorbox`, ou le Saint-Graal pour faire des boîtes !

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor}

\usepackage[breakable]{tcolorbox}
% Le package suprême pour faire des boîtes !
% breakable : pour pouvoir couper les boîtes qui font plus d'
% une page ou qui se retrouve en bas de page (sinon c'est
% moche)

\begin{document}

\begin{tcolorbox}[oversize, breakable, colframe = Orange,
  colback = Apricot, boxrule = 2pt, arc = 6pt, title = Un
  titre, coltitle = Black]
% La quantité d'options est incroyable

% oversize : les bords de la boîtes s'alignent avec le texte
% Sinon, la boîte est un peu plus courte
```

1. Soit durant l'été 2016.

2. Oui, je n'ai pas pu résister à l'envie de mettre un lien "troll" ... Navré.



```
% Personnellement, je ne l'utilise pas

% breakable : pour couper la boîte si elle dépasse
% colframe : couleur de la bordure
% colback : couleur du fond
% boxrule : épaisseur du trait
% arc : rayon de l'arrondi aux coins
% title : obvious if you speak English a little !
% coltitle : couleur du titre (mais tu avais sûrement compris)

J'adore ce package ! De toute mon âme !
\end{tcolorbox}

\end{document}
```

Et comme je l'ai dit, ce n'est que la partie émergée de l'iceberg. La documentation technique sur ce passage propose beaucoup plus d'options, plus de boîtes de types différents, . . . bref, plus de tout, preuve que L<sup>A</sup>T<sub>E</sub>X, c'est la vie !

Mais je crois m'être légèrement emporté. Si tu es intéressé par ce package, je te laisse donc aller regarder les parties qui t'intéressent dans la documentation (disponible sur Internet je rappelle, plus de 466 pages de bonheur).

Passons plutôt à un point plus sympathique mais que j'avais envie de garder pour la fin. Oh, mais je suis persuadé que tu l'attendais depuis un petit moment : insérer une image !



# Chapitre 7

## Insérer des images

### 7.1 Les formats d'images

Il existe deux sortes d'images : les images matricielles et les images vectorielles. Les premières sont les plus courantes et portent généralement les extensions `.jpg` (*Joint Photographic Group*) ou `.png` (*Portable Network Graphics*).

La base d'une image matricielle est le pixel, d'une couleur donnée et figée. Si tu zoomes sur l'image à la page précédente, tu devrais tomber sur ces fameux pixels. *A priori*, rien de méchant : dès lors que ton image contient "suffisamment" de pixels par rapport à la taille affichée, elle ne devrait pas apparaître trop floutée.

Tu dois sûrement te demander pourquoi j'aborde un tel sujet. Peut-être que tu sais déjà où je veux en venir. Une image vectorielle est définie par l'intermédiaire d'outils géométriques (arcs de cercle, traits, courbes de Bézier, ...). Je ne vais pas faire un cours dessus : d'abord parce que je n'en sais pas plus et ensuite parce que ce n'est pas le but de guide.

Ce qu'il faut retenir c'est que, peu importe à quel point tu zoomes, tu ne tomberas jamais sur un pixel et l'image vectorielle reste lisse et belle<sup>1</sup>. Et inversement, si l'image est grande de base, aucun pixel ne sera donc visible.

#### Un exemple d'images vectorielles

Si tu veux un exemple, retourne sur la page de garde et regarde les deux images que j'ai insérées. Tu auras beau zoomer, aucun pixel à la ronde.

---

1. J'ai l'impression de faire de la pub' pour l'Oréal ...



Un exemple encore plus simple est le texte. En effet, peu importe le texte que tu lis et le niveau de zoom, là encore, pas de pixel. C'est normal : les polices d'écriture sont définies comme une image vectorielle.

Et heureusement ! Imagine un peu le résultat : conçois (mentalement) une “police matricielle” et augmente sa taille. Le résultat n'est pas terrible, n'est-ce pas ?

Un format usuel d'images vectorielles est le `.svg` (*Scalable Vector Graphics*), dont la création, édition et visualisation sont possibles grâce à des logiciels spécialisés, comme **Inkscape** par exemple. Il existe aussi un autre format, moins connu et un peu délaissé, développé par Adobe à l'origine : le format `.eps` (*Encapsulated PostScript*). Un logiciel simple comme **EPS Viewer** suffit pour les visualiser.

Et voilà, j'ai fini par arriver à introduire cette nouvelle notion. Le format `.eps` fait un peu vieux jeu et reste surtout utilisé dans le domaine scientifique. Cependant, même s'il est difficile à modifier avec des outils standards (comme **Paint**), il est plus facile à implanter sous  $\text{\LaTeX}$  que le format `.svg`, pour un résultat identique.

### Conversion `.png` → `.eps`

Une image au format `.eps` n'est pas automatiquement vectorielle. Supposons que tu ouvres sous **GIMP** une image matricielle et que tu l'enregistres au format `.eps`. Le rendu final sous  $\text{\LaTeX}$  reste une image matricielle.

Le format `.eps` ne garantit pas automatiquement une image vectorielle derrière. C'est bel et bien un format qui peut gérer ce type d'image mais il ne faut pas s'attendre à ce qu'il fasse de lui-même une belle conversion.

Pour conserver une véritable image vectorielle au format `.eps`, il faut vectoriser l'image matricielle (passage du matriciel au vectoriel), sous **Inkscape** par exemple<sup>a</sup>, puis enregistrer cette image vectorielle au format `.eps`.

<sup>a</sup>. Fonctionne parfaitement pour des formes simples, avec peu de variations de couleur. Résultat à travailler pour des images plus complexes.



## 7.2 Les longueurs

Après cette première introduction, nous allons continuer par un petit passage barbare, mais qui va se révéler utile pour la suite. J'en ai déjà brièvement parlé plus tôt ... mais c'est l'occasion parfaite pour proprement présenter la notion de « longueur » sous  $\text{\LaTeX}$ .

Sous  $\text{\LaTeX}$ , il est possible de travailler avec toutes sortes d'unités : `mm`, `cm` pour citer les plus courantes ; `pt`, `in` pour citer quelques cas moins usités ; `ex` pour citer l'unité de distance la plus amusante que j'ai découverte à ce jour en informatique.

Dès lors qu'une commande requiert une longueur en paramètre d'entrée, nous l'indiquons très clairement. Par exemple, `\vspace{13mm}`. Cependant,  $\text{\LaTeX}$  permet d'aller plus loin, beaucoup plus loin en mettant des longueurs prédéfinies sous forme de commande.

### Un exemple de longueurs sous $\text{\LaTeX}$

$\text{\LaTeX}$  utilise des longueurs nativement, dans chaque nouveau document. Par exemple, à chaque nouveau paragraphe,  $\text{\LaTeX}$  met un alinéa. La taille de cet alinéa est une longueur définie par défaut et  $\text{\LaTeX}$  utilise cette longueur.

Il en va par exemple de même pour les marges ou les sauts de ligne. Naturellement, toutes ces longueurs peuvent être modifiables, même si ce n'est pas vraiment recommandé.

Du coup, sans entrer plus dans les détails, voici deux longueurs fondamentales qui sont plutôt utiles :

- `\linewidth` : longueur qui correspond à la largeur totale du texte sur la page (largeur de la page marges exclues),
- `\baselineskip` : longueur qui correspond à un saut de ligne sous  $\text{\LaTeX}$ .

Il faut aussi savoir qu'un coefficient est toléré devant les longueurs. Par exemple, `\vspace{2\baselineskip}` correspond à un double saut de ligne ; `0.5\linewidth` correspond à une longueur égale à la moitié de la page (marges exclues).

Voilà, je ne vais pas aller plus loin. Si tu veux en savoir plus sur les longueurs (création, utilisation, modification ; longueurs définies par défaut ;



définition du `ex`), je te recommande d’aller lire la page suivante : <http://en.wikibooks.org/wiki/LaTeX/Lengths>.

Bien, allons maintenant insérer des images. Retiens juste la longueur suivante : `\linewidth`. C’est celle qui va beaucoup nous servir ici.

## 7.3 Insérer une image

Je pense que tu devais attendre ce point depuis pas mal de temps. Ne traînons pas plus dans ce cas : place aux insertions d’images !

Travailler avec des images sous  $\text{\LaTeX}$  est possible. Il faut au préalable charger le package `graphicx`. Pour insérer une image, c’est très simple. Il faut utiliser la commande :

```
\includegraphics[options]{nom_img.extension}
```

En revanche, tu risques d’avoir des difficultés pour placer une légende. Cette commande reste cependant utilisable si tu veux juste insérer une image telle quelle.

Sous  $\text{\LaTeX}$ , il est plutôt recommandé de travailler avec un élément flottant. Il s’agit d’un “cadre invisible” qu’il est possible de positionner n’importe où sur la page. Là encore, c’est une façon propre à  $\text{\LaTeX}$  de gérer la forme tandis que tu te concentres uniquement sur le fond. Un élément flottant est défini grâce à l’environnement `figure`.

Mais je crois qu’un exemple sera plus parlant. Pour ce faire, prends une image plutôt grande de préférence, soit au format `.jpg` ou `.png` (si tu ne sais pas quel est le format de ton image, un clic droit dessus puis aller sur les propriétés), puis renomme-la `fond`. De cette manière, tu auras moins de souci avec le code qui suit. Place cette image au même endroit que le fichier `.tex` avec lequel tu travailles.

### Les images

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}
```



```

\usepackage{graphicx}

\begin{document}

\includegraphics{fond.png} \\
% J'ai supposé que l'image est un png de GRANDE taille
% Sinon, écrire nom_image.extension
% LaTeX te signale si tu t'es trompé d'extension/format car il
% ne va pas trouver l'image

C'est un peu grand, n'est-ce pas ? L'image doit déborder ! \\

\begin{center}
\includegraphics[scale = 0.4]{fond.png}
\end{center}

C'est mieux, non ? Que donne l'environnement \textit{figure} ?
\\

\begin{figure}[!h]
\centering
\includegraphics[scale = 0.4]{fond.png}
\caption{Une légende}
% Commande caption pour une légende
\end{figure}

Une rotation ? \\

\begin{figure}[!h]
\centering
\includegraphics[scale = 0.4, angle = 45]{fond.png}
\caption{Une image tournée}
\end{figure}

\end{document}

```

C'est pratique, n'est-ce pas ?  $\text{\LaTeX}$  s'occupe de positionner l'image et le texte en fonction de la place qu'il reste. **Petite précision** sur l'environnement `figure` et une option que je n'ai pas détaillée. Tu as sûrement vu que j'ai écrit `\begin{figure}[!h]`. Tu peux mettre quatre lettres entre les





crochets en guise d’option, pour indiquer à  $\text{\LaTeX}$  où positionner l’image :

- **t** pour **top** : l’image se retrouve en haut de page,
- **b** pour **bottom** : l’image se retrouve en bas de page,
- **p** pour **page** : l’image se retrouve sur une page particulière réservée aux éléments flottants,
- **h** pour **here (le plus pratique)** : l’image se retrouve là où elle est positionnée dans le code.

Cependant, il arrive à  $\text{\LaTeX}$  d’être un peu capricieux et l’option “!” devant la lettre lui indique que l’utilisateur a raison. Si tu mets donc l’option **!h**,  $\text{\LaTeX}$  fait tout son possible pour mettre l’image là où elle est écrite dans le code.

Si le code de l’image est écrit entre une zone de texte A et une autre zone de texte B, elle le sera aussi sur le document final ... à condition qu’il y ait suffisamment de place, naturellement. Dans le cas contraire, l’image se retrouve à la page suivante et le texte est remonté en conséquence pour combler les blancs.

La **solution ultime** reste sinon d’utiliser le package **float** et de renseigner un **H** à la place de **!h**. L’image est *vraiment* contrainte d’être à cet endroit. S’il n’y a pas la place,  $\text{\LaTeX}$  laisse un blanc.

### Une question ?

« Ce n’est pas très pratique ton option **scale**. Un coup je dois prendre 0.4, un autre 0.8. N’y a-t-il pas mieux ? »

Bien sûr,  $\text{\LaTeX}$  est le lieu de tous les possibles. Je recommande plutôt d’utiliser l’option **width** qui te permet de régler la largeur de l’image. Associée à la commande **linewidth** (écrire donc **width = \linewidth** : l’image sera de largeur celle de la page (marges exclues), tu peux insérer des images sans souci.

Et si c’est toujours trop grand, tu peux aussi choisir **width = 0.7\linewidth** : l’image sera de largeur 70 % de celle de la page.

Si les images sont trop grandes,  $\text{\LaTeX}$  les réduit donc pour toi. Si elles sont trop petites, ... tu verras les pixels !

Et si jamais tu as d’autres questions sur les images et les flottants,



tu peux aller sur [http://fr.wikibooks.org/wiki/LaTeX/%C3%891%C3%A9ments\\_flottants\\_et\\_figures](http://fr.wikibooks.org/wiki/LaTeX/%C3%891%C3%A9ments_flottants_et_figures).

### Répertoire d'images

Si jamais tu as beaucoup d'images, tu peux vite noyer le dossier de travail où se situe ton fichier `.tex`.

Dans ce cas, tu peux placer tes images dans un dossier, situé au même endroit que ton fichier `.tex`, puis utiliser la commande suivante dans le préambule :

```
\graphicspath{{./nom_du_dossier/}}
```

De cette façon, tu indiques à  $\text{\LaTeX}$  où aller chercher les images. Fais attention à bien placer cette commande **après** le package `graphicx`, car il s'agit d'une commande de ce même package.

Si le dossier est placé à un autre endroit, la commande s'applique toujours mais, dans ce cas, il faut renseigner le chemin complet pour accéder jusqu'au dossier.

### Nom des images et des dossiers

Le nom de tes images ou des dossiers où tu places tes images ne doit contenir **ni accent ni espace**. Autrement, tu risques de ne pas pouvoir compiler ton document et tu ne vas pas comprendre l'erreur.

Le nom `texte mathématiques` est donc à bannir. Tu peux par contre appeler ton image `texte_maths`, `textemathematiques`, `texte-maths`, etc.

Bien, si ce point est clair, passons au suivant. La fin est encore un peu loin mais tu t'en approches.

## 7.4 Les références

Une image, une équation, un tableau, une partie ... tous ces outils sont bien pratiques mais que valent-ils si tu ne peux y faire référence (cf. l'image



n° $x$  page  $y$  par exemple) ?

Naturellement, je n'en parlerais pas si L<sup>A</sup>T<sub>E</sub>X ne proposait pas une solution. Déjà, c'est implanté par défaut sous L<sup>A</sup>T<sub>E</sub>X donc pas besoin de nouveaux packages.

Si tu veux créer une référence, il faut utiliser la commande `label{nom_ref}`. Puis, tu peux y faire appel avec les commandes `ref{nom_ref}` (pour le numéro de l'objet concerné) et `pageref{nom_ref}` (pour la page où se situe l'objet concerné). Un petit exemple pour mieux comprendre le principe :

### Les références

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{graphicx, float}
% Attention, utilisation du package float désormais

\begin{document}

% Penser à enlever le % la ligne en-dessous
%\section{Que dire ?}

\label{que_dire}Je ne sais pas quoi dire ici \dots{}

\begin{figure}[H]
\centering
\includegraphics[width = 0.6\linewidth]{fond.png}
% Tu peux mettre l'image que tu veux maintenant que tu sais
  comment faire
\caption{Un petit titre}
\label{une_image}
\end{figure}

Je fais du \LaTeX{} et je ne sais pas quoi dire. \\
```



```
Tu peux t'en rendre compte en \ref{que_dire},
p. \pageref{que_dire}. \\

La \figurename{} \ref{une_image} est similaire, en p. \pageref
{une_image}.
% Utiliser le même nom que LaTeX pour les légendes : \
figurename{}

\end{document}
```

C'est extrêmement puissant et extrêmement pratique car tu te moques du numéro :  $\LaTeX$  a tout en mémoire et l'adapte si besoin. **Petite précision :** le fonctionnement est similaire pour les formules mais les références se font avec la commande `eqref`.

### Une question ?

« Je ne comprends pas. J'ai compilé et j'ai ?? à la place de mes références. Pourquoi ? »

Ce n'est rien de grave. C'est le même problème que pour le sommaire.  $\LaTeX$  stocke les références dans un fichier à la première compilation et ne s'en sert que lors de la seconde. Il faut donc juste compiler deux fois.

### Remarque personnelle

Le package `hyperref`, utilisé pour faire des liens entre le sommaire et les parties, permet aussi d'utiliser une commande supplémentaire pour les références : `\nameref{nom_ref}`.

Cette commande te permet de recopier le nom auquel est associée la référence (titre de parties, chapitres, sections, images, etc.). C'est toujours bon à savoir.

Allez, un peu de courage. Tu touches presque au but de ce guide : atteindre l'autonomie en  $\LaTeX$  (et taper de magnifiques rapports). Et la suite n'est pas compliquée : c'est du code personnel pour t'éviter de chercher pendant des heures comme j'ai eu à le faire!



## 7.5 Un peu de montage

Tu dois sûrement te dire : « Cool, je peux mettre des images mais j'ai l'impression que mon rapport a un balai dans la structure avec toutes ces images centrées sans texte autour ... ». Et tu as bien raison !

Il existe des solutions pour mettre du texte autour d'une image, comme le package `wrapfigure` qui fonctionne plutôt bien mais qui doit être utilisé avec des pincettes. Il est fortement recommandé d'aller jeter un coup d'œil à l'aide en ligne.

Concrètement, pour expliquer le fonctionnement de ce package, il permet de positionner une image sur la droite ou sur la gauche, dans un bloc de taille fixée par l'utilisateur. Le texte qui suit la commande épouse alors le contour de l'image avant de reprendre son cours initial.

Heureusement, il existe d'autres solutions plus simples comme les `minipage`. Tout est dans le nom : tu gères une page dans la page ! C'est un peu le pendant de la zone de texte sous Word. Un exemple et tout sera clair :

### wrapfig & minipage

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{graphicx, float, wrapfig}

\begin{document}

% L : position de l'image (à gauche) ; R possible
% Largeur de l'image à insérer
\begin{wrapfigure}{L}{0.3\linewidth}
% Remonter l'image peut servir parfois
%\vspace{-\baselineskip}
\includegraphics[width = \linewidth]{fond.png}
% \linewidth ici correspond à la largeur de la ZONE considérée
% soit 0.3\linewidth
```



```
\caption{Légende possible}
\end{wrapfigure}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc est leo, facilisis non nisi eget, auctor eleifend metus. Sed nec condimentum erat. Aliquam pulvinar feugiat enim et porttitor.

Vestibulum porttitor, ligula vitae suscipit bibendum, lorem ligula vestibulum ipsum, sed ultricies tellus dolor sit amet odio. Sed sodales eros vitae pulvinar venenatis.

Aliquam in massa eu tortor pellentesque posuere. Nam sit amet tellus eleifend, ornare augue ut, congue ex. Morbi neque dolor, tincidunt sed est id, imperdiet tempus ipsum. Ut sed dignissim tellus, id efficitur dui. \\

% Les sauts de ligne sont interdits à partir d'ici. Sinon, la compilation ne donne pas le résultat attendu !

Un montage avec deux images côte à côte :

```
\begin{figure}[H]
\begin{minipage}[t]{0.45\linewidth}
% Création d'un bloc de largeur 45% de celle de la page
% Paramètre t optionnel (position "top" de la minipage) -->
% Permet d'aligner les minipages sur la légende
\centering
\includegraphics[width = 0.6\linewidth]{fond.png}
% L'image prend 60% des 45%
% Elle est placée dans la minipage de taille linewidth mais ce
% linewidth correspond à 45% du linewidth de la page
\caption{Légende 1}
\end{minipage}
\hfill % Remplir l'espace horizontal qui reste (par du blanc)
\begin{minipage}[t]{0.45\linewidth}
% Idem - Pas de centering ici car l'image prend la largeur
% totale fournie
\includegraphics[width = \linewidth]{fond.png}
\caption{Légende 2}
\end{minipage}
```



```

\end{figure}

% Petit bilan ici :
% zone de 45% avec une image qui prend 60% (de ces 45%)
% espace blanc (10% ici : 100 - 45 - 45)
% zone de 45% avec une image qui prend toute la largeur (de
  ces 45%)
% [t] utile si une légende tient sur deux lignes : alignement
  par rapport aux légendes
% ==> toujours définir des minipages dont la somme des
  largeurs est strictement inférieure à \linewidth

Avec du texte ?

\begin{figure}[H]
\begin{minipage}{0.55\linewidth}
%\vspace{-2\baselineskip}
% Il faut parfois remonter le texte
J'aime écrire n'importe quoi ! \\

J'aime le chocolat ! J'adore \LaTeX{}. \\

J'adore les jeux vidéos !
\end{minipage}
\hfill
\begin{minipage}{0.4\linewidth}
\centering
\includegraphics[width = 0.86\linewidth]{fond.png}
\caption{Légende 1}
\end{minipage}
\end{figure}

\end{document}

```

Tu peux faire à peu près ce que tu veux avec cet exemple. À toi de l'agencer et de l'adapter en fonction de tes besoins (`centering`, taille des images, etc.).

Et voilà, tu approches de la fin de ce guide. Tu peux clairement t'arrêter après le chapitre 8 Traitement des erreurs et revenir à ce guide beaucoup plus tard selon tes besoins.

# Chapitre 8

## Traitement des erreurs

Les erreurs peuvent être nombreuses sous L<sup>A</sup>T<sub>E</sub>X et pas toujours évidentes à corriger. Maintenant que tu connais le minimum syndical des outils accessibles, voici une petite liste des erreurs possibles et mes conseils personnels pour les éviter.

Sache aussi que tu peux te rendre sur [http://fr.wikibooks.org/wiki/LaTeX/%C3%80\\_1%27aide\\_!](http://fr.wikibooks.org/wiki/LaTeX/%C3%80_1%27aide_!) si tu veux des informations complémentaires.

### Les erreurs courantes :

- 1) Missing \$ inserted,
- 2) Missing } inserted (nous avons plutôt tendance à oublier les accolades fermantes mais valable aussi pour les accolades ouvrantes),
- 3) There's no line to end here,
- 4) undefined control sequence,
- 5) Package inputenc Error: Unicode char, suivi éventuellement d'un caractère et de son code UTF-8,
- 6) Option clash for package nom\_package,
- 7) Extra alignment tab has been changed to \cr.

### Comment les corriger ?

- 1) Soit tu as oublié de fermer le mode mathématiques soit tu as employé sans faire attention un symbole propre à ce mode ( $\hat{}$  ou  $\_$  par exemple). Pour corriger le second point, supprimer le  $\hat{}$  ou utiliser  $\\_$  pour afficher un *underscore* en mode texte,





- 2) Tu as sûrement oublié une } sur une commande. Il faut d'abord commencer par chercher les erreurs parmi les lignes de codes écrites ou modifiées depuis la dernière compilation.

De plus, **au début, je recommande de compiler souvent son code**. C'est plus pratique de corriger plein de petites erreurs que de s'arracher les cheveux sur un très grand nombre. Avec l'expérience, tu verras que tu feras de moins en moins d'erreurs ou que tu les corrigeras très rapidement,

- 3) Tu as essayé un saut de ligne que L<sup>A</sup>T<sub>E</sub>X ne comprend pas (après un environnement `center` par exemple).

Dans ce cas, il faut déjà regarder le résultat sans saut de ligne : certains environnements laissent un peu de blanc avant et après (comme `center` justement). Autrement, il faut utiliser la commande `\vspace`,

- 4) Soit tu as oublié un élément à un endroit (comme une virgule “,” lors d'un espace insécable ; essaye d'écrire `\13` et `\,13` et tu verras).

Sinon, tu as inventé un nom de commande qui n'existe pas. Ou alors tu as bel et bien défini une nouvelle commande mais tu as juste fait une faute de frappe lors de l'utilisation de ta commande personnelle,

- 5) Tu as utilisé un caractère du clavier interdit dans ce mode de compilation. L'exemple le plus courant est le symbole “°”, généré par la commande `\degres{}` sous PDFLaTeX alors qu'il faut l'écrire normalement au clavier sous XeLaTeX<sup>1</sup>.

Cette erreur revient aussi si tu effectues des copier-collers depuis Word : il faut reprendre tous les accents et les apostrophes. **Dans ce cas, la fonction Remplacer de Texmaker peut se révéler très utile . . .**,

- 6) Il faut parfois charger certains packages dans un ordre bien précis, sous peine d'avoir cette erreur. C'est par exemple le cas pour les packages `wallpaper` et `xcolor`.

Dans ce cas, l'erreur provient généralement lors de l'ajout d'un nouveau package. Il s'agit donc de trouver le package en question, de lire l'erreur pour comprendre avec quel package il entre en conflit puis de le charger avant ou après le package conflictuel.

**Dans tous les cas, il faut toujours charger le package `hyperref` en**

1. Les modes de compilation sont abordés dans la partie suivante si tu es intéressé.



dernier,

- 7) L'erreur concerne un tableau. Tu as probablement oublié d'indiquer un saut de ligne avec la commande “`\`” en fin de ligne de ton tableau.

Enfin, mon conseil le plus important : dès que tu ouvres un `$` ou un `\[` ou un `{` ou un délimiteur, ferme-le en suivant. Puis, tu reviens en arrière et tu écris ton code. Les erreurs devraient diminuer.

L'auto-complétion de `Texmaker` est aussi très pratique pour éviter ce genre de désagréments.

Enfin, `LATEX`, par l'intermédiaire des informations du compilateur (affichées par `Texmaker`, tout en bas), te renvoie au paragraphe qui lui pose problème. 90 % du temps, c'est dans les environs qu'il faut relire son code et chercher l'erreur.

Et voilà, la première partie de ce guide est bel et bien terminée. Toutes mes félicitations si tu es arrivé jusqu'ici. J'espère avoir pu t'être d'une aide quelconque et que mes explications étaient assez claires.

Ce n'est pas absolument pas évident de débiter en `LATEX`. Et si je commence à m'y faire, la route est encore longue avant de pouvoir maîtriser les innombrables facettes de cet outil incroyable.

Tu trouveras dans la partie suivante mes expériences `LATEX` sur des sujets plus poussés. Je tenais initialement à les regrouper dans ce guide pour mon usage personnel mais je me suis rendu compte qu'elles peuvent aussi aider mes potentiels lecteurs.

**Bon courage pour la suite et, surtout,  
n'oublie pas :**

**`LATEX`, c'est la vie !**

**Troisième partie**  
**Aller plus loin avec L<sup>A</sup>T<sub>E</sub>X**

# Préambule - Le retour

Ce guide prend désormais une toute nouvelle tournure. À partir de cette page, tu décides de mettre un pied dans la cour des grands chez  $\text{\LaTeX}$ . Jusqu'à présent, je t'ai présenté des solutions simples, fonctionnelles et courantes. Désormais, je te rassure, elles seront toujours fonctionnelles mais moins faciles à comprendre et bien plus sophistiquées quant au résultat.

Toutefois, si un peu d'expérience en  $\text{\LaTeX}$  est donc nécessaire pour comprendre ce qui va suivre<sup>2</sup>, je continuerai à expliquer un minimum des notions ou certains points. Déjà, je le fais parce que sinon ce document ne mérite plus sinon d'être un guide ; ensuite parce que c'est dans ma nature ; enfin parce que je peux poser "sur le papier" mes connaissances et faciliter leur diffusion.

Du coup, si dans les parties qui vont suivre, un point ne te semble pas clair, je te recommande vivement à aller fouiner un peu sur Internet pour comprendre ce que je fais, avant d'aller crier à l'aide depuis le  $\text{\GGform}$  disponible avec ce guide, toujours à l'adresse suivante :

[http://drive.google.com/drive/folders/  
0BzU2BdcGjfU5Tk1XaXhxbk5JcEE?usp=sharing](http://drive.google.com/drive/folders/0BzU2BdcGjfU5Tk1XaXhxbk5JcEE?usp=sharing)

Comme je suis aussi extrêmement attentionné, voici une petite liste d'endroits très pratiques pour aller chercher de l'information sur  $\text{\LaTeX}$  :

- [http://www.xmlmath.net/texmaker/doc\\_fr.html](http://www.xmlmath.net/texmaker/doc_fr.html) : l'aide officielle de *Texmaker*, qui fournit aussi des indications sur  $\text{\LaTeX}$ ,
- <http://www.grappa.univ-lille3.fr/FAQ-LaTeX/> : une FAQ simple mais bien fournie,
- <http://fr.wikibooks.org/wiki/LaTeX> : un Wiki sur le  $\text{\LaTeX}$  en français,
- <http://en.wikibooks.org/wiki/LaTeX> : le "même" Wiki mais en anglais. L'information y est des fois plus précise et complète que sur le précédent,

---

2. Je ne vais pas revenir sur les bases : ce guide possède une première partie à ce sujet.



- <http://www.ctan.org/> : le site qui centralise tous les packages L<sup>A</sup>T<sub>E</sub>X ... et leur documentation officielle! Une référence absolue donc,
- <http://tex.stackexchange.com/> : les forums, c'est cool. Un forum sur L<sup>A</sup>T<sub>E</sub>X, c'est encore plus cool ... à condition de bien formuler sa demande.

Et, encore une fois, n'hésite pas à laisser des commentaires ou à signaler des fautes sur ce même GGform.

Désormais j'ai fini de blablater. Ok pour toi? Es-tu prêt? Oui? Alors plongeons un peu plus profondément dans l'univers (fabuleux) de L<sup>A</sup>T<sub>E</sub>X!

Adrien BOUZIGUES  
I13 CL215

# Chapitre 9

## Les modes de compilation sous L<sup>A</sup>T<sub>E</sub>X

### 9.1 Présentation des différents modes

Revenons un instant sur l'interface `Texmaker`. Jusqu'à présent, tu as toujours compilé sous PDF`LaTeX`, comme je t'ai dit de faire. Normalement, à l'exception du code et de l'aperçu, ta fenêtre `Texmaker` devrait ressembler à :

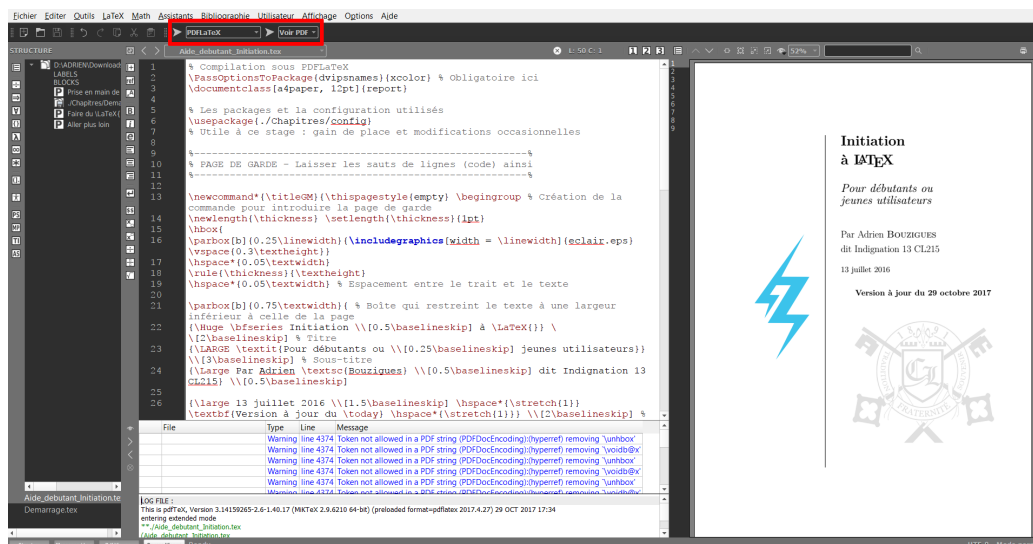


FIGURE 9.1 – Ta fenêtre `Texmaker`

Il est là dans ce but donc intéressons-nous au cadre rouge sur cette image. Normalement, c'est l'état dans lequel il se trouve. Depuis tes débuts sous `Texmaker`, il y a de grandes chances que tu cliques sur la flèche horizontale à



gauche pour lancer la compilation et que tu demandes l’affichage du résultat avec la flèche de droite.

Déjà, sache qu’il est possible d’aller beaucoup plus vite avec le clavier. En effet, **Texmaker** a mis en place des raccourcis, configurables dans **Options** → **Configurer Texmaker** → **Raccourcis**. Par défaut, appuyer sur **F6** permet de lancer la compilation sous PDFLaTeX et, **une fois que cette dernière est terminée**, appuyer sur **F7** permet d’afficher le résultat.

Il est même possible d’aller encore plus vite grâce à la compilation rapide (propriété propre à **Texmaker**, pas à L<sup>A</sup>T<sub>E</sub>X). **Texmaker** est un bon logiciel car il permet justement de combiner à notre guise différentes options pour la compilation et de les réunir sous un seul bouton. La compilation rapide est paramétrable depuis **Options** → **Configurer Texmaker** → **Compil rapide**. Par défaut, elle est assigné au bouton **F1**.

Prenons un fichier `.tex` quelconque et choisissons de mettre la compilation rapide sous **PdfLaTeX + Voir PDF**. Appuyons sur **F1** : normalement, la compilation s’est lancée toute seule et le résultat apparaît<sup>1</sup>. C’est plus pratique, ne trouves-tu pas ?

Si tu es un peu curieux, tu as pu te rendre compte qu’il existe plein d’autres possibilités pour la compilation rapide. Certaines d’entre elles contiennent justement de nouveaux modes de compilation L<sup>A</sup>T<sub>E</sub>X. Un mode de compilation, c’est un peu comme une grosse boîte qui prend en entrée un fichier `.tex` pour donner un fichier `.pdf` en sortie. Je n’en sais pas plus sur leur fonctionnement respectif mais je vais tâcher d’être clair par la suite avec quelques schémas :



FIGURE 9.2 – Schématisation d’un mode de compilation

Ensuite, si nous devons lister les différents modes de compilation, nous pouvons en relever 3 principaux :

- le mode PDFLaTeX,
- le mode LaTeX puis Dvi → PS puis PS → PDF,
- le mode XeLaTeX.

1. Hormis à l’ouverture d’un document, il peut arriver qu’il faille appuyer deux fois sur **F1** pour forcer l’affichage (petit bug technique de **Texmaker**).



Il en existe d'autres comme LuaLaTeX, voire des solutions plus exotiques (KaTeX pour du web par exemple). Ces 3 vont nous suffire pour tout le reste.

**Une question ?**

« Je ne comprends pas à quoi peuvent bien servir ces nouveaux modes de compilation. Après tout, PDFLaTeX fonctionne très bien jusqu'à présent ... »

En effet, tu as tout à fait raison. Et la conclusion de ce chapitre ira dans ce sens. Mais il existe aussi des classes ou des situations particulières qui ne fonctionnent pas sous PDFLaTeX ... mais chaque chose en son temps !

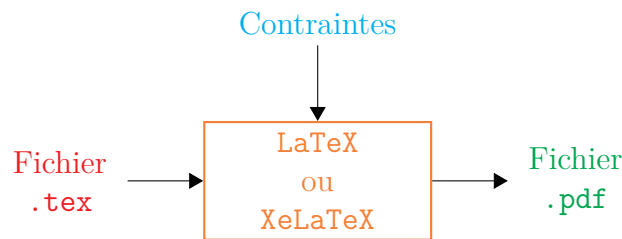


FIGURE 9.3 – Schématisation de l'utilisation d'un mode de compilation

Bon, voyons un peu ces “contraintes” (classes, packages, ...) qui nous obligent à utiliser un mode de compilation plutôt qu'un autre.

**Attention :** je risque de citer des packages ou classes jamais abordés jusqu'à présent. Certes, selon le préambule, tu peux/dois aller te renseigner à leur sujet. Toutefois, sache aussi que je les présente plus loin dans ce guide.

## 9.2 Utilisation des différents modes de compilation

Si L<sup>A</sup>T<sub>E</sub>X est très puissant, il connaît quelques limites et incompatibilités qui empêchent de générer certains résultats. C'est d'ailleurs le but du projet d'amélioration de L<sup>A</sup>T<sub>E</sub>X : s'affranchir de ces limites, entre autres. Nom de code : L<sup>A</sup>T<sub>E</sub>X3Project ; site officiel : <http://www.latex-project.org/latex3/>.





À l’heure actuelle, j’ai recensé les différents cas d’utilisation possibles suivants :

- 1) générer un PDF “simple” (rapport normal, avec juste des images et des commandes “basiques”),
- 2) générer un PDF avec des dessins faits sous PStricks (schémas, circuits électriques, ...),
- 3) changer la police d’écriture (cf. p. 93),
- 4) inclure et/ou fusionner des fichiers PDF dans le rapport (cf. p. 98),
- 5) insérer des images vectorielles .eps,
- 6) générer une bibliographie,
- 7) générer un index (cf. p. 113) ;

et du coup, pour chaque cas, il est donc possible de compiler de la manière suivante, **avec les contraintes associées** :

- 1) PDFLaTeX ⇒ images au format png ou jpg  
OU LaTeX ⇒ images au format eps ⇒ convertir les images si besoin,
- 2) LaTeX ⇒ le plus rapide  
OU XeLaTeX ⇒ le plus long ...
- 3) XeLaTeX : attention, certaines commandes ne sont plus interprétées (le symbole ° ne fonctionne plus avec `\degres{}` mais s’écrit directement au clavier ; idem pour `\og` et `\fg` qui peuvent être appelés de manière plus générique par `\guillemotleft{}` et `\guillemotright{}`),
- 4) PDFLaTeX OU XeLaTeX,
- 5) PDFLaTeX OU LaTeX,
- 6) PDFLaTeX OU LaTeX,
- 7) PDFLaTeX OU LaTeX ou XeLaTeX.

Il faut donc bien comprendre que, *a priori*, certaines combinaisons sont impossibles à réaliser :

- dessiner et inclure des PDF ⇒ passer sous XeLaTeX ⇒ génération plus longue ...



- dessiner et générer une bibliographie ⇒ passer sous L<sup>A</sup>T<sub>E</sub>X ⇒ génération rapide,
- dessiner, générer une bibliographie ET inclure des PDF : **IMPOSSIBLE** ... avec P<sub>S</sub>Tricks.

Après plusieurs recherches et tentatives de réussir à tout générer sous PDFL<sup>A</sup>T<sub>E</sub>X, comme en témoigne la génération d'index et ce guide, il existe une et une seule manière de réussir à tout combiner (dessins, bibliographie, index, inclusion de PDF et même écrire du code comme je l'ai fait tout le long de ce guide) : dessiner avec TikZ.

Si tu es intéressé, tu peux aller te renseigner sur Internet ou profiter du chapitre à ce sujet, p. 163.

#### Et les classes ?

« Tu as dit que des classes peuvent poser problèmes mais tu n'as cité que des packages pour l'instant. Essaierais-tu de nous tromper ? »

Ah, il y en a qui suit, cela fait plaisir. Non, je n'ai pas oublié. Je profite au contraire de la possibilité de générer de telles apartés pour en parler.

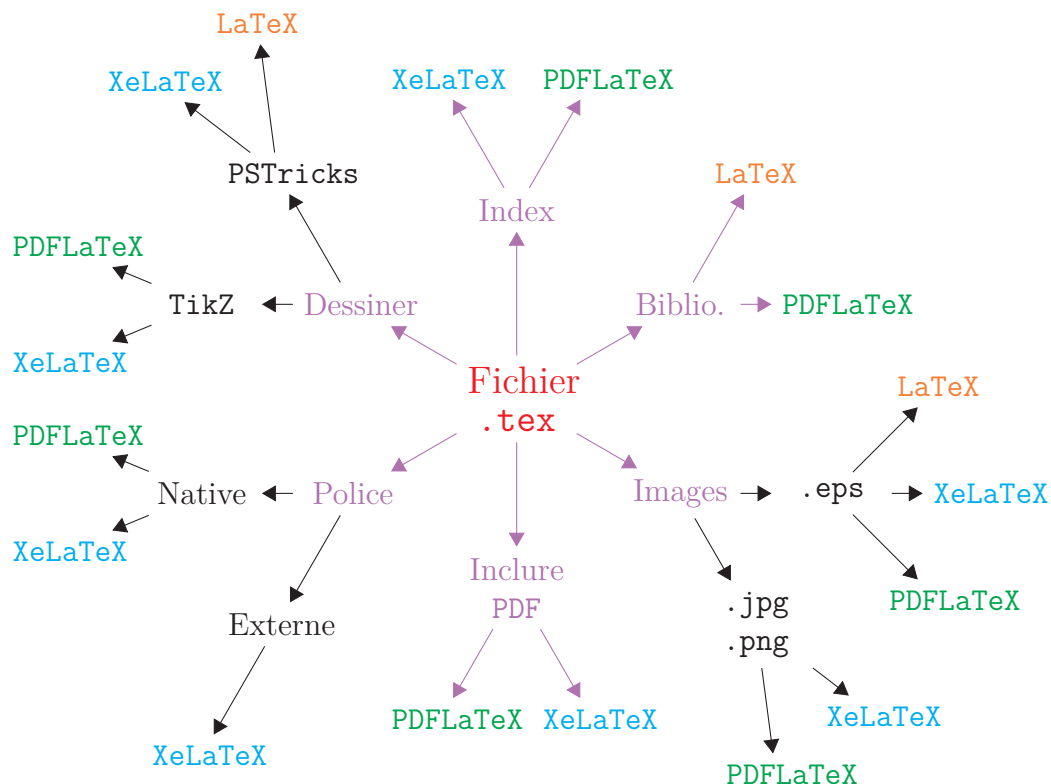
Actuellement, je n'ai rencontré qu'une seule classe qui oblige à compiler sous L<sup>A</sup>T<sub>E</sub>X : Powerdot, pour faire des diaporamas professionnels. Comme il peut être vite pénible d'avoir à convertir une multitude d'images au format .eps, une solution consiste à utiliser Beamer, qui a le bon goût de fonctionner sous PDFL<sup>A</sup>T<sub>E</sub>X.

Pour plus d'informations sur Beamer, je te renvoie à la page ?? de ce guide.

### 9.3 Bilan

Toujours en vie ? Je veux bien croire cette partie un peu technique. Elle demande un peu de pratique et de se frotter aux erreurs pour comprendre le fonctionnement et l'utilisation des modes de compilation.

Hormis le changement de police, il existe toujours un moyen de compiler sous PDFL<sup>A</sup>T<sub>E</sub>X (mode rapide et simple à utiliser pour rappel). Et si les explications précédentes sont un peu indigestes de prime abord, j'ai finalisé ce chapitre par un arbre synthétique quant au choix d'un mode de compilation :



Du coup, si tu veux utiliser plusieurs fonctionnalités, dès lors qu'elles débouchent sur le même mode de compilation, c'est bon, tu peux les utiliser facilement. Sinon, il faut trouver une autre solution ou essayer une autre approche.

### Petit bilan

Nous avons à disposition 3 modes de compilation :

- ❖ PDFLaTeX : le plus pratique et simple à utiliser ; conversion directe du `.tex` en `.pdf`,
- ❖ LaTeX et co. : série de transformations (efficaces sur de petits documents mais conversion PS  $\rightarrow$  PDF plus longue sur des documents plus importants) ; emploi obligatoire du format `.eps` pour les images,
- ❖ XeLaTeX : équivalent à PDFLaTeX (moins rapide, quelques rares



différences dans les commandes pour le texte); utilisation possible de n'importe quelle police d'écriture (*Calibri*, *Arial*, ...).

Bien, je crois avoir assez parlé de théorie désormais. Repassons à un peu plus de pratique sous L<sup>A</sup>T<sub>E</sub>X.

# Chapitre 10

## Texte et mise en forme

### 10.1 Changer la police d'écriture

Changer de police peut être un vaste débat sous  $\text{\LaTeX}$ . Déjà, pourquoi vouloir changer ? La police proposée ne te convient-elle pas ? Ou est-ce par habitude d'utiliser auparavant une autre police sous Word ?

Ensuite, quelle police préfères-tu utiliser ? Avec empattement<sup>1</sup>, sans empattement ? Mais surtout, si tu dois écrire des formules mathématiques, la nouvelle police possède-t-elle les caractères nécessaires sous un format équivalent (symboles  $\sum$  ou  $\int$ , sans parler des lettres grecques, par exemple) ?

Du coup, il existe de multiples façons de changer de police sous  $\text{\LaTeX}$ . Dans un premier temps, de manière locale, pour pouvoir juste profiter d'un effet de style ; puis, de manière globale, donc sur tout le document.

Tout comme il est possible de mettre un bout de texte en gras grâce à la commande `\textbf`, tu peux choisir d'enlever l'empatement d'un bout de texte (*sans serif*) grâce à la commande `\textsf`. Tu peux aussi choisir de le mettre en valeur différemment grâce à la mise en forme « machine à écrire » (*typewriter*) avec la commande `\texttt`.

Si tu veux appliquer un tel changement sur plusieurs paragraphes, comme l'équivalent du `\textbf` et le `\bfseries`, nous avons donc à disposition le `\sffamily` et le `\ttfamily`. Naturellement, il faut délimiter les paragraphes concernés avec des accolades “{” et “}” ou utiliser de telles commandes à l'intérieur d'un environnement (séparateur en soi).

Enfin, si tu veux faire de telles modifications sur tout le document, tu peux aussi renseigner la commande suivante dans le préambule :

---

1. Mode de terminaison d'un caractère : [http://fr.wikipedia.org/wiki/Empatement\\_\(typographie\)](http://fr.wikipedia.org/wiki/Empatement_(typographie)).



```
\renewcommand{\familydefault}{\sfdefault}
```

Voyons plutôt un exemple pour comprendre le fonctionnement de ces nouvelles commandes :

### L'empatement et la machine à écrire

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

%\renewcommand{\familydefault}{\sfdefault} % sans serif sur
  tout le document
%\renewcommand{\familydefault}{\ttdefault} % typewriter sur
  tout le document

% N.B. : les commandes locales ont la priorité sur la commande
  globale

\begin{document}

Texte \textsf{sans empatement} ou au format \texttt{machine à
  écrire}. \[\backslashbaselineskip]

{\sffamily}C'est amusant à faire surtout avec plusieurs
  paragraphes. \[

Il faut bien tout encadrer avec des accolades !} \[\backslash
  baselineskip]

\begin{large}
\bfseries\ttfamily{Sinon, je peux aussi la jouer retro sur \
  dots{}} \[

\noindent{plusieurs paragraphes, à l'intérieur d'un
  environnement !
\end{large}
% ATTENTION : le format "machine à écrire" ne respecte pas les
```



```
marges si le texte est trop long !
```

```
\end{document}
```

Bien, maintenant que nous avons vu quelques spécificités quant à l'empatement du texte, passons au changement de police. Déjà, depuis le début de ce guide, je recommande d'utiliser le package `lmodern`. Même s'il n'est pas nécessaire, il améliore le rendu et peut aussi être substitué par d'autres polices :

- ❖ `times` : un équivalent au *Times New Roman*,
- ❖ `palatino` : une autre police,
- ❖ `helvet` : un équivalent à l'*Helvetica*,
- ❖ `courier` : encore de la machine à écrire.

Les deux dernières possibilités n'ont rien donné d'intéressant pour ma part donc il faut aller creuser le sujet. Mais les deux premières fonctionnent à merveille. Pour plus d'informations sur les polices disponibles par défaut, tu peux aller sur <http://en.wikibooks.org/wiki/LaTeX/Fonts>.

### La petite subtilité

Certaines polices ne font pas la pluie et le beau temps. Par exemple, tu peux être amené à cumuler les commandes, comme mettre du texte en gras et en italique. Jusque là, tout va bien, tu peux même le faire de deux façons :

```
\textit{\textbf{test}} ou \textbf{\textit{test}}
```

As-tu déjà utilisé les petites majuscules? C'est très propre et vraiment agréable à lire. La commande? `\textsc{Texte}` ou `\scshape{Paragraphes}` sont deux solutions possibles. Toutefois, essaye maintenant :

```
\textbf{\textsc{Texte}} et \textsc{\textbf{Texte}}
```

Ben zut alors, aucune petites majuscules sous `lmodern`. C'est parce que cette police ne gère pas une telle configuration. Utiliser `times` ou `palatino` est une solution possible, mais il en existe d'autres sur Internet.

Fort heureusement, c'est un cas d'utilisation extrêmement rare. C'était surtout pour te faire toucher du doigt cette petite subtilité.



Autrement, il n'y a apparemment qu'un seul moyen de pouvoir *véritablement* changer de police d'écriture sous L<sup>A</sup>T<sub>E</sub>X. Loin de moi l'idée que la police par défaut me déplaie, bien au contraire. Toutefois, quand il faut taper un rapport officiel et qu'une police spécifique de Word est imposée, il n'y a pas d'autres solutions !

**Cette solution fonctionne uniquement grâce à une compilation sous XeLaTeX** (ou LuaLaTeX). Le résultat est à la hauteur de nos exigences : accents affichés et utilisation de toutes les autres commandes exactement de la même façon que sous PDFLaTeX (mathématiques, images, tableaux, ...). Le code est donc le suivant, sans plus d'explication technique de ma part car je n'en sais pas plus (hormis qu'il fonctionne) :

### Changer de police d'écriture

```
\documentclass[a4paper, 12pt]{report}

% XeLaTeX
\usepackage{fontspec}
\usepackage{xunicode}
\usepackage{xltextra}
\setmainfont{nom_police}
% Le paramètre nom_police est un nom de police présent dans le
% dossier Font de Windows : Arial, Calibri, Cambria, French
% Script MT, ...

\usepackage{polyglossia}
\setdefaultlanguage{french}

% Autres packages : idem

\begin{document}

J'aime écrire en \LaTeX{} ! Surtout des maths :  $i = 13\$$ .

% Taper son rapport comme d'habitude

\end{document}
```





### *Nota Bene*

La modification de la police d'écriture n'est pas recommandée dans le cas d'un document qui contient des formules. Les symboles utilisés par  $\LaTeX$  peuvent ne pas être (ou ne sont généralement pas) définis dans cette nouvelle police.



Normalement,  $\LaTeX$  générera malgré tout tes formules mathématiques avec la police par défaut soit `lmodern`.

Enfin, si une erreur incompréhensible surgit, il faut aller sur le gestionnaire de packages  $\text{MiKTeX}$  (*MiKTeX Package Manager*), désinstaller puis réinstaller le package `fontspec`. Au moment où j'écris ces lignes et que je vérifie qu'elles fonctionnent, c'est ce que j'ai dû faire en tout cas !

Dans le cas d'une **modification locale de la police** d'écriture, **toujours sous XeLaTeX**, il faut procéder ainsi :

```
texte {\fontspec{nom_police} essai} suite texte
```

Il faut toutefois savoir qu'utiliser  $\text{XeLaTeX}$  entraîne de légères modifications sur des commandes à utiliser. Actuellement, j'ai relevé les cas suivants :

- la commande `\degres{}` ne fonctionne pas et il faut directement taper le symbole ° au clavier (sous Windows,  + ),
- les guillemets français `\og` et `\fg` deviennent inopérants et il faut passer par les commandes plus génériques suivantes : `\guillemotleft{}` et `\guillemotright{}`.

Enfin, le changement de police est souvent problématique car la taille du texte n'est pas automatiquement la même. Certes, il existe des commandes comme `\large`, `\Large`, `\LARGE`, `\huge` ou `\Huge` pour augmenter la taille du texte mais il peut arriver que ce soit totalement insuffisant. Heureusement, il y a une commande toute prête dans ce cas :

```
\fontsize{taille1}{taille2}\selectfont{ }texte
```

avec `taille1` la taille de la police (13pt, 215pt, ... bref, ce que tu veux<sup>2</sup>) et `taille2` l'espacement entre les lignes : `taille2` remplace alors la valeur par défaut de `\baselineskip`.

<sup>2</sup> Le `pt` est l'unité fréquemment utilisée pour les tailles de police mais rien ne t'empêche d'en choisir une autre.



Naturellement, cette commande s'utilise surtout pour faire des gros titres, délimités par un environnement (`center` par exemple) pour ne pas affecter le reste du document. Si tu ne sais pas quoi choisir pour `taille2`, je ne réfléchis plus personnellement et prends toujours la moitié de `taille1` (arrondie si besoin).

### Bilan : le changement de police sous $\LaTeX$

Nous avons donc vu :

- comment modifier l'empatement du texte,
- comment utiliser quelques unes des autres polices présentes par défaut sous  $\LaTeX$ ,
- comment changer drastiquement la police d'écriture, en compilant sous  $\XeLaTeX$ ,
- comment personnaliser la taille de la police.

Changer la police peut se révéler amusant pour certaines réalisations personnelles mais il faut passer sous  $\XeLaTeX$  dans le pire des cas, ce qui peut augmenter le temps de compilation <sup>a</sup>.

Le format par défaut proposé depuis le début de ce guide convient tout à fait et permet de te démarquer des autres réalisations. Après tout, ne s'agit-il pas d'une marque de fabrique signée  $\LaTeX$  ?

<sup>a</sup>. Qui reste raisonnable malgré tout, je te rassure, tout au plus de l'ordre de quelques minutes, sinon de quelques secondes.

## 10.2 Inclure des fichiers PDF

Tu es pauvre ? Tu ne veux pas payer Adobe et toutes les options inutiles qu'il propose, dont la fusion de fichiers PDF ? Peu importe, c'est mon cas ! Et  $\LaTeX$  sait faire du très bon boulot, gratuitement.

Pour fusionner des fichiers PDF, il faut utiliser le package `pdfpages` et utiliser la commande :

```
includepdf[pages = debut - fin]{nom_fichier.pdf}
```

Le document `nom_fichier.pdf` doit être inclus à l'endroit où la ligne de code est insérée. Si tu écris `[pages = -]`, le document entier sera inclus. Du



coup, il ne te reste plus qu'à placer cette commande autant de fois que le nombre de fichiers à fusionner et le résultat  $\LaTeX$  contient donc tes différents fichiers les uns à la suite des autres.

Tu peux aussi juste t'en servir pour insérer une page d'un rapport PDF dans ton propre rapport ou un tableau Excel exporté en PDF : c'est des fois plus simples que de le recopier sous  $\LaTeX$ .

### *Nota Bene*

Le paramètre `nom_fichier` ne doit contenir ni espace ni accent. C'est le même principe que pour les images.

Pour rappel, la **compilation** doit se faire **uniquement** avec PDFLaTeX ou XeLaTeX.

Il existe d'autres options, comme faire une rotation à la page, ajuster automatiquement le format (`fitpaper`), réorganisation des pages. Pour les connaître ou si tu en as besoin, je te laisse aller consulter la documentation du package.

Pour présenter un point plus technique, il est possible de réaliser des références sur un tel fichier (renvoi de pages) mais il existe une astuce pour faire en sorte que le renvoi et le numéro de page correspondent.

Pour ce faire, il faut créer un compteur<sup>3</sup> dans le préambule de la manière suivante : `\newcounter{pdfpage}`, puis procéder de la manière suivante (utilisation de l'option `pagecommand` de `\includepdf`) :

### Référence des fichiers .pdf insérés

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{pdfpages} % Le package
\newcounter{pdfpage} % Le compteur
```

3. Si tu es curieux, tu peux commencer à approcher la notion de compteur ici : [http://fr.wikibooks.org/wiki/LaTeX/Programmer\\_avec\\_LaTeX#Compteurs](http://fr.wikibooks.org/wiki/LaTeX/Programmer_avec_LaTeX#Compteurs)



```

\usepackage[hidelinks, breaklinks]{hyperref}

\begin{document}

J'ai beaucoup de texte à écrire \dots{}

% Enlever le % ci-dessous
%\includepdf[pages = -, pagecommand = {\refstepcounter{pdfpage
  } \label{reference}}]{nom_fichier.pdf}

% pagecommand : faire un peu ce qu'on veut
% refstepcounter : permet le bon renvoi
% label : création de la référence

Je viens d'intégrer un document très important. Pour le
  consultant, aller à la page \pageref{reference}.

% Fonctionne bien pour un document d'une page
% Sinon, mettre plusieurs \includepdf du même document :

%\includepdf[pages = -1, pagecommand = {\refstepcounter{
  pdfpage} \label{ref_deb}}]{nom_fichier.pdf} % 1 page
%\includepdf[pages = 2-13]{nom_fichier.pdf}
%\includepdf[pages = 13-, pagecommand = {\refstepcounter{
  pdfpage} \label{ref_fin}}]{nom_fichier.pdf} % 1 page

Le document important est disponible de la page \pageref{ref_
  deb} à \pageref{ref_fin}.

\end{document}

```

**Attention :** par défaut, la numérotation des pages se poursuit lors de l'inclusion de fichiers PDF avec cette méthode. Il faut donc veiller à ce que ces derniers n'en aient pas ... ou supprimer l'insertion du numéro de page grâce à `pagecommand`.

Cette solution tombe à pic : études désormais la gestion des en-têtes et pieds de page!



## 10.3 En-têtes et pieds de page

Par défaut, L<sup>A</sup>T<sub>E</sub>X propose principalement trois styles pour les en-têtes et pieds de page. Ceux-ci sont appelés dans le préambule grâce à la commande `\pagestyle{style}` :

- le style `empty` : aucune en-tête et aucun pied de page ne seront affichés. Pour utiliser ce style, il faut donc renseigner dans le préambule la commande `\pagestyle{empty}`,
- le style `plain`, *style par défaut* : seul le numéro de page est affiché, au centre du pied de page. Comme c'est le style par défaut, il n'y a aucune commande à taper. Autrement, son appel se fait grâce à la commande `\pagestyle{plain}` dans le préambule,
- le style `headings` : les en-têtes et les pieds de page sont définis automatiquement selon la classe de document utilisée. Pour utiliser ce style, il suffit donc d'inclure la commande `\pagestyle{headings}` dans le préambule.

Si jamais tu désires personnaliser de ta main les en-têtes et pieds de page, le package `fancyhdr` est très pratique. Il faut appeler le package dans le préambule **suivi de la commande** `\pagestyle{fancy}`.

Ensuite, tu peux définir les en-têtes grâce à `\fancyhead[zone]{contenu}` et les pieds grâce à `\fancyfoot[zone]{contenu}`. Si `contenu` est totalement libre (choix de l'utilisateur), `zone` est défini de la manière suivante :

| zone | Description                           |
|------|---------------------------------------|
| L    | champ gauche pour toutes les pages    |
| LE   | champ gauche pour les pages paires    |
| LO   | champ gauche pour les pages impaires  |
| C    | champ central pour toutes les pages   |
| CE   | champ central pour les pages paires   |
| CO   | champ central pour les pages impaires |
| R    | champ droit pour toutes les pages     |
| RE   | champ droit pour les pages paires     |
| RO   | champ droit pour les pages impaires   |



Un moyen mnémotechnique très simple pour s'en souvenir : **L**, **C** et **R** pour *Left*, *Center* et *Right* (je pense que ce n'est pas le plus compliqué) ; **E** et **O** pour *Even* et *Odd*.

Il existe ensuite quelques commandes fort pratiques pour `contenu` :

- `\thepage` : affiche le numéro de la page courante,
- `\thesection` : affiche le numéro de la section courante,
- `\thechapter` avec un document de classe `book` ou `report` : affiche le numéro du chapitre courant,
- `\leftmark` : avec un document de classe `article`, affiche le nom de la section courante ; avec un document de classe `book` ou `report`, affiche le nom du chapitre courant,
- `\rightmark` : avec un document de classe `article`, affiche le nom de la sous-section courante ; avec un document de classe `book` ou `report`, affiche le nom de la section courante.

Enfin, tu peux définir les épaisseurs des traits de séparation grâce à la commande :

```
\renewcommand{\headrulewidth}{épaisseur}
```

pour les en-têtes et :

```
\renewcommand{\footrulewidth}{épaisseur}
```

pour les pieds de page.

Un petit exemple ne fait jamais de mal :

#### Les en-têtes et pieds de page

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}
```



```

\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhf{} % Effacer tout

\renewcommand{\headrulewidth}{0.5pt} % Trait en-têtes
\fancyhead[LE]{Initiation à \LaTeX{}}
\fancyhead[RE]{Adrien \textsc{Bouzigues}}
\fancyhead[LO]{\leftmark}

\renewcommand{\footrulewidth}{0pt} % Pas de trait = épaisseur
nulle
\fancyfoot[C]{\thepage}

\begin{document}

Bla bla bla

\newpage

Bla bla bla % Taper son rapport comme d'habitude

\end{document}

```

### Une question ?

« Je viens d'essayer ton exemple mais l'affichage sur les pages paires et impaires ne fonctionne pas ! Comment puis-je être remboursé ? »

Et oui, cela ne fonctionne pas et je voulais que tu touches du doigt ce problème, au moins une fois, pour pouvoir en être conscient.

Un document de classe `report` est, par défaut, considéré pour être imprimé uniquement en recto, même si tu peux choisir recto-verso au moment de l'impression.  $\LaTeX$  ne fait donc pas la différence entre les pages paires et impaires.

Pour résoudre ce problème, il faut donc dire à  $\LaTeX$  que ton document est en recto-verso. C'est possible grâce à une option supplémentaire dans le `documentclass` : `twoside`, soit :

```
\documentclass[a4paper, 12pt, twoside]{report}
```



Par exemple, dans le cas d'un document de classe `book`, le recto-verso est compris par défaut. Cette option entraîne alors des marges différentes selon les pages paires et impaires. À la lecture depuis ton écran, le résultat peut paraître déstabilisant ... et pourtant, si tu y réfléchis 2 minutes 15, c'est parfaitement logique si tu veux faire relier ton rapport !

Toutefois, *si tu tiens à avoir une feuille centrée*, avec des marges identiques à gauche et à droite, le package `geometry` est alors recommandé.

Enfin, il est possible de définir un "nom de style". Il est ainsi possible de changer à tout moment de style de page, et ce facilement. Cette astuce permet d'éviter bien des tracas et des copier-collers, tandis qu'elle respecte un des principes de L<sup>A</sup>T<sub>E</sub>X : séparer le fond et la forme. La preuve ici et maintenant :

#### Définir complètement son style de page

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{fancyhdr, fourier-orns} % En-têtes et pieds de
    pages, avec un bonus pour les en-têtes (\headrule)
\fancyhf{} % Efface tout

% Définition de notre style
\fancypagestyle{mainstyle}{
\renewcommand{\headrulewidth}{0pt} % Pas de trait en haut
\renewcommand{\footrulewidth}{0pt} % Ni en bas
\fancyhead[L]{\textsc{\nouppercase{\leftmark}}} % Mettre l'en-
    tête en small capitals --> obligatoire pour ne pas avoir
    un résultat pixellisé !
\fancyfoot[C]{\thepage}} % Fin de la définition

\pagestyle{mainstyle} % Appel de notre style
```





```

\renewcommand{\headrule}{\hrulefill\raisebox{-2.1pt}[10pt][10
  pt]{\quad\decofourleft\decofourright\quad}\hrulefill} % Bonus sur le headrule

\begin{document}

Un petit essai.

\newpage

\pagestyle{empty}Alors ? Quel est le style ?

\newpage

\pagestyle{mainstyle}Et maintenant ? Qu'avons-nous ? C'est de
  la magie, n'est-ce pas ?

\end{document}

```

Bon, que me reste-t-il à présenter ? Ah, je sais : une petite astuce, simple et courte, histoire de ce se reposer un peu.

## 10.4 Centrer verticalement du texte

À l'heure actuelle, la seule solution simple et fonctionnelle pour centrer verticalement du texte, comme un résumé par exemple, revient à utiliser la commande `\vspace*{\stretch{1}}` de la manière suivante :

```

\vspace*{\stretch{1}}
Paragraphe à centrer
verticalement
\vspace*{\stretch{1}}

```

Techniquement, la commande `\vfill` existe et reste bien plus simple à écrire ... sauf que, pour ma part, elle fonctionne une fois sur deux, voire jamais !

S'il faut aussi centrer le texte horizontalement, l'environnement `center` convient parfaitement. La commande `\hfill` (et, par la même occasion,



`\hspace*{\stretch{1}}`) doit être utilisée plutôt pour équilibrer l'espace-ment entre des blocs, comme des `minipage` ou des `subfigure` par exemple.

## 10.5 Générer une bibliographie

Comme  $\text{\LaTeX}$  peut vraiment tout faire<sup>4</sup>, pourquoi se priver et ne pas générer facilement de magnifiques bibliographies ?

Même si je commence à avoir un peu d'expérience avec les bibliographies, je n'en ai pas fait énormément pour en comprendre toutes les subtilités. Si mes explications te semblent un peu confuses, les sites :

- <http://www.tuteurs.ens.fr/logiciels/latex/bibtex.html>
- <http://www.xmlmath.net/doculatem/bibtex.html>

sont assez complets pour saisir les bases et compléter ce qui va suivre.

La première étape pour ajouter une bibliographie à un document généré sous  $\text{\LaTeX}$  est de créer la dite bibliographie, sous forme d'une base de données. Pour cela, ouvrons notre éditeur  $\text{\LaTeX}$  (`Texmaker` pour ce guide) puis créons un nouveau document. Là, renseignons les lignes suivantes qui nous serviront d'essais par la suite :

### Un début de bibliographie

```
% Renseigner le type de document pour la bibliographie ==>
% utiliser un "@" suivi du nom du document (parmi ceux
% disponibles)
@Article{Johnson,
author = {Edgar G. Johnson and Alfred O. Nier},
title = {Angular Aberrations in Sector Shaped Lenses},
journal = {Physical Review},
year = {1953},
volume = {91},
number = {1},
}
% Le premier élément est la clef (similaire à une étiquette
% pour les références)
```

4. Pour le traitement de texte, j'entends. J'attends toujours l'option ménage et repassage ...



```
% ATTENTION : il faut séparer les noms d'auteurs par "and" et
non par une virgule ou un &

@Phdthesis{Zoran,
author = {Zoran Racic},
title = {\ 'Etude et essais du spectromètre à plasma},
publisher = {Université Pierre et Marie Curie},
year = {1996}
}

% Pour renseigner un nom composé ou un groupe de mots dans "
author", il faut doubler les accolades : author = {André {
Mouche Baie}}
% Apparemment, nécessaire seulement pour les documents de type
@Misc
@Misc{opensource,
author = {{Open Source Initiative}},
title = {The Open Source Definition},
howpublished = {\url{http://opensource.org/osd}},
note = {accès le 10/10/2017}
}
```

Ensuite, il faut enregistrer le fichier. Ce doit sûrement être un peu la même recette pour tous les éditeurs de texte en général. Sous *Texmaker*, il faut procéder ainsi :

- 1) barre de menus,
- 2) Fichier,
- 3) Enregistrer sous,
- 4) `nom_fichier.bib`, comme, par exemple, `biblio_type.bib` pour ma part.

L'extension `.bib` correspond au format employé sous  $\text{\LaTeX}$  pour gérer une base de données bibliographique. C'est très important car, lors de la compilation,  $\text{\LaTeX}$  va chercher un fichier `.bib`, le lire et créer la bibliographie.

### Le conseil personnel

Je recommande de conserver le fichier `biblio_type.bib` dans un dossier à part en tant que *template* : tu n'auras pas ainsi à le créer à chaque fois mais juste à faire un copier-coller du fichier en question.



Ensuite, je ne vais pas m'étendre sur les diverses possibilités de renseigner une bibliographie (article, thèse, livre, site internet, ...). Internet fournit suffisamment d'exemples et `Texmaker` propose des commandes à trous (dans la barre de menus, `Bibliographie` puis `Bibtex` puis choisir le type de document à renseigner pour la bibliographie).

Personnellement, je viens d'en apprendre un peu plus sur les bibliographies grâce à des explications développées dans l'ouvrage *L<sup>A</sup>T<sub>E</sub>X - How To*, disponible au format numérique ici même : [http://www.latex-howto.be/book/download\\_fr](http://www.latex-howto.be/book/download_fr) (chapitre 10 pour les bibliographies).

Bon, nous avons créé la bibliographie. Il faut maintenant indiquer à `LATEX` de la générer et de l'introduire dans le document. Pour ce faire, il faut utiliser les commandes suivantes, dans cet ordre et **à l'endroit où doit apparaître la bibliographie** :

- `\bibliographystyle{smfplain}` : pour générer la bibliographie avec des normes françaises. Sans le `smf`, la bibliographie est générée selon des normes américaines.  
Par exemple, le `and` dans le fichier `.bib` pour séparer les auteurs reste tel quel, alors qu'il est remplacé par un "et" avec les normes françaises (entre autres modifications),
- `\bibliography{nom_fichier}` : pour indiquer le fichier `.bib` qui contient notre bibliographie. Ici, `nom_fichier = biblio_type`.

Puis vient la compilation. Il faut alors procéder de la manière suivante :

- 1) lancer la compilation sous `PDFLaTeX` (première compilation du document),
- 2) lancer `Bibtex` (génération de la bibliographie dans des fichiers annexes ; cf. les options `Texmaker` pour connaître le raccourci clavier associé),
- 3) relancer `PDFLaTeX` deux fois (intégration de la bibliographie et bonne implémentation des références et du sommaire),
- 4) afficher le résultat.



### *Nota Bene*

Le schéma est le même pour une compilation avec LaTeX. Il faut juste ne pas oublier à la fin de transformer son fichier DVI en PS puis de convertir le PS en PDF.

Pour gagner en efficacité et ne pas se compliquer la vie, il est préférable d'utiliser la **Compilation rapide** de Texmaker qui prend en compte la bibliographie, soit qui comprend le terme `Bib(la)tex`.

**ATTENTION :** avec cette méthode, il est impossible de générer une bibliographie sous XeLaTeX!

Maintenant, si tu as lancé la compilation, tu dois te dire que c'est nul car rien n'apparaît ... et c'est normal! L<sup>A</sup>T<sub>E</sub>X ne va pas générer une bibliographie si tu n'y fais pas référence. Te souviens-tu de la clef/étiquette créée lors de la création de ta base de données `.bib`? C'est ici qu'elle entre en jeu avec la commande `\cite{clef}`. Il te suffit donc de placer cette commande à l'endroit où tu souhaites faire une référence à ta bibliographie.

Et voilà, c'est tout. Il ne faut pas en savoir plus pour générer une bibliographie. Tu as donc désormais toutes les cartes en main pour t'y mettre et expérimenter.

### Faire fonctionner hyperref avec la bibliographie

Il peut être frustrant que les liens dans la bibliographie ne soient pas coupés, ou qu'un clic sur un élément cité ne renvoie pas à sa ligne dans la bibliographie.

Pour ce faire, il faut charger les packages ci-après, **dans l'ordre suivant :**

```
\usepackage[nottoc, notlof, notlot]{tocbibind}
% Pour inclure la bibliographie dans le sommaire

\usepackage[hyphens]{url}
% Pour couper les urls dans la bibliographie
\usepackage[hidelinks, breaklinks, linktoc = all]{hyperref}
\usepackage[hyphenbreaks]{breakurl}
% Idem package url
```



Du coup, pour résumer tout ce qui a été vu durant cette partie, ton fichier `.tex` doit ressembler à :

### Bibliographie : code complet

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[nottoc, notlof, notlot]{tocbibind} % Pour inclure
    la bibliographie dans le sommaire

\usepackage[hyphens]{url} % Pour couper les urls dans la
    bibliographie
\usepackage[hidelinks, breaklinks, linktoc = all]{hyperref}
\usepackage[hyphenbreaks]{breakurl} % Idem package url

\begin{document}

Mon document se réfère à \cite{Johnson, Zoran} mais des ré
    ponses sont aussi disponibles sur Internet \cite{
    opensource}.
% Possibilités de cumuler les \cite dans une même commande

\bibliographystyle{smfplain}
% Enlever le % ci-après
%\bibliography{biblio_type}

\end{document}
```

Tout ces éléments te semblent compliqués ? C'est bien normal au début. Là encore, il faut comprendre que tu sépares le fond de la forme : tu crées une base de données bibliographique, totalement désordonnée, puis  $\text{\LaTeX}$  affiche uniquement les éléments auxquels tu fais référence. En parallèle,  $\text{\LaTeX}$  gère automatiquement la mise en forme de la bibliographie et classe les auteurs par ordre alphabétique !

C'est donc un peu compliqué au début mais c'est extrêmement puissant.



Toutefois, si jamais tu n'as que 2-3 références à faire, il existe une manière plus simple de générer une bibliographie. Je te laisse reprendre le code ci-après et digérer mes commentaires :

### Une autre possibilité pour la bibliographie

```

\documentclass[a4paper, 12pt]{report}

% LaTeX ou PDFLaTeX
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

% XeLaTeX
%\usepackage{fontspec}
%\usepackage{xunicode}
%\usepackage{xltextra}
%\setmainfont{Calibri}
%\usepackage{polyglossia}
%\setdefaultlanguage{french}

\usepackage{enumitem, pifont} % Pour les listes à puces
% Le item par défaut, un rond, n'est pas toujours défini dans
% toutes les polices

\begin{document}

Notre bibliothèque propose trois livres \cite{latexpratique,
texbook, latexcompanion} :

\begin{itemize}[label = \ding{213}]
\item Les livres \cite{latexpratique,latexcompanion} traitent
de \LaTeX.

\item Le livre \cite{texbook} traite de \TeX{}.
\end{itemize}

% \begin{thebibliography}{affichage} : "affichage" le plus
% long (taille du texte) pour produire un alignement correct
% "affichage" est OPTIONNEL --> numérotation automatique par

```



```

LaTeX par défaut

% Ajout d'un élément bibliographique : \bibitem[affichage]{
  clef} Auteur. Titre, etc.
\begin{thebibliography}{KNU90}
\bibitem{latexpratique} Christian \textsc{Rolland}. \emph{\
  LaTeX{}} par la pratique}. O'Reilly, 1999.

\bibitem[KNU90]{texbook} Donald E. \textsc{Knuth}. \emph{The \
  TeX{}}book}. Addison-Wesley, 1990.

\bibitem{latexcompanion} Frank \textsc{Mittelbach} et Michel \
  textsc{Goosens}. \emph{The \LaTeX{}} Companion}. Addison-
  Wesley, 2004.
\end{thebibliography}

\end{document}

```

### Bilan sur la bibliographie

Pourquoi se priver d'un tableau synthétique pour résumer ces nouvelles notions ? Comparons donc l'usage d'un fichier `.bib` à l'environnement `thebibliography` :

| <code>.bib</code>   | <code>thebibliography</code>                                     |
|---|--|
| Base de données <code>.bib</code>   | Dans le fichier <code>.tex</code>                                |
| Fichier supplémentaire à gérer  | Dans le code   |
| Gestion automatique de la mise en forme (remplissage de champs dans la base de données) | Mise en forme par l'utilisateur                                  |
| Classement par ordre alphabétique des auteurs   | Entrées bibliographiques affichées dans l'ordre de leur création |





Cohérence de la  
bibliographie (affichage des  
références citées)

Compilation sous LaTeX ou  
PDFLaTeX, par  
l'intermédiaire de Bibtex

Affichage de toute la  
bibliographie créée

Tous les modes de  
compilation tolérés  
(#XeLaTeX #changement  
de police) ; pas besoin de  
Bibtex

Pour conclure, il n'y a pas de bonnes ou de mauvaises méthodes. Tout dépend uniquement de l'utilisation que tu dois en faire.

*Personnellement*, désormais, pour une courte bibliographie, je la ferai à la main, quitte à la générer avec Bibtex initialement pour voir la mise en forme, comme c'est conçu pour.

Sinon, à partir de 4-5 ouvrages à citer, je ne réfléchirai plus et implémenterai tout sous un fichier `.bib`.

Allez, passons à la cerise sur le gâteau : la génération d'un index. Ce serait dommage de s'en priver !

## 10.6 Générer un index

Un index est « un outil du livre qui consiste en une liste organisée d'éléments appelés termes (mots, concepts, objets, ...) jugés pertinents pour le lecteur, accompagnés de leur adresse - c'est-à-dire la place où ils sont évoqués dans l'ouvrage. Il permet au lecteur de localiser rapidement un élément dans l'ouvrage, sans être contraint de le lire intégralement. » (*Wikipédia*).

Pour être un poil plus précis, l'organisation d'un index se fait dans l'ordre alphabétique. Tu conviendras que c'est quand même vachement plus pratique.

Pour cette fois, je ne vais pas présenter le cas minimal qui fonctionne sous L<sup>A</sup>T<sub>E</sub>X car le résultat est relativement moche et la compilation demande d'utiliser MakeIndex (comme mode de compilation, entre deux compilations sous PDFLaTeX).



Il vaut donc mieux charger le package `imakeidx` qui permet de personnaliser l'index et même d'en gérer plusieurs. La compilation intermédiaire devient même superflue et une compilation simple sous PDFLaTeX suffit à tout générer<sup>5</sup>.

**Juste après** avoir chargé le package, il faut indiquer à L<sup>A</sup>T<sub>E</sub>X de préparer l'index, grâce à la commande `\makeindex`, avec les options `intoc` pour inclure l'index dans le sommaire et `options = {-s index-style.mst}` pour le fichier de personnalisation de l'index.

Je ne vais pas entrer dans les détails concernant la personnalisation de l'index. Un exemple fonctionnel et assez joli est disponible ci-après, tandis que le site <http://winnt.developpez.com/tutoriels/latex-index/> explique tout très bien en détail, complété par <http://fr.sharelatex.com/learn/Indices>.

Toutefois, je préciserai juste que le fichier est bel et bien au format `.mst` car c'est celui qui fonctionne pour ma part. Sur Internet, le même code est utilisé mais avec un fichier `.ist`... Je ne saurais dire, à l'heure actuelle, pourquoi un cas fonctionne chez moi et pas l'autre.

Ensuite, il suffit de placer un mot de l'index à l'endroit souhaité dans le document avec la commande `\index{mot}`. Toutefois, il faut savoir que le classement alphabétique de `makeindex` ne fonctionne pas avec les accents. Il faut donc ruser avec un “@”, de la manière suivante :

```
\index{mot_sans_accident@mot_avec_accident}
```

Il est possible de créer une sous-liste dans l'index avec un “!” :

```
\index{mot_principal!mot_secondaire}
```

Enfin, il faut dire à L<sup>A</sup>T<sub>E</sub>X de générer l'index. Pour cela, il faut utiliser la commande `\printindex` à l'endroit voulu, un peu comme pour le sommaire ou la bibliographie. Voilà, c'est tout.

### Bilan sur l'index

Pour répertorier tous les sites intéressants pour générer un index, nous avons donc :

→ commandes propres à l'index : <http://en.wikibooks.org/wiki/LaTeX/Indexing>,

5. Bon, ok, en vérité, comme l'atteste la zone propre à la compilation sous `Texmaker`, le package `imakeidx` lance en fond le mode de compilation `MakeIndex`.



- personnalisation de l'index : <http://winnt.developpez.com/tutoriels/latex-index/>,
- toutes les options de personnalisation : <http://fr.sharelatex.com/learn/Indices>.

Ok pour toi ? Voyons un petit exemple alors.

### Générer un index

```
% Environnement filecontents* : évite d'avoir à conserver un
  fichier en plus du fichier .tex
% Index style
\begin{filecontents*}{index-style.mst}
headings_flag 1 % Autoriser une lettre majuscule comme repère

heading_prefix "\\vspace{26pt}{\\bfseries\\huge " % Mise en
  forme de cette lettre (en-tête) - Début du code
heading_suffix "}\\vspace{13pt}\\nopagebreak\\n" % Fin du
  code

% En-têtes pour les cas particuliers
symhead_positive "Symboles"
symhead_negative "symboles"
numhead_positive "Nombres"
numhead_negative "nombres"

% Séparateur entre "mot de l'index" & "numéro de page" (selon
  les niveaux)
delim_0 "\\hspace{6pt}\\dotfill\\hspace{6pt}"
delim_1 "\\hspace{5pt}\\dotfill\\hspace{5pt}"
delim_2 "\\hspace{4pt}\\dotfill\\hspace{4pt}"

% Mise en forme du numéro de page
encap_prefix "{\\color{Red}\\n"
encap_infix "{"
encap_suffix "}"
\end{filecontents*}

\documentclass[a4paper, 12pt]{report}
```



```

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[makeindex]{imakeidx} % Ajout de [makeindex] pour
    préciser la compilation avec MakeIndex
\makeindex[options = {-s index-style.mst}, intoc]
% Index par défaut sur 2 colonnes
% Autre option possible : column = nbre (1, 2, 3 ...)
% Ajout de : columnsep = distance (13pt, ...) pour gérer l'é
    cart entre les colonnes

\begin{document}

Texte\index{Texte}

\newpage

Autre essai \index{Essai a confirmer@Essai à confirmer} et j'
    ajouterai aussi \index{Ajout!Ajout1} que les combinaisons
    sont possibles \index{Ajout!Ajout teste@Ajout testé}

\indexprologue{Ceci est mon index !} % Pour ajouter un petit
    texte au début de l'index
\printindex{}

\end{document}

```

Grâce au package `imakeidx`, il est possible de générer plusieurs index. Il suffit juste de leur donner un nom lors du `\makeindex` puis d'indiquer ce même nom lors de `\index` :

### Générer plusieurs index

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}

```



```

\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[makeindex]{imakeidx}
\makeindex[name = nom1, intoc]
\makeindex[name = nom2, intoc]

\begin{document}

Texte\index[nom1]{Texte} % Va dans l'index 1

\newpage

Autre essai \index[nom1]{Essai a confirmer@Essai à confirmer}
  et j'ajouterai aussi \index[nom2]{Ajout!Ajout1} que les
  combinaisons sont possibles \index[nom2]{Ajout!Ajout
  teste@Ajout testé}

\indexprologue{Ceci est mon index !} % Pour ajouter un petit
  texte au début de l'index
\printindex[nom1]{ }

\printindex[nom2]{ } % indexprologue valable que pour le plus
  proche index

\end{document}

```

Enfin, il est possible d'utiliser un index pour classer des données (liste de films, animes, jeux, pistes musicales par nom d'auteur, ...). Au lieu de tenir un Excel tout moche et de te prendre la tête pour trier le résultat, tu renseignes tous tes éléments et  $\text{\LaTeX}$  se charge de faire le classement.

Il y a une toute petite astuce à connaître : supprimer le numéro de page. En effet, ici, tous nos références doivent être affichées sur une page bidon. Dans le cas contraire, l'index ne sera pas généré (ou sera vide). Toutefois, nous n'avons pas besoin des numéros de page. Il faut donc procéder ainsi :



### Index sans les numéros de page

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[makeindex]{imakeidx}
\makeindex[intoc]

\begin{document}

Texte bidon\index{Film 1}\index{Film 2}

\newpage

% \begingroup [...] \endgroup : permet d'appliquer des
% modifications uniquement à l'intérieur des deux balises
% --> aucune influence sur le reste du document
\begingroup
\def\hyperpage#1{} % Suppression du numéro de page dans l'
index

\printindex{}

\endgroup

\end{document}
```

# Chapitre 11

## Mathématiques

La commande officielle pour écrire un vecteur est `\vec` et non pas l'immonde `\overrightarrow`. Il est aussi possible d'utiliser la commande `\vv` du package `esvect`, adapté pour l'écriture de vecteurs.

Pour placer des barres verticales, ne pas utiliser `Alt Gr + 6` mais la commande `\lvert` pour la gauche ou `\rvert` pour la droite (`\lVert` et `\rVert` pour placer des doubles barres).

Dans le cas de délimiteurs, il faut utiliser `\left\lvert` ou `\right\lvert` (`\left\lVert` et `\right\lVert` pour des doubles barres).

Pour mettre des accolades en-dessous d'une formule en mode mathématiques, la commande `underbrace` est disponible. De même au-dessus avec `overbrace`.

Le package `mathrsfs` permet d'utiliser la commande `\mathscr` pour donner un style différent de celui fourni par `\mathcal`.

Pour un intervalle avec des doubles barres, le package `stmaryrd` et les commandes `\llbracket` et `\rrbracket` fonctionnent à merveille!

### Exemples

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
```



```

\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb, upgreek, mathrsfs,
  esvect, stmaryrd} % Présentation de nouveaux packages

\begin{document}

\everymath{\displaystyle}

Pour les vecteurs, utiliser  $\vec{u}$  ou  $\mathbf{v}$  est mieux
que  $\overrightarrow{u}$ . \\

 $\left\| \vec{u} \right\|$  pour une norme ou :  $\left\| \frac{\vec{u}}{13} \right\|$  pour les délimiteurs. \\

Un cas bidon :  $\underbrace{1 - 1 + 1 - 1}_{= 0} + 13 = 13$  \\

De même :  $\overbrace{1 - 1 + 1 - 1}^{= 0} + 13 = 13$  \\

Changer la forme des lettres en mode mathématiques est inté-
ressant, comme avec  $\mathcal{X}$  pour le polynôme caracté-
ristique ou  $\mathcal{C}^0$  pour l'ensemble des fonctions
continues.

Les possibilités sont nombreuses et ne demandent qu'à être
explorées \dots \\

Besoin d'avoir un intervalle d'entiers (doubles barres) ? Du
style  $\llbracket 0, ; \rrbracket$  en clair.

\end{document}

```

Il existe aussi d'autres cas plus techniques, comme l'écriture de limites ou plusieurs lignes d'indigage dans une somme ou un produit ...

### Écriture de limites - Indigages plus complexes

Pour écrire une limite, comme  $\lim_{x \rightarrow +\infty} f(x) = 0$ , il faut procéder ainsi :

```
 $\underset{x \to +\infty}{\lim} f(x) = 0$ 
```





Dans certains cas, il faut même considérer un espace insécable. Personnellement, je trouve  $\lim_{x \rightarrow 13} f(x) = 215$  plus esthétique que  $\lim_{x \rightarrow 13^*} f(x) = 215$ . Par rapport à précédemment, la différence provient du  $+\infty$  qui dépasse de « lim » et permet donc “d’aérer” la formule. Ici, il faut donc procéder ainsi :

$$\underset{x \rightarrow 13}{\lim} f(x) = 215$$

Enfin, pour écrire plusieurs lignes l’une sous l’autre, comme  $i = 1$  et  $i \neq k$  – sous une somme, un produit ou une intégrale – il faut utiliser la commande `\substack`. Un exemple avec les polynômes d’interpolation de Lagrange (pour les connaisseurs) sera bien plus pratique qu’une longue explication :

$$L_k = \prod_{\substack{j=0 \\ j \neq k}}^n \frac{X - a_j}{a_k - a_j}$$

ce qui donne  $L_k = \prod_{\substack{j=0 \\ j \neq k}}^n \frac{X - a_j}{a_k - a_j}$ .

**Dans tous les cas**, la littérature montre l’existence d’une commande `\limits` qui peut servir pour ces exemples. Non seulement j’ai lu qu’il était déconseillé de l’utiliser <sup>a</sup>, mais je viens aussi de montrer qu’obtenir un résultat est possible sans cette commande donc évitons de nous compliquer la vie !

<sup>a</sup>. Si tu veux savoir pourquoi, je te laisse chercher. Moi, j’ai oublié.

# Chapitre 12

## Tableaux & boîtes

### 12.1 Un autre format de cellules

Dans la partie précédente de ce guide, j'ai présenté les formats par défaut pour les cellules d'un `tabular` : `l`, `c` et `r`. Il en existe un 4<sup>ème</sup>, qui prend en compte un argument : le format `p{longueur}`.

`p` permet de définir une colonne de largeur définie par `longueur` et dont le contenu est aligné à gauche par défaut.

Si, comme nous le verrons par la suite, la commande `linebreak` est utile pour écrire du texte sur une nouvelle ligne (dans la même cellule), elle peut parfois créer de grandes espaces blancs entre les mots (pour justifier le texte).

Du coup, il existe une commande spécifique au format `p` qui permet d'éviter tous ces désagréments : `\newline`. Comme toujours, un exemple sera bien plus explicite :

#### Le format de cellule `p`

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{array}
```



```

\begin{document}

\begin{center}
\begin{tabular}{p{0.2\linewidth}|p{0.4\linewidth}}
Du texte \linebreak Un très très très long texte & \hfil Titre
centré \hfil \\ \hline
Nouvel essai \newline Plus joli \newline Espacement correct \
newline Utile pour des dates : \newline XXXX \newline YYYY
& \begin{center} Ok ? Mieux centré ! Mais pas encore top
\dotsc{} \end{center}
\end{tabular}
\end{center}

\end{document}

```

### *Nota Bene*

Pour une raison que j'ignore, le centrage du texte dans une cellule au format `p` ne fonctionne pas avec la commande usuelle `\hfill`, qui encadre le texte.

Une commande équivalente, `\hfil` ou `\hspace*{\stretch{1}}`, fonctionne parfaitement, ainsi que l'environnement `center`. La première commande possède l'avantage d'être la plus concise à écrire. Il s'agit toutefois d'un point encore délicat donc cette solution peut encore être améliorée.

## 12.2 Cellules centrées verticalement et horizontalement

Au lieu d'utiliser la lettre `c`, une commande plus complexe permet de satisfaire un centrage vertical et horizontal du contenu de la case :

```
>{\centering\arraybackslash}m{taille_case}
```

Dès lors, l'agencement peut se faire de la manière suivante :



### Un exemple avec un SWOT

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{graphicx} % Pour la commande rotatebox
\usepackage{array}

\begin{document}

\begin{center}
\begin{tabular}{>{\centering\arraybackslash}m{0.5cm}|>{\centering\arraybackslash}m{7cm}|>{\centering\arraybackslash}m{7cm}}
~& {\Large Atouts} & {\Large Handicaps} \\ \hline
\rotatebox{90}{{\Large Interne}} & \vspace{\baselineskip} \textbf{Forces (Strengths)} \linebreak \linebreak - rapidité de production \linebreak - etc. \linebreak & \vspace{\baselineskip} \textbf{Faiblesses (Weaknesses)} \linebreak \linebreak - idem \linebreak \\ \hline
\rotatebox{90}{{\Large Marché}} & \vspace{\baselineskip} \textbf{Opportunités (Opportunities)} \linebreak \linebreak - idem \linebreak & \vspace{\baselineskip} \textbf{Menaces (Threats)} \linebreak \linebreak - idem \linebreak
\end{tabular}
\end{center}

\end{document}

```

Le raccourci pour définir  $N$  colonnes identiques est toujours utilisable. La formulation ne change pas : c'est toujours `*{nbre_col}{commande}`. Un exemple est donné plus loin.

En revanche, il peut vite être pénible d'écrire (ou de copier-coller) cette ligne à chaque fois. C'est pourquoi il existe une commande pour définir de nouveaux formats de colonne :



```
\newcolumntype{nom_format}[nbre_arg]{def_format}
```

soit, dans notre cas :

```
\newcolumntype{C}[1]{>{\centering\arraybackslash}m{#1}}
```

Cette **commande se place dans le préambule** du document  $\text{\LaTeX}$ . Il faut donc procéder de la manière suivante :

### Application

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{array}
\newcolumntype{C}[1]{>{\centering\arraybackslash}m{#1}} % L'
    argument #1 est une longueur (2cm, etc.)

\begin{document}

\begin{tabular}{|C{2.5 cm}||C{0.5\linewidth}||}
Ligne bidon 1 & Ligne bidon 2
\end{tabular}

C'est plus pratique, n'est-ce pas ?

\end{document}
```

## 12.3 Fusion et coloriage de cellules

Si la fusion des colonnes et des lignes est comprise de base avec le package `array`, la fusion des lignes laisse un peu à désirer. C'est pourquoi il vaut mieux charger le package `multirow`.

La fusion des colonnes a lieu grâce à la commande :

```
\multicolumn{nbre_col}{position}{texte}
```



et la ligne du tableau s'achève par la commande `\cline{col_debut-col_fin}`.

Quant à la fusion des lignes, elle est possible grâce à la commande :

```
\multirow{nbre_ligne}*{texte}
```

Afin de pouvoir colorier les cellules, il faut ajouter le package `makecell` dans le préambule. Grâce à la commande `\rowcolor{couleur}`, tu peux colorier une ligne entière mais il existe d'autres commandes pour cibler les cases. Pour connaître les dites commandes, il faut aller se documenter sur Internet. Pour le reste, un exemple sera plus clair :

### Coloriage et fusion

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{array, multirow, makecell} % Packages pour la
    fusions des lignes et le coloriage
\newcolumntype{C}[1]{>{\centering\arraybackslash}m{#1}}
\renewcommand{\arraystretch}{1.3} % Pour augmenter la hauteur
    d'une ligne - Meilleure lisibilité
\usepackage[table, dvipsnames]{xcolor} % Ajout de table pour
    appliquer dvipsnames aux tableaux

\begin{document}

\begin{center}
\begin{tabular}{|*{4}{C{3cm}}|}
\hline
\rowcolor{Peach} \multicolumn{4}{|c|}{\textbf{Comparaison des
    configurations}} \linebreak \\\hline
\multirow{2}{*{\textbf{Critères}}} & \multicolumn{3}{c|}{\textbf{
    Structures}} \linebreak \\\cline{2-4}
~& \textbf{Fonctionnelle} & \textbf{Divisionnelle} & \textbf{
    Matricielle} \\\hline
Stabilité & ++ & + & - \\\hline
Flexibilité - Adaptabilité & - & - & ++ \\\hline
\end{tabular}
\end{center}
\end{document}
```



```

\rowcolor{black} ~ & ~ & ~ & ~ & ~ \\ \hline
\textbf{Cas pratiques} & \multicolumn{3}{|c|}{\textbf{Qui ré
    ussira le mieux à répondre aux besoins ?}} \\ \hline
Projet (long) à réaliser & - & - & & ++ \\ \hline
Fabrication en grande série & ++ & - & & -- \\ \hline
\end{tabular}
\end{center}

\end{document}

```

Il se peut que la visualisation des bordures noires du tableau ne s'affiche pas bien avec la cellule colorée. Il s'agit juste d'un problème d'affichage avec ton écran d'ordinateur, tellement la ligne est fine.

## 12.4 longtable & booktabs

Il s'agit de deux packages que j'ai découverts en 2017. Le premier se révèle très utile pour des tableaux dont la longueur dépasse une page. Il suffit juste de remplacer l'environnement `tabular` par `longtable` et tout le reste fonctionne exactement pareil, y compris les nouveaux formats de colonnes définies par l'utilisateur, comme abordé précédemment.

D'autres options existent aussi pour rappeler lors d'un changement de page les titres des colonnes. Si tu es intéressé, je te renvoie à la documentation officielle.

Quant au second package, c'est le seul à ma connaissance qui permette d'obtenir des tableaux suffisamment aérés. En théorie, la commande :

```
\renewcommand{\arraystretch}{1.3}
```

permet de modifier la hauteur de ligne, pour tous les tableaux créés après cette commande. Mais, avec des fractions, impossible de modifier quoi que ce soit ... sauf avec `booktabs` ! Pourquoi ? Je n'en sais rien. Hormis l'absence de barres verticales, le rendu est meilleur et convient.

### Un exemple d'application

```
\documentclass[a4paper, 12pt]{report}
```



```

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{array, longtable, booktabs}
\newcolumntype{C}[1]{>{\centering\arraybackslash}m{#1}}
\renewcommand{\arraystretch}{1.3} % Pour augmenter la hauteur
    de ligne des tableaux

\begin{document}

\everymath{\displaystyle}

Allons faire un peu de maths \dots{} Mais avant, mangeons un
    peu de place : \\

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi
    venenatis orci vitae odio varius ultrices. Pellentesque
    habitant morbi tristique senectus et netus et malesuada
    fames ac turpis egestas. Vivamus eget interdum ipsum. Ut
    et odio nec ipsum luctus tempor. In dapibus sapien ut
    sagittis congue. Suspendisse scelerisque molestie leo vel
    tempus. Vivamus facilisis ipsum quis hendrerit vestibulum.
    Fusce malesuada dictum nunc non molestie. Proin maximus
    neque neque, eu molestie nulla condimentum eget. Integer
    tincidunt ut velit at tristique. Suspendisse quis erat ac
    ligula facilisis commodo quis sit amet est. Maecenas a
    nulla nec lorem vehicula tempor. Donec tristique purus
    vitae nibh pretium venenatis. Fusce pellentesque convallis
    neque eu sagittis. Sed eu ante egestas felis sagittis
    tincidunt.

\begin{center}
\begin{longtable}{C{0.3\linewidth}C{0.3\linewidth}}
 $f(x)$  &  $\int f(x)\,dx$  \\ \midrule % \midrule : commande
    propre à booktabs - Création d'une ligne horizontale (= \
    hline)
 $x^\alpha$ , avec  $\alpha \neq -1$  &  $\frac{x^{\alpha + 1}}{\alpha + 1}$  \\ \midrule

```





```

 $\frac{1}{x}$  &  $\ln |x|$  \\ \midrule
 $\cos(ax)$ , avec  $a \neq 0$  &  $\frac{\sin(ax)}{a}$  \\ \midrule
 $\sin x$  &  $-\cos x$  \\ \midrule
 $\frac{1}{1+x^2}$  &  $\arctan x$  \\ \midrule
 $\cosh x$  &  $\sinh x$  \\ \midrule
 $\sinh x$  &  $\cosh x$  \\ \midrule
 $e^{\omega x}$ , avec  $\omega \neq 0$  &  $\frac{e^{\omega x}}{\omega}$  \\ \midrule
 $\frac{u'}{u}$  &  $\ln |u|$  \\ \midrule
 $\tan x$  &  $-\ln |\cos x|$  \\ \midrule
 $\frac{1}{\sqrt{1-x^2}}$  &  $\arcsin x$  \\ \midrule
 $\frac{-1}{\sqrt{1-x^2}}$  &  $\arccos x$  \\ \midrule
 $\frac{1}{\sqrt{x^2+1}}$  &  $\operatorname{argsinh} x$  \\ \midrule
 $\frac{1}{\sqrt{x^2-1}}$  &  $\operatorname{argcosh} x$  \\ \midrule
 $\frac{1}{1-x^2}$  &  $\operatorname{argtanh} x$  \\ \end{longtable}
\end{center}

\end{document}

```

Ici, j'ai cumulé l'emploi de `longtable` et de `booktabs` pour optimiser l'exemple mais rien n'empêche d'utiliser uniquement l'un ou l'autre.

## 12.5 Créer sa propre boîte

Le package `tcolorbox` propose de très nombreux outils pour créer des boîtes. Il propose même une commande pour créer soi-même son propre environnement, et donc sa propre boîte entièrement personnalisable.

La structure est très particulière et propre à `tcolorbox` mais permet de respecter la philosophie  $\text{\LaTeX}$  : créer des commandes. Un changement sur la commande entraîne un changement sur tout le document. Tu admettras que c'est vachement plus pratique que de devoir tout passer en revue !

Le code fonctionne selon le format suivant :

```
\newtcolorbox{nom}[nbre][options, #1]
```

avec :

→ `nom` : nom de l'environnement,



- `nbre` : le nombre de paramètres, supérieur ou égal à 1 (et probablement inférieur à 9),
- `[]` : obligatoire pour permettre le changement ou l'ajout d'options,
- `options` : les options disponibles avec `tcolorbox`,
- `#1` : formulation particulière mais correspond tout simplement au texte de l'environnement.

Nous avons donc le code complet suivant, avec quelques exemples :

### Ma petite boîte personnelle

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor}

\usepackage[breakable]{tcolorbox}

\newtcolorbox{boite}[3][\]{breakable, before upper = {\parindent17pt}, beforeafter skip = \baselineskip, colframe = #3, colback = White, boxrule = 2pt, arc = 6pt, title = \textbf{#2}, coltitle = Black, #1} % Commande pour créer sa boîte

% breakable : pour casser la boîte si elle arrive en fin de page
% before upper : pour contrôler la taille de l'alinéa
% beforeafter skip : longueur à appliquer avant et après la boîte
% colframe : couleur du contour
% colback : couleur du fond
% boxrule : épaisseur du contour
% arc : rayon des bords
% title et coltitle : obvious !

\begin{document}
```



```

\begin{boite}{Un premier exemple}{Orchid}
C'est pratique, n'est-ce pas ?
\end{boite}

\begin{boite}[colback = Apricot]{Un deuxième exemple}{Orange}
Apportons un petit changement, juste pour cette fois.
\end{boite}

\begin{boite}{Un troisième exemple}{Cyan}
Retour sur un cas normal d'utilisation.
\end{boite}

\end{document}

```

Le principe est donc de créer une nouvelle boîte (même schéma que pour une commande) pour chaque cas (boîte pour les définitions, boîte pour les remarques, etc.) au lieu de devoir changer les options à chaque fois.

Utiliser  $\LaTeX$ , c'est tout automatiser pour se concentrer sur le fond et moins sur la forme, une fois qu'elle est définie en tout cas.

## 12.6 Afficher du code $\LaTeX$

Il n'y a pas 13 000 façons d'afficher du code  $\LaTeX$  simplement (oui, parce qu'écrire `\texttt{\textbackslash}` pour produire `\`, ce n'est vraiment pas pratique). La première est d'utiliser l'environnement `verbatim` : tout ce qui est compris dans cet environnement ne sera pas interprété par  $\LaTeX$  et sera recopié tel quel. Cependant, une trop longue commande sort des marges voire de la page ... C'est donc vite compliqué de lire la fin ou même de copier un morceau de code.

Toutefois, il existe déjà un moyen simple pour les “petites” formules : la commande `\verb?\commande?`. Il faut juste écrire `\verb` et tout ce qui est compris entre les deux délimiteurs (ici, des `?`) subit le même traitement que sous l'environnement `verbatim`. `?` n'est pas le seul délimiteur envisageable : `!` convient tout à fait par exemple.

Autrement, le seul package qui permette de mettre en forme du code ( $\LaTeX$ , mais aussi Python, Perl, C, C++, Java, SQL, HTML, ...) avec un retour à la ligne s'appelle `listings`. Beaucoup d'options de mise en forme sont possible (numéro de ligne de code sur le côté, commandes en couleur



selon le langage, ...) mais il y a mieux. Et oui, c'est maintenant que revient le Saint Graal : `tcolorbox` ! Ce dernier intègre le package `listings` et permet de faire pas mal de trucs sympatiques.

Malgré tout,  $\LaTeX$  est loin d'être parfait et renvoie une erreur si des accents sont présents dans le code. Cela peut déjà s'expliquer parce que la programmation est généralement en anglais et ne comporte pas d'accents donc il n'y a aucun souci. Mais je suis Français et donc, personnellement, il m'arrive de rédiger mes commentaires de code en français. Pour résoudre ce problème, il faut soit compiler sous `XeLaTeX` soit sous `PDFLaTeX` sous réserve d'introduire les accents.

Un exemple sera sûrement plus clair et m'évitera de longues explications :

### Écrire du code sous $\LaTeX$

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor}
\usepackage[breakable, listings]{tcolorbox} % Ajout de
      listings en option

% Création d'une boite avec listings
\newtcblisting{codeLaTeX}[2][]{breakable, beforeafter skip = \
  baselineskip, colframe = LimeGreen, colback = White,
  boxrule = 2pt, arc = 6pt, title = \textbf{\#2}, coltitle =
  Black, listing options = {basicstyle = \ttfamily\small,
  keepspaces = true, columns = flexible, breaklines = true,
  inputencoding = utf8, language = TeX, numbers = none},
  listing only, \#1} % options tcolorbox : OK

% listing options : options du package listings
% Ici, dans l'ordre : petite police de type "machine à écrire"
  (ttfamily) ; conserver les espaces ; nécessaire pour les
  espaces ; retour à la ligne du code ; UTF-8 ; pour le
  LaTeX ; pas de numéro
```



```

% Pour les accents
\lstset{literate = {à}{\`a}1 {â}{\^a}1 {é}{\`e}1 {è}{\`e}1
  {ê}{\^e}1 {ï}{\^i}1 {ô}{\^o}1 {ç}{\c{c}}1} %
  Format : {lettre UTF8}{[lettre LaTeX]}1
% A compléter si besoin

\begin{document}

\begin{verbatim}
Très pratique d'écrire du code sous \LaTeX{} \

Mais c'est encore plus compliqué quand le code en question est
extrêmement long \dots{}
\end{verbatim}

\begin{codeLaTeX}{Titre}
Très pratique d'écrire du code sous \LaTeX{} \

Mais c'est encore plus compliqué quand le code en question est
extrêmement long \dots{} Ah non, là c'est mieux !
\end{codeLaTeX}

Le dernier résultat est mieux, non ? Tu peux même le
personnaliser (changement ponctuel) :

\begin{codeLaTeX}[colframe = BrickRed]{Second test}
Code vraiment important qui demande de changer de couleur.
  Juste pour lui.
\end{codeLaTeX}

\end{document}

```

Il est possible d'aller encore plus loin, toujours grâce aux packages `tcolorbox` et `listings`, et de compléter le précédent code pour avoir les options suivantes :

- numérotation automatique des boîtes,
- coloriage syntaxique du code (mots clés, chaînes de caractères, commentaires, numéros de ligne),
- création d'un sommaire dédié à ces boîtes.



La documentation `tcolorbox` fournit toutes les indications nécessaires. Comme c'est difficile de tout faire tenir avec l'environnement verbatim, je vais balancer le code et allègrement tout commenter :

### Intégrer du code - Aller plus loin

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor} % Pour les couleurs

\usepackage[breakable, listings]{tcolorbox} % Pour des boîtes
arrondies - breakable : pour couper les boîtes - listings
: pour le code

\usepackage[hidelinks, breaklinks, linktoc = all]{hyperref}

% Création d'une boîte numérotée pour le code
% Utilisation de Python pour changer
\newtcblisting[auto counter, number within = chapter, list
inside = pythoncode]{codePython}[2][]{breakable,
beforeafter skip = \baselineskip, colframe = LimeGreen,
colback = White, boxrule = 2pt, arc = 6pt, title = \textbf
{Code Python \thetcbcounter{} : #2}, list entry = {\
protect\numberline{\thetcbcounter}#2}, coltitle = Black,
listing options = {basicstyle = \ttfamily\small,
keepspaces = true, breaklines = true, inputencoding = utf
8, language = Python, tabsize = 4, xleftmargin = 17pt,
numbers = left, numbersep = 13pt, numberstyle = \ttfamily\
footnotesize\color{OrangeRed}, showstringspaces = false,
commentstyle = \color{Gray}, keywordstyle = \color{Blue},
stringstyle = \color{Orange}}, listing only, #1}

\lstset{literate = {\à}{\`a}1 {\â}{\^a}1 {é}{\`e}1 {è}{\`e}
e}1 {ê}{\^e}1 {ï}{\`i}1 {ô}{\^o}1 {ç}{\c{c}}1} %
Pour résoudre le problème des accents dans le code
```



```

listings (tcolorbox)

% --- Les explications ---
% Intégration d'un compteur dans l'en-tête et liste des codes
  : [auto counter, number within = chapter, list inside =
  pythoncode] <==> numérotation automatique ; numéro de
  chapitre utilisé ; étiquette pour le sommaire dédié
%\newtcblisting[...] : création d'un environnement pour le
  code avec tcolorbox + listings

% Options tcolorbox : inchangées
% \thetcbcounter{} : compteur tcolorbox
% Intégrer une entrée dans le sommaire dédié : list entry = {\
  protect\numberline{\thetcbcounter}#2} (dans les options
  tcolorbox)

% Options listings : début inchangé
% Options listings propre au langage : {language = Python,
  tabsize = 4, xleftmargin = 17pt, numbers = left, numbersep
  = 13pt, numberstyle = \ttfamily\footnotesize\color{
  OrangeRed}, showstringspaces = false, commentstyle = \
  color{Gray}, keywordstyle = \color{Blue}, stringstyle = \
  color{Orange}}

% Détails : langage utilisé ; taille de l'indentation (nbre d'
  espaces) ; marge à gauche ; numéro de ligne à gauche ; sé
  paration entre le code et les numéros ; style des numéros
  ; bonus pour les chaînes de caractères ; mise en couleur
  du code

% Affichage uniquement du code : listing only --> "listing and
  text" = code et résultat (utile pour du code LaTeX mais
  que du code pur : pas de préambule ou de \begin{document})

\begin{document}

\renewcommand{\contentsname}{Sommaire} % Renommer le sommaire
\tableofcontents % Pour créer le sommaire - Penser à compiler
  deux fois

\newpage

```



```

\phantomsection % Renvoi correct (hyperref)
\addcontentsline{toc}{chapter}{Liste des codes Python} % Ajout
dans le sommaire
\tcblistof[\chapter*]{pythoncode}{Liste des codes Python} %
Sommaire dédié --> cf. aide tcolorbox
% Ajout de "Liste des codes Python" en tant que chapitre non
numéroté (chapter*)

\newpage

% Enlever le commentaire ci-dessous
%\chapter{Codes Python}

Blablaba

\begin{codePython}{Essai}
#!/usr/bin/python3
# -*- coding: utf-8 -*-

S = 0
for i in range(215):
    S += i
print(S)
\end{codePython}

\begin{codePython}[colframe = Red]{Essai important}
#!/usr/bin/python3
# -*- coding: utf-8 -*-

S = 0
for i in range(13):
    for j in range(13):
        S += i + j
print(S)
\end{codePython}

Alors ? Plutôt propre, n'est-ce pas ?

\end{document}

```





Il existe un autre package que `listings` qui s'occupe automatiquement du coloriage du code : `minted`. Il fonctionne grâce à `pygments`, une bibliothèque Python. Par contre, même après son installation, je n'ai toujours pas réussi à le faire fonctionner ...

# Chapitre 13

## Images

### 13.1 Une référence toute prête

Les références, c'est bien. Les automatiser, c'est mieux. J'étais plutôt agacé d'écrire constamment « (cf. `FIGURE ref` p. `page-ref`) », d'autant plus que le mot « `FIGURE` » peut varier selon la classe.

Puis, j'ai découvert la commande `\figurename{}` : elle contient justement le nom utilisé dans la légende. Il est donc possible d'automatiser mon problème initial de la manière suivante :

#### Une référence fraîche ! Une !

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{graphicx, float}

\usepackage[hidelinks, breaklinks, linktoc = all]{hyperref}

\newcommand{\reffig}[1]{(cf. \figurename{} \ref{#1} p. \
  pageref{#1})}

\begin{document}
```



```

\begin{figure}[H]
\centering
%\includegraphics[width = 0.5\linewidth]{test.png}
\caption{La légende}
\label{essai}
\end{figure}

Toutes les explications sont disponibles ci-avant \reffig{
  essai}.

\end{document}

```

## 13.2 L'environnement subfigure

Il peut être intéressant d'afficher plusieurs images avec chacune sa légende, ainsi qu'une légende globale pour toutes les images. Une solution très simple à implémenter est possible grâce au package `subcaption` :

- apport de l'environnement `subfigure`, au fonctionnement identique à une `minipage`,
- apport de la commande `\subcaption`, pour mettre une légende sous chaque image.

Un aperçu de cette solution prend la forme suivante :

### Utilisation de `subfigure`

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{graphicx, float, subcaption}

\begin{document}

```



```

\begin{figure}[H]
% [t] en option : permet d'aligner correctement les images si
% les sous-légendes sont longues
\begin{subfigure}[t]{0.32\linewidth}
%\includegraphics[width = \linewidth]{image1.png}
\subcaption{Image 1}
\end{subfigure}
\hfill
\begin{subfigure}[t]{0.32\linewidth}
%\includegraphics[width = \linewidth]{image2.png}
\subcaption{Image 2 : avec une très longue légende}
\end{subfigure}
\hfill
\begin{subfigure}[t]{0.32\linewidth}
%\includegraphics[width = \linewidth]{image3.png}
\subcaption{Image 3}
\end{subfigure}
\caption{Images de test}
\end{figure}

% Pas le même rendu au niveau des légendes
\begin{figure}[H]
\begin{minipage}{0.32\linewidth}
%\includegraphics[width = \linewidth]{image1.png}
\caption{Image 1}
\end{minipage}
\hfill
\begin{minipage}{0.32\linewidth}
%\includegraphics[width = \linewidth]{image2.png}
\caption{Image 2}
\end{minipage}
\hfill
\begin{minipage}{0.32\linewidth}
%\includegraphics[width = \linewidth]{image3.png}
\caption{Image 3}
\end{minipage}
\caption{Images de test}
\end{figure}

% Les références doivent toujours fonctionner --> faire des

```



```

    essais et lire la documentation

\end{document}

```

### La petite subtilité

La commande `\subcaption` ne requiert pas l'utilisation systématique de l'environnement `subfigure`. Elle peut très bien être utilisée avec une `minipage`.

Toutefois, il faut bien comprendre qu'une `\caption` ou `\subcaption` n'est valide que dans un élément flottant soit dans un environnement global `figure`.

De même, l'environnement `subfigure` ne peut être utilisé que dans un environnement `figure`, contrairement à une `minipage` qui peut être utilisée n'importe quand. Cette dernière n'est donc pas considérée comme un flottant.

Enfin, le package `subcaption` charge aussi le package `caption`, ce qui permet d'utiliser la commande `\caption*`. Cette dernière permet d'avoir une légende sans numéro. C'est toujours pratique de temps en temps.

## 13.3 Insérer un grand nombre de fichiers

Il est possible d'être amené, ponctuellement, à regrouper un grand nombre de fichiers (images, PDF, ...) dans un seul et unique PDF.

Si écrire toutes les lignes de code ou faire des copier-coller pour n'avoir qu'à modifier les noms de fichiers à la fin peut fonctionner, il existe une méthode plus élégante et efficace qui consiste à utiliser une boucle `for`.

Ce n'est pas bien compliqué et tout est résumé dans le code ci-dessous :

### Insertion avec une boucle for

```

\documentclass[a4paper, 12pt]{report}

\usepackage{graphicx, float} % Si images
\graphicspath{./Images/} % Chemin des images
\DeclareGraphicsExtensions{.jpg} % Pour définir l'extension

```



```
des images

\usepackage{pgffor} % Pour la boucle for

\begin{document}

% Page de garde ou ce que tu veux

% Commande d'insertion avec la boucle for
% #1 = numéro début
% #2 = numéro fin
% #3 = nom devant le numéro
\newcommand*\insertgraphicsfiles[3]{\foreach \i in
    {#1,...,#2} {%
\begin{figure}[H]
\vspace*{\stretch{1}}
\centering
\includegraphics[width=0.99\linewidth]{#3\i}
\vspace*{\stretch{1}}
\end{figure}
}}

% Insertion images 001 à 009
\insertgraphicsfiles{1}{9}{00}

% Insertion images 010 à 099
\insertgraphicsfiles{10}{99}{0}

% Insertion images 100 à 151
\insertgraphicsfiles{100}{151}{}

% Il est possible de faire de même avec des PDF et \includepdf

\end{document}
```

Et voilà ! Si tu ne juges pas ce passage intéressant, la réalisation d'un trombinoscope peut constituer une application plus concrète de l'utilisation d'une boucle for :



### Une autre application : le trombinoscope

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{graphicx}
\graphicspath{{./Images/}} % Chemin des images
\DeclareGraphicsExtensions{.png} % Pour définir l'extension
    des images

\usepackage{array}
\usepackage{pgffor} % Pour les boucles

\setlength\parindent{0pt} % Pour supprimer les indentations (
    inutiles ici)

\newcommand*\affichechoriste[3]{\begin{tabular}{c} \
    includegraphics[width = 0.32\linewidth]{#1} \\ #2 \textsc
    {#3} \end{tabular} }
% Attention, l'espace après le tabular est indispensable pour
    les renvois
% Sinon, tout s'affiche sur une seule ligne

\begin{document}

\begin{center}
\Huge{\textsc{Titre}}
\end{center}

\foreach \prenom/\nom/\fichier in {%
Prénom/Nom/001,%
Prénom/Nom/002,%
Prénom/Nom/003,%
Prénom/Nom/004}{\affichechoriste{\fichier}{\prenom}{\nom}}
% Et ainsi de suite. Il vaut mieux ne pas mettre d'accent ni d
    'espace dans les noms de fichiers
% Le % en fin de ligne est indispensable au bon fonctionnement
    de \foreach pour éviter l'insertion de blancs qui

```



```
troubleraient l'appel du fichier (et permet de faciliter
la relecture du code)
```

```
\end{document}
```

Nous reviendrons plus tard, avec le chapitre sur TikZ, sur des utilisations de la boucle `for`.

## 13.4 Insérer un fichier `.svg`

Si, comme moi, tu apprécies ne pas avoir de gros pixels immondes au moindre zoom de ton fichier `.pdf`, il est possible d'importer un fichier `.svg` (image vectorielle donc pas de pixels au zoom) dans ton document.

Pour ce faire, aucun package supplémentaire n'est requis et il faut juste suivre la procédure suivante :

- ❖ exporter le fichier `.svg` sous Inkscape au format `.pdf`,
- ❖ dans les options, choisir “Exclure le texte ...” et “Utiliser la taille ...”,
- ❖ garder les DEUX fichiers générés (`.pdf` et `.pdf_tex`),
- ❖ utiliser le code suivant :

```
\begin{figure}[H]
\centering
\def\svgwidth{\columnwidth}
%\input{nom_fichier.pdf_tex}
\caption{Légende éventuelle}
\end{figure}
```

- ❖ compiler, peu importe le mode (LaTeX, PDFLaTeX, ...).

Et voilà, c'est tout ce qu'il y a à faire. Après, c'est vraiment se prendre le chou pour pas grand chose. Autant rester sous Inkscape, exporter l'image au format `.eps` et l'intégrer comme n'importe quelle image.

Les pixels ne se verront toujours pas au zoom ... et la compilation se fait sous LaTeX comme sous PDFLaTeX (création d'un fichier intermédiaire supplémentaire mais génération bien plus rapide).





Bref, c'était surtout une volonté personnelle d'explorer de nouveaux domaines sous  $\text{\LaTeX}$  mais il faut aussi savoir utiliser des solutions simples parfois.

# Chapitre 14

## Dessiner avec PSTricks

Il n’y a rien de pire que d’apprendre une notion, de l’appréhender, d’expérimenter ... pour se rendre compte qu’une autre est meilleure et qu’il faille tout recommencer depuis le début.

C’est ce qui m’est arrivé avec PSTricks. J’ai appris à dessiner avec ce package, qui requiert de compiler sous LaTeX ou XeLaTeX. Puis, j’ai découvert TikZ, qui fonctionne avec n’importe quel mode de compilation.

Je ne vais pas supprimer mon travail initial. Tu peux le consulter. Mais je te recommande de passer directement au chapitre suivant sur TikZ.

### 14.1 Fonctionnement général

Selon le dessin à réaliser, il faut charger un ou plusieurs packages :

- `pstricks` : la base pour dessiner avec PSTricks,
- `pst-circ` : pour dessiner des circuits électriques,
- `pst-node` : pour dessiner des diagrammes,
- `pst-eucl` : pour dessiner des figures géométriques,
- `pstricks-add` : pour ajouter de nouvelles commandes, comme la rotation d’objets par exemple.

Ensuite, pour indiquer à L<sup>A</sup>T<sub>E</sub>X que nous souhaitons dessiner une image sous PSTricks, il faut utiliser l’environnement `pspicture`, suivi de la taille maximale de l’image au format «  $(x_{max}, y_{max})$  ».

Une option supplémentaire, `[showgrid = true]`, est très utile pour visualiser le résultat avec un quadrillage en arrière-plan. Ce dernier permet de



corriger des points mal placés ou de faciliter les décalages à faire.

Bon, allons faire quelques essais pour mieux saisir le principe de fonctionnement.

## 14.2 Dessiner des circuits électriques

Le principe de fonctionnement est très simple. Imagine que tu dessines ton circuit électrique sur une feuille de papier. Dans le coin inférieur gauche, tu places un repère et son origine puis tu considères qu'un composant, un fil, etc. revient à se déplacer d'une unité.

Honnêtement, si tu es arrivé jusqu'à cette partie du guide, tu devrais pouvoir aller jeter un coup d'œil à l'aide sans problème, surtout pour avoir accès à toutes les options disponibles. Je n'ai ni l'envie ni le besoin de le faire ici : quelqu'un l'a parfaitement bien fait à ma place !

Mais comme je ne suis pas un monstre, voici deux petits exemples pour te mettre en bouche :

### Un cas minimaliste

```
\documentclass[a4paper, 12pt]{report}

% LaTeX
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb, upgreek}

\usepackage[dvipsnames]{xcolor} % Pour les couleurs si besoin
\usepackage{pst-circ} % Pour les circuits électriques

\begin{document}

\everymath{\displaystyle}

\begin{pspicture}[showgrid = true](5,2)
```



```

% showgrid affiche le quadrillage
% Permet de se repérer au début et en cas d'erreur
% A mettre sur false lors de la génération du résultat final

% Composants
\resistor(1,1)(2,1){R$}
% Les coordonnées à renseigner sont celles des extrémités du
  composant
\coil[dipolestyle = curved](3,1)(4,1){L$}

% Fils
\wire[intensitylabel = I$, intensitycolor = red,
      intensitylabelcolor = red](0,1)(1,1) % Ou mettre le texte
      en rouge avec \textcolor
\wire(2,1)(3,1)
\wire(4,1)(5,1)

% Annotations
\tension[labeloffset = -0.5](0.5,0.5)(2.5,0.5){V$}
% Si coordonnées non entières, utiliser un point
\end{pspicture}

\end{document}

```

### Un cas plus complet

```

\documentclass[a4paper, 12pt]{report}

% LaTeX
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonts, amssymb, upgreek}

\usepackage[dvipsnames]{xcolor} % Pour les couleurs si besoin
\usepackage{pst-circ} % Pour les circuits électriques

```



```

\begin{document}

\everymath{\displaystyle}

\begin{pspicture}[showgrid = true](8,3)
% Composants
\resistor(2,1)(2,2){R}
\coil[dipolestyle = curved](4,1)(4,2){L} % Un affichage
    possible pour une bobine
\coil[dipolestyle = elektor](6,3)(7,3){1} % Un autre format
    d'affichage
\resistor(8,1)(8,2){\cfrac{r}{g}}

% Fils
\wire[intensitylabel = I$, intensitylabeloffset = 0.5](0,3)
    (2,3)
\wire(2,3)(4,3)
\wire[intensitylabel = I'$(4,3)(6,3)
\wire(7,3)(8,3)
\wire(0,0)(8,0)
\wire(2,0)(2,1)
\wire(2,2)(2,3)
\wire(4,0)(4,1)
\wire(4,2)(4,3)
\wire(8,0)(8,1)
\wire(8,2)(8,3)

% Annotations
\tension(0,0)(0,3){V}
\end{pspicture}

\end{document}

```

### Conseil personnel

La génération sous XeLaTeX peut se révéler assez longue, surtout si tu cumules de nombreux circuits.

Après des essais, le temps d'attente est négligeable avec une compilation sous LaTeX puis Dvi -> PS puis PS -> PDF.



Tu peux donc éventuellement rédiger tout ton rapport avec ce mode de compilation. En revanche, ce dernier ne tolère pas les fichiers `.png` ou `.jpg` pour les images. Il faut donc les convertir en fichier `.eps`, grâce au logiciel GIMP par exemple.

### *Nota Bene*

Tu pourras remarquer que le guide de `pst-circ` utilise une commande `pnode` pour définir les nœuds et leur donner une lettre.

Sache que ce n'est pas absolument pas nécessaire (la preuve avec mes exemples) et, personnellement, je ne te recommande pas de le faire. Je trouve que c'est plus beaucoup plus long s'il faut déplacer des points.

Bon, si tu viens de te rendre compte que  $\text{\LaTeX}$  est extrêmement puissant pour dessiner des circuits d'aussi bonne qualité, sache que ce n'est pas fini. Allons dessiner tout court.

## 14.3 Dessiner tout court

Pour dessiner avec PSTricks, le principe est extrêmement similaire : tu définis des traits ou des formes à partir de coordonnées et  $\text{\LaTeX}$  trace le tout. C'est parti avec un exemple :

### Deux dessins

```
\documentclass[a4paper, 12pt]{report}

% LaTeX
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{amsmath, amsfonTS, amssymb, upgreek}

\usepackage{pstricks}
% Pas besoin du package xcolor ici
```



```

% pstricks l'importe automatiquement

\begin{document}

% Coefficient de transmission thermique d'une paroi
\begin{pspicture}(7,4)
\psline[linecolor = green](1.5,4)(1.5,0) % Pour tracer une
  ligne
\psline(2.5,4)(2.5,0)
\psline(3.5,4)(3.5,0)
\psline(4.5,4)(4.5,0)
\psline[linecolor = green](5.5,4)(5.5,0)
\psline[linecolor = red]{->}(0,2)(7,2)

\psframe[fillstyle = hlines](1.5,0)(2.5,4) % Pour tracer un
  rectangle
\psframe[fillstyle = vlines](3.5,0)(4.5,4)
\psframe[fillstyle = crosshatch](4.5,0)(5.5,4)

\rput(2,-0.25){1} % Pour placer une information
\rput(3,-0.25){2}
\rput(4,-0.25){3}
\rput(5,-0.25){4}
\rput(1.5,4.25){\textcolor{green}{ $T_{S_a}$ }}
\rput(5.5,4.25){\textcolor{green}{ $T_{S_b}$ }}
\rput(7,1.75){\textcolor{red}{ $\Phi$ }}
\rput(0,3){Ambiance a}
\rput(7,3){Ambiance b}
\rput(0,2.5){ $T_a$ }
\rput(7,2.5){ $T_b$ }
\end{pspicture}

\vspace{2\baselineskip}

% Création d'un domaine hachuré
\begin{pspicture}(7,7)
% Repère
\psline{->}(0,1)(7,1)
\psline{->}(1,0)(1,7)

```



```

\uput[d](7,1){\$t_A\$} % Plus pratique pour placer une
    indication décalée
% d = down ; u = up ; l = left ; r = right
\uput[ul](1,7){\$t_B\$} % Combinaison de position possible DANS
    CETTE ORDRE (lu ne fonctionne pas)

\pscircle[fillcolor = black, fillstyle = solid](6,1){0.1} %
    Pour tracer un cercle + le remplir
\uput[d](6,1){30}

% Carré et Delta_t (domaine hachuré)
\psline(6,1)(6,6)(1,6)
\pspolygon[linecolor = red, hatchcolor = red, fillstyle =
    hlines](3,1)(6,4)(6,6)(4,6)(1,3)(1,1)(3,1)
\rput(6.4,6.4){\textcolor{red}{\$Delta_t\$}}

\pscircle[linecolor = red, fillcolor = red, fillstyle = solid
    ](3,1){0.1}
\rput(3,0.6){\textcolor{red}{\$t\$}}
\pscircle[linecolor = red, fillcolor = red, fillstyle = solid
    ](1,3){0.1}
\rput(0.6,3){\textcolor{red}{\$t\$}}
\end{pspicture}

\end{document}

```

### Pour plus de commandes

Je ne vois aucun intérêt à faire une liste des commandes et des options possibles. Je t'ai fourni deux exemples pour que tu aies un aperçu du rendu et des possibilités mais à toi d'aller te documenter par la suite.

Je te recommande particulièrement d'aller sur : [http://fr.wikibooks.org/wiki/LaTeX/Dessiner\\_avec\\_LaTeX/Dessiner\\_avec\\_PSTricks](http://fr.wikibooks.org/wiki/LaTeX/Dessiner_avec_LaTeX/Dessiner_avec_PSTricks). C'est assez complet.





## 14.4 Éclair

Il s'agit d'un dessin que j'ai trouvé sur Internet mais dans un format trop petit. Comme j'en avais besoin, j'ai décidé de le dessiner. Depuis, il est même devenu mon logo personnel ! Cependant, la nature ne m'a pas doté de talents artistiques incroyables, que ce soit sur papier ou une souris à la main.

Heureusement, Dieu (alias Donald KNUTH et Leslie LAMPORT) a inventé L<sup>A</sup>T<sub>E</sub>X, tandis que Spencer KIMBALL et Peter MATTIS ont créé GIMP. J'ai donc pris un papier, reproduit le dessin rapidement et placé les points nécessaires.

Avec l'outil mesure de GIMP, j'ai déterminé toutes les distances et donc toutes les coordonnées des points en question.

Avec l'outil pipette, j'ai sélectionné la couleur et suis allé regarder les options pour obtenir le code RGB.

Trois minutes de code plus tard sous L<sup>A</sup>T<sub>E</sub>X, le dessin est prêt. Il y a juste un petit problème d'affichage donc j'ai ajouté quelques packages pour avoir une page à la taille voulu (au revoir le A4).

### Le code en question

```
% Compiler sous le mode LaTeX
\documentclass[12pt]{report}

\usepackage[french]{babel}

\usepackage[paperwidth = 50cm, paperheight = 90cm,
left = 2cm, top = 2cm]{geometry}

\pagestyle{empty}

\usepackage{pstricks}
\definecolor{bleucyan}{RGB}{59,195,235} % Rappel : pstricks
charge xcolor

\begin{document}

\begin{pspicture}(34,77)
\pspolygon*[linecolor = bleucyan] (13,0) (33,27) (19,27) (33,50)
```



```
(16,50) (30,76) (0,34) (9,34) (6,25) (15,39) (9,39) (13,45)
(22,45) (8,22) (24,22) (13,0)
\end{pspicture}

\end{document}
```

## 14.5 Tête

De même, voici une autre “création”, ou du moins copie personnelle. Dessinée de la même manière que décrite précédemment, le travail complémentaire a juste consisté à la mettre aux bonnes dimensions grâce à GIMP puis à l’insérer dans un document L<sup>A</sup>T<sub>E</sub>X en tant qu’image pour impression.

Un peu de papier calque m’a ensuite permis de reproduire le motif là où j’en avais envie ... Simple mais diablement efficace !

### Le code en question - Le retour

```
% Compiler sous le mode LaTeX
\documentclass[12pt]{report}

\usepackage[french]{babel}

\usepackage[paperwidth = 70 cm, paperheight = 60 cm, left = 2
cm,
right = 2 cm, top = 2 cm, bottom = 2 cm]{geometry}

\pagestyle{empty}

\usepackage{pstricks}

\begin{document}

\begin{pspicture}(64,50)
\psset{linewidth = 0.15} % Réglage des paramètres

% Partie supérieure - Début sommet corne gauche
\pscurve(0,45) (1,43) (3,39.5) (6,38)
```



```

\psline(6,38)(15,38)
\pscurve(15,38)(17.5,43)(20,45)(24,46)
\psline(24,46)(36,46)
\pscurve(36,46)(40,45)(42.5,43)(45,37)
\psline(45,37)(53,37)
\pscurve(53,37)(55,37.5)(60,43) % Sommet corne droite
\pscurve(60,43)(60,37)(56,31)(53,30)
\psline(53,30)(47,30)(49,18)(44,24)(40,18)(36,24)(31,19)
    (27,24)(22,18)(18,24)(13,20)(13,29)(10,29)
\pscurve(10,29)(7,30)(0,38)(0,45)

% Partie inférieure - De la gauche vers la droite
\psline(14,8)(14,10)(18,16)(23,11)(27,17)(32,10)(36,17)(40,10)
    (45,17)(48,12)(48,10)
\pscurve(48,10)(43,5)(34,0)
\psline(34,0)(29,0)
\pscurve(29,0)(20,4)(14,8)

% Oeil gauche
\psline(21,36)(21,32)
\pscurve(21,32)(22.5,30)(24,29)
\psline(24,29)(27,29)(21,36)

% Oeil droit
\psline(40,37)(34,29)(36,29)
\pscurve(36,29)(38,30)(39.5,31.5)(40,33)
\psline(40,33)(40,37)
\end{pspicture}

\end{document}

```

## 14.6 Utiliser des coordonnées

Dans une optique d'automatisation des dessins (un système d'amortisseur en mécanique ou un circuit RLC, utilisés de nombreuses fois, par exemple), il est possible de créer une commande.

L'argument principal de cette commande serait alors un point de départ pour le schéma (en bas à gauche, en haut à droite ou ailleurs, au choix). Sous PSTricks, il s'agirait d'un nœud (**node**) et tous les autres sont définis à partir de ce nœud d'origine (décalage des abscisses et des ordonnées).

L'origine sert donc de "point d'ancrage" pour positionner le dessin et le



reste est construit automatiquement.

Pour ce faire, il faut procéder de la manière suivante :

- ❖ en plus de `pstricks`, charger le package `pst-node`,
- ❖ définir tous les nœuds grâce à la commande :

$$\backslash\text{psnodes}(x_1, y_1)\{\text{noeud1}\} \dots (x_N, y_N)\{\text{noeudN}\}$$

En l'occurrence, le nœud 1 est l'origine;  $(x_1, y_1)$  est donc remplacé par  $(\#1)$  (argument de la commande),

- ❖ définir les  $(x_i, y_i)$  en commençant par un ! et selon la méthode NPI (cf. encadré ci-après),
- ❖ récupérer les coordonnées selon l'une des deux manières suivantes :
  - utiliser la commande `\psGetNodeCenter{noeudi} noeudi.Z`, où Z correspond à x ou y (respectivement, récupération de l'abscisse ou de l'ordonnée),
  - OU BIEN, introduire la commande `saveNodeCoors` dans les options de l'environnement `pspicture` et utiliser ensuite la commande `N-noeudi.Z`.

Il est aussi possible de définir des longueurs pour continuer de généraliser la commande, comme nous le verrons dans l'exemple qui va suivre.

### La Notation Polonaise Inverse

Selon Wikipédia, « la notation polonaise inverse (NPI) (en anglais RPN pour Reverse Polish Notation), également connue sous le nom de notation post-fixée, permet d'écrire de façon non ambiguë les formules arithmétiques sans utiliser de parenthèses. »

Concrètement, pour utiliser un exemple, l'opération  $((1+2) \times 4) + 3$  peut être notée en NPI `1 2 + 4 x 3 +`. Il suffit de partir de la gauche, de prendre deux éléments et un opérateur, de faire le calcul et de le remplacer. Pour détailler, nous avons donc ici :

- `1 2 + 4 x 3 +` : prendre `1 2 +` qui devient `1 + 2` soit 3,
- passage à `3 4 x 3 +` : prendre `3 4 x` qui devient `3 × 4` soit 12,
- passage à `12 3 +` qui devient `12 + 3` soit 15.

Dans le cadre de `PSTricks`, le fonctionnement est le même sauf que



les opérateurs suivants sont utilisés : `add`, `sub`, `mul` et `div`, respectivement pour addition, soustraction, multiplication et division.

Avec un exemple commenté, nous obtenons :

### Exemple d'utilisation des coordonnées

```
% Compiler sous le mode LaTeX
\documentclass[a4paper, 12pt]{report}

\usepackage[french]{babel}

\usepackage{amsmath, amsfonts, amssymb}

\usepackage[dvipsnames]{xcolor}
\usepackage{pstricks, pst-node, pstricks-add}
% pst-node pour les noeuds et le calcul de nouvelles coordonnées
% pstricks-add pour la commande \psrotate

\newcommand{\textedbox}[4]{\pnodes(#1){origine}(#2){fin}
\psframe[#3](origine)(fin)
\rput(!N-fin.x N-origine.x add 2 div N-fin.y N-origine.y add 2
div){\parbox{\linewidth}{\centering #4}}}

\newcommand{\amortisseur}[3]{\pnodes(#1,#2){A}(!#1 1 sub #2){B
}(!#1 1 sub #2 1 add){C}(!#1 #2 1 add){D}(!#1 0.5 sub #2 1
add){F}(!#1 0.5 sub #2){E}(!#1 0.5 sub #2 0.5 add){G}(!#1
0.5 add #2 0.5 add){H}(!#1 1 sub #2 0.5 add){I}(!#1 2 sub
#2 0.5 add){J}(!#1 0.5 sub #2){K}\psline(A)(B)(C)(D) \
psline(F)(E) \psline(G)(H) \psline(I)(J) \uput[d](K){#3}}

\begin{document}

\begin{pspicture}[showgrid = true, saveNodeCoors](10,5)
% saveNodeCoors ssi utilisation de N-node_name.x/y
\def\longueur{4 } % Espace OBLIGATOIRE (sinon rien ne s'
affiche)
\def\decalage{0.5 }
% Définition de longueurs
```



```

% Possibilité de les mettre en argument d'une commande

% Une option brute
\pnodes(1,1){origine}(!\psGetNodeCenter{origine} origine.x
\longueur add origine.y \longueur add){fin}
\psframe(origine)(fin)
\psline[linecolor = Orchid]{|<->|}(!N-origine.x N-origine.y
\decalage sub)(!N-fin.x N-fin.y \longueur \decalage add sub)

% Une commande créée avec l'option saveNodeCoors
\rput(4,0){\psrotate(2.5,2.5){90}{\textedbox{0,2}{5,3}{
  linecolor = Red, framearc = 0.5, linestyle = dashed,
  fillstyle = hlines, hatchcolor = gray}{\textcolor{Cyan}{
  Texte}}}}

% Une autre possibilité
\rput(9,2.5){\psframebox[linecolor = Orange, framesep = 13pt
] {\Large Test}}
\end{pspicture}

\vspace{2\baselineskip}

Un cas plus concret avec un amortisseur, moins élégante mais
qui fonctionne :

\begin{pspicture}[showgrid = true](3,2)
\amortisseur{2}{0.5}{\mu}
\end{pspicture}

\end{document}

```

Comme tu peux le constater, la définition des nœuds avec cette méthode est, certes, laborieuse mais peut se révéler très pratique avec la possibilité de créer des commandes : au lieu d'avoir une entrée pour l'abscisse de l'origine et une autre pour son ordonnée, tout passe avec un argument et PSTricks fait le reste.

Autrement, dans la définition des nœuds, avec cette notation, il ne faut **pas oublier** le ! et il est important de noter que la séparation des abscisses et des ordonnées se fait SANS virgule<sup>1</sup>.

1. Pourquoi ? Je n'en sais rien, ça marche comme ça et c'est très bien. Mais il doit bien



## 14.7 Des boîtes pour le texte

Peut-être l'as-tu remarqué dans mon précédent exemple mais il est possible de créer des boîtes avec le texte centré et tout et tout.

Ma commande, définie dans l'exemple précédent, serait "parfaite" (de mon point de vue, après, tout est relatif) s'il était possible d'extraire la longueur de la boîte pour l'intégrer comme argument de la `parbox` ... Sans succès pour l'instant.

Mais il semblerait qu'elle fonctionne grâce à un petit `\linewidth`. Tant mieux.

Sinon, il existe d'autres possibilités sous `PSTricks` comme la commande `PSTextFrame`. Une piste à explorer ...

## 14.8 Réaliser des intersections

Tu as envie de tracer un contour qui correspond à l'intersection de deux cercles mais tu ne sais pas comment faire ... Pas de panique, il existe une solution. Je vais présenter celle disponible sous `PSTricks`, même s'il en existe une aussi sous `TikZ` (comme elles portent le même nom, la documentation est facile à trouver).

Il s'agit d'utiliser le "`clip`". Le fonctionnement est très simple : tu définis la zone d'intersection puis tu places un objet assez grand (comme un rectangle) et paf! Tu obtiens des Chocapics ... bon ok, quand même pas mais le résultat escompté est là et c'est déjà ça.

### Réaliser des intersections

```
% Compiler sous le mode LaTeX
\documentclass[a4paper, 12pt]{report}

\usepackage[french]{babel}

\usepackage[dvipsnames]{xcolor}
\usepackage{pstricks}

\begin{document}
```

y avoir une raison ...



```

\begin{pspicture}[showgrid = true](5,5)
\psclip{
  \pscircle[linestyle = none](1,2){2}
  \pscircle[linestyle = none](4,2){2}
}
  % Chemin de coupure
  % linestyle = none pour ne pas le dessiner

\psframe*[linecolor = Cyan](0,0)(4,4)
% Remplir l'intérieur du chemin
\endpsclip

% Affichage du contour du chemin de coupure
\pscircle(1,2){2}
\pscircle(4,2){2}
\end{pspicture}

\vspace{2\baselineskip}

\begin{pspicture}[showgrid = true](5,5)
\psclip{
  \rput{-30}(0,2){\psframe[linestyle = none](0,0)(3,2)}
  % psrotate inopérant --> travailler avec rput (partir
  de (0,0), rotation puis décalage)
  \pscircle[linestyle = none](4,2){2}
}
% Chemin de coupure
\endpsclip

\psframe*[linecolor = Cyan](0,0)(4,4)
\end{pspicture}

\end{document}

```

## 14.9 Extraction du contour d'une image

Il existe une image, une icône bien spécifique que tu as envie de réexploiter mais elle est trop petite (zoom  $\Rightarrow$  pixels immondes) ou bien tu tiens à la créer sous L<sup>A</sup>T<sub>E</sub>X pour obtenir une image vectorielle ...

Même si le résultat n'est pas encore parfait, c'est possible, plus parti-





culièrement sur des images monochromes. Le plus dur et le seul point qui nous importe est l'obtention du contour de l'image. Dès que nous avons les coordonnées des points, `psline` suffit, quitte à ajouter des options pour le remplissage.

Il doit être possible de trouver une commande similaire sous TikZ, qui permet aussi de lisser la courbe mais je n'ai rien trouvé de très pratique jusqu'à présent.

Par contre, pour réussir à obtenir les coordonnées du contour en question, il faut bidouiller de la manière suivante :

- ❖ vectorialiser l'image sous Inkscape, la lisser si nécessaire (enlever les bosses superflues, ...),
- ❖ enregistrer le résultat au format `.tex`,
- ❖ ouvrir le code obtenu, vérifier les dimensions utilisées puis épurer le code, c'est-à-dire ne garder que les coordonnées et enlever les commandes s'il y en a,
- ❖ simplifier les coordonnées (beaucoup de décimales pas forcément utiles, surtout à la vue des dimensions utilisées) → possibilité de laisser le traitement à un programme Python,
- ❖ insérer les coordonnées obtenues dans le code L<sup>A</sup>T<sub>E</sub>X final de ton image,
- ❖ bien reporter les dimensions ou les ajuster si besoin :

`\psset{xunit = 0.5pt, yunit = 0.5pt}` par exemple

- ❖ dans le cas où TikZ est utilisé, grâce à l'outil « Remplacer » de Texmaker, remplacer les `)` des coordonnées en `--` (règle sous TikZ pour faire les tracés).

Normalement, le résultat n'est pas trop moche. Tu peux le lisser sous PSTricks sans effort en utilisant la commande `pscurve` au lieu de `psline`. À toi de jouer<sup>2</sup> désormais!

### Simplifier les coordonnées - Le code Python

```
1 def reduction(chaine, nombre) :
2     """Réduire la valeur des décimales à "nombre" d'une
   liste de coordonnées (x,y) (variable chaine)"""
```

2. Yu-Gi-Oh oh oh ...



```
3     resultat = ""
4     i = 0
5     while i <= len(chaine) - 1 :
6         # Coordonnée x
7         while chaine[i] != "." :
8             # On implémente et on cherche le point (séparateur
des décimales)
9                 resultat = resultat + chaine[i]
10                i = i + 1
11            for j in range(0, nombre + 1) :
12                # On implémente la quantité de décimales voulues (
nombre)
13                    resultat = resultat + chaine[i + j]
14                i = i + nombre + 1
15            while chaine[i] != "," :
16                # On a implémenté le nombre souhaité de décimales
17                # --> aller à l'autre coordonnées
18                    i = i + 1
19
20            # Coordonnée y
21            while chaine[i] != "." : # Idem
22                resultat = resultat + chaine[i]
23                i = i + 1
24            for j in range(0, nombre + 1) : # Idem
25                resultat = resultat + chaine[i + j]
26                i = i + nombre + 1
27            while chaine[i] != ")" : # Idem
28                i = i + 1
29            resultat = resultat + chaine[i]
30            i = i + 1
31        print(resultat)
32
33    chaine = "(13.10458,13.112)(13.10458,13.112)
(13.10458,13.112)"
34    nombre = 2
35    reduction(chaine, nombre)
```

# Chapitre 15

## Dessiner avec TikZ

Le seul moyen de pouvoir profiter de toutes les fonctionnalités  $\LaTeX$  précédemment décrites nécessite de compiler, *a minima*, sous PDF $\LaTeX$  (ou Xe $\LaTeX$  s'il y a un changement de police). Dans ce cas, pour dessiner, il faut utiliser TikZ.

Si, comme moi, tu étais un habitué de PSTricks, il peut sembler déroutant de passer à TikZ mais, avec la pratique, il devient facile de réaliser simplement quelques figures. Mais ce n'est pas tout. TikZ est aussi extrêmement puissant, avec énormément de possibilités, comme tu vas pouvoir le découvrir.

Et si jamais tu t'intéresses à la documentation officielle<sup>1</sup>, sache qu'il faut mieux aller d'abord regarder le sommaire ou l'index. Avec plus de 1000 pages d'aide et de code, elle est plutôt bien fournie !

### 15.1 Démarrer sous TikZ

Tu vas difficilement pouvoir utiliser TikZ si tu ne charges pas le package associé : `tikz`. Si, comme nous le verrons plus tard, tu dois charger des fonctionnalités supplémentaires de TikZ, il faut utiliser, **juste après** avoir chargé le package, la commande `\usetikzlibrary{nom}`.

Quant au dessin en lui-même, tout comme pour PSTricks, il faut charger un environnement. Ici, il se nomme `tikzpicture` et n'a pas besoin d'options supplémentaires comme la taille du cadre. Bien au contraire, TikZ produit toujours le résultat le plus compact possible, comme nous le verrons dans un exemple juste après.

---

1. Disponible sur le site du CTAN par exemple, directement sur : <http://www.ctan.org/pkg/pgf>



**Il existe une règle capitale sous TikZ :** chaque qui utilise une commande propre à TikZ se termine par un “;”. Toujours. C’est le seul point important à retenir. Tout le reste va finir par rentrer avec un peu de pratique.

Ensuite, s’il est plus courant de travailler avec des coordonnées cartésiennes  $(x, y)$ , sache aussi que les coordonnées polaire  $(\theta : R)$  sont disponibles, écrites dans le même format que précédemment.

Ensuite, pour commencer en douceur, le tracé d’un trait sous TikZ se fait de la manière suivante :

```
\draw (x0,y0) -- (x1,y1);
```

La commande `\draw` permet de garantir le tracé tandis que les points à relier par un trait sont donc séparés par “--”. Il existe des fonctions propres à TikZ pour tracer un rectangle – `(x0,y0) rectangle (x1,y1)` – ou un cercle – `(x,y) circle (R)`. Il faut continuer d’utiliser la commande `\draw` au préalable.

Il est aussi possible d’augmenter l’épaisseur du trait ou de changer sa couleur grâce à des options à introduire entre crochets “[ ]”, de la manière suivante :

```
\draw[options] ...;
```

Je ne vais pas m’amuser à lister toutes les options possibles et envisageables. Les plus basiques sont présentées ci-après. Les autres sont à chercher en fonction des besoins.

Bien, voici un premier exemple pour avoir un aperçu concret des bases :

#### Démarrer sous TikZ

```
\documentclass[a4paper, 12pt]{report}

\usepackage{tikz}

\begin{document}

\begin{tikzpicture}
\draw (0,0) -- (1,1); % Trait entre (0,0) et (1,1)
\end{tikzpicture}
```



```

\hfill
\begin{tikzpicture}
\draw (2,2) -- (3,3); % Trait entre (2,2) et (3,3) ... sur le
    code !
% Dans le rendu final, aucune différence avec le code précé
    dent
\end{tikzpicture}
\hfill
\begin{tikzpicture}
\draw (0,0) rectangle (1,1);
\end{tikzpicture}
\hfill
\begin{tikzpicture}
\draw (0.5,0.5) circle (0.5);
\end{tikzpicture}

\vspace{\baselineskip}

% Augmenter l'espace blanc autour de l'image (nécessaire de
    temps en temps)
% --> Créer une forme "incolore" (blanche par défaut)
    suffisamment grande
\begin{tikzpicture}
\draw[white] (0,0) rectangle (13,2);
%\draw[red] (0,0) rectangle (13,2);

\draw (6,0) -- (7,1);
\end{tikzpicture}

\vspace{\baselineskip}

% Un aperçu des options de base
\begin{tikzpicture}
% Epaisseur du trait : line width = "longueur"
\draw[line width = 1.3mm] (0,0) -- (1,1);
\draw[line width = 5pt] (2,1) -- (3,0);

% Le "pt" est l'unité par défaut dans les options
\draw[line width = 8] (4,0) -- (5,1);
% Mais les coordonnées sont par défaut en "cm"
\end{tikzpicture}

```



```

%\hfill
\begin{tikzpicture}
% Epaisseur du trait : tailles prédéfinies
\draw[thin] (0,0) -- (1,1);
\draw[thick, dashed] (2,1) -- (3,0); % Ligne en tirets
\draw[ultra thick, dotted] (4,0) -- (5,1); % Ligne en pointill
és
\end{tikzpicture}
%\hfill
\begin{tikzpicture}
% Changement de couleur d'un trait
\draw[blue] (0,0) -- (1,1); % Plus concis et écriture
implicite
\draw[color = orange] (2,1) -- (3,0); % Le nom de l'option
\end{tikzpicture}

\vspace{\baselineskip}

\begin{tikzpicture}
% Changement de couleur d'un contour fermé (contour)
\draw[red] (0,0) rectangle (1,1);
\draw[color = green] (2,1) rectangle (3,0);
\draw[draw = purple] (4,0) rectangle (5,1); % Option étrange
mais va prendre son sens après
\end{tikzpicture}
\hfill
\begin{tikzpicture}
% Changement de couleur d'un contour fermé (remplissage)
\draw[fill = red] (0,0) rectangle (1,1); % draw de base (donc
contour noir) avec remplissage rouge
\fill[color = green] (2,1) rectangle (3,0); % Remplissage pur
(sans contour)
\fill[fill = purple] (4,0) rectangle (5,1); % Idem
\end{tikzpicture}

\vspace{\baselineskip}

\begin{tikzpicture}
% Changement de couleur d'un contour fermé (contour ET
remplissage)
\draw[red, fill = blue] (0.5,0.5) circle (0.5);

```



```

\draw[draw = green, fill = orange] (2.5,0.5) circle (0.5);
\filldraw (4.5,0.5) circle (0.5); % Nouvelle commande :
    contour et remplissage
\filldraw[brown] (6.5,0.5) circle (0.5);
\filldraw[pink, draw = gray] (8.5,0.5) circle (0.5); % etc.
\end{tikzpicture}

\end{document}

```

Tu as tout compris ? Il existe plein d'options extrêmement pratiques mais la couleur et l'épaisseur du trait sont les plus couramment utilisées au début. Il existe aussi des épaisseurs prédéfinies, qui fonctionnent très bien et évitent de perdre du temps à trouver la "bonne" épaisseur :

- ❖ `ultra thin` : 0.1pt,
- ❖ `very thin` : 0.2pt,
- ❖ `thin` : 0.4pt (défaut),
- ❖ `semithick` : 0.6pt,
- ❖ `thick` : 0.8pt,
- ❖ `very thick` : 1.2pt,
- ❖ `ultra thick` : 1.6pt.

Essayons maintenant de tracer des figures un peu plus complexes désormais, avec des coordonnées polaires pour changer et les manipuler un peu.

## 15.2 Un polygone régulier

Je pense que tu dois avoir déjà entendu parler d'un polygone régulier. Pour la faire simple et éviter de faire mon pédant trop longtemps, il s'agit d'une figure géométrique fermée, à  $N$  côtés de même longueur.

Une façon très simple d'en créer consiste à passer par des coordonnées polaires. En effet, les sommets  $S_i$  d'un polygone régulier sont tous placés sur un cercle de centre  $O$  quelconque, de rayon  $R$  et la droite  $(OS_i)$  forme un angle de  $\frac{i \times 360}{N}$  avec l'axe des abscisses.

Enfin, pour revenir sur l'utilisation de TikZ en elle-même, il faut savoir que toute figure définie par des traits et dont le point d'arrivée coïncide avec le point de départ (figure fermée) doit se conclure de la manière suivante : `-- cycle;`

Cette commande permet de proprement fermer la figure. Je te laisse aller faire des recherches ou des essais pour voir la différence avec une fermeture



manuelle. Allons plutôt dessiner un polygone régulier, comme un triangle équilatéral pour commencer simplement :

### Un triangle équilatéral

```
\documentclass[a4paper, 12pt]{report}

\usepackage{tikz}

\begin{document}

% Triangle équilatéral, inscrit dans un cercle de rayon R
% Coordonnées polaires ==> centre (0,0)
\begin{center}
\begin{tikzpicture}
% Rayon R choisi arbitrairement à 3cm
\draw (90:3) -- (210:3) -- (330:3) -- cycle;
\draw[green] (60:3) -- (180:3) -- (300:3) -- cycle; % Une
    autre possibilité
\draw[red] circle (3); % Pas de centre ==> (0,0) par défaut
\end{tikzpicture}
\end{center}

\end{document}
```

Avouons que c'est plus pratique que de devoir placer 2 points et calculer la position du dernier, surtout si les calculs ne donnent pas une valeur exacte. Ici, notre triangle est bel et bien équilatéral. Il est aussi possible de passer par des points définis au préalable :

### Définir des points

```
\documentclass[a4paper, 12pt]{report}

\usepackage{tikz}

\begin{document}
```





```
% Le principe :
% \coordinate (nom_point) at (x,y)/(theta:R);
\begin{center}
\begin{tikzpicture}
\coordinate (A) at (90:3);
\coordinate (B) at (210:3);
\coordinate (C) at (330:3);

\draw (A) -- (B) -- (C) -- cycle;
\end{tikzpicture}
\end{center}

\end{document}
```

### 15.3 Automatiser les dessins

Bon, tracer un triangle équilatéral, c'est bien. Tracer un hexagone, avec un copier-coller et un peu de patience, c'est faisable. Un tridécagone (polygone régulier à 13 côtés) ... bon, rien d'impossible mais le copier-coller et les modifications ne constituent clairement pas une solution optimale.

Fort heureusement, il existe le principe des coordonnées absolues et relatives. Pour faire simple, tracer un dessin grâce à une série de coordonnées absolues revient à connaître les positions de toutes les coordonnées par rapport à un repère, l'origine (0,0) généralement mais il peut aussi s'agir d'un autre point.

Avec les coordonnées relatives, peu importe la position exacte de tous les points : il suffit juste de connaître la position d'un point par rapport à celui qui le précède !

Sous TikZ, les coordonnées absolues ne requiert aucune option spécifique, hormis la position du point. Les coordonnées relatives sont reconnaissables grâce au “++” et il existe un mix des deux, un peu subtil, qui utilise un “+”.

L'aide officielle peut servir à commencer à digérer mes explications : « You can add a single + sign in front of a coordinate or two of them as in +(1cm,0cm) or ++(0cm,2cm). Such coordinates are interpreted differently. The first form means “1cm upwards from the previous specified position” ; the second means “2cm to the right of the previous specified position, **making this the new specified position.**” »

Bien, je pense qu'un petit exemple ne sera pas de trop pour aborder cette



notion :

### Coordonnées absolues et relatives

```

\documentclass[a4paper, 12pt]{report}

\usepackage{tikz}

\begin{document}

% Sans les + ou ++
\begin{tikzpicture}
\draw[gray, dotted] (0,-1) grid (3,1); % Une trame de fond,
    pour aider

\draw (0,0) node[circle, fill = red, inner sep = 2pt] {} --
    (1,1) -- (2,0) -- (0,-1) node[circle, fill = blue, inner
    sep = 2pt] {}; % Le point de départ est toujours le point
    à partir duquel est appliqué le déplacement
\end{tikzpicture}
\hfill
% Avec le +
\begin{tikzpicture}
\draw[gray, dotted] (0,-1) grid (3,1); % Une trame de fond,
    pour aider

\draw (0,0) node[circle, fill = red, inner sep = 2pt] {} --
    (1,1) --- (2,0) --- (0,-1) node[circle, fill = blue, inner
    sep = 2pt] {}; % Le dernier point sans "+" est toujours
    le point à partir duquel est appliqué le déplacement
\end{tikzpicture}
\hfill
% Avec le ++
\begin{tikzpicture}
\draw[gray, dotted] (0,-1) grid (3,1); % Une trame de fond,
    pour aider

\draw (0,0) node[circle, fill = red, inner sep = 2pt] {} ---+
    (1,1) ---+ (2,0) ---+ (0,-1) node[circle, fill = blue,
    inner sep = 2pt] {}; % Chaque nouveau point est le point
    de départ pour le déplacement d'après

```



```

\end{tikzpicture}

% Je reviendrai sur les "node" par la suite
% Ici, ils permettent un point de repère pour distinguer le dé
  part du tracé de sa fin

\end{document}

```

Et cette méthode s'applique aussi pour les coordonnées polaires. Appliquons alors cette découverte pour nos polygones réguliers. Après tout, il s'agit de prendre le point précédent et de le faire pivoter du bon angle :

### Coordonnées relatives polaires

```

\documentclass[a4paper, 12pt]{report}

\usepackage{tikz}

\begin{document}

% Cas d'un triangle équilatéral
\begin{tikzpicture}
\draw (0,0) -- (2,0) ---++ (120:2) -- cycle;
\end{tikzpicture}
\hfill
% Cas d'un carré
\begin{tikzpicture}
\draw (0,0) -- (2,0) ---++ (90:2) ---++ (180:2) -- cycle;
\end{tikzpicture}
\hfill
% Cas d'un pentagone
\begin{tikzpicture}
\draw (0,0) -- (2,0) ---++ (72:2) ---++ (144:2) ---++ (216:2) --
  cycle;
\end{tikzpicture}

\end{document}

```



### Une question ?

« C'est marrant ton astuce pour tracer le polygone en polaire mais ce n'est toujours pas pratique. Il faut quand même changer à la main les valeurs pour chaque polynôme ... »

En effet ... mais j'allais justement annoncer une magnifique solution automatisée !

Il existe trois outils que j'ai découverts suite à mon passage à TikZ et qui se révèlent très utiles dans ce cas :

- **la définition de variable** : tu peux créer toi-même ta propre variable sous L<sup>A</sup>T<sub>E</sub>X<sup>2</sup>. Appliquée à TikZ, tu peux l'associer en tant que nombre (nombre de côtés d'un polygone régulier par exemple) ou que longueur (rayon du cercle dans lequel le dit polygone est inscrit). Il suffit d'utiliser la commande suivante :

```
\def\nom{valeur}
```

- **le calcul de nouvelles variables** : propre à TikZ, cette possibilité peut parfois servir. En l'occurrence, nous dessinons un polygone régulier inscrit dans un cercle de rayon fixé, sans connaître la valeur d'un côté (même si c'est bien plus simple de considérer la taille d'un cercle pour l'affichage). Du coup, si tu tiens à avoir un polygone avec une taille d'arête bien spécifique, tu peux calculer le rayon nécessaire ! Pour ce faire, il faut utiliser la commande :

```
\pgfmathsetmacro\nom{\calcul}
```

Il est aussi possible d'utiliser des variables déjà définies pour les intégrer dans le calcul. Les possibilités offertes deviennent alors très intéressantes,

- **la boucle for** : oui, comme en informatique, il est possible d'indiquer à L<sup>A</sup>T<sub>E</sub>X, et plus particulièrement à TikZ dans notre cas, des tâches répétitives. La formulation est la suivante :

```
\foreach \nom_var in {1,...,N} {commandes}
```

---

2. Très exactement, il s'agit d'une macro. J'apporterai sûrement un correctif et une explication plus poussée lors de la prochaine mise à jour de ce guide et après quelques recherches.



Naturellement, j'ai mis  $\{1, \dots, N\}$  pour l'exemple mais tu peux mettre n'importe quelle valeur numérique, comme  $\{2, 3, 4\}$ , ou même des lettres !

Voici donc une solution simple qui fonctionne. Il y a sûrement encore moyen de l'améliorer, comme permettre à chaque trait d'avoir une couleur différente (avec `cycle` en fin de ligne sinon c'est moche) mais elle fonctionne déjà plutôt bien :

### Une solution automatisée

```
\documentclass[a4paper, 12pt]{report}

\usepackage{tikz}

\begin{document}

% Un polygone régulier
\begin{tikzpicture}
% Paramétrage
\def\poly{13} % Nombre entier supérieur à 1
% % Limite de calcul LaTeX fixée à 16 383 ...
\pgfmathsetmacro\polyg{\poly - 1}
\def\R{2} % 0.25\linewidth est aussi une distance ...

% Tracé du polygone
\draw[orange] (90:\R) \foreach \i in {1,...,\polyg} {-- (90-\i
  /\poly*360:\R)} -- cycle; % Usage de \polyg pour bien
  fermer avec un "cycle"
\end{tikzpicture}
\hfill
% Un amortisseur
\begin{tikzpicture}
\coordinate (0) at (0,0); % Possibilité de changer le 0,0 en
  argument d'une nouvelle commande ...

\draw (0) ---+ (2,0) ---+ (0,-1) ---+ (2,0) ++ (-2,1) ---+
  (0,1) ---+ (2,0) node[above left] {\Large{\mu}} ++
  (-1,0) ---+ (0,-2) ++ (0,1) ---+ (2,0);
% Utilisation de "+" sans "--" pour déplacer la coordonnée
  relative (on rebrousse chemin dans le tracé) sans tracer
```



```

un trait
\end{tikzpicture}

\end{document}

```

### La gestion des unités

Il peut arriver que tu définisses une variable mais que sa valeur ne donne pas le résultat attendu, en terme de taille. Par exemple, un rayon `\def\R{50}` de 50pt ou 50mm au lieu de 50cm par défaut, un peu grand, surtout sur une feuille A4 ; une épaisseur de trait `\def\sep{13}` de 13mm au lieu de 13pt par défaut.

Seulement, écrire `circle (\R{pt})` ou `line width = \sep mm` ne fonctionne pas car L<sup>A</sup>T<sub>E</sub>X n'arrive pas à combiner une variable avec du texte ...

Heureusement, il existe donc un moyen très simple de résoudre ce problème. Il faut définir une variable unité : `\def\unit{unité}`. Par exemple, nous pouvons écrire `\def\unit{pt}` ou `\def\unit{mm}` s'il y a plusieurs unités et que tu ne veux pas les confondre.

Il faut ensuite écrire, par exemple, `circle (\R\unit)` ou `line width = \sep\unit`, et le tour est joué !

### La limite de calcul sous TikZ

Avec les commandes `\def` et `\pgfmathsetmacro`, il existe une limite de calcul, fixée à 16 383, très exactement  $\frac{2^{30} - 1}{2^{16}}$ . Du coup, si tu veux tracer un polygone de 17 000 côtés, c'est impossible. Et je n'aborde pas l'intérêt d'un tel tracé : autant utiliser un cercle dans ce cas !

Généralement, pour des cas raisonnables, il ne devrait pas y avoir de problème mais il est bon de connaître cette notion.

Dans le cas où une telle erreur apparaît, le compilateur devrait afficher l'erreur « ! Dimension too large. ». **Il peut aussi arriver que cette limite apparaisse alors que les calculs ne dépassent pas la valeur interdite.**

Par exemple, trace un polygone de 50 côtés avec mon code précé-



dent et essaye les deux possibilités suivantes dans la boucle `for` :

- `{-- (90-\i/\poly*360:\R)}` : aucun problème,
- `{-- (90-\i*360/\poly:\R)}` : problème ... alors que, d'un point de vue purement formel, le calcul est le même !

De ce que j'ai compris, il s'agit d'une erreur due à un dépassement de pile (*stack overflow*) sous TikZ. Pour l'éviter, il faut **toujours privilégier les divisions au début du calcul**.

## 15.4 Dessiner des figures mathématiques

Je ne vais pas m'attarder sur cette section, juste donner deux pistes de recherche. Si tu as beaucoup de figures géométriques à dessiner, et surtout des figures mathématiques, avec beaucoup de sommets, des intersections, ..., tu peux :

- utiliser le logiciel gratuit `GeoGebra` et exporter les figures en code TikZ,
- utiliser le package `tkz-euclide`, qui possède une documentation bien fournie et beaucoup de commandes intéressantes.

Bien, maintenant que nous connaissons le fonctionnement de TikZ et l'avons un peu manipulé, voyons maintenant des méthodes élégantes pour gérer la forme soit toutes les options accessibles sous `\draw`.

## 15.5 Gestion des styles

Pour commencer, avant même de définir le principe du style sous TikZ, je tiens à aborder le cas toujours assez délicat de l'importation du package `xcolor`, et encore plus de ses options, dont `dvipsnames` pour ma part.

Il faut déjà savoir qu'il faut toujours charger `xcolor` avant `tikz`. Mais, dans certains cas (utilisation d'autres packages principalement), il peut arriver qu'il y ait un conflit et que l'erreur `Option clash for package xcolor` surgisse.

Dans ce cas, la seule solution trouvée jusqu'à présent remplace le chargement de `xcolor` dans le préambule par la ligne :

```
\PassOptionsToPackage{dvipsnames}{xcolor}
```



Et si jamais cette solution ne fonctionne toujours pas, placer cette ligne avant même le `\documentclass` convient en dernier recours.

Venons-en maintenant aux styles. Imaginons un instant que nous avons plein de traits, de rectangles et de cercles à tracer. Bref, pleins de figures qui requiert d'utiliser beaucoup de `\draw`. Nous voulons derrière que toutes les figures aient le même format (couleur, épaisseur de trait, ...).

Il est alors possible de définir un style global pour un dessin (environnement `tikzpicture`). Au lieu d'écrire `\draw[draw_options]` à chaque fois et de devoir tout changer manuellement, il est possible d'ajouter des options à l'environnement de la manière suivante :

```
\begin{tikzpicture}[options]
```

Au contraire, si nous avons besoin de définir plusieurs styles distincts, c'est possible avec la syntaxe suivante :

```
\tikzstyle{nom_style} = [draw_options]
```

avant d'appeler le style en question dans les options : `\draw[nom_style]`.

Enfin, il est toujours possible de procéder à des changements ponctuels dans les options d'un `\draw` : **placés après un style**, ils prédomineront à coup sûr.

Un petit exemple pour bien comprendre, comme d'habitude :

#### Les styles sous TikZ

```
\documentclass[a4paper, 12pt]{report}

\usepackage[dvipsnames]{xcolor}

%\PassOptionsToPackage{dvipsnames}{xcolor} % Si erreur avec
      xcolor --> Enlever le commentaire sur cette ligne et
      mettre la ligne précédente en commentaire
\usepackage{tikz}

\begin{document}

% Style global
\begin{tikzpicture}[thick, Red, dashed]
\def\R{1.5}
```





```

\draw circle (\R);
\draw (\R,0) ---+ (-2*\R,0);
\draw (0,\R) ---+ (0,-2*\R);
\end{tikzpicture}
\hfill
% Styles locaux
\begin{tikzpicture}[thick, Red, dashed]
\def\R{1.5}
\draw[solid, thin] circle (\R); % solid = trait plein

\draw (\R,0) ---+ (-2*\R,0); % Bien mettre un * pour les
    calculs
% A ne pas confondre avec l'utilisation des longueurs : 0.5\
    linewidth, ...
\draw[Cyan, ultra thick] (0,\R) ---+ (0,-2*\R);
\end{tikzpicture}
\hfill
% Style groupé
\begin{tikzpicture}[thick, Red, dashed]
\def\R{1.5}

\tikzstyle{style} = [Orchid, line width = 4pt, line cap =
    round, dash pattern = on Opt off 2.5\pgflinewidth] % Style
    dotted pas très "dot" --> utilisation de line cap & dash
    pattern

\draw circle (\R);

\draw[style] (\R,0) ---+ (-2*\R,0);
\draw[style, LimeGreen] (0,\R) ---+ (0,-2*\R); % Toujours
    possible de modifier un style prédéfini
% A bien placer après le nom du style : options lues et
    appliquées de gauche à droite
\end{tikzpicture}

\end{document}

```

Bien évidemment, ici, le code est très simple et cette notion n'a vraiment d'intérêt quand tu as beaucoup de `\draw ...` ou que tu te rends compte que tu fais beaucoup de changements dans les options. Il devient alors plus intéressant d'automatiser les options avec des styles.



Maintenant que la mise en forme est bien définie et que nous savons tracer quelques figures élémentaires, pimentons un peu les possibilités : ajoutons du texte.

## 15.6 Insérer du texte

Il n'y a qu'une seule façon d'écrire dans un dessin réalisé sous TikZ : les `node`. Très exactement, les `node` permettent de placer à peu près tout et n'importe quoi à l'endroit souhaité dans le dessin, en particulier du texte.

Un `node` s'appelle par une commande, selon la syntaxe suivante :

```
\node[draw_options] at (x,y) {Texte};
```

avec plein de possibilités pour `draw_options` comme par exemple :

- `draw` : pour afficher le cadre du `node`,
- `circle` : pour avoir un cercle comme cadre au lieu d'un rectangle (défaut). Différents formats sont disponibles et sont explicités dans les exemples ci-après,
- `draw = color` : la couleur de la bordure du cadre,
- `fill = color` : la couleur de remplissage du cadre,
- `text = color` : la couleur du texte,
- `minimum width = taille` : largeur minimale du cadre,
- `text width = taille` : largeur de la boîte (invisible) dans laquelle est placée le texte. Si `text width` est inférieur à `minimum width`, la boîte en question est centrée,
- `minimum height = taille` : hauteur minimale du cadre,
- `align = position`, avec `position` qui peut prendre les valeurs `left`, `center` ou `right` : positionnement du texte dans sa boîte (invisible).

Un exemple très simple d'application peut prendre la forme suivante :

### Utilisation des `node`

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
```



```

\usepackage{lmodern}

\usepackage{tikz}

\begin{document}

% La base sur les node
\begin{tikzpicture}
\draw (0,0) -- (1,0);

\node at (0.5,0.5) {Texte};
\end{tikzpicture}
\hfill
% Quelques options
\begin{tikzpicture}
\node[draw] at (0,0) {some text}; % Affichage de la bordure
    rectangulaire du node

\node[draw, circle, align = left] at (4,0) {some text \\
    spanning three lines \\ with manual line breaks}; % circle
    : format du cadre (rectangle par défaut)

\node[draw = Green, fill = Red, text = White, thick, minimum
    width = 5cm, text width = 4cm, minimum height = 2cm, align
    = center] at (2,-2) {some text spanning three lines with
    automatic line breaks};
\end{tikzpicture}

\end{document}

```

Pour avoir un aperçu des différents formats disponibles, c'est par ici :

### Les différents formats de node

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

```



```

\usepackage{lmodern}

\usepackage{tikz}
\usetikzlibrary{shapes} % Pour certains formats de node

\begin{document}

% Formats : rectangle, circle, ellipse, diamond, circle split,
% forbidden sign, cross out, strike out
% regular polygon, regular polygon sides = 5
% star, star points = 7 + star point ratio = 0.8 (bonus)
\begin{tikzpicture}
\tikzstyle{ann} = [draw = none, fill = none, right]

% Affichage sous forme d'un tableau
\matrix[nodes = {draw, ultra thick, fill = blue!20}, row sep =
  0.3cm, column sep = 0.5cm] {%
  \node[draw = none, fill = none] {Plain node}; &
  \node[rectangle] {Rectangle}; &
  \node[circle] {Circle}; \\
  \node[ellipse] {Ellipse}; &
  \node[circle split] {Circle \nodepart{lower} split}; &
  \node[forbidden sign, text width=4em, text centered] {
Forbidden sign}; \\
  \node[diamond] {Diamond}; &
  \node[cross out] {Cross out}; &
  \node[strike out] {Strike out}; \\
  \node[regular polygon, regular polygon sides = 5] {$n
= 5$}; &
  \node[regular polygon, regular polygon sides = 7] {$n
=7$}; &
  \node[regular polygon, regular polygon sides = 9] {$n
=9$}; &
  \node[ann]{Regular polygon}; \\
  \node[star, star points = 4] {$p = 4$}; &
  \node[star, star points = 7, star point ratio = 0.8]
{$p=7$}; &
  \node[star, star points = 10] {$p = 9$}; &
  \node[ann]{Star}; \\
};

```



```
\end{tikzpicture}

\end{document}
```

Mais tu peux tout faire avec des `node`. Par exemple, tu peux les placer à l'intérieur d'un `\draw` pour ajouter de l'information (texte ou symbole). L'intérêt? Pendant que tu traces ton dessin, tu associes l'information, au lieu de l'ajouter manuellement. C'est très pratique si tu modifies ton dessin ou si les coordonnées sont difficiles à déterminer.

Reprenons un ancien exemple, qui devrait te sembler plus clair désormais :

### Ajouter de l'information avec des `node`

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{tikz}

\begin{document}

\begin{tikzpicture}
\draw[gray, dotted] (0,-1) grid (3,1); % Une trame de fond,
pour aider

% Les node en cascade, c'est le nec plus ultra !
\draw (0,0) node[circle, fill = red, inner sep = 2pt] {} node[
left, inner sep = 5pt] {Début} --- (1,1) --- (2,0) ---
(0,-1) node[circle, fill = blue, inner sep = 2pt] {} node[
right] {Fin}; % Option radius possible pour définir la
taille du point mais inner sep plus efficace

\draw (0,0) -- (1,-1) node[right, align = left, font = \small]
{Nouvelle \ branch}; % Positionnement du texte par
rapport au centre du noeud
```



```

% Possibilités de positionnement : above, below, left et
  right
% Des combinaisons comme "above left" sont possibles (ordre à
  respecter)

% Saut de ligne "\\\" licite dans le node si et seulement si "
  align = ..." précisé
\end{tikzpicture}

\end{document}

```

Bon, je crois avoir à peu près fait le tour en ce qui concerne la base pour les `node`. Voyons une dernière application, plus poussée : la création de graphes et d'organigrammes.

## 15.7 Création d'un organigramme

Moi-même j'expérimente ces nouveautés donc je suis encore loin d'en maîtriser toutes les subtilités. Je risque donc ne pas être aussi exhaustif que je le souhaite. Je vais donc expliquer le principe de base avant de te laisser examiner trois exemples.

Sous TikZ, un `node` est constitué de points d'ancrage, répartis de la manière suivante :

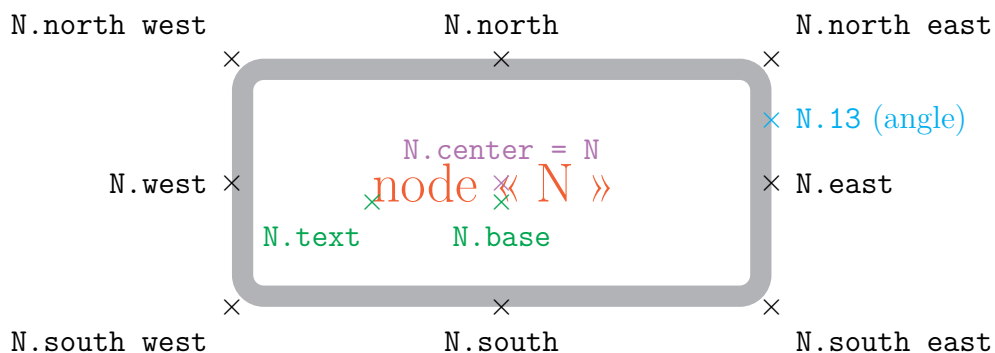


FIGURE 15.1 – Vue d'un `node` et de ses points d'ancrage

Voyons maintenant comment utiliser cette notion pour placer deux boîtes. Nous allons :



- 1) placer un premier `node`, par défaut en  $(0, 0)$ ,
- 2) paramétrer ce `node` pour qu'il ressemble à une boîte,
- 3) placer un second `node` relativement au premier.

L'avantage de cette méthode ? Il faut juste placer un `node` et tout peut se faire relativement à ce dernier. Et grâce aux points d'ancrage, relier deux `node` devient très aisé.

Enfin, **il est possible de créer un style global pour tous les `node`**. Il faut juste respecter la syntaxe suivante après la déclaration de l'environnement `tikzpicture` :

```
[every node/.style = {options}]
```

Le code d'initiation est alors le suivant :

### Initiation aux points d'ancrage

```
\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage{tikz}
\usetikzlibrary{calc}

\begin{document}

\begin{tikzpicture}[every node/.style = {draw = orange, very
    thick, minimum width = 3cm, minimum height = 2cm}]
% Création du node "master" - Options pour avoir une boîte
\node (master) at (0,0) {Boîte 1};

% Création du deuxième node relativement à "master"
\node[anchor = west] at (master.east) {Boîte 2};
% Positionnement de l'ancrage ouest de la boîte 2 sur l'
    ancrage est de "master"

% Troisième node différent relié à "master"
\node[draw = green, fill = gray, minimum width = 2cm] (box) at
    (2,-3) {Boîte 3};
```



```

% Tracé du trait automatisé --> placement très précis entre
  les deux boîtes (milieu vertical)
\coordinate (middle) at ($(master.south)!0.5!(box.north)$); %
  Création du point milieu entre les ancres master.south et
  box.north --> tikzlibrary : calc
\draw (master.south) -- (master.south |- middle) -- (middle |-
  box.north) -- (box.north);

% Tracé du trait manuellement
%\draw (master.south) --++ (0,-0.3) -| (box.north);
% -| : départ horizontal, arrivée verticale du trait
\end{tikzpicture}

\end{document}

```

Et voici maintenant 3 exemples un peu plus détaillés que je te laisse analyser si tu es intéressé :

#### Organigramme 1 : positionnement de boîtes par ancrage

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor}

%\PassOptionsToPackage{dvipsnames}{xcolor} % Si erreur avec
  xcolor --> Enlever le commentaire sur cette ligne et
  mettre la ligne précédente en commentaire
\usepackage{tikz}

\begin{document}

% Boite principale en (0,0)
% Positionnement des en-têtes de chaque sous-boîte

```





```

% Tracé des traits automatisés - Gestion des styles

\begin{tikzpicture}[every node/.style = {align = center, draw
= Black, fill = RoyalPurple!70, line width = 1.5pt, text
width = 3cm, minimum width = 3.5cm, minimum height = 1cm,
text = White}]
\node[text width = 2.5cm, minimum width = 3cm] (master) at
(0,0) {\Large{}Manager};

\tikzstyle{bigbox} = [text width = 3.2cm, minimum width = 3.5
cm, minimum height = 3cm]

\node (boxa) at (-6,-3) {\large{}Team A};
\node[bigbox, anchor = north] at (boxa.south) {Commercial \
linebreak \linebreak Support function};

\node (boxb) at (-2,-3) {\large{}Team B};
\node[bigbox, anchor = north] at (boxb.south) {PHP \linebreak
\linebreak JavaScript};

\node (boxc) at (2,-3) {\large{}Team C};
\node[bigbox, anchor = north] at (boxc.south) {Support \
linebreak \linebreak Supervision};

\node (boxd) at (6,-3) {\large{}Team D};
\node[bigbox, anchor = north] at (boxd.south) {Report \& KPI \
linebreak \linebreak Financial management};

\foreach \point in {a, b, c, d} {\draw[very thick] (master.
south) ---+(0,-1cm) -| (box\point.north);} % Tracé
automatisé : -| = départ horizontal, arrivée verticale
%\draw (master.south)---+(0,-1) ---+ (-2,0) ---+ (0,-1) ++
(0,1) ---+ (-4,0) ---+ (0,-1) ++ (6,1) ---+ (2,0) ---+
(0,-1) ++ (0,1) ---+ (4,0) ---+ (0,-1); % Ancien tracé
manuel moins pratique
\end{tikzpicture}

% Pour un meilleur centrage de l'organigramme : environnement
center et agrandir un peu les marges

\end{document}

```



### Organigramme 2 : utilisation d'un arbre

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor}

%\PassOptionsToPackage{dvipsnames}{xcolor} % Si erreur avec
      xcolor --> Enlever le commentaire sur cette ligne et
      mettre la ligne précédente en commentaire
\usepackage{tikz}
\usetikzlibrary{trees}

\begin{document}

% Utilisation d'un arbre
% Syntaxe : child { node {Texte} }
% Jouer sur l'encapsulation de plusieurs child pour faire des
      ramifications

% Pleins d'options globales disponibles :
% --> gestion de la distance (général ou pour chaque niveau) =
      sibling distance
% --> format du trait entre deux niveaux = edge from parent

\begin{tikzpicture}[level 1/.style = {sibling distance = 17em
      }, level 2/.style = {sibling distance = 8em}, every node/.
      style = {shape = rectangle, rounded corners, draw, align =
      center, top color = white, bottom color = blue!20}, edge
      from parent/.style = {draw, edge from parent path = {(\tikzparentnode.south) -- +(0,-8pt) -| (\tikzchildnode)}}},
      level distance = 50pt] % Pas de gestion automatique de la
      taille et de l'espacement
\node {Prenom Nom \ \ Chef}
      child { node {Prenom Nom \ \ Sous-chef}
        child { node {Prenom Nom \ \ Esclave}}
        child { node {Prenom Nom \ \ Esclave}}}

```



```

    }
    child { node {Prenom Nom \\ Sous-chef}
        child { node {Prenom Nom \\ Mineur}
            child { node {Prenom Nom \\ Stagiaire
                child { node {Prenom Nom \\ Stagiaire
                    child { node {Prenom Nom \\ Stagiaire
                }
            }
        }
    }
};
\end{tikzpicture}

\end{document}

```

### Organigramme 3 : autre utilisation d'un arbre

```

\documentclass[a4paper, 12pt]{report}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{lmodern}

\usepackage[dvipsnames]{xcolor}

%\PassOptionsToPackage{dvipsnames}{xcolor} % Si erreur avec
xcolor --> Enlever le commentaire sur cette ligne et
mettre la ligne précédente en commentaire
\usepackage{tikz}
\usetikzlibrary{trees}

\begin{document}

% L'arbre précédent est plutôt bien mais est plus adapté au
format paysage et s'il y a peu de sous-divisons
% Ou alors amélioration des sous-divisions comme ci-après

```



```

% Création de styles avec un nom dans l'appel de l'
environnement

\begin{tikzpicture}[man/.style = {rectangle, draw, fill = blue
!20}, woman/.style = {rectangle, draw, fill = red!20,
rounded corners = .8ex}, grandchild/.style = {grow = down,
xshift = 1em, anchor = west, edge from parent path = {(\
tikzparentnode.south) |- (\tikzchildnode.west)}}, first/.
style = {level distance = 6ex}, second/.style = {level
distance = 12ex}, third/.style = {level distance = 18ex},
level 1/.style = {sibling distance = 5em}]
% Parents
\coordinate
    child[grow = left] {node[man, anchor = east] {Jim}}
    child[grow=right] {node[woman,anchor = west] {Jane}}
    child[grow = down, level distance = 0ex][edge from
parent fork down]
% Children and grandchildren
    child{node[man] {Alfred}
        child[grandchild, first] {node[man] {Joe}}
        child[grandchild, second] {node[woman] {
Heather}}}
        child[grandchild, third] {node[woman] {Barbara
}}
    }
    child{node[woman] {Berta}
        child[grandchild, first] {node[man] {Howard}}}
    }
    child {node[man] {Charles}}
    child {node[woman] {Doris}
        child[grandchild, first] {node[man] {Nick}}
        child[grandchild, second] {node[woman] {Liz}}}
    };
\end{tikzpicture}

\end{document}

```

## 15.8 Les possibilités offertes par TikZ

TikZ met à dispositions de très nombreuses bibliothèques pour réaliser “tes envies les plus folles” ... enfin, tout ce que tu as besoin de dessiner !



Pour utiliser ces bibliothèques, il suffit d'utiliser la commande suivante dans le préambule, juste après l'appel à `tikz` :

```
\usetikzlibrary{nom_bibliothèque}
```

Ces bibliothèques sont détaillées à la partie V du guide officiel de TikZ, ainsi que sur le site suivant : <http://tex.stackexchange.com/questions/42611/list-of-available-tikz-libraries-with-a-short-introduction>. En voici un aperçu résumé :

- ❖ `angles` : pour faciliter le dessin d'angles,
- ❖ `arrows` : pour de nouvelles pointes de flèches, personnalisables,
- ❖ `automata` : pour dessiner des automates finis (diagrammes d'état) et des machines de Turing,
- ❖ `babel`,
- ❖ `backgrounds` : pour créer des arrière-plans colorés dans l'environnement `tikzpicture`. À essayer : peut-être qu'il y a un intérêt,
- ❖ `calc` : pour faire des calculs, généralement sur les coordonnées,
- ❖ `calendar` : pour créer des calendriers,
- ❖ `chains` : pour créer des chaînes,
- ❖ `circuits` : pour dessiner des circuits électriques. Il existe aussi le package `circuitikz`,
- ❖ `decorations` : pour pouvoir appliquer des transformations à des chemins (`path`) et les décorer.,
- ❖ `er` : pour des dessiner des diagrammes entité-association (`er` = `entity-relationship`),
- ❖ `fadings` : pour estomper les couleurs,
- ❖ `fit` : pour créer un `node` qui doit contenir des coordonnées définies,



- ❖ `folding` : pour créer des patrons ou des objets à plier,
- ❖ `intersections` : pour calculer des intersections de chemins (`path`),
- ❖ `math` : pour définir des fonctions mathématiques et exécuter des opérations mathématiques,
- ❖ `matrix` : options et styles supplémentaires pour la création d'une matrice de `node`,
- ❖ `mindmap` : pour créer des cartes heuristiques. Le cœur est placé au centre, tandis que les éléments qui en découlent sont créés à partir de branches (`children`, comme pour les organigrammes ou les arbres). Personnalisation complète possible et annotations disponibles,
- ❖ `patterns` : pour définir de nouveaux motifs pour du remplissage,
- ❖ `petri` : pour dessiner des réseaux de Petri,
- ❖ `plothandlers`,
- ❖ `plotmarks`,
- ❖ `shadings` et `shadows` : pour réaliser des ombrages,
- ❖ `shapes` : pour de nouveaux formats de `node` (ellipse, diamant, étoile, polygone, ...),
- ❖ `spy` : pour faire des zooms,
- ❖ `trees` : pour dessiner des arbres (de possibilités).

Enfin, il existe aussi d'autres packages pour agrémenter les dessins sous L<sup>A</sup>T<sub>E</sub>X. C'est par exemple le cas de `pgfornament`, qui mérite le détour pour fournir beaucoup d'ornements intéressants.

FIGURE 15.2 – Un premier aperçu du package `pgfornament`

## 15.9 Le fond d'écran $\text{\LaTeX}$

Loin d'avoir tout expliqué sur `TikZ`<sup>3</sup>, les exemples que j'ai élaborés et mis à disposition au sein de ce guide donnent quand même beaucoup d'informations et de possibilités.

Naturellement, je suis loin d'être complètement exhaustif et je me suis efforcé d'aborder un maximum de notions fondamentales. Lors de la prochaine modification de ce guide, il y a des chances que je revienne sur cette partie, tant pour améliorer mes explications, mes exemples et l'affichage du code.

En tout cas, je me suis amusé à réaliser un petit fond d'écran pour mon ordinateur. Naturellement, il prône l'utilisation du  $\text{\LaTeX}$  et joue un peu sur la fibre patriotique.

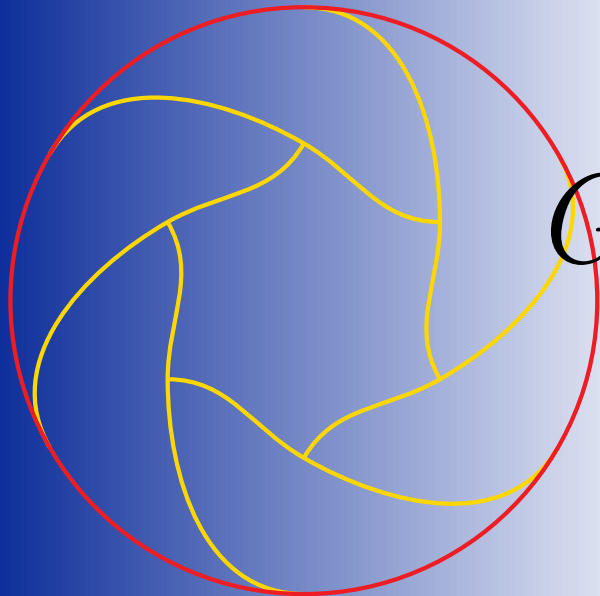
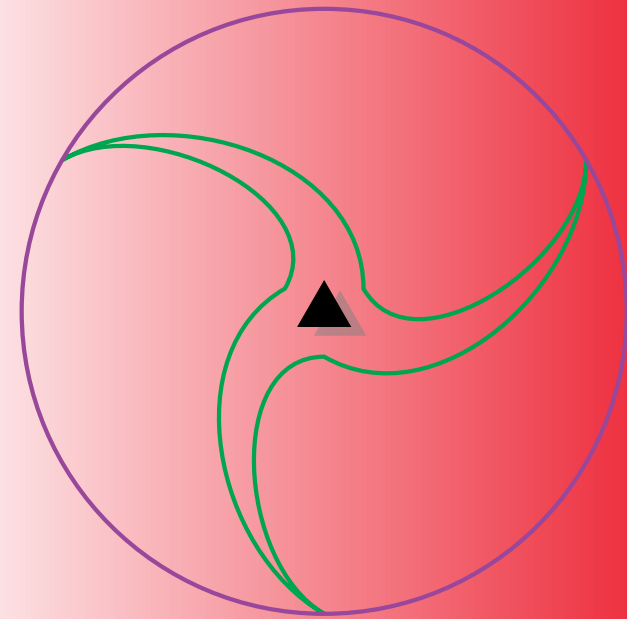
Il est à disposition si après si tu veux réaliser une capture d'écran pour l'utiliser toi aussi. Il peut aussi constituer un bon entraînement si tu veux dessiner sous `TikZ`.

---

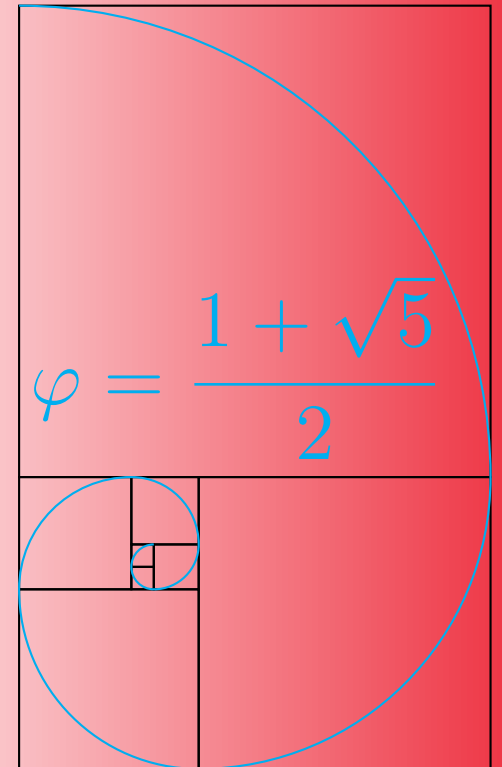
3. Le guide officiel fait plus de 1000 pages donc tu penses bien que je n'ai fait qu'effleurer le champ des possibles.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras euismod fringilla felis, ac mollis nulla pellentesque eget. Vivamus blandit porta tincidunt. Quisque ullamcorper ipsum a dui posuere, congue placerat justo vestibulum. Maecenas eleifend neque posuere gravida dignissim. Quisque quis tortor sed elit rutrum aliquam. Nulla facilisi. Mauris est sapien, viverra vitae purus pretium, mollis posuere neque. Duis rutrum lectus vel nunc tincidunt condimentum. Suspendisse est sem, sodales eu metus id, fringilla vulputate risus. Proin et ex at nunc consectetur tempor sit amet accumsan tellus. Vestibulum quis ultricies orci. Morbi mollis quam neque, eu vulputate libero volutpat sagittis. Phasellus scelerisque mauris id lorem viverra rhoncus. Fusce dictum velit arcu, eget congue mi convallis pulvinar.

LATEX  
LATEX



*Limitless creation*  
*Good-looking reports*





# Chapitre 16

## Faire des présentations avec Beamer

À venir ...