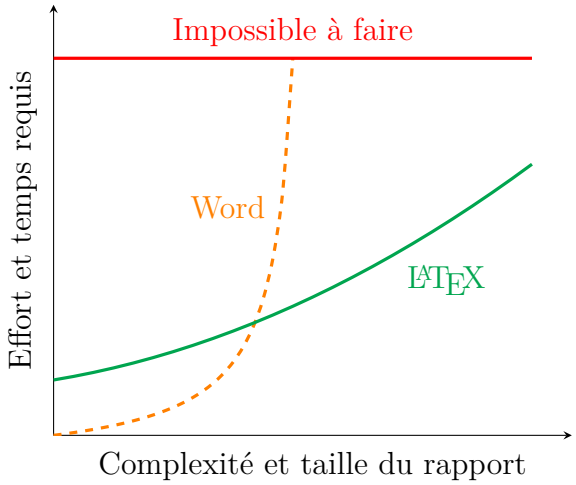
Initiation à LaTeX

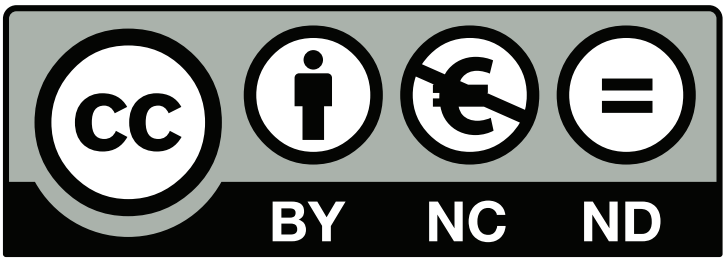
guide-latex-fr

Pour débutants ou jeunes utilisateurs

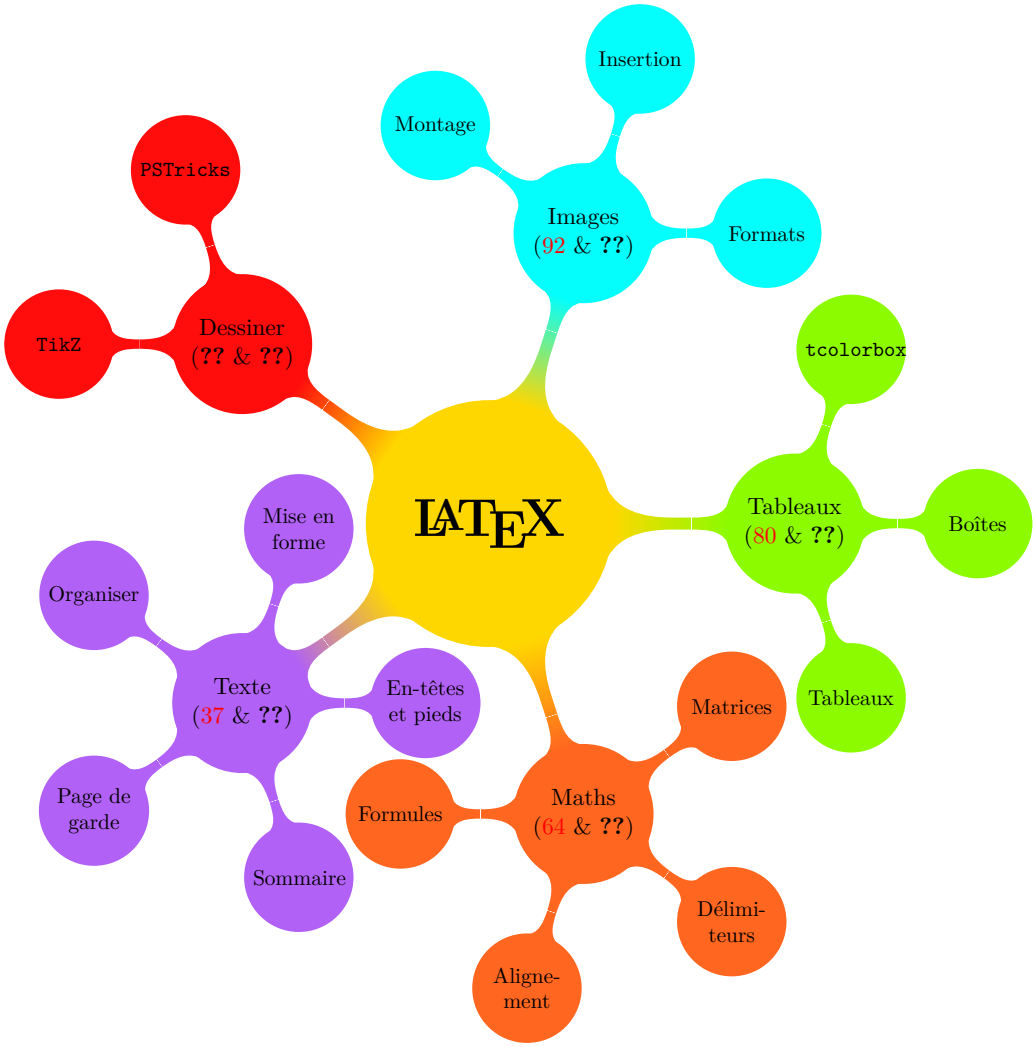


Par Adrien Bouzigues
Indignation 13 ₵215

13 juillet 2016
Version 3.0 à jour au :
25 décembre 2018



Toutes les versions de ce guide sont soumises à cette licence Creative Commons, y compris les plus anciennes qui peuvent circuler et qui n'y font pas explicitement mention.


LaTeX
Images (92 & ??)
Insertion
Montage
Formats
Dessiner (?? & ??)
PSTricks
TikZ
Tableaux (80 & ??)
tcolorbox
Boîtes
Tableaux
Texte (37 & ??)
Mise en forme
Organiser
En-têtes et pieds
Page de garde
Sommaire
Maths (64 & ??)
Matrices
Formules
Délimiteurs
Alignement

Sommaire

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.

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.

D'autre part, mes connaissances en LaTeX restent limitées. Je n'ai pas la science infuse et ce guide est loin d'être exhaustif. **Je propose juste des solutions qui fonctionnent.** N'hésite donc pas à aller te documenter ailleurs si un point ne te semble pas clair ou si tu cherches d'autres informations.

S'ils ne sont pas légion, il existe d'autres guides en français pour apprendre le LaTeX. Pour ma part, je recommande l'excellent *LaTeX... pour le prof de maths !* d'Arnaud Gazagnes[1], très complet et bien expliqué. Je suis aussi tombé plus récemment sur *Rédaction avec LaTeX* de Vincent Goulet[2], très agréable à lire et bien détaillé lui aussi.

Et si jamais tu désires retrouver mon 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`

N'hésite pas à laisser des commentaires ou à signaler des fautes dans le GGForm mis à disposition. Sur ce, bonne lecture !

Adrien Bouzigues
I13 Cl215

1. Disponible sur : `http://math.univ-lyon1.fr/irem/spip.php?article340`.
2. Disponible sur : `https://ctan.org/pkg/formation-latex-ul`.

Lien de mon `Drive` LaTeX... sous la forme d'un code QR !

Ô Capitaine ! Mon Capitaine !
Pendant chaque traversée,
Tu restes à mes côtés
Et soutiens mon avancée.

Première partie

LaTeX : histoire & premier contact

Chapitre 1

Pourquoi (utiliser) LaTeX ?

Comme toute chose, LaTeX possède une histoire qui lui est propre, des avantages mais aussi des inconvénients – rien n'est parfait en ce monde. Toutefois, LaTeX est aussi un langage qui continue d'exister à l'heure actuelle et qui reste une référence dans le milieu scientifique.

C'est pourquoi je te propose un petit interlude culturel avant d'entrer dans le vif du sujet... et peut-être aussi pour finir de te convaincre de son utilité !

1.1 Historique : de TeX à LaTeX

La (petite) histoire

Donald Knuth est un mathématicien et informaticien américain, professeur émérite à l'université de Stanford. Il est l'auteur d'une bible de la programmation intitulée *The Art of Computer Programming* (TAOCP).[1]

Le premier volume paru en 1965, a été publié à l'ancienne avec des caractères en plomb. Quand en 1976 Donald Knuth décide de publier la seconde édition du volume 2 de TAOCP, les caractères en plomb ont été abandonné au profit de la photocomposition.

Donald Knuth trouve alors la qualité d'impression de ces machines, médiocre (notamment pour l'écriture des formules mathématiques) et décide de créer deux logiciels pour pouvoir produire ses publications avec une qualité typographique professionnelle.

1. Cet historique est extrait des « Fiches à Bébert », dont le texte complet est disponible sur : `http://lesfichesabebert.fr/divers/tex.html`.

Le premier, TeX, sert à la composition de documents ; le second, METAFONT, à produire des polices vectorielles. Donald Knuth va mettre plusieurs années avant de sortir en 1983 la version définitive de TeX qui utilise la police Computer Modern qu'il a créé à l'aide de METAFONT.

En effet, Donald Knuth s'était fixé comme but d'arriver à un produit qui devrait être parfait et qui devrait le rester au cours du temps. C'est cette version qui est toujours utilisée et qui fonctionne depuis 30 ans.

Donald Knuth est quand même intervenu sur TeX à plusieurs reprise, notamment en 1989 pour l'adapter aux caractères nécessaires pour la composition de texte avec d'autres langues que l'anglais (version 2.991). La version actuel de TeX est la 3.14159265 (janvier 2014).

L'autre trait de génie de Donald Knuth est de confier TeX à l'American Mathematical Society et d'en faire un logiciel libre.

À partir de là, d'autres informaticiens vont s'emparer de TeXpour l'adapter (sortie de document au format PDF, utilisation de format d'image inconnue en 1983, adaptation à d'autres langues que l'anglais...) et l'enrichir (module permettant la création de formule chimique, de partition musicale, de diagrammes électrique ou physique...).

En 1982, Leslie Lamport, un chercheur en informatique américain, écrit LaTeX (Lamport TeX) un nouveau jeu de macros beaucoup plus simple à utiliser que TeX.

C'est un succès et pratiquement plus personne n'utilise TeX. L'apparition des packages, qui permettent facilement d'augmenter les fonctionnalités, ont rendu LaTeX incontournable (édition d'ouvrages scientifiques ou article de recherches, notamment).

La version actuelle de LaTeX est LaTeX2ε, qui date de 1994. Elle est maintenue par le LaTeX3 Project team qui nous prépare la version 3 de LaTeX depuis 20 ans !

À la fin des années 90, Hàn Thê Thành crée le moteur pdfTeX qui permet de sortir les documents au format PDF, plus convivial que le format d'origine de TeX le DVI.

La dernière version la 1.40.11 date de 2011. pdfTeX n'est plus développé, seules des corrections de bug y sont apportées.

C'est ce moteur que nous allons utiliser par la suite, qui permet de passer du fichier LaTeX au fichier PDF final voulu.

Étymologie et prononciation

Si je remercie encore une fois Bébert pour ce magnifique historique, je me dois désormais d'intervenir sur un point qu'il ne traite pas sur cette page : l'étymologie et la prononciation de LaTeX.

C'est un point extrêmement crucial qui peut te permettre de briller lors de soirées mondaines et d'éviter de passer pour un blaireau lors de conversations avec d'autres utilisateurs de LaTeX.

De ce que j'ai lu un jour quelque part sur Internet, Donald Knuth a nommé son logiciel TeX comme pour « technologie ».

Mais, il s'avère qu'il est aussi féru de grec. Et « technologie », en grec, s'écrit « $\tau\varepsilon\chi\nu o\lambda o\gamma\iota\alpha$ », le χ correspondant au « chi » mais que l'on prononce « khi ».

Et c'est donc pourquoi TeX se prononce « tech » mais s'écrit avec un "X".

Quant à LaTeX, il s'agit juste d'ajouter les premières lettres du nom de son créateur, Leslie Lamport. TeX est donc devenu LaTeX... et se prononce *a priori* de la même façon.

Toutefois, Leslie Lamport indique explicitement dans son livre *LaTeX : A Document Preparation System* qu'il n'encourage aucune prononciation particulière pour LaTeX... mais là encore, si tu ne veux pas passer pour un blaireau, je t'encourage vivement à t'en tenir à la prononciation usuelle, soit « latech » !

Bien, maintenant que ce point a été abordé, venons-en aux avantages à utiliser LaTeX avec, pour commencer, des témoignages !

1.2 Pourquoi utiliser LaTeX ?

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 (scientifical 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 6 pages de commentaires... Je vais donc faire un petit résumé :

→ certains pensent qu'utiliser LaTeX 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 LaTeX est appréciée (clarté du message, mise en page propre, simplicité...) ;

→ beaucoup reconnaissent que LaTeX possède une forte courbe d'apprentissage, surtout au début [2]. Toutefois, ils utilisent aussi LaTeX dans leur quotidien (lettres, CV, rendus...) car ils préfèrent sa facilité d'utilisation par rapport à Word une fois l'apprentissage bien avancé ;

→ quasiment tous considèrent qu'apprendre à utiliser LaTeX 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 :

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

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

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.

Toujours pas convaincu ? Voici alors une ribambelle d'arguments qui devraient, j'espère, finir de te convaincre d'utiliser LaTeX.

1.3 Autres arguments

Utiliser LaTeX au lieu d'un autre logiciel de traitement de texte plus… conventionnel présente un certain nombre d'avantages, dont voici la liste (non exhaustive) :

→ LaTeX est entièrement gratuit et utilisable sur n'importe quel système d'exploitation ;

→ un fichier LaTeX est utilisable par n'importe qui (à condition d'avoir les logiciels adaptés à LaTeX) et sous n'importe quelle version de LaTeX ;

→ LaTeX génère un fichier PDF prêt à l'impression et lisible par n'importe qui ;

→ LaTeX propose une mise en page professionnelle et déjà paramétrée. La gestion de la numérotation des pages, des en-têtes et des pieds de page est relativement simple ;

→ écrire des formules mathématiques devient assez facile (avec un peu de pratique) ;

→ LaTeX gère intégralement les notes de bas de pages, les renvois, le sommaire, les images, les tableaux, les légendes et la numérotation, les références bibliographiques ou la mise en place d'un index ;

→ LaTeX réalise aussi les césures les plus appropriées et prend en compte les ligatures.

Convaincu cette fois ? Pas vraiment ? Tu hésites encore ? Dans ce cas, continuons sur notre lancée et installons LaTeX sur notre ordinateur. Tu ne peux pas savoir avant d'essayer, n'est-ce pas ?

Chapitre 2

Installation de LaTeX

Avant de commencer, je suppose que tu utilises un système d'exploitation `Windows`. Dans le cas contraire, un utilisateur `Linux` devrait savoir se débrouiller pour tout installer.

Si tu es un utilisateur d'`Apple`, je considère déjà ta cause perdue d'avance et tu trouveras des équivalents grâce à Google... enfin, c'est ce que je disais initialement. Désormais, les programmes que je présente par la suite te sont aussi accessibles.

2.1 Installation de `MiKTeX`

`MiKTeX` est une distribution LaTeX. Bon, je dois t'avouer que je ne sais pas moi-même ce qu'est une distribution... Ce qui m'intéresse, c'est d'arriver à faire fonctionner l'outil en question. Je vais donc sortir mon joker Wikipédia pour cette fois :

Définition d'une distribution (informatique)

« On parle souvent de distribution pour désigner un ensemble de logiciels formant un tout cohérent et prêt à installer, incluant des jeux de paquetages, le noyau du système d'exploitation, en particulier le noyau Linux pour les distributions GNU/Linux (comme Debian, Mandriva, Red Hat, Ubuntu, etc.), un système d'installation et des utilitaires de configuration.

Cela désigne aussi un ensemble de paquets et d'outils utiles à la création d'un document au format LaTeX et pour en faciliter l'utili-

sation. Parmi les distributions LaTeX courantes, on trouve MiKTeX, TeXLive, MacTeX2.

Par ailleurs, une base de données distribuées est répartie sur plusieurs nœuds, généralement sur différents serveurs.

»

Wikipédia – Disponible sur :
`http://fr.wikipedia.org/wiki/Distribution#Informatique`

Je ne sais pas si c'est plus clair ainsi... Ce qui est certain, c'est que le seul élément intéressant à retenir est le suivant : `MiKTeX` est l'outil qui te permet de transformer tes futures lignes de code LaTeX en un PDF propre et lisible par tous.

Pour installer `MiKTeX`, il faut procéder de la manière suivante[1] :

1) aller sur : `http://miktex.org/download` et télécharger l'exécutable ;

2) lancer l'exécutable et suivre les instructions d'installation ;

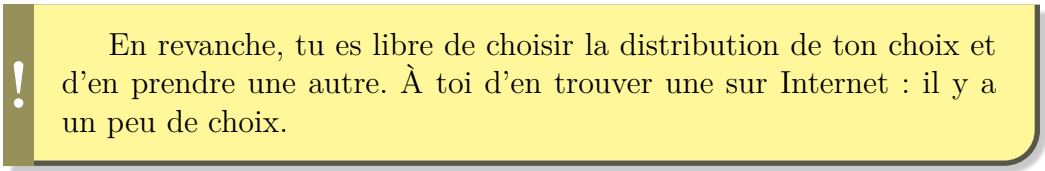
3) **laisser les options par défaut DONT** le "choix de poste" « `Install MiKTeX only for me` ».
   Pour débuter, elles conviennent parfaitement et le choix « `only for me` » permet d'éviter tout problème par la suite.

! ***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.

Elle a aussi l'avantage de proposer un gestionnaire de packages, via `MiKTeX Console` ou l'interface de `MiKTeX`. Nous aurons l'occasion d'y revenir plus tard dans ce guide, une fois que la notion de packages aura été introduite.

1. Si besoin, un descriptif encore plus détaillé et imagé est disponible à l'adresse suivante : `http://miktex.org/howto/install-miktex`.



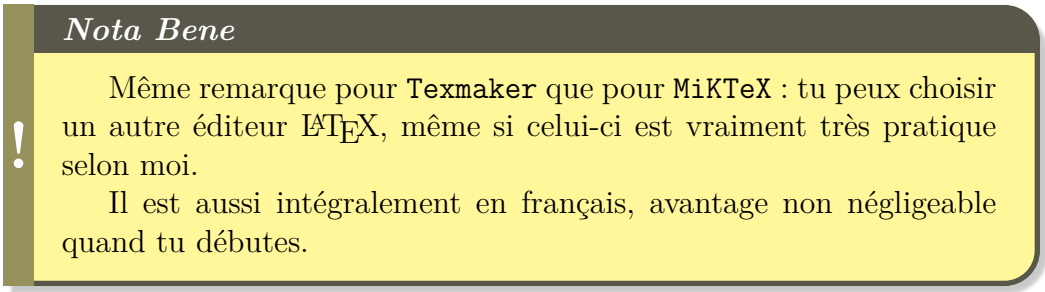
En revanche, tu es libre de choisir la distribution de ton choix et d'en prendre une autre. À toi d'en trouver une sur Internet : il y a un peu de choix.

2.2 Installation de Texmaker

Techniquement, cette étape n'est pas nécessaire car tu pourrais écrire ton fichier LaTeX dans un fichier `.txt` (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` – via des commandes dans le `CMD` – de transformer ton code en PDF grâce à `MiKTeX`.

Avec `Texmaker`, tous ces tracas sont épargnés : tu as à disposition un éditeur de fichiers LaTeX performant, de la coloration syntaxique, 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`. Là encore, il suffit de télécharger l'exécutable, le lancer, suivre les instructions et laisser les options par défaut (comme pour `MiKTeX`).



Nota Bene

Même remarque pour `Texmaker` que pour `MiKTeX` : tu peux choisir un autre éditeur LaTeX, même si celui-ci est vraiment très pratique selon moi.

Il est aussi intégralement en français, avantage non négligeable quand tu débutes.

2.3 Vérification finale

Si tu tiens à t'assurer que tout fonctionne, tu peux d'ores et déjà procéder à une vérification finale comme décrit ci-après.

Si jamais tu rencontres le moindre problème, ne t'attarde pas sur cette partie et poursuis au chapitre suivant, qui détaille l'utilisation des logiciels récemment installés.

1) Ouvrir `Texmaker`.

2) En haut à gauche : `Fichier` puis `Nouveau` (ou Ctrl + N pour les connaisseurs).

3) Recopier le code « Bonjour monde ! », fourni en-dessous, et sauvegarder **dans un dossier** (le nom importe peu).

4) Appuyer sur F6, attendre un peu, puis aller dans le dossier où tu as sauvegardé le fichier : tu devrais y trouver un PDF avec la ligne « Bonjour monde ! » écrite.

Bonjour monde !

```
\documentclass[]{report}


\begin{document}

Bonjour monde !

\end{document}
```

Tout fonctionne donc parfaitement ! Tu peux poursuivre sereinement la suite du guide.

Dans le cas contraire, ne perds pas ton temps et passe directement à la suite. Nous allons rapidement aborder le fonctionnement de `Texmaker`.

Si jamais des problèmes persistent par la suite, je ne peux que te conseiller de tout désinstaller et de bien tout réinstaller comme indiqué précédemment.

Chapitre 3

Compiler avec LaTeX

Pour faire du LaTeX, il faut déjà connaître le point suivant : LaTeX est un langage et un système de composition de documents. Généralement, en informatique, un langage requiert une étape obligatoire : la compilation. Et LaTeX n'échappe pas à cette règle.

3.1 Principe de la compilation

Quand tu vas rédiger un document sous LaTeX, tu vas devoir procéder en 3 temps :

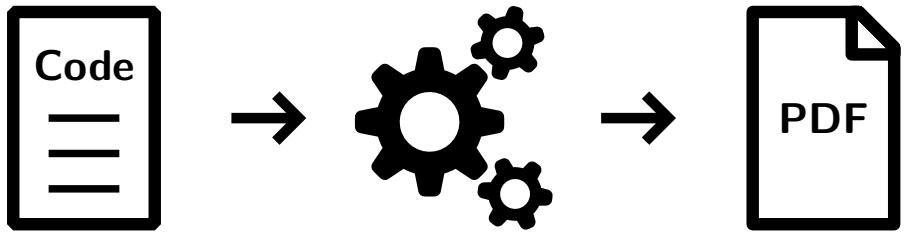


Figure 3.1 – Les 3 étapes pour rédiger un document sous LaTeX

Pour entrer un peu plus dans les détails, tu dois donc :

1) écrire ton document en LaTeX (respect de ses conventions et utilisation de commandes spécifiques) ;

2) demander à un moteur LaTeX de transformer ton document et ses commandes en un fichier lisible et utilisable : c'est la compilation ;

3) profiter du résultat fourni (format PDF) ou l'évaluer pour ensuite apporter des modifications au document, et ainsi de suite.

Quant au moteur LaTeX utilisé, il en existe plusieurs. Pour débuter, je recommande d'utiliser plutôt pdfTeX (intitulé apparemment PDFLaTeX sous `Texmaker`), qui permet de passer d'un coup du document LaTeX au fichier PDF final.

Quant aux autres moteurs, je les aborde bien plus loin dans ce guide, en page **??**. Je recommande plutôt de t'y rendre une fois que tu as un peu d'expérience sous LaTeX, pour ne pas perdre du temps et acquérir des bases solides.

Nous savons désormais que nous devons compiler avec le moteur pdfTeX… mais nous ne savons toujours pas comment faire ! Pas de panique : les logiciels que je t'ai fait installer prennent tout en charge.

3.2 Démarrer avec `Texmaker`

Pour gérer et éditer ses fichiers LaTeX, `Texmaker` est un excellent logiciel. 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 :

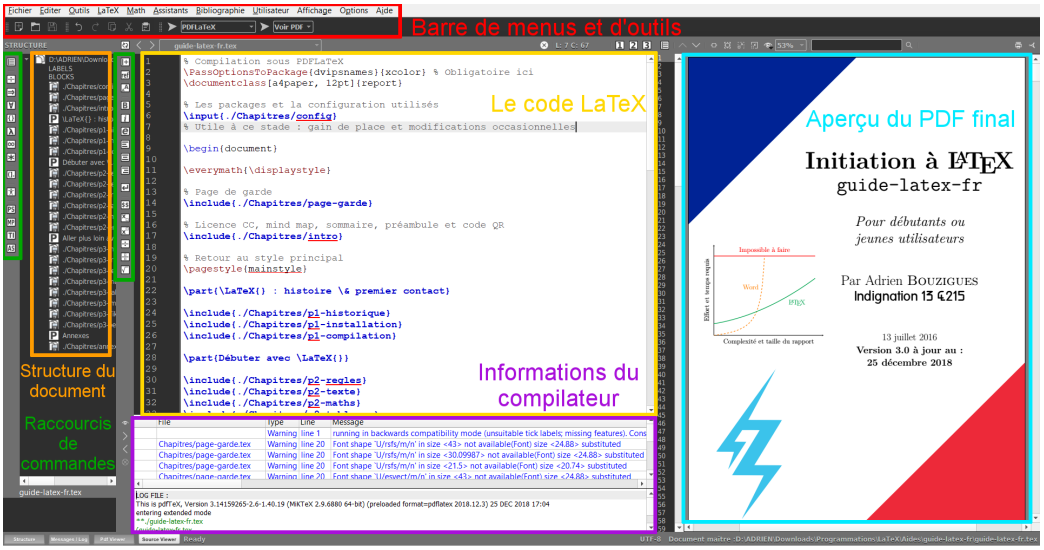


Figure 3.2 – Aperçu de `Texmaker`

Revenons sur chaque point :

→ barre de menus et d'outils : plein de commandes LaTeX préremplies. Personnellement, je l'utilise très rarement (y compris le bouton de sauvegarde). Je préfère utiliser les raccourcis clavier (en l'occurrence, Ctrl + S) ;

→ structure du document : très pratique pour naviguer dans le code du document ouvert ;

→ 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 LaTeX : c'est ici que tu tapes le texte de ton document et les commandes LaTeX 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.

Si jamais cet aperçu n'est pas disponible (fenêtre d'affichage inexistante comme sur mon image), il faut procéder de la manière suivante :

1) Aller dans `Options` puis dans `Configurer Texmaker`.

2) Dans l'onglet `Afficheur Pdf`, choisir les options `Afficheur Pdf interne` et `Intégré à la fenêtre`. Valider.

3) Un bouton `Pdf Viewer` est alors disponible en bas à gauche et te permet d'activer ou non cette fenêtre d'aperçu.

Si jamais ce n'est pas clair, j'espère que cette capture d'écran permettra de lever le moindre doute :

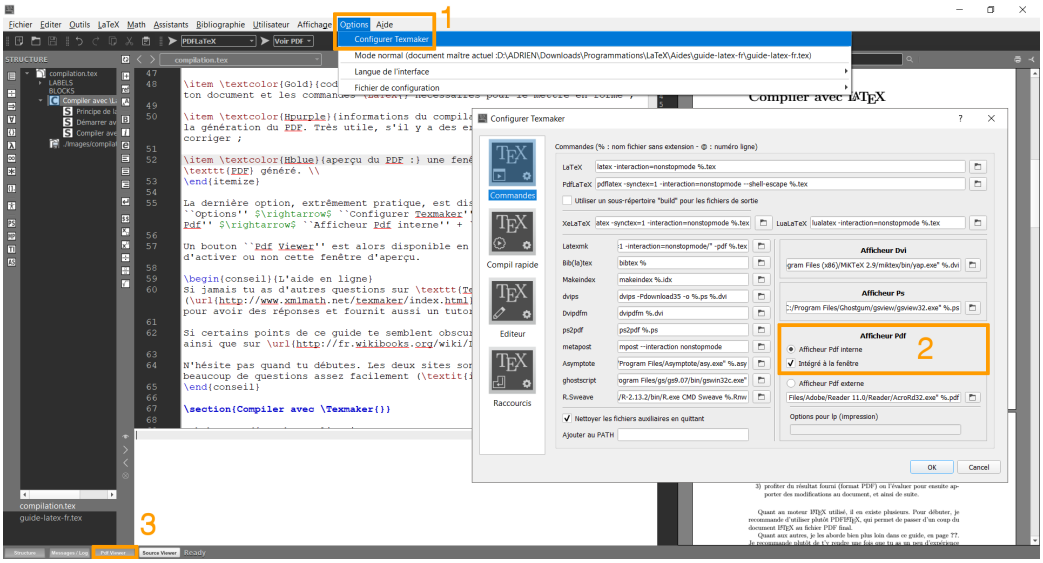


Figure 3.3 – Procédure pour obtenir l'aperçu du PDF

L'aide en ligne

Si jamais tu as d'autres questions sur `Texmaker`, son site officiel (`http://www.xm1math.net/texmaker/index_fr.html`) est le meilleur endroit pour avoir des réponses et fournit aussi un tutoriel pour débuter avec LaTeX.

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).

3.3 Compiler avec `Texmaker`

Maintenant que l'environnement propre à `Texmaker` a été présenté, voyons un peu plus dans le détail un dernier point : la compilation. Pour commencer, reprenons le code « Bonjour monde ! » utilisé en page 18 :

Bonjour monde !

```
\documentclass[]{report}


\begin{document}

Bonjour monde !

\end{document}
```

Je suppose que tu as suivi les premières indications fournies en page 18, soit ouvrir `Texmaker`, recopier le code « Bonjour monde ! » fourni et enregistrer ton document.

Si tu n'as pas précisé d'extension, tu remarqueras au passage que ton fichier a été sauvegardé avec l'extension `.tex`, qui correspond à l'extension pour des fichiers LaTeX.

Il existe ensuite 3 façons de lancer la compilation de ton document LaTeX :

→ via l'invite de commandes de ton système d'exploitation (le CMD pour les utilisateurs de `Windows`)... maisje n'en parlerai pas pendant ce guide.
Sache cependant que c'est possible mais ne présente aucun intérêt comme `Texmaker` propose des solutions plus pratiques ;

→ via `Texmaker` avec des clics souris ;

→ via `Texmaker` avec des raccourcis clavier (le plus rapide à mon sens).

Revenons sur les 2 derniers points plus dans le détail, pour que tu comprennes bien les actions à effectuer.

Pour une compilation via `Texmaker` avec des clics souris, il faut procéder en 3 temps (cf. Figure 3.4 si besoin) :

1) Choisir le moteur de compilation, PDFLaTeX dans notre cas, en haut dans la barre d'outils.

2) Lancer la compilation en cliquant sur la flèche à gauche du choix du moteur de compilation. Attendre que la compilation soit terminée.

3) Juste à droite du choix du moteur de compilation, bien choisir l'option `Voir PDF` et cliquer sur la flèche associée pour afficher le résultat.

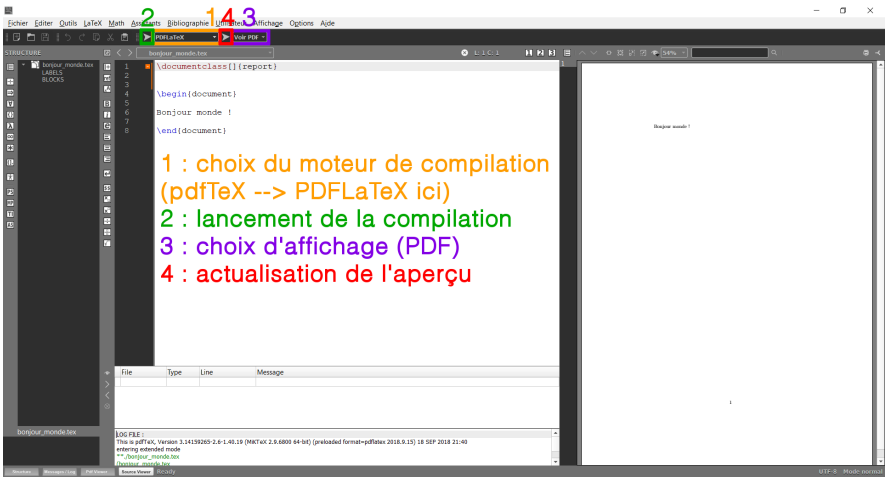


Figure 3.4 – Lancer la compilation avec des clics souris (`Texmaker`)

Pour une compilation via `Texmaker` avec des raccourcis clavier, il faut procéder en 2 temps (cf. Figure 3.5 si besoin) :

1) Lancer la compilation avec le moteur PDFLaTeX avec la touche F6.

2) Afficher le résultat avec la touche F7.

Ces raccourcis sont personnalisables dans les options de `Texmaker`, comme décrit ci-après (Figure 3.5).

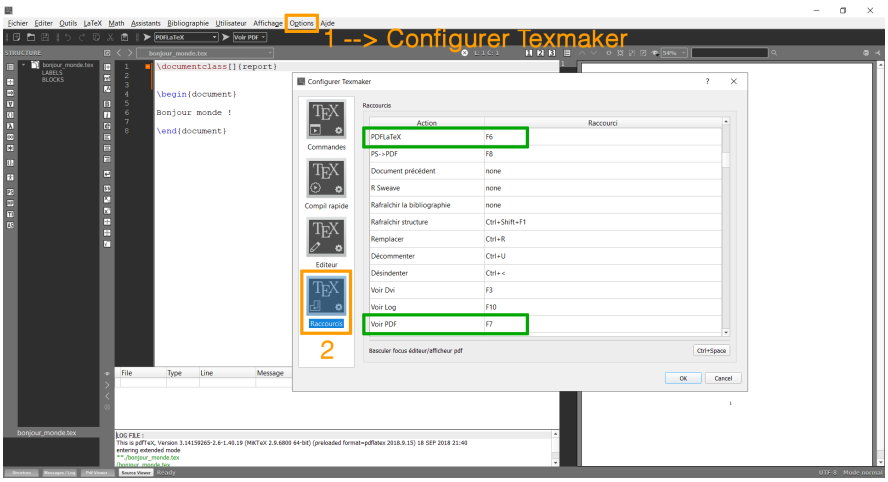


Figure 3.5 – Configuration des raccourcis clavier pour lancer la compilation (`Texmaker`)

Mais il y a encore plus rapide : lancer la compilation ET avoir l'aperçu du PDF actualisé en un seul raccourci clavier. C'est ce que `Texmaker` appelle la « compilation rapide » [1].

Tout d'abord, il faut s'assurer que la compilation rapide est bien programmée. Pour ce faire, il faut configurer `Texmaker` de la manière suivante :

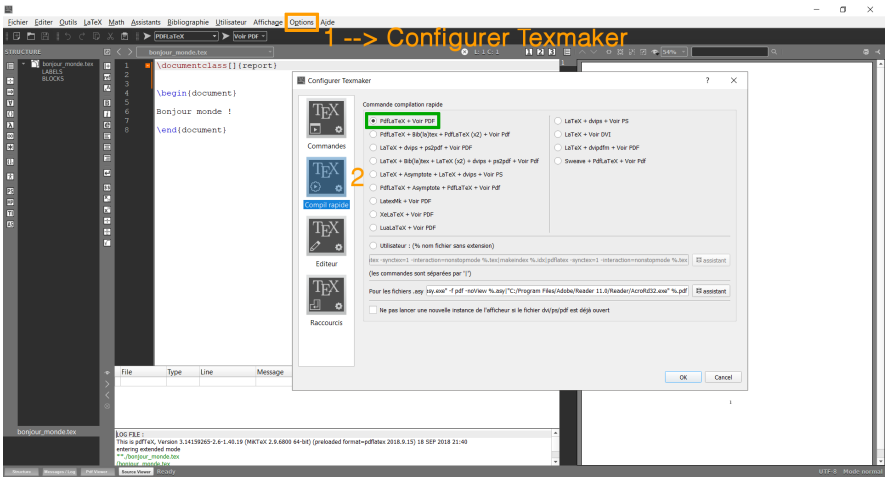


Figure 3.6 – Configuration de la compilation rapide (`Texmaker`)

Ensuite, il suffit d'appuyer sur la touche F1 pour lancer la compilation rapide. Il s'agit de la touche par défaut, paramétrable dans les options `Texmaker` comme indiqué en Figure 3.5.

Enfin, pour terminer les explications, il faut savoir que les compilations réalisées sous `Texmaker` sont équivalentes à l'utilisation de l'invite de commandes. Grâce à `Texmaker`, cette utilisation est transparente et grandement simplifiée... pour les non-initiés nous dirons !

La compilation : le conseil personnel

! Peu importe le moteur utilisé pour la compilation, tu peux remarquer qu'un fichier `.tex` entraîne toujours la génération d'autres fichiers. **C'est pourquoi je recommande toujours de travailler avec le fichier `.tex` placé dans un dossier,** pour éviter de submerger tes autres dossiers et de te perdre parmi les fichiers.

1. Il s'agit du nom attribué par le concepteur de `Texmaker`. Le temps nécessaire pour compiler le document n'est en rien diminué.

Enfin, ces fichiers secondaires n'ont besoin d'être conservés que le temps de travailler sur un document LaTeX. **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 final.

Attaquons désormais la raison première de ce guide : faire du LaTeX.

Deuxième partie

Débuter avec LaTeX

Chapitre 4

Les règles de base

4.1 Les règles pour faire du LaTeX

Ces règles sont officieuses : je les ai élaborées à partir de mon expérience personnelle avec LaTeX. Elles restent donc pragmatiques et peuvent paraître un peu farfelues mais sont importantes à mes yeux :

Les 5 règles pragmatiques

Règle n° 1 : Tout est possible en LaTeX [a].

Règle n° 2 : La règle n° 1 est toujours vraie.

Règle n° 3 : 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 LaTeX passe par une recherche régulière sur Internet.

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

a. Y compris écrire des partitions de musique : `http://fr.wikibooks.org/wiki/LaTeX/%C3%89crire_de_la_musique`

Passons maintenant sur des règles plus concrètes vis-à-vis de l'écriture d'un code LaTeX.

4.2 Les 3 règles d'or en LaTeX

Pour écrire du code LaTeX, il existe 3 règles, suffisamment importantes à mon sens pour être en or :

Les 3 règles d'or en LaTeX

Règle d'or n° 1 : Toute commande LaTeX débute par un *backslash* "\".
`Windows` : Alt Gr + 8
`Apple` : Alt + ⇧ + /

Règle d'or n° 2 : Tout texte concerné par une commande LaTeX est délimité par des accolades "{" et "}".
`Windows` : Alt Gr + 4 et Alt Gr + =
`Apple` : Alt + (et Alt +)

Règle d'or n° 3 : Toute commande LaTeX 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 pour écrire du code LaTeX. 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 LaTeX (cf. 5.2 Un peu de mise en forme, p. 41).

4.3 La base d'un document LaTeX

Pour commencer, démarrons un fichier LaTeX : ouvrons `Texmaker`, créons un nouveau fichier et enregistrons-le au format `.tex`[1].

1. Pour information/rappel, un fichier LaTeX possède toujours l'extension `.tex`

Codes LaTeX fournis

Tout au long de ce guide, des exemples de code LaTeX sont fournis dans des encadrés verts clairs. Ils ont été testés par mes soins avec le moteur PDFLaTeX : tout devrait donc fonctionner aussi de ton côté.

Toutefois, la copie du code 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 et l'apprentissage selon moi – ou si tu préfères copier-coller et habilement utiliser la fonction « Remplacer » de `Texmaker`.

La base d'un document LaTeX est la suivante :

La base d'un document LaTeX

```
\documentclass[options]{classe}

% 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 interprété par LaTeX.
```

Plusieurs points importants sont à retenir :

→ `\documentclass` permet de définir le type de document (appelé « *classe* » en LaTeX) sur lequel tu vas travailler ;

→ la zone entre `\documentclass` et `\begin{document}` s'appelle le préambule. Je décris cette partie en 4.4 Les packages, p. 33 ;

→ un premier exemple d'illustration de la règle d'or n°3 : un `begin` implique un `end`.
Seuls le texte et les commandes LaTeX écrits entre `\begin{document}` et `\end{document}`, hormis les commentaires, sont interprétés par LaTeX lors de sa création du fichier PDF final ;

→ tout ce qui peut être écrit après `\end{document}` n'est pas pris en compte par LaTeX.

Que mettre maintenant dans la ligne `\documentclass` ? Il s'agit ici de définir le type de document à mettre en forme. En LaTeX, le terme consacré est « *classe* ». Définir la classe d'un document LaTeX revient à utiliser un gabarit spécifique pour le document, défini par défaut dans le code source de LaTeX, et entièrement personnalisable par la suite si besoin.

Il existe plusieurs classes, à renseigner à l'endroit où il y a écrit `classe` dans mon exemple générique : `report` pour taper des rapports ; `article` pour des articles scientifiques ; `book` pour des livres et `letter`, tu as compris je pense, pour des lettres.

La partie `options` permet ensuite de renseigner toutes les options propres à une classe. Les plus communs sont la taille du papier (A4 : `a4paper`, A5 : `a5paper`) ainsi que la taille de police de base (`10pt`, `11pt` ou `12pt`). Mais il en existe d'autres, en fonction des classes utilisées.

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

Néanmoins, il faut bien 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`.

Ce qu'il faut bien comprendre et surtout retenir, c'est que **la forme finale de ton document est intrinsèquement liée à sa classe et aux options choisies.**

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 5.2 Un peu de mise en forme, p. 41.

C'est bon ? Toujours là ? Tu verras, avec de la pratique, les bases vont rentrer. Plus qu'un dernier point un peu théorique à aborder et nous passerons à la pratique. Promis !

4.4 Les packages

Si le lecteur 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
    lisibilité (règle pragmatique n°3)
```

Si jamais tu ne sais pas quoi faire du code, je ne peux que t'inviter à te rendre en page 20. Tu y trouveras tout un chapitre consacré à la compilation sous LaTeX, soit l'étape pour transformer ton code en un PDF !

Normalement, suite à la compilation, tu as dû obtenir :

J'aime crire en LaTeX !

Analysons le résultat. La règle d'or n°1 commence à prendre du sens : une commande LaTeX commence par un *backslash* "\" (Ou contre-oblique pour les puristes). Cette commande me permet d'écrire le mot « LaTeX » d'une manière plus élégante.

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 LaTeX d'écrire en `UTF-8`[2] !

Pour ce faire, il faut dire à LaTeX de charger des options supplémentaires. Dans le jargon LaTeX, ces options sont appelées des *packages*. Dans la littérature française, le terme de « paquetages » est parfois employé.

Les packages sont toujours renseignés dans le préambule, soit entre les lignes `\documentclass[options]{classe}` et `\begin{document}`. Pour charger un package, il faut utiliser la commande :

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

Pour rédiger des documents en français, il est recommandé de remplir le préambule de la manière suivante :

Un exemple qui fonctionne bien

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

\usepackage{lmodern} % Police standard sous LaTeX : Latin
    Modern
% (alternative à la police d'origine développée par Donald
    Knuth : Computer Modern)
\usepackage[french]{babel} % Pour la langue française
\usepackage[utf8]{inputenc} % Pour l'UTF-8
\usepackage[T1]{fontenc} % Pour les césures des caractères
    accentués


\begin{document}

J'aime écrire en \LaTeX{} !
```

2. L'`UTF-8` est un codage de caractères informatiques, qui tolère les accents : `http://fr.wikipedia.org/wiki/UTF-8`.

```
\end{document}
```

Je sens la curiosité briller dans ton regard donc je vais essayer de te donner un peu plus de détails que les commentaires fournis[3] :

- le package `inputenc`, avec l'option `utf8`, permet de prendre en compte l'utilisation de caractères accentués dans le fichier source (soit ton fichier `.tex`). Concrètement, `inputenc` se contente en fait de faire lui-même la conversion entre les caractères accentués et les commandes d'accentuation propres à LaTeX ;

- si `inputenc` gère l'affichage des caractères accentués, la césure reste catastrophique ! Pour indiquer au compilateur les règles de césure pour les mot accentués, il faut donc utiliser le package `fontenc`, avec l'encodage `T1` en option ;

- là encore, le résultat est loin d'être parfait. Le fait de charger `fontenc` remplace les polices par défaut par des fontes de type 3, c'est-à-dire non vectorielles. En clair, si tu zoomes sur un caractère accentué de ton PDF, il sera pixélisé. D'où le chargement en amont de la police *Latin Modern*, via le package `lmodern` ;

- enfin, pour s'adapter à la langue de Molière, le package `babel` avec l'option `french` est indispensable !

Une question ?

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

Tout simplement parce qu'il s'agit d'une commande présente de base dans le code source de LaTeX. Il n'y a donc pas besoin de charger quoi que ce soit au préalable.

3. Les explications qui suivent proviennent du site `http://blog.dorian-depriester.fr/latex/mais-a-quoi-bon-servent-les-packages-fontenc-et-inputenc`

! Sache aussi que les packages sont construits par les utilisateurs LaTeX. C'est pourquoi tout est possible avec 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 écrire des paragraphes. Passons donc à la pratique !

Chapitre 5

Gestion du texte et mise en forme

MAINTENANT que nous connaissons les règles de base pour faire du LaTeX et que les packages ont été introduits, nous allons pouvoir commencer à écrire du texte sous LaTeX.

Par la suite, pour alléger les exemples, le préambule ne sera plus renseigné dans les codes LaTeX mis à disposition. Ces derniers seront basés sur l'architecture du code minimal fourni ci-après. L'ajout de nouveaux packages sera signalé au début du code par un commentaire.

Le code minimal

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

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


\begin{document}

% Ecrire le code ici !

\end{document}
```

5.1 Notre premier texte

Prêt à enfin écrire un roman ? Bon, nous allons attaquer en douceur. Et, petit à petit, tu auras suffisamment d'outils à ta disposition pour profiter des fonctionnalités offertes par LaTeX. Poursuivons avec le code suivant :

Un nouvel essai

```
J'aime écrire en \LaTeX{} ! Vraiment ! Surtout avec des
    phrases longues qui prennent de la place. % 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
```

Tu devrais normalement obtenir :

Le résultat

J'aime écrire en LaTeX ! Vraiment ! Surtout avec des phrases longues qui prennent de la place.
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 et donc de créer un nouveau paragraphe ;

→ les espaces et saut de ligne intempestifs ne sont pas pris en compte ;

→ les alinéas sont automatiques (pas besoin de faire de tabulations).

Contrairement à Word, à première vue, LaTeX possède une façon un peu

curieuse d'aller à la ligne ou de faire un saut de ligne (nouveau paragraphe). Mais rappelons que la raison d'être de LaTeX est de séparer le fond de la forme : nous tapons le fond pour laisser LaTeX le mettre en forme lors de la compilation.

Mais l'utilisateur garde le contrôle et peut influer sur la forme grâce à des commandes LaTeX. C'est bien ce qui se passe ici. Si l'utilisateur veut un retour à la ligne, il doit sauter une ligne dans son code. S'il veut sauter une ligne, il doit utiliser la commande `\\` et sauter une ligne dans son code.

Nous avons aussi pu remarquer que LaTeX gère tous les problèmes liés à l'espacement entre les mots. Il est donc inutile de faire un grand nombre d'espaces ou de saut de ligne pour aérer son texte. Encore une fois, ce n'est pas la politique de LaTeX et il faut passer par des commandes si besoin.

Si le saut de ligne est disponible grâce à la commande `\\` – qui, au passage, est cumulable –, l'utilisateur peut jouer sur l'espacement vertical grâce à la commande `\vspace{longueur}`, avec `v` pour vertical et `space` pour espace. Il en va de même pour un espacement horizontal avec la commande `\hspace{longueur}`.

La longueur est totalement libre, à condition de renseigner correctement l'unité : `13mm` ou `215pt`, par exemple. Concrètement, nous pouvons procéder de la manière suivante :

Gérer l'espacement

```
J'aime toujours écrire en
    \LaTeX{}.

\vspace{1cm}

Surtout \hspace{8mm} quand je
    laisse du blanc !
```

J'aime toujours écrire en LaTeX.

Surtout quand je laisse du blanc !

Si j'ai rapidement annoncé qu'il était possible de cumuler la commande `\\` pour engendrer la création de plusieurs sauts de ligne, il existe aussi une longueur [1] définie nativement sous LaTeX et qui correspond à un saut de ligne.

Cette longueur est disponible grâce à la commande `\baselineskip`. Voyons son utilisation sur un cas pratique :

1. Nous aurons l'occasion de revenir sur ce point en page 94.

Sauts de ligne et longueur `baselineskip`

```
Il est possible de sauter
    plusieurs \\ \\ % Double
    saut de ligne

lignes \\ \\ \\ % Triple saut de
    ligne

ainsi.

\vspace{2\baselineskip} % Double
    saut de ligne

Cette solution est aussi
    possible, tout comme
    celle-ci ! \\[\baselineskip]
    % Double saut de ligne

Bref, beaucoup de façons de
    sauter des lignes, de
    manière plutôt concise.
```

Il est possible de sauter plusieurs

lignes

ainsi.

Cette solution est aussi possible, tout comme celle-ci !

Bref, beaucoup de façons de sauter des lignes, de manière plutôt concise.

Enfin, sache qu'il est possible de rentrer des valeurs négatives, comme `-13mm` ou `-215pt`. C'est surtout pratique pour remonter du texte lors de montages, voire des images si besoin. Je recommande juste de limiter cette pratique : tu risques de perdre beaucoup de temps à ajuster ton document.

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.

Cependant, pour obtenir un retour à la ligne sur ton document, tu peux aussi terminer ta phrase par la commande `\\` et faire un simple retour à la ligne dans ton code.

Bon, maintenant que nous avons toutes les cartes en main pour écrire des paragraphes, passons à de la mise en forme.

5.2 Un peu de mise en forme

Comme je l'ai déjà annoncé, avec LaTeX, tu rédiges le fond et lui laisses le soin de s'occuper de la forme. Si nous en avons eu un premier aperçu, tu vas pouvoir t'en rendre véritablement compte dès à présent.

Si tu veux mettre un texte en gras ou en italique, il faut donc l'indiquer à LaTeX par le biais de commandes bien spécifiques :

Gras & italique

```
\textbf{texte en gras} \\
\textit{texte en italique}
```

texte en gras

texte en italique

Comme tu peux le constater, tu écris la commande – qui débute par un *backslash* – et tu encadres le texte concerné par des accolades. Fort heureusement, LaTeX ne propose pas uniquement le gras et l'italique :

Table 5.1 – Les différentes possibilités de mise en forme du texte

Texte	Rendu	Environnement
`\textbf{gras}`	**gras**	`bfseries`
`\textit{italique}`	*italique*	`itshape`
`\emph{emphase}`	*emphase*	`em`
`\textsl{penché}`	*penché*	`slshape`
`\textsc{Petites Capitales}`	Petites Capitales	`scshape`
`\textsf{sans empattement}`	sans empattement	`sffamily`
`\texttt{machine}`	`machine (à écrire)`	`ttfamily`

Je vais revenir plus en détail sur l'emphase, avec la commande `\emph{texte}`, qui ne correspond *pas* à de l'italique.

En typographie, l'emphase permet d'accentuer un mot ou une phrase grâce à un style ou une police différente de celle du reste du texte. Essayons avec un exemple répandu de faux texte : le *lorem ipsum*[2].

2. Pour plus de renseignements : `http://fr.wikipedia.org/wiki/Faux-texte` et `http://fr.lipsum.com/`.

L'emphase ou *emphasis*

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

\textit{Vestibulum porttitor,
    ligula vitae suscipit
    bibendum, \emph{lorem ligula
    vestibulum ipsum,} sed
    ultricies tellus dolor sit
    amet odio.}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. *Nunc est leo, facilisis non nisi eget,* auctor eleifend metus.

Vestibulum porttitor, ligula vitae suscipit bibendum, lorem ligula vestibulum ipsum, *sed ultricies tellus dolor sit amet odio.*

Comme tu peux le constater, LaTeX a adapté la mise en forme du texte avec emphase en fonction de la mise en forme du paragraphe !

Voyons maintenant sur un exemple comment augmenter ou réduire la taille de la police, ainsi que les notes de bas de page :

Taille de police et note de bas de page

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

% Note
Note de bas de page\footnote{La
    note en question.}.
```

tiny scriptsize footnotesize small normalsize large Large LARGE huge Huge

Note de bas de page [a].

a. La note en question.

Comme tu peux le constater, LaTeX gère lui-même la numérotation des notes de bas de page, sans rien avoir besoin de lui indiquer. Pratique, n'est-ce pas ?

Si jamais tu as besoin d'appliquer une mise en forme ou une taille de police à plusieurs paragraphes, LaTeX ne saura pas interpréter le changement de paragraphe. Dans ce cas, il faut passer par un environnement [3] :

Mise en forme et taille – Environnement

```
\begin{Large}
Paragraphe 1.

Paragraphe 2. \\
\end{Large}

\begin{bfseries}
Paragraphe 3.

Paragraphe 4.
\end{bfseries}
```

Paragraphe 1.
Paragraphe 2.

Paragraphe 3.
Paragraphe 4.

LaTeX propose aussi un moyen très simple pour accentuer les majuscules. Il suffit d'utiliser un *backslash*, suivi de l'accent désiré. Puis, tu écris ton mot normalement, avec une majuscule.

Accentuation

```
% Accent sur les majuscules
\'E, \`E, \^E et \c{C} \\

% "o pris dans e"
\OE{}il, c\oe{}ur
```

É, È, Ê et Ç

Œil, cœur

Il existe aussi des commandes spécifiques pour des symboles fréquemment utilisés. Je pense notamment aux guillemets et aux points de suspension. La preuve par l'exemple :

3. Pour la mise en forme du texte, utiliser les noms définis dans la colonne « Environnement » de la Table 5.1.

Autres symboles utiles

```
% Guillemets
\og guillemets français \fg{} et
    ``guillemets anglais'' \\

% Points de suspension
Points de suspension\dots{} \\

% Tirets
Tiret court : - \\
Tiret moyen : -- \\
Tiret long : --- \\

% Esperluette - Pourcentage
\& et \textit{\&} ; \%
```

« guillemets français » et
“guillemets anglais”

Points de suspension...

Tiret court : -
Tiret moyen : –
Tiret long : —

& et *&* ; %

Une liste plus complète des symboles utiles sous LaTeX est disponible en annexes, p. 119. Nous remarquons au passage que le *backslash* sert aussi de **caractère d'échappement** pour tous les symboles utilisés lors de l'écriture du code LaTeX (&, $, #, _, { ou } par exemple).

Les marges

Les gens me demandent souvent comment modifier les marges sous LaTeX. Personnellement, j'ai fini par m'habituer aux marges natives de LaTeX : je les modifie donc que pour des besoins très particuliers.

Si tu tiens à savoir pourquoi les marges sont plus grandes que celles d'un document Word, c'est parce que LaTeX a initialement été inventé par des Américains (conventions américaines).
De plus, LaTeX sert pour rédiger des rapports scientifiques : leur reliure demande alors une marge plus importante s'ils sont épais.

Autrement, si tu as absolument besoin de modifier les marges de ton document, je te recommande le package `geometry`, ainsi que la page suivante : `http://fr.wikibooks.org/wiki/LaTeX/Mise_en_page#Modification_des_marges`.

Bien, nous avons fait un premier tour d'horizon des premières possibilités offertes par LaTeX pour mettre en forme le texte. Passons à un peu d'organisation.

5.3 Organiser son document

La base

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 :

- → `\part{titre}` ;
- → `\chapter{titre}` ;
- → `\section{titre}` ;
- → `\subsection{titre}` ;
- → `\subsubsection{titre}` ;
- → `\paragraph{titre}` ;
- → `\subparagraph{titre}`.

Nota Bene

La commande `\part` n'est disponible que pour un document de classe `report` (cf. 4.3 La base d'un document LaTeX p. 30).

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

```
% Enlever les % ici

%\part{Partie I}

%\section{Section 1.1}

Comment est-ce numéroté ?
```

```
%\section{Section 1.2}

D'une manière bizarre !

%\part{Partie II}

%\chapter{Chapitre 1}

%\section{Section 1.1}

Lorem ipsum\dots{}

\newpage

Bis repetita\dots{}
```

Configuration de la numérotation

Comme tu peux le constater, il y a quelques problèmes de numérotation. Si les compteurs tournent normalement, il faut juste donner un coup de pouce à LaTeX pour faire correctement les choses. Retente le même code avec ces commandes dans le préambule :

La numérotation des titres

```
% ATTENTION : écriture de ces commandes dans le PREAMBULE !!!

% RAZ des numéros de section après un chapitre
\makeatletter\@addtoreset{section}{chapter}\makeatother

% Pour mettre des I, II, etc. aux parties
\renewcommand{\thepart}{\Roman{part}}

% Pour mettre des 1, 2, etc. aux chapitres
\renewcommand{\thechapter}{\arabic{chapter}}

% Idem pour les sections et avoir le numéro de chapitre
\renewcommand{\thesection}{\thechapter.\arabic{section}}
```

Il existe plusieurs possibilités pour numéroter les différents titres de ton document. La liste complète des commandes utilisables est la suivante :

- `\arabic` : pour avoir des chiffres arabes soit 1, 2, 3. . . ;
- `\roman` : pour avoir des chiffres romains minuscules soit i, ii, iii. . . ;
- `\Roman` : pour avoir des chiffres romains majuscules soit I, II, III. . . ;
- `\alph` : pour avoir des lettres minuscules soit a, b, c. . . ;
- `\Alph` : pour avoir des lettres majuscules soit A, B, C. . ..

Titre sans numérotation

Enfin, dans le cas où tu veux juste écrire un titre sans numéro, il suffit d'ajouter une `*` à la commande du titre : `\part*{titre}`, etc.

Bon, c'est bien gentil d'avoir des titres. Comment obtenir un sommaire désormais ?

5.4 Gestion du sommaire

La base

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

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 obtenir le résultat final attendu.

À la première compilation, LaTeX créé 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 générer 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, juste avant `\tableofcontents` par exemple :

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

Le nec plus ultra

Avoir un sommaire, c'est bien. Pouvoir interagir avec, c'est encore mieux ! Pour pouvoir se déplacer rapidement dans le document grâce à un clic sur un titre du sommaire, il faut utiliser un nouveau package : `hyperref`.

Cependant, le résultat par défaut n'est pas très esthétique et peut entraîner une crise d'épilepsie aux plus sensibles, c'est pourquoi je recommande d'utiliser l'option `colorlinks`[4].

Mais `hyperref` va beaucoup plus loin et permet de personnaliser les options de lecture par défaut du PDF généré. C'est pourquoi je recommande d'utiliser aussi les nombreuses options comme indiqué dans l'exemple ci-après.

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. Des fois, charger le package supplémentaire `url`, avec l'option `hyphens`, est obligatoire pour traiter les derniers cas de figure problématiques.

4. Si tu ne me crois pas, génères un sommaire avec juste le package `hyperref`, sans option, et regarde le rendu du fichier PDF. Tu comprendras.

Le sommaire – Bilan

```
% Ajout au PREAMBULE
%\usepackage[hyphens]{url} % Pour des césures correctes dans
    les URLs
%\usepackage[pdfauthor = {{Prénom Nom}}, pdftitle = {{Titre
    document}}, pdfstartview = Fit, pdfpagelayout =
    SinglePage, pdfnewwindow = true, bookmarksnumbered =
    true, breaklinks, colorlinks, linkcolor = red, urlcolor =
    black, citecolor = cyan, linktoc = all]{hyperref} %
    Renvois -- Options Adobe/lecteur PDF


\begin{document}

% Page de garde

% Sommaire -- Penser à compiler deux fois
{
\hypersetup{hidelinks} % Sommaire en "noir"
\renewcommand{\contentsname}{Sommaire} % Remplacer "Table des
    matières"
\tableofcontents % Affichage du sommaire
}

% Si nécessaire
%\clearpage % Mieux qu'un \newpage ou des erreurs dans le
    sommaire parfois

% Parties, chapitres, texte, etc.

% Commande fournie avec le package hyperref
\url{https://www.ctan.org/}

\end{document}
```

Pour revenir rapidement sur les options du package `hyperref`, en voici un descriptif :

→ `pdfauthor` & `pdftitle` : pour renseigner correctement les champs des options du fichier PDF.

Il est possible de remplir les autres champs disponibles : cf. la docu-

mentation du package `hyperref` [5] ;

→ `pdfstartview` & `pdfpagelayout` : pour les options d'affichage du PDF à sa lecture.
Pour connaître toutes les options disponibles, cf. la documentation du package `hyperref` ;

→ `pdfnewwindow = true` : si ton document contient un lien vers un autre fichier PDF, cliquer sur le lien entraîne l'ouverture du PDF dans un nouvel onglet (et non à la place du premier PDF) ;

→ `bookmarksnumbered` : pour les signets du lecteur PDF ;

→ `breaklinks` : pour permettre la césure des liens insérés trop longs ;

→ `colorlinks` et toutes les couleurs qui suivent : pour colorer correctement les références du document ;

→ `linktoc = all` : pour faire un renvoi du sommaire avec les numéros de page.

Ajout d'un titre étoilé

Enfin, dans le cas des titres étoilés, ces derniers n'apparaissent pas dans le sommaire. Il existe malgré tout un moyen de l'ajouter manuellement, si tu y tiens. Cette solution requiert l'utilisation du package `hyperref`, que nous connaissons déjà.

Ajout d'un titre étoilé dans le sommaire

```
% Ajout dans le préambule
%\usepackage{hyperref}


% Ajout d'un titre sans numéro

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

5. Et comme je suis adorable, voici le lien : `https://www.ctan.org/pkg/hyperref`.

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

Lorem ipsum dolor\dots{}
```

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.).

Le numéro de page correspond à l'endroit où est tapée la commande, d'où son positionnement **après** `\section*`, pour éviter une mauvaise numérotation si le titre étoilé débute sur une nouvelle page.

Il faut donc faire bien attention avec cette situation pour garantir la cohérence du document : il faut renseigner le même titre dans `\section*` et dans `\addcontentsline...`. C'est pourquoi, personnellement, j'utilise les titres étoilés le moins possible.

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

5.5 La page de garde

La base

Comme pour le sommaire, il faut d'abord créer la page de garde puis indiquer à LaTeX de l'afficher. Pour la créer, il faut remplir les informations suivantes :

La page de garde – Création

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


% Titre
\title{Titre}
%\title{\textbf{Titre}} % Ressort mieux selon moi
```

```
% Auteur
\author{Prénom \textsc{Nom} \\ Profession}

% Date
\date{\today} % Date du jour (compilation du document)
%\date{date_à_afficher} % Date fixe
```

Naturellement, tout n'a pas besoin d'être renseigné. Si tu veux uniquement le titre, tu laisses juste la commande `\title{Titre}`.

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`, ce qui donne :

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

LaTeX s'occupe ensuite de la mise en forme de tous ces noms. Pratique, n'est-ce pas ?

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

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 $N+1$. C'est parce que LaTeX ne numérote pas la page de garde et commence ensuite la numérotation à 1 au lieu de 2...

C'est peut-être un détail mais, personnellement, je trouve extrêmement irritant d'aller dans le sommaire, de trouver le numéro de page de la section qui t'intéresse, de la saisir dans ton lecteur PDF... et d'arriver à la mauvaise page !

Fort heureusement, il suffit d'ajouter après `\maketitle` la commande `\setcounter{page}{2}`, pour réajuster correctement la numérotation des pages.

S'il fallait synthétiser les différentes options de base pour la page de garde, nous pourrions alors nous servir du code suivant :

La page de garde – Bilan

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

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


\begin{document}

% Informations de la page de garde
\title{\textbf{Initiation à \LaTeX{}} \\ \textit{Pour
    débutants ou jeunes utilisateurs}}
\author{Adrien \textsc{Bouzigues} \\ Salarié \and John
    \textsc{Doe} \\ Profession}
\date{\today}

% Générer la page de garde
\maketitle


% Changer le titre du résumé
%\renewcommand{\abstractname}{\Large{}\textbf{Résumé}}

% Résumé
\begin{abstract}
Résumé du document
\end{abstract}
% Classe report : sur une page à part
% Classe article : sur la page de garde (si pas de newpage)

\clearpage\setcounter{page}{2}

Lorem ipsum dolor\dots{}

\end{document}
```

Une question ?

« Ce n'est pas pratique ta page de garde. C'est sobre, impossible de mettre une image ! Est-il possible d'avoir mieux »

... Tu as parfaitement raison ! Il est tout à fait possible d'avoir une page de garde plus personnelle et plus décorée. Pour satisfaire ta curiosité, je te propose une première solution "simple" à utiliser.

!

Mais prends garde ! Dès l'instant où tu commences à arpenter ce chemin – construire une page spécifique à partir de rien –, tu peux très vite y passer beaucoup de temps. LaTeX n'est pas un outil de création graphique à la base.

Pour un rapport officiel ou si le temps t'est précieux, je recommande d'utiliser les commandes de base que je viens de présenter.

Autrement, tu peux te permettre, comme je le fais pour ce guide, de construire ta propre page personnalisée. Il n'y a pas une seule bonne façon de faire et tout dépend de ce que tu veux faire.

Personnalisation de la page de garde

Comme promis, voici un exemple "simple" pour avoir une première page de garde personnalisable. Les possibilités sont très nombreuses avec LaTeX : tout dépend donc de ce que tu veux faire.

Une solution personnalisable

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

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

\usepackage{graphicx} % Pour insérer une image (logo)
% cf. les chapitres suivants pour plus de détails


\begin{document}
```

```
% Auteur : WikiBooks
    (http://en.wikibooks.org/wiki/LaTeX/Title_Creation)
% License : CC BY-NC-SA 3.0
    (http://creativecommons.org/licenses/by-nc-sa/3.0/)
% Adaptation du document d'origine

% Environnement titlepage : permet de créer une page de garde
    et de la personnaliser à volonté
\begin{titlepage}
\newcommand{\HRule}{\rule{\linewidth}{0.5mm}} % Ligne
    horizontale (épaisseur modifiable)

\begin{center} % Centrer le contenu de la page
% En-têtes
\textsc{\LARGE{}Université} \\[1.5cm]
\textsc{\Large{}En-tête principal} \\[0.5cm] % Nom du cursus
    (par exemple)
\textsc{\large{}En-tête secondaire} \\[0.5cm] % Intitulé du
    cours (par exemple)

% Titre
\HRule \\[0.6cm]
{\huge\bfseries{}Titre} \\[0.25cm]
\HRule \\[1.5cm]

% Auteur
\begin{minipage}{0.45\linewidth}
\begin{flushleft}
\Large\textit{Auteur :} \\
John \textsc{Smith} % Nom auteur
\end{flushleft}
\end{minipage}
\hfill
\begin{minipage}{0.45\linewidth}
\begin{flushright}
\Large\textit{Superviseur :} \\
Dr. John \textsc{Smith} % Nom superviseur
\end{flushright}
\end{minipage} \\[2cm]
```

```
% Si aucun superviseur, utiliser les lignes ci-après et
    supprimer les lignes précédentes
%\Large\textit{Auteur :} \\
%John \textsc{Smith} \\[3cm] % Nom auteur

% Date
{\large\today} \\[2cm] % Date : \today ou date saisie à la
    main

% Logo
%\includegraphics{logo.png} \\[1cm] % Logo à utiliser
\end{center}

\vfill % Remplir le reste de la page avec du blanc
\end{titlepage}

\clearpage\setcounter{page}{2}

Lorem ipsum dolor\dots{}

\end{document}
```

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

5.6 Création de commandes

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

Pour ce faire, il suffit d'ajouter la ligne suivante, de préférence dans le préambule même si tu peux l'insérer n'importe où dans ton document (avant le premier appel de ta nouvelle commande) :

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

\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 plus concrète lorsque nous aborderons les mathématiques, voici un cas sans argument :

Un second cas

```
% Création de la commande après
    \begin{document} : OK
\newcommand{\SAV}{\textbf{Service
    Après-Vente}}

Notre \SAV vous aide. \\

Grâce à notre \SAV, vous serez
    comblés.
```

Notre **Service Après-Vente**vous aide.

Grâce à notre **Service Après-Vente**, vous serez comblés.

J'ai décidé de mettre le texte en gras, mais rien ne m'empêche en cours de rédaction de mon rapport de modifier ce choix. L'avantage ? Tu as juste à

modifier la commande et, lors de la compilation, la modification se répercute sur tout le document ! Pratique, non ?

C'est aussi pourquoi je recommande de placer tous les `newcommand` dans le préambule : c'est plus pratique pour les retrouver s'ils sont tous au même endroit, au lieu d'être dispersés dans le document.

Une question ?

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

Tu as l'œil ! 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.

Pour indiquer à LaTeX la fin de la commande, il faut donc la fermer avec des accolades. C'est ce que je fais par exemple lorsque j'écris LaTeX : le code derrière est `\LaTeX{}`.

Il aurait donc fallu écrire dans mon exemple : `notre \SAV{} vous` et `notre \SAV{}, vous`.

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.

Bien, continuons. C'est quoi déjà la suite ? Ah oui, les listes.

5.7 Les listes

La base

Les listes (à puces ou numérotées) sont des outils fort pratiques quand il s'agit d'énumérer des éléments, faire un inventaire ou décrire des étapes.

Les listes peuvent donc être soit non numérotées (listes dites « à puces » : tiret, rond, autres symboles) : il faut alors utiliser l'environnement `itemize` ; soit numérotées (numéro ou lettre) : il faut donc passer par l'environnement `enumerate`.

Pour faire apparaître une puce ou un numéro, il faut utiliser la commande `\item`.

Voyons plutôt le résultat avec un exemple :

Les listes

```
Détail de la chambre :

\begin{itemize}
\item un lit ;
% Le saut de ligne est optionnel
% --> aération du code LaTeX

\item une armoire ;

\item et un bureau. \\
% Saut de ligne (\\) licite pour
    aérer le texte
\end{itemize}

Pour écrire facilement en
    \LaTeX{}, il faut :

\begin{enumerate}
\item Apprendre les bases.

\item Pratiquer les bases.

\item \^Etre curieux !
\end{enumerate}

% Imbrication des environnements
    (et donc des listes) possible
```

Détail de la chambre :

- un lit ;
- une armoire ;
- et un bureau.

Pour écrire facilement en LaTeX, il faut :

1) Apprendre les bases.
2) Pratiquer les bases.
3) Être curieux !

Normalement, tu devrais avoir un résultat un peu différent du mieux : les tirets sont plus grands et tes numéros se terminent par un point et non par une parenthèse.

Tout va bien ! J'ai juste une configuration par défaut de LaTeX qui génère ce rendu. Tu peux donc constater que créer des listes sous LaTeX se révèle très facile. Voyons maintenant comment les personnaliser à notre guise ?

5.7.1 Personnalisation des listes

Pour personnaliser ses listes, il existe un package incontournable : le package `enumitem`. Combiné avec le package `pifont` pour obtenir des symboles supplémentaires et donc avoir de nouvelles puces, c'est la meilleure combinaison possible pour personnaliser simplement ses listes.

Concrètement, le package `enumitem` offre quelques options, dont les suivantes que je recommande particulièrement :

→ `label = puce` : pour changer la puce ou la numérotation utilisée ;

→ `leftmargin = *` : pour supprimer l'indentation de la liste ;

→ `itemsep = <distance>` : pour insérer `<distance>` entre 2 puces. C'est plus commode ainsi que de devoir sauter des lignes avec `\\[<distance>]` à chaque fin de puce.

Pour ce qui est du package `pifont`, son utilisation est très simple. Il faut utiliser la commande `\ding{<num>}` avec `<num>` pris dans la liste ci-après pour afficher un symbole du package.

Table 5.2 – Liste des symboles du package `pifont`

32		33	✁	34	✂	35	✃	36	✄	37	☎	38	✆	39	✇
40	✈	41	✉	42	☛	43	☞	44	✌	45	✍	46	✎	47	✏
48	✐	49	✑	50	✒	51	✓	52	✔	53	✕	54	✖	55	✗
56	✘	57	✙	58	✚	59	✛	60	✜	61	✝	62	✞	63	✟
64	✠	65	✡	66	✢	67	✣	68	✤	69	✥	70	✦	71	✧
72	★	73	☆	74	✪	75	✫	76	✬	77	✭	78	✮	79	✯
80	✰	81	✱	82	✲	83	✳	84	✴	85	✵	86	✶	87	✷
88	✸	89	✹	90	✺	91	✻	92	✼	93	✽	94	✾	95	✿
96	❀	97	❁	98	❂	99	❃	100	❄	101	❅	102	❆	103	❇
104	❈	105	❉	106	❊	107	❋	108	●	109	❍	110	■	111	❑
112	❒	113	❑	114	❒	115	▲	116	▼	117	◆	118	❖	119	◗
120	|	121	❙	122	❚	123	❛	124	❜	125	❝	126	❞		
		161	❡	162	❢	163	❣	164	❤	165	❥	166	❦	167	❧
168	♣	169	♦	170	♥	171	♠	172	①	173	②	174	③	175	④
176	⑤	177	⑥	178	⑦	179	⑧	180	⑨	181	⑩	182	❶	183	❷
184	❸	185	❹	186	❺	187	❻	188	❼	189	❽	190	❾	191	❿
192	①	193	②	194	③	195	④	196	⑤	197	⑥	198	⑦	199	⑧
200	⑨	201	⑩	202	❶	203	❷	204	❸	205	❹	206	❺	207	❻
208	❼	209	❽	210	❾	211	❿	212	➔	213	→	214	↔	215	↕
216	➘	217	➙	218	➚	219	➛	220	➜	221	➝	222	➞	223	➟
224	➠	225	➡	226	➢	227	➢	228	➤	229	➥	230	➦	231	➧
232	➨	233	➩	234	➪	235	➫	236	➬	237	➭	238	➮	239	➯
		241	➱	242	➲	243	➳	244	➴	245	➵	246	➶	247	➷
248	➸	249	➹	250	➺	251	➻	252	➼	253	➽	254	➾		

Mais le plus appréciable avec ce package `enumitem`, c'est la possibilité de configurer globalement les liste dès le préambule (`\setlist`), voire d'en créer de nouvelles avec ces propres règles, puces, distances, etc. (`\newlist`).

Voici un rapide aperçu des possibilités désormais accessibles en quelques touches de clavier :

Des listes proprement personnalisables

```
% Ajout dans le préambule
%\usepackage{enumitem, pifont}
%\setlist[itemize, 1]{label =
    {--}, itemsep =
    \baselineskip}
%\setlist[enumerate, 1]{label =
    \arabic*), itemsep =
    \baselineskip}


J'ai envie de dire :
    \begin{itemize}
\item une chose ;

\item[\ding{118}] avec une puce
    ponctuelle ! \\
\end{itemize}

Je peux aussi énumérer :
    \begin{enumerate}[label =
    {\bfseries{}\'Etape \Alph*
    :}, leftmargin = *]
\item marcher ;

\item lire ;

\item écrire.
\end{enumerate}
```

J'ai envie de dire :

- une chose ;

❖ avec une puce ponctuelle !

Je peux aussi énumérer :

Étape A : marcher ;

Étape B : lire ;

Étape C : écrire.

OK pour toi ? Toujours d'attaque ? Finissons désormais ce chapitre sur la gestion du texte.

5.8 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 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 de base disponibles pour `nom_couleur` sont alors les suivantes :

- → `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 5.1 pour `nom_couleur` :

Apricot	*Emerald*	*OliveGreen*	*RubineRed*
Aquamarine	*ForestGreen*	*OrangeRed*	*Salmon*
Bittersweet	*Fuchsia*	*Orange*	*SeaGreen*
Black	*Goldenrod*	*Orchid*	*Sepia*
BlueGreen	*Gray*	*Peach*	*SkyBlue*
BlueViolet	*GreenYellow*	*Periwinkle*	*SpringGreen*
Blue	*Green*	*PineGreen*	*Tan*
BrickRed	*JungleGreen*	*Plum*	*TealBlue*
Brown	*Lavender*	*ProcessBlue*	*Thistle*
BurntOrange	*LimeGreen*	*Purple*	*Turquoise*
CadetBlue	*Magenta*	*RawSienna*	*VioletRed*
CarnationPink	*Mahogany*	*RedOrange*	*Violet*
Cerulean	*Maroon*	*RedViolet*	*White*
CornflowerBlue	*Melon*	*Red*	*WildStrawberry*
Cyan	*MidnightBlue*	*Rhodamine*	*YellowGreen*
Dandelion	*Mulberry*	*RoyalBlue*	*YellowOrange*
DarkOrchid	*NavyBlue*	*RoyalPurple*	*Yellow*

Figure 5.1 – 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}{def_couleur}
```

Le `modèle` correspond à `RGB` par exemple et `def_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`.

Tu peux aussi renseigner `nom_couleur` ou `def_couleur` par l'intermédiaire de mélanges de couleur. Pour ce faire, il faut utiliser la syntaxe `couleurA!x!couleurB`, pour $\texttt{x} \in [0\,;\,100]$, qui te permet de mélanger `x` % de `couleurA` et $(100 - \texttt{x})$ % de `couleurB`.

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 5.2 – Non, je ne suis pas un fan d'Evangelion !

Toujours des nôtres ? Si tu te sens prêt, nous allons pouvoir aborder un nouveau chapitre !

Chapitre 6

Les mathématiques sous LaTeX

Comme tu peux le constater, il y a fort à faire sous LaTeX. Les combinaisons sont déjà impressionnantes. Je te laisse maintenant découvrir la raison d'être de LaTeX, ce pourquoi il a été créé : écrire proprement des formules mathématiques !

Par la suite, pour alléger les exemples, le préambule ne sera plus renseigné dans les codes LaTeX mis à disposition. Ces derniers seront basés sur l'architecture du code minimal fourni ci-après. L'ajout de nouveaux packages sera signalé au début du code par un commentaire.

Le code minimal

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

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


\begin{document}

% Ecrire le code ici !

\end{document}
```

6.1 Le mode mathématiques

Bon, c'est bien beau de vouloir faire écrire des maths à LaTeX, 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

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

De même si j'écris \[x = 2 !\] %
    Pas de \\ car déjà un saut
    de ligne

On obtient la même chose avec  :

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

mais l'équation est numérotée !
    \\

Que donne $a b c d$ ?
```

Écrire x = 2 ! et $x = 2!$ ne donnent pas le même résultat !

De même si j'écris

$$x = 2!$$

On obtient la même chose avec :

$$x = 2! \tag{6.1}$$

mais l'équation est numérotée !

Que donne $abcd$?

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 !

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, LaTeX 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 et permette d'appeler les commandes que nous verrons par la suite, pour écrire des formules mathématiques (fraction, 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.

6.2 Vers les espaces insécables

Pour pouvoir librement écrire du texte dans le mode mathématiques, la commande `\text{texte_à_écrire}` est très utile. **C'est vraiment la commande la plus simple qui existe : à utiliser en priorité pour ce genre de situation !**

Mais, tu peux aussi avoir envie de jouer un peu sur l'espacement entre les différents symboles, si tu trouves qu'ils sont trop rapprochés. Il existe alors des commandes bien plus efficaces et pratiques que `\hspace{<distance>}` pour le mode mathématiques.

Ces commandes portent le nom d'**espaces insécables** – insécables car LaTeX ne peut y toucher et se plie à la volonté de l'utilisateur. **Ces espaces sont utilisables aussi bien dans le mode mathématiques que sur du texte normal.**

Ils permettent aussi de bien imposer l'espace souhaité et évite d'avoir un symbole ou un signe de ponctuation qui se balade seul en début de phrase. Nous pouvons relever :

- `\!` : espace très petit,
- `\,` : espace fin,
- `\:` : espace moyen,
- `~` (tilde) : espace normal,
- `\;` : espace large,
- `\quad` : espace très large,
- `\qquad` : espace encore plus large.

Toutefois, si j'utilisais initialement les espaces insécables à outrance, ils peuvent vite se révéler pénibles à écrire. Il faut donc généralement faire confiance à LaTeX 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 ta préférence :

$$\forall x \quad vs \quad \forall\, x$$

`$\forall x$` *vs* `$\forall\,x$`

Commande et espaces insécables

```
Nous obtenons donc $x + y = 3$
    et $y = 2$ donc : \[x = 1
    \text{ (obvious)}\]
% Présence d'un espace au début
    dans \text : séparation du
    texte de l'équation

% Utiliser \quad aussi possible
    : exemple d'utilisation
    assez fréquent
Nous obtenons alors : \[x = 1
    \quad \text{et} \quad y = 2\]
```

Nous obtenons donc $x + y = 3$ et $y = 2$ donc :

$$x = 1 \text{ (obvious)}$$

Nous obtenons alors :

$$x = 1 \quad \text{et} \quad y = 2$$

Si jamais tu veux appliquer un espace insécable de manière définitive sur une commande LaTeX, il existe des moyens de remplacer la définition initiale de la commande par la même avec l'espace insécable.

Ainsi, tu continuerais à écrire `$\forall x$` mais le résultat serait identique à `$\forall\,x$`. Il faut procéder de la manière suivante dans le préambule :

Changement de la définition d'une commande

```
% Renommer la commande initiale (sinon bug : boucle infinie)
\let\oldforall\forall
% Modification de la commande
\renewcommand{\forall}{\oldforall\,}
```

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

6.3 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. Commençons donc par un premier exemple :

Les premiers symboles mathématiques

```
% Ajout au préambule !
%\usepackage{amsmath, amsfonts,
    amssymb}


Indice : $i_2$ \\
% Encadrement avec des {}
$i_{13}$ différent de $i_13$ \\

Exposant : $i^3$ ou $i^{13}$ \\

Fraction : $\frac{x}{y}$ \\

Racine carrée : $\sqrt{13}$ \\
Racine énième : $\sqrt[n]{13}$ \\

Mix de formules (exemple) :
    $\sqrt{\frac{a}{b}}$
```

Indice : i_2
i_{13} différent de i_13

Exposant : i^3 ou i^{13}

Fraction : $\frac{x}{y}$

Racine carrée : $\sqrt{13}$
Racine énième : $\sqrt[n]{13}$

Mix de formules (exemple) : $\sqrt{\frac{a}{b}}$

Pardon ? Il n'y en a pas assez ? Ok, navré, poursuivons :

D'autres symboles mathématiques

```
Intégrale : $\int_0^{13}
    f(x)\,dx$ \\
% Attention aux bornes : les {}
    sont vite oubliées
Somme : $\sum_{i = 13}^n x^i$ \\

\'Equation : $x + y - z = 3
    \times t + f$ \\
% 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)
```

Intégrale : $\int_0^{13} f(x)\,dx$

Somme : $\sum_{i=13}^{n} x^i$

Équation : $x + y - z = 3 \times t + f$

$x < y$, $y \leq z$, $z \geqslant c$, $c > d$ mais $d \neq f$ alors que $f \simeq g$!

Ok pour toi ? Comment ? J'ai oublié de mentionner les lettres grecques ? Toutes mes excuses. Les voici :

Les lettres grecques

```
Les lettres grecques ? Facile :
    $\alpha$, $\beta$, $\mu$,
    etc. \\

En majuscules ? $\Omega$,
    $\Delta$, $\Lambda$, etc.
% Ne fonctionne pas pour toutes
    les majuscules : \Alpha
    entraîne une erreur
```

Les lettres grecques ? Facile : α, β, μ, etc.

En majuscules ? Ω, Δ, Λ, etc.

Si jamais tu souhaites connaître la liste exacte des commandes pour écrire

les lettres grecques, la voici :

Table 6.1 – La liste complète des lettres grecques sous LaTeX

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

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 le mérite d'être allé fouiner une nouvelle commande). Tout d'abord, `\mathrm` est une commande qui ne fonctionne qu'en mode mathématiques et qui permet de redresser le texte (enlever l'italique). Il faut donc bien écrire :

```
$\mathrm{\mu}$
```

Cependant, `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 Ω est déjà considérée comme droite. 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.

6.4 L'affichage et les délimiteurs

Tu l'as peut-être remarqué : écrire une somme doit donner un résultat un peu différent de ce que tu peux lire sur cette page. Il doit en aller de même si tu écris une intégrale ou un empilement de fraction :

$$\sum_k x^k \quad \int_0^x (ft)\,dt \quad \cfrac{a}{b}{c}{d}$$

Pour avoir un affichage "normal", il faut indiquer à LaTeX de forcer toutes les équations en mode mathématiques avec l'affichage `displaystyle`.

La commande `\everymath{\displaystyle}` *juste après* `\begin{document}` suffit donc pour avoir le même rendu que moi... sauf pour les fractions. Pour ces dernières, il faut utiliser la commande `\cfrac{x}{y}` ou `\dfrac{a}{b}` Dès lors, l'affichage de tes équations devrait être meilleur :

Forcer l'affichage

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts, amssymb}


\everymath{\displaystyle} % Commande indispensable !
```

```
Somme : $\sum_k x^k$ \\

Intégrale : $\int_0^x (ft)\,dt$ \\

Fractions : $\frac{\frac{a}{b}}{\frac{c}{d}} \neq
\cfrac{\cfrac{a}{b}}{\cfrac{c}{d}}$
```

Bien, maintenant que les choses sont correctement posées, tu peux avoir le meilleur rendu au monde mais LaTeX reste toujours extrêmement puissant, à condition de le lui dire.

En effet, écrire $(\frac{a}{b})$ et $\left(\frac{a}{b}\right)$ sont deux choses totalement différentes. LaTeX 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

Règle n° 1 : Un délimiteur n'existe qu'en mode mathématiques.

Règle n° 2 : **Un délimiteur entrant implique un délimiteur sortant.**

Règle n° 3 : Un délimiteur entrant est défini par la commande `\left` suivi du nom du délimiteur ; pour le sortant, de même avec `\right`.

Règle n° 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

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts,
    amssymb}


Parenthèses : $\left(
    \cfrac{a}{b} \right)$ \\
Crochets : $\left[ \cfrac{a}{b}
    \right]$ \\
Mix possible : $\left(
    \cfrac{a}{b} \right]$ \\
% Aucun problème tant que la
    règle 2 est respectée

Accolade à gauche : $\left\{
    \cfrac{a}{b} \right.$ \\
Accolade à droite : $\left.
    \cfrac{a}{b} \right\}$ \\

Bonus $\left\langle \left\lbrace
    \cfrac{a}{b} \right\rbrace
    \right\rangle$ \\
% \lbrace ou \rbrace équivalent à
     \{ ou \}
% Selon moi : \{ plus logique
```

Parenthèses : $\left(\cfrac{a}{b}\right)$

Crochets : $\left[\cfrac{a}{b}\right]$

Mix possible : $\left(\cfrac{a}{b}\right]$

Accolade à gauche : $\left\{\cfrac{a}{b}\right.$

Accolade à droite : $\left.\cfrac{a}{b}\right\}$

Bonus $\left\langle\left\{\cfrac{a}{b}\right\}\right\rangle$

Il faut aussi savoir que les délimiteurs sont parfois inutiles :

$$(a \times b) \quad vs \quad (a \times b)$$

`$(a \times b)$` *vs* `$\left( a \times b \right)$`

Les délimiteurs sont donc pratiques et intéressants à utiliser dès lors qu'il y a un "étage" dans l'équation. Autrement, mieux vaut les éviter, pour simplifier l'écriture des équations et réduire les erreurs.

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...

6.5 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. Je préfère donc bien les poser maintenant car nous en aurons besoin un peu plus loin dans ce guide :

Règles de base pour les matrices – Introduction aux tableaux

Règle n° 1 : Il faut considérer une matrice $n \times m$ comme un tableau vide à $n \times m$ cases.

Règle n° 2 : Une matrice est générée par l'environnement `pmatrix`.

Règle n° 3 : Les colonnes sont séparées par une esperluette "&" (touche [1] sous `Windows`).

Règle n° 4 : Le passage à la ligne suivante se fait grâce à `\\`.

De même, ne nous privons pas d'un petit exemple pour comprendre et digérer le tout :

Les matrices – 1ers exemples

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts,
    amssymb}


Matrice 2 x 2 : $\begin{pmatrix}
a & b \\ c & d
\end{pmatrix}$ \\

Matrice 2 x 4 : \[\begin{pmatrix}
a & b & c & d \\
e & f & g & h
\end{pmatrix}\]
```

Matrice 2 x 2 : $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Matrice 2 x 4 :

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \end{pmatrix}$$

Il restera toujours des cas un peu plus délicats à traiter :

Les matrices – Cas plus technique

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts,
    amssymb}


Matrice à trou :
    \[\begin{pmatrix}
1 & 1 & \cdots & 1 \\
1 & 2 & \cdots & 2 \\
\vdots & \vdots & \ddots &
    \vdots \\
1 & 2 & \cdots & n
\end{pmatrix}\]
```

Matrice à trou :

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \cdots & n \end{pmatrix}$$

Comme indiqué dans les règles, il faut donc utiliser le symbole `&` pour changer de colonne et la commande `\\` pour passer à la ligne suivante. Si l'espace entre les `&` est optionnel, il est quand même recommandé pour faciliter la relecture de ton code.

Surtout au début, pour des matrices plus complexes comme des matrices à trou, il ne faut pas hésiter à faire un dessin pour bien visualiser l'agencement des différents éléments de la matrice.

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

- `matrix` : aucun délimiteurs ;
- `pmatrix` : parenthèses ;
- `vmatrix` : barres verticales ;
- `Vmatrix` : doubles barres verticales ;
- `bmatrix` : crochets ;
- `Bmatrix` : 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

`vcol` de la manière suivante :

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

qui s'appelle de cette façon : `$\vcol{a}{b}{c}$`. Plus pratique, n'est-ce pas ? Je te garantis que c'est vrai, surtout que tu dois écrire un très grand nombre de fois un vecteur colonne sur la même page !

Bon, finissons-en avec les mathématiques sous LaTeX par la présentation et l'alignement des équations.

6.6 Aligner des équations

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

« Nous cherchons a tel que :

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

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 [1] :

$$\begin{cases} I &= [74,98 - 0,0428\,;\,74,98 + 0,0428] \\ n &= 20 \\ 1-\alpha &= 0,9 \end{cases} \quad . \text{ »}$$

Pour obtenir ce résultat avec des équations bien alignées, tu dois utiliser l'environnement `align` (ou `align*` pour éviter la numérotation de chaque ligne).

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

Ne pas utiliser `eqnarray` ! ! !

Après avoir vagabondé sur Internet et essayé différents rendus, je préfère utiliser `align`. Après d'autres recherches, Lars Madsen préconise lui aussi très fortement l'usage de `align` et recommande de bannir toute utilisation de `eqnarray` (un autre environnement pour aligner des équations) [a].

S'il y a donc un point à retenir : « **Avoid `eqnarray` !** » et utilise bien l'environnement `align`.

a. Article disponible sur : `http://www.tug.org/pracjourn/2006-4/madsen/madsen.pdf`.

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` s'emploie sans ! C'est parti pour un exemple. Reproduisons le cas présent en page 76 :

Aligner des équations – 1ère partie

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts,
    amssymb}


Nous cherchons $a$ tel que :
    \begin{align*}
P\,(\mu \in I) & = 1 - \alpha \\
        & = 0,9 \intertext{car
    l'énoncé indique que $1 -
    \alpha = 0,9$}
        & = P \left(
    \cfrac{\bar{X} - \mu}{S}
    \dots{} \right) \\
        & = 2 \mathcal{S}_{n -
    1} \left( \cfrac{a}{S}
    \dots{} \right)
\end{align*}
```

Nous cherchons a tel que :

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

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

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

Que relevons-nous de concret sur ce premier cas de figure ?

- l'environnement `align` (ou `align*`) utilise un `&` – **et un seul** – comme point de repère pour l'alignement. C'est pourquoi il est plutôt recommandé de le placer avant le signe `=` ;
- une nouvelle ligne est annoncée par un saut de ligne `\\`, comme pour une matrice ;
- la commande `\intertext{texte}` permet d'ajouter une remarque entre 2 équations. Notons au passage l'absence (volontaire) de saut de ligne (`\\`) : `\intertext` entraîne déjà un espacement suffisant ;
- plus anecdotique : la commande `\mathcal{texte}` permet de "transformer" les caractères (utilisation d'une autre police adaptée aux symboles mathématiques).

Aligner des équations – 2nde partie

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts,
    amssymb}


Nous pouvons donc conclure par :
    \[\left\{\begin{array}{rcl}
I & = & [74,98 \dots{} ] \\
n & = & 20 \\
1 - \alpha & = & 0,9
\end{array} \right.\]
```

Nous pouvons donc conclure par :

$$\left\{\begin{array}{rcl} I & = & [74, 98\dots] \\ n & = & 20 \\ 1-\alpha & = & 0,9 \end{array}\right.$$

Nous constatons que l'environnement `array` fonctionne de manière très similaire aux matrices. Il faut indiquer le nombre de colonnes via des lettres (`l`, `c` ou `r`).

Le nombre de lettres correspond au nombre de colonnes et le nom parle de lui-même pour positionner le contenu à l'intérieur de la colonne : `left`, `center` ou `right`.

L'utilisation d'un délimiteur est parfaitement envisageable (et recommandé) pour avoir l'accolade de taille variable à gauche. **L'usage du mode**

mathématiques devient dès lors obligatoire et justifie l'emploi de l'environnement `array` au lieu d'`align`.

Enfin, si jamais tu désires avoir une résolution d'équations avec un seul numéro global comme référence (ce que ne permet pas l'environnement `align`), tu peux procéder de la façon suivante :

Des équations – Un numéro

```
% Toujours dans le préambule
%\usepackage{amsmath, amsfonts,
    amssymb}


\begin{equation}
\begin{split}
x       & = y + z \\
        & = 13
\end{split}
\end{equation}
```

$$\begin{aligned} x &= y + z \\ &= 13 \end{aligned} \tag{6.2}$$

Il faut donc utiliser l'environnement `equation` pour passer en mode mathématiques avec un numéro pour l'équation, puis utiliser l'environnement `split` pour écrire tes équations bien alignées.

L'environnement `split` fonctionne de la même manière que l'environnement `align` (ou `align*`).

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...

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 7

Les tableaux et boîtes sous LaTeX

Synthétiser l'information n'est pas toujours évident. Et pourtant, un bon tableau suffit parfois à véhiculer un message ou à lister des éléments. Voyons comment en créer sous LaTeX.

Par la suite, pour alléger les exemples, le préambule ne sera plus renseigné dans les codes LaTeX mis à disposition. Ces derniers seront basés sur l'architecture du code minimal fourni ci-après. L'ajout de nouveaux packages sera signalé au début du code par un commentaire.

Le code minimal

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

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


\begin{document}

% Ecrire le code ici !

\end{document}
```

7.1 Conventions

C'est le passage un peu rébarbatif du guide mais qu'il faut rappeler si tu t'intéresses un peu à la mise en forme de documents. Les conventions que je vais énoncer ci-après proviennent à l'origine du guide du package `booktabs`[1], qui permet d'améliorer la qualité des tableaux sous LaTeX.

Les conventions pour rédiger des tableaux

«

Tu ne feras pas de graves erreurs si tu te rappelles à tout moment de deux simples commandements :

1) Ne jamais, au grand jamais, utiliser de filets verticaux.

2) Ne jamais utiliser de filets doubles.

Ces commandements peuvent sembler extrêmes mais en des années d'expérience je n'ai jamais trouvé un bon argument pour passer outre.

Par exemple, si tu estimes que les informations dans la moitié gauche d'une table sont si différentes de celles de la droite qu'il faut les séparer par une ligne verticale, alors tu devrais plutôt utiliser deux tables.

Le second commandement est très, très occasionnellement violé. [...]

Il y a trois autres conseils que je pourrai citer ici car ils sont si peu connus en dehors des cercles des typographes et éditeurs professionnels :

1) Place les unités dans l'en-tête de la colonne (pas dans le corps de la table).

2) Fais toujours précéder un point décimal (une virgule décimale en français) par un chiffre ; donc 0.1 (ou 0,1) et pas simplement .1 (,1).

3) N'utilises pas de signes « ditto » ou toute convention analogue pour répéter une valeur précédente. Dans la plupart des cas, un

1. Disponible sur : `https://www.ctan.org/pkg/booktabs`.

blanc fait aussi bien l'affaire. Sinon, répètes la valeur.

Est-ce que c'est moi qui suis pédant ? Ces derniers conseils sont de plus en plus souvent ignorés dans les travaux publiés. Pour moi, ceci montre simplement que la typographie est celle d'un amateur. »

Guide du package `booktabs`
`https://www.ctan.org/pkg/booktabs`

Maintenant que les conventions sont posées, voyons désormais comment créer un tableau.

7.2 Création de tableaux

Nativement, tous les éléments sont disponibles sous LaTeX pour créer des tableaux extrêmement simples. Pour ce faire, il faut utiliser l'environnement `tabular`.

Si tu te souviens bien de la construction de l'environnement `array` (tableaux en mode mathématiques), tu vas vite te rendre compte que le fonctionnement de base de l'environnement `tabular` (tableaux en mode texte) est identique. Un petit exemple, comme toujours :

Premiers tableaux

```
\begin{tabular}{cc}
Centrage & Ici aussi \\
Ok ! & $\alpha = 13$
\end{tabular} \\ \\

\begin{tabular}{ll} \hline
Tableau & simple \\ \hline
sous & \LaTeX{} \\ \hline
\end{tabular}
```

Centrage	Ici aussi
Ok !	$\alpha = 13$

Tableau	simple
sous	LaTeX

Comme tu peux le constater, je ne t'ai pas menti : la construction est rigoureusement identique à celle de l'environnement `array`. La seule diffé-

rence ? Comme il s'agit d'un tableau en mode texte, il est tout à fait licite d'introduire un mode mathématiques local (avec des `$`) pour écrire des mathématiques dans une cellule du tableau.

Pour la séparation avec un filet [2] horizontal, il faut donc appeler la commande `\hline` **après** un saut de ligne (hormis au début du tableau).

Une question ?

« Par rapport à ton exemple, mes tableaux sont plus resserrés. Pourquoi n'avons-nous pas le même résultat ? »

Question très pertinente : j'ai en effet une corde supplémentaire à mon arc. J'ai indiqué dans le **préambule** de ce guide une commande qui impacte tous mes tableaux et permet de les aérer un peu plus. Cette commande est la suivante :

```
\renewcommand{\arraystretch}{1.3}
```

Elle permet d'agrandir la hauteur minimale d'une ligne, ce qui permet d'aérer les tableaux. Le coefficient de `1.3` est un choix personnel : libre à toi de le modifier à ta convenance.

La petite astuce

Si jamais tu as un "grand nombre" de colonnes à déclarer lors de la création de ton tableau, il existe un petit raccourci.

Au lieu d'écrire `c...c` (`N` fois), tu peux écrire `*{N}{|c}|`. Ainsi, tu crées `N` colonnes centrées. Pratique, non ?

Si tu veux des options plus poussées sur les tableaux (fusion de cellules, remplissage, mise en gras d'une colonne entière...), je te renvoie à la 3ème partie de ce guide où tu peux trouver des réponses. Internet peut aussi t'aider si besoin.

Sache encore que, dès l'instant où tu arpentes le chemin d'une personnalisation très poussée et sophistiquée, tu risques de perdre beaucoup de temps à faire en sorte que le code LaTeX fonctionne. Avec les éléments de base que je viens de te présenter, j'estime que tu peux déjà faire 70 % des tableaux nécessaires.

Pour les 30 % restants, à titre indicatif et si tu es curieux, tu peux te tour-

2. Terme consacré apparemment : c'est l'équivalent d'un "trait".

ner vers les packages suivants : `array`, `booktabs`, `longtable` et `multirow` (fusion de lignes ; fusion de colonnes possible de base avec la commande `\multicolumn`).

Maintenant que les éléments de base ont été présentés, passons à un autre élément important.

7.3 Insérer une légende

Avoir un tableau, c'est bien. Avoir un tableau avec une légende, c'est mieux. Et avoir une légende avec une numérotation automatique, c'est encore mieux ! Fort heureusement, LaTeX propose tous ces éléments nativement.

Je ne vais pas rentrer dans les détails du concept, que je développe plus amplement dans le prochain chapitre. Sache juste, pour commencer, qu'il te faut procéder de la manière suivante :

1) Insertion du tableau dans un environnement `table`.

2) Insertion de la légende avant ou après le tableau (au choix) grâce à la commande `\caption{Légende}`.

Un exemple minimaliste serait alors le suivant :

Tableau & légende

```
\begin{table}
\centering
\caption{Légende du tableau}
\begin{tabular}{ccc}
Tableau & de & test \\ \hline
sous & \LaTeX{}
\end{tabular}
\end{table}
```

Passons maintenant à un autre élément disponible sous LaTeX pour faire un peu de mise en forme sans nécessairement passer par un tableau : les boîtes.

7.4 Les boîtes

La théorie

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 LaTeX. En l'occurrence, parlons des boîtes.

Sous LaTeX, tout tient dans une boîte : les lettres, les paragraphes, les tableaux, les images, les équations... Bref, tout ! Concrètement, une *box* (boîte) est le terme technique en LaTeX pour un contenant invisible qui peut contenir soit un élément visible, soit une autre boîte, soit rien du tout. Ensuite, chaque boîte est connecté grâce à de la *glue* (colle), qui détermine la séparation entre les boîtes.

Dans un document traditionnel, les "lettres-boîtes" sont donc collées à d'autres pour former des mots, eux-mêmes collés élastiquement à d'autres mots pour former des phrases. Ces phrases sont découpées en lignes et placées dans un paragraphe (boîte encore une fois), écarté ou collé à d'autres paragraphes de manière élastique là encore, cette fois pour former des pages suffisamment aérées et remplies.

C'est donc ainsi que LaTeX construit un document et les pages qui le compose, en collant les boîtes ensembles et grâce aux règles de base (natives) et à celles définies par l'utilisateur.

Concrètement, une boîte ressemble à :

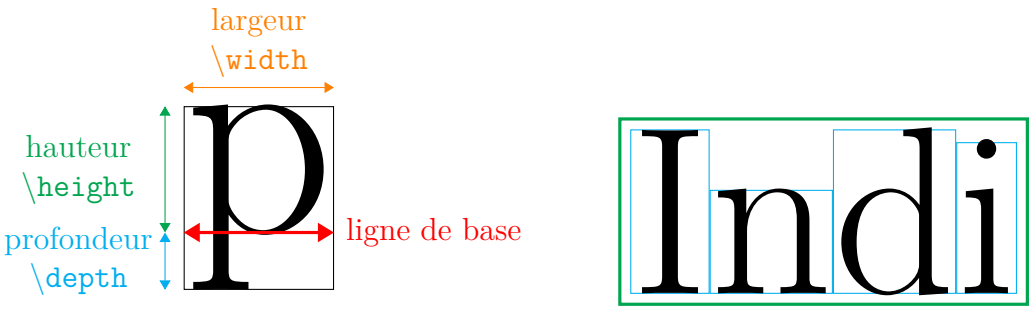


Figure 7.1 – Schéma d'une boîte et de ses composants

Les différentes dimensions de la boîte sont contenues au sein de 4 variables : `\width` pour la largeur de la boîte ; `\height` pour sa hauteur ; `\depth` pour sa profondeur ; et `\totalheight` pour sa hauteur totale soit `\height` + `\depth`.

Ces variables ne peuvent être utilisées que pour des boîtes. Passons à un peu de pratique pour voir comment appeler les boîtes en question et se servir de ces dimensions qui leur sont propres.

La pratique

Tout d'abord, les `framebox` constituent la base en LaTeX. La commande est assez simple :

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

avec les choix suivants pour `pos`[3] :

- `l` pour aligner `texte` sur la gauche de la boîte ;
- `r` pour aligner `texte` sur la droite de la boîte ;
- `s` pour aligner `texte` sur toute la longueur de la boîte.

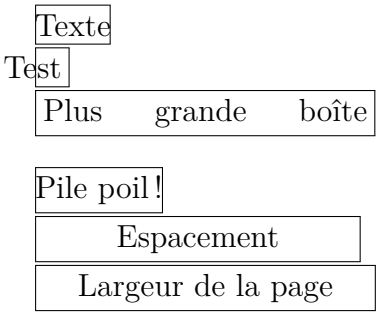
Si `texte` fait référence à l'objet à encadrer (texte, image, formule...), `largeur` fait référence à la largeur de la boîte. Tu peux renseigner une unité de distance (`13pt`, `215mm`, etc.).

Naturellement, si tu renseignes `1cm` alors que le texte en fait 2, le résultat risque de ne pas être satisfaisant. Les dimensions propres aux boîtes peuvent alors être utilisées, comme `\width` :

Utilisation des `framebox`

```
\framebox[1cm]{Texte} \\
\framebox[13pt][r]{Test} \\
\framebox[45mm][s]{Plus grande
   boîte} \\

\framebox[\width]{Pile poil !} \\
\framebox[2\width]{Espacement} \\
\framebox[\linewidth]{Largeur de
   la page}
```



Comme démontré avec le dernier cas de figure, la longueur `\linewidth`, présente nativement sous LaTeX, peut être utilisée pour créer une boîte de

3. Par défaut, si rien n'est indiqué pour `pos`, le texte est centré à l'intérieur de la boîte.

largeur égale à celle de la page.[4]

Il existe un raccourci pour appeler plus simplement une `framebox` avec la commande `\fbox`. Aucune option n'a besoin d'être indiqué, juste la partie `texte`. La taille de la boîte s'adapte alors au contenu :

Le petit raccourci sympathique

```
\fbox{Un peu de texte} \\
\fbox{Un peu plus de texte} \\

\fbox{Des maths : $i =
    \sqrt{169}$}
```

Un peu de texte
Un peu plus de texte

Des maths : $i = \sqrt{169}$

De l'utilisation des `fbox`

Les `fbox` sont très pratiques, pour comprendre comment un paragraphe ou une image est agencé, ainsi que la taille qu'il occupe.

Lors de montages ou de nouvelles créations, il peut se révéler très utile d'encadrer les différents éléments grâce à des `fbox` pour comprendre comment LaTeX les agence et pouvoir apporter les correctifs nécessaires afin d'avoir le résultat souhaité !

Tu trouveras aussi dans la littérature les `makebox`, dont l'appel est rigoureusement identique à une `framebox`. Il s'agit tout simplement d'une `framebox` sans cadre, ce qui ne présente que peu d'intérêt selon moi.

Techniquement, la `framebox` est construite à partir d'une `makebox` mais j'ai trouvé plus judicieux de présenter directement la première. Bien, terminons avec une autre boîte bien utile.

La plus utile

La boîte qui se révèle bien utile pour faire quelques montages reste la `parbox` et l'environnement qui lui est associé : la `minipage`. Sa syntaxe est la suivante :

```
\parbox[ext][hauteur][int]{largeur}{texte}
```

4. La notion de « longueur » sous LaTeX est abordée plus amplement en page 94.

avec `hauteur` et `largeur` respectivement la hauteur et la largeur de la boîte (distance manuelle comme `13mm` ou une longueur propre aux boîtes comme `\width` ou une longueur LaTeX comme `\linewidth`).

Fonctionnement de base d'une parbox

```
\parbox{13mm}{Texte} \\

\fbox{ % Intérêt de la fbox !
\parbox{\linewidth-2cm}{Partie A
   \\ Partie B}
}
```

Texte

Partie A
Partie B

Ensuite, `ext` correspond à l'alignement externe de la `parbox` par rapport à la ligne de base[5], avec les choix suivants :

- `m` (par défaut) ou si aucune option n'est donnée pour centrer la boîte sur la ligne de base ;
- `b` pour aligner le bas (*bottom*) de la boîte sur la ligne de base ;
- `t` pour aligner le haut (*top*) de la boîte sur la ligne de base.

Alignement externe

```
A : \parbox[b]{2cm}{Par. 1 \\ Par. 2} \hfill
B : \parbox{2cm}{Par. 3 \\ Par. 4} \hfill
C : \parbox[t]{2cm}{Par. 5 \\ Par. 6}
```

Par. 1
A : Par. 2 B : Par. 3
Par. 4 C : Par. 5
Par. 6

Enfin, `int` désigne l'alignement interne de la boîte, pour pouvoir positionner verticalement le texte dans la boîte, **sous réserve qu'une hauteur ait été indiquée**. Elle peut prendre quatre valeurs :

- `b` pour repousser le texte vers le bas de la boîte ;

5. Ligne sur laquelle reposent les lettres.

- `t` pour situer le texte en haut de la boîte ;
- `c` (par défaut) ou si aucune option n'est donnée pour centrer verticalement le texte ;
- `s` pour étirer verticalement le texte.

Toutefois, comme tu as pu le constater avec mes exemples, il n'est pas évident de gérer plusieurs paragraphes dans une `parbox`. Il est même impossible d'y introduire d'autres environnements !

Il faut donc inclure le tout dans l'environnement équivalent : `minipage`. L'appel à cet environnement se fait de la manière suivante, avec des paramètres identiques à ceux d'une `parbox`, mais dans un ordre différent :

```
\begin{minipage}[ext][hauteur][int]{largeur}
texte
\end{minipage}
```

Il convient donc de définir correctement une `minipage` si tu veux éviter les erreurs. Dès l'instant où tu renseignes une des options non obligatoires (`ext`, `hauteur` ou `int`), il faut toutes les indiquer ou le rendu ne sera pas conforme :

Appel de `minipage`

```
\begin{minipage}{0.8\linewidth}
Texte avec un \\
retour à la ligne !
\end{minipage} \\ \\

\fbox{
\begin{minipage}[m][1cm][b]{2cm}
Lorem
\end{minipage}
} \& ipsum
```

Texte avec un
retour à la ligne !

& ipsum

De l'utilisation des `minipage`

Il ne faut pas utiliser une `minipage` pour simplement avoir un texte sur 2 colonnes !

D'abord, le résultat ne correspondra pas à tes attentes, ne sera pas esthétique et sera difficile à gérer. Ensuite, LaTeX met à disposition l'option `twocolumn` lors de la définition de la classe.

Une `minipage` sert donc exclusivement pour des montages, par exemple une image à côté d'une autre image ou d'un texte, comme nous aurons l'occasion de le voir par la suite.
Enfin, il faut savoir qu'il n'y a pas d'alinéa dans une `minipage` : dans sa définition, par défaut, l'indentation est nulle.

Le petit bonus

Si tu veux continuer à arpenter le chemin des boîtes et avoir encore plus de personnalisation, je te recommande le package `fancybox`. Il permet, entre autres, d'ajouter du surlignage et de l'ombrage aux boîtes.

Mais il existe un autre package bien plus puissant...

7.5 Le Saint-Graal des boîtes

Découvert lors de la rédaction de la première version de ce guide (été 2016), le package `tcolorbox` est extrêmement complet et permet une personnalisation totale des boîtes. Tous les encadrés que tu as pu rencontrer jusqu'à présent dans ce guide sont générés grâce à ce package !

Si tu fouines un peu sur Internet, tu devrais trouver la documentation officielle... allez, je suis gentil, je te donne le lien : `http://fr.lmgtfy.com/?q=tcolorbox+help`.

C'est actuellement 500 pages complexes mais qui assez illustrées, pour te permettre donc de réaliser des boîtes aussi jolies que celles présentes dans ce guide et bien plus. Beaucoup plus !

Pour te donner un premier aperçu, voici un exemple extrêmement simple :

Première utilisation de `tcolorbox`

```
% Ajout dans le préambule
%\usepackage{tcolorbox}

\begin{tcolorbox}[colframe =
    orange, colback = orange!50,
    boxrule = 2pt, arc = 6pt,
    title = {Un titre}, coltitle
    = black]
J'adore ce package ! \\
De toute mon âme !
\end{tcolorbox}
```

Un titre

J'adore ce package !
De toute mon âme !

Et il ne s'agit que la partie émergée de l'iceberg ! La documentation officielle décrit toutes les options disponibles, les différentes boîtesmise à disposition, la création d'environnement pour appeler ses propres boîtes...

Mais je crois m'être légèrement emporté. Je reviens sur ce package plus en détail dans la partie suivante. Le but de cette partie reste de te présenter les bases sous LaTeX.

Passons désormais 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 : comment insérer une image !

Chapitre 8

Insérer des images

Une bonne image suffit des fois à remplacer 13 lignes de texte. Découvrons quelques spécificités à leur sujet et comment en insérer sous LaTeX.

Par la suite, pour alléger les exemples, le préambule ne sera plus renseigné dans les codes LaTeX mis à disposition. Ces derniers seront basés sur l'architecture du code minimal fourni ci-après. L'ajout de nouveaux packages sera signalé au début du code par un commentaire.

Le code minimal

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

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


\begin{document}

% Ecrire le code ici !

\end{document}
```

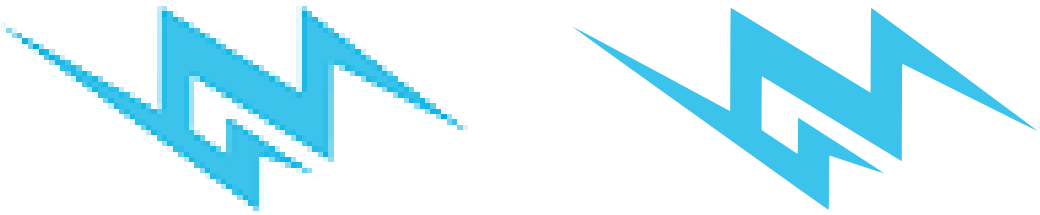
8.1 Les formats d'images

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*). Les secondes sont les plus propres et utilisent des extensions comme `.svg` (*Scalable Vector Graphics*) ou `.eps` (*Encapsulated PostScript*).[1]

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.

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[2]. Et inversement, si l'image est grande de base, aucun pixel ne sera donc visible.



Cachez cette image matricielle (à gauche) que je ne saurais voir !

Un exemple plus courant d'images vectorielles

Dans un fichier PDF, tu trouves en réalité des images vectorielles partout. En effet, le texte affiché utilise une police spécifique, définie vectoriellement.

Et heureusement ! Quel enfer serait sinon la lecture si tout le texte était flou voire illisible faute d'avoir suffisamment de pixels.

1. La création, édition et visualisation des fichiers `.svg` sont possibles grâce à des logiciels spécialisés, comme `Inkscape`. Le format `.eps`, moins connu et un peu délaissé de nos jours, peut être visualisé simplement grâce à des logiciels comme `EPS Viewer`.
2. J'ai l'impression de faire de la pub' pour l'Oréal...

Le format `.eps` fait un peu vieux jeu et reste surtout utilisé dans le domaine scientifique. Cependant, même s'il difficile à modifier avec des outils standards (comme `Paint`), il est plus facile à implanter sous LaTeX que le format `.svg`, pour un résultat identique.

Conversion au format `.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 reste une image matricielle.

Le format `.eps` ne garantit pas automatiquement une image vectorielle derrière. Il n'y a pas non plus de transformation miraculeuse en arrière-plan. 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 [a], sauvegarder le résultat au format `.svg` (sécurité) puis enregistrer cette image vectorielle au format `.eps`.

a. Cette méthode fonctionne parfaitement pour des formes simples, avec peu de variations de couleur. Le résultat est à travailler pour des images plus complexes, voire à créer directement au format vectoriel.

8.2 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 LaTeX.

Sous 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 [3].

Dès lors qu'une commande requiert une longueur en paramètre d'entrée, nous l'indiquons très clairement. Par exemple, `\vspace{13mm}`. Cependant,

3. Hauteur d'un « `x` » : cette unité de longueur dépend donc de la police utilisée.

LaTeX permet d'aller plus loin, beaucoup plus loin en mettant des longueurs prédéfinies sous forme de commande.

Quelques longueurs sous LaTeX

LaTeX utilise des longueurs nativement, dans chaque nouveau document. Par exemple, à chaque nouveau paragraphe, LaTeX met un alinéa. La taille de cet alinéa est une longueur définie par défaut et LaTeX utilise sa valeur.

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. C'est aussi ce qui garanti l'homogénéité (ou la cohérence, si tu préfères) d'un document réalisé avec LaTeX.

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

→ `\linewidth` : longueur qui correspond à la largeur "locale" du texte (vis-à-vis de la page, dans un tableau, dans une boîte, etc.) ;

→ `\baselineskip` : longueur qui correspond à un saut de ligne sous LaTeX.

! **`\linewidth` *vs* `\textwidth`**

Tu trouveras des fois dans la littérature ou dans des exemples sur Internet des gens qui emploie la longueur `\textwidth`. À première vue, lors de son utilisation, elle présente peu de différences avec `\linewidth`. Et pourtant, il y a bel et bien une différence ! [a]

❖ `\textwidth` représente la largeur d'un bloc de texte (valeur constante, globale) ;

❖ `\linewidth` représente la largeur locale du texte, que ce dernier soit présent dans une colonne, un tableau, une liste ou une `minipage`.

En règle générale, il vaut mieux utiliser `\linewidth` pour spécifier la taille relative d'une image ou d'une boîte. Cette longueur s'adapte

mieux à la situation et aux potentiels montages que tu peux réaliser (avec des `minipage`, par exemple).

a. Les points à venir ont été extraits et traduits de la page suivante : `https://tex.stackexchange.com/questions/16942/difference-between-textwidth-linewidth-and-hsize`.

Enfin, il peut être intéressant de 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 de longueurs, longueurs définies par défaut, etc.), je te recommande d'aller lire la page suivante : `http://en.wikibooks.org/wiki/LaTeX/Lengths`.

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

8.3 Insérer une image

La commande de base

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 LaTeX est possible. Il faut au préalable charger le package `graphicx`[4]. Pour insérer une image, c'est très simple. Il faut utiliser la commande suivante, à l'endroit où tu souhaites afficher ton image :

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

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`[5], puis renomme-là `fond`. De cette manière, tu auras moins de souci avec le code qui suit. Place cette image dans le même dossier que le fichier `.tex` avec lequel tu travailles.

4. Ne pas confondre avec le package de base `graphics` dont `graphicx` (avec un « x » donc) est une version améliorée !

5. Si tu ne connais pas le format de ton image, `clic droit` puis `Propriétés`.

Si jamais tu te trompes de format d'images ou que l'image n'est pas dans le dossier, LaTeX va te renvoyer un message d'erreur, du genre « File `nom_img.format` not found ».

Première insertion d'images

```
% Ajout dans le préambule !!!
%\usepackage{graphicx}

\includegraphics{fond.jpg}
```

Bon, si l'utilisation de la commande de base est aussi simple, tu conviendras que ce n'est pas très pratique avec une image très grande et qui déborde pas mal du document ! Voyons donc maintenant comment judicieusement utiliser les longueurs pour avoir un affichage convenable.

Utilisation des longueurs

Pour ajuster la taille d'une image, **2 options *utiles*** sont disponibles avec la commande `\includegraphics` :

1) `width = <distance>` : forcer la largeur de l'image à `<distance>`. Cette option se révèle salvatrice combinée avec la longueur `linewidth`.

2) `height = <distance>` : forcer la hauteur de l'image à `<distance>`. Essentiellement utile pour des images dont le format « portrait » est très prononcé, ou si tu veux remplir intégralement la page.

Et c'est tout ce qu'il faut savoir ! Il existe bien une autre option comme

`scale` mais sans intérêt car la valeur à utiliser dépend de la taille de l'image.

Avec l'option `width`, peu importe la taille de ton image, elle sera toujours bien insérée dans ton document. Bien entendu, si ton image reste petite et matricielle, elle risque d'être floue à l'affichage. Autrement, tu n'as plus à te soucier de retraiter tes images pour les avoir à une taille appropriée.

Une image bien taillée

```
% Ajout dans le préambule
%\usepackage{graphicx}


\includegraphics[width =
    \linewidth]{fond.jpg}

\begin{center}
\includegraphics[height =
    0.25\linewidth]{fond.jpg}
\end{center}
```

Comme afficher ci-dessus, tu peux centrer ton image avec un environnement `center`. La commande `\centering` fonctionne aussi et va se révéler utile par la suite.

C'est déjà mieux, non ? Pardon ? Tu voudrais aussi une magnifique légende pour accompagner ton image ? Ta demande est légitime !

Légende et environnement flottant

Tout comme pour les tableaux, l'insertion d'une légende à une image demande de placer celle-ci dans un *environnement flottant*. Il s'agit d'une obligation sous LaTeX pour garantir la qualité du document. Tu conviendras que le rendu ne serait pas très esthétique si l'image était en bas de page et la légende à la page suivante faute de place.

Visuellement, nous pouvons considérer l'environnement flottant comme une boîte qui va englober ton image et ta légende et dont la position est

variable, selon la place restante sur ta page :

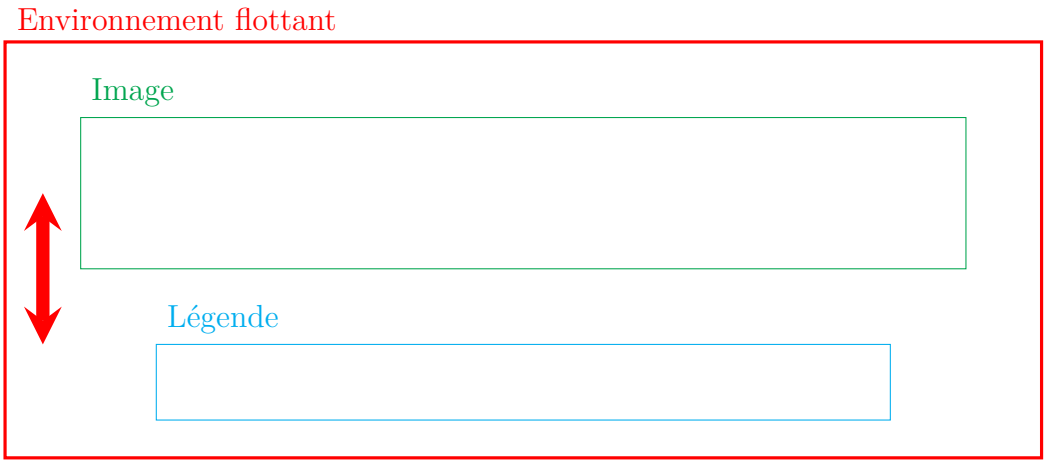


Figure 8.1 – Schématisation d'un environnement flottant

L'environnement flottant pour les images s'appelle `figure`. Et comme LaTeX fait bien les choses, il met à ta disposition différentes options pour positionner correctement cet environnement :

→ `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 LaTeX d'être un peu capricieux et l'option "`!`" devant la lettre lui indique que l'utilisateur a raison. Si tu utilises donc l'option `!h`, LaTeX fait tout son possible pour placer l'environnement flottant là où il est placé dans le code.

Ainsi, 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.

Enfin, pour placer une légende, la commande `\caption{légende}` est toujours d'actualité pour les images et se place donc à l'intérieur de l'environnement flottant `figure`.

Environnement flottant (`figure`) & légende (`\caption`)

```
% Ajout dans le préambule
%\usepackage{graphicx}


% La base
\begin{figure}
\includegraphics[width = 0.5\linewidth]{fond.jpg}
\caption{Une première légende}
\end{figure}

% !h + centrage
\begin{figure}[!h]
\centering
\includegraphics[width = 0.5\linewidth]{fond.jpg}
\caption{Une autre légende}
\end{figure}
```

Si je rencontre quelques soucis pour afficher le résultat du code dans cette exemple, tu dois constater que la numérotation de la légende est automatisée par LaTeX. Parfait, une tâche à laquelle nous n'aurons pas à nous soucier !

Toutefois, certaines complications peuvent parfois se produire avec cette option `!h`. C'est pourquoi la **solution ultime** – que j'utilise constamment – consiste en l'utilisation du package `float` et de renseigner un `H` à la place de `!h`. L'image est *vraiment* contrainte d'être à cet endroit.

Et s'il n'y a pas la place, LaTeX laisse un blanc, ce qui laisse parfois un gros trou dans ton document... Difficile d'arriver à tout concilier !

La meilleure solution (selon moi)

```
% Ajout dans le préambule
%\usepackage{graphicx, float}


% Package float
\begin{figure}[H]
\centering
\includegraphics[width =
    0.5\linewidth]{fond.jpg}
\caption{Avec une légende !}
\end{figure}
```

Figure 8.2 – Avec une légende !

Disposition des images & marges

Quand une image ne rentre pas en bas d'une page, il peut être tentant dans un premier temps d'augmenter les marges du document pour laisser plus de place à l'image. **C'est exactement ce qu'il *ne faut pas* faire !**

Tout d'abord, augmenter les marges permet en effet de gagner quelques lignes de texte et donc de gagner la place attendue pour placer l'image. Mais la longueur `\linewidth` est aussi augmentée en conséquence donc ton image est plus grande ! Ce n'est donc pas une solution.

De plus, changer les marges bouscule toute la structure et l'agencement de ton document, les blancs laissés par les images qui ne rentrent pas en bas de page (si choix de l'option `H` avec le package `float`). Tu risques donc de perdre un temps considérable à tout réajuster à chaque fois.

Je recommande donc de procéder de la manière suivante :

1) Régler les marges à la création du document (marges natives inchangées, choix personnel ou consigne de mise en page du rapport).

2) Rédiger ton document, inclure les images.

3) Revenir sur ton document, réagencer les images, les réduire, faire des montages, etc. pour limiter les blancs et avoir le meilleur rendu (subjectif).
   Tu peux aussi t'occuper de cette dernière étape chapitre par chapitre par exemple (changement de page par défaut entre deux chapitres).

Bien abordons désormais un dernier point capital : la gestion des images.

Bien ranger ses images

Si jamais tu as beaucoup d'images dans ton rapport, tu peux vite noyer le dossier de travail où se trouve 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/}}
```

Fais attention à bien placer cette commande **après** le package `graphicx`, car il s'agit d'une commande de ce même package.

Grâce à cette commande au nom assez explicite, tu indiques à LaTeX le répertoire/dossier où tu as rangé tes images.[6] Tu n'es pas limité à un seul chemin, tu peux en indiquer autant que nécessaire si besoin.

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.

6. En informatique, la "commande" `./` fait référence au dossier où se trouve le fichier avec lequel tu travailles. Pour revenir au dossier parent, il faut utiliser `../`.

Bien, voyons maintenant comment faire référence à une image.

8.4 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 ? Par exemple, comment écrire **automatiquement** « cf. l'image n°x page y » ?

Le but est bel et bien d'avoir une numérotation automatique : c'est bien plus pratique et moins fatiguant que de devoir corriger tout ton document à la main (et même impossible et impensable sur un rapport de plusieurs centaines de pages).

Naturellement, LaTeX propose nativement une solution, ou je n'aborderais pas le sujet. Donc pas de nouveaux packages pour cette fois !

Si tu veux créer une référence, il faut procéder en 2 étapes :

1) Création de la référence avec la commande `\label{nom-ref}`. Cette commande est à placer **après** une légende par exemple (`\caption` pour rappel).

   Tu peux aussi l'utiliser dans un environnement mathématiques (si tu veux faire référence à une équation ou dans un paragraphe (pour renvoyer à un bout de texte en particulier).

2) Appel de la référence avec la commande `\ref{nom-ref}` (numéro de la légende, de l'équation ou section dans lequel se situe le texte).

La commande `\pageref{nom-ref}` est disponible nativement (appel du numéro de page où se situe la référence et donc l'objet référence), tandis que les commandes `\nameref` et `\autoref` sont présentes avec le package `hyperref`.

Et si tu veux encore d'autres fonctionnalités, il paraît que le package `cleveref` est LA solution. Je dois encore le tester donc je ne vais pas m'épancher sur le sujet.

Voici un petit exemple pour mieux comprendre le fonctionnement des références :

Faire référence à une image

```
% Ajout dans le préambule
%\usepackage{graphicx, float}


\begin{figure}[H]
\centering
\includegraphics[width =
    0.5\linewidth]{fond.jpg}
\caption{Légende}
\label{exemple-ref-img}
\end{figure}

Mon image est la \figurename{}
    \ref{exemple-ref-img},
    située en page
    \pageref{exemple-ref-img}. \\

Avec le package \verb?hyperref?
    : \nameref{exemple-ref-img}
    \& \autoref{exemple-ref-img}.
```

Figure 8.3 – Légende

Mon image est la Figure 8.3, située en page 104.

Avec le package `hyperref` : Légende & Figure 8.3.

Comme tu peux le constater, les références se mettent à jour automatiquement. Cette fonctionnalité est très puissante et extrêmement pratique : tu n'as plus à te soucier de devoir tout mettre à jour manuellement à chaque ajout d'une image. LaTeX a tout en mémoire et l'adapte si besoin.

Pour information, le fonctionnement est similaire pour les formules mais l'appel de la référence se fait avec la commande `\eqref{nom-ref}`.

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 part à la première compilation et ne s'en sert que lors de la seconde.

Il faut donc juste compiler deux fois pour afficher correctement les références ou les mettre à jour..

Allez, un peu de courage. Tu touches presque à la fin de ce guide. Tu pourras alors être autonome sous LaTeX, et taper de magnifiques rapports.

Et la suite n'est pas compliquée : c'est du code pour faire des montages d'images et t'éviter de chercher pendant des heures comme j'ai eu à le faire !

8.5 Un peu de montage

Je suis sûr que, si tu n'y penses pas maintenant, tu souhaiteras à l'avenir faire quelques montages avec des images. Par exemple, placer 2-3 images l'une à côté de l'autre ou une image avec du texte autour.

Pour ce dernier cas (image avec du texte autour), il existe des solutions, 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.

Je m'arrache toujours les cheveux à chaque fois que je l'utilise car je trouve que le rendu n'est jamais à la hauteur et beaucoup de problèmes se posent dès qu'une légende est ajoutée à l'image. Je ne fournirai donc pas un exemple ici.

Heureusement, il existe d'autres solutions plus simples comme les `minipage`. Si jamais tu as besoin de te remémorer le fonctionnement des `minipage`, je te renvoie à la page 89. Sinon, pour aligner côte à côte deux images, il y a déjà une règle absolument primordiale, ou la compilation ne donnera pas le résultat espéré :

NE PAS laisser une seule ligne blanche !

Ensuite, la petite recette de cuisine avec les `minipage` fonctionne de la manière suivante :

1) Création d'une 1ère `minipage` de largeur X_1`\linewidth`, avec $X_1 \in]0\,;\,1[$.

2) Insertion classique de l'image avec la commande `\includegraphics` et l'option `width`. Utiliser la longueur `\linewidth` (ou une valeur réduite) devient très pratique dans cette situation.

Par exemple, une image de largeur `0.6\linewidth` dans une `minipage` de largeur `0.45\linewidth` (par rapport à la page ici) sera de largeur totale `0.27\linewidth`, par rapport à la page du coup.

3) Séparation (espace blanc) avec la commande `\hfill` : remplissage de l'espace horizontal restant après la création de la 2nde `minipage`.
   Cette séparation permet d'avoir l'image à gauche collée sur la marge de gauche, celle de droite sur la marge de droite, et d'avoir un beau séparateur (espace blanc) entre les deux.

4) Création de la 2nde `minipage` de largeur X_2`\linewidth`, avec $X_2 \in]0\,;\,1[$ **et**. $X_1 + X_2 < 1$ (ou le `\hfill` n'a aucun intérêt).

5) Si insertion de légende(s), encadrement de toutes les étapes précédentes par un environnement `figure` et placement des légendes respectives au sein de chaque `minipage`.

Un exemple ici et maintenant et tout sera plus clair :

`minipage` & montage d'images

```
% Ajout dans le préambule
%\usepackage{graphicx, float}


\begin{figure}[H]
\begin{minipage}[t]{0.45\linewidth}
\centering
\includegraphics[width =
    0.6\linewidth]{fond.jpg}
\caption{Lég. 1}
\end{minipage}
\hfill
\begin{minipage}[t]{0.45\linewidth}
\includegraphics[width =
    \linewidth]{fond.jpg}
\caption{Lég. 2}
\end{minipage}
\end{figure}
```

Figure 8.4 – Lég. 1

Figure 8.5 – Lég. 2

Une fois que tu as saisi le principe pour 2 images, rien ne t'empêche d'en aligner autant que tu le souhaites, à condition d'avoir la place (ou tes images risquent d'être très petites).

Tu peux aussi moduler à ta guise la largeur des différentes `minipage` : rien ne t'oblige à toutes les avoir de la même largeur, par exemple. À toi d'adapter cet exemple en fonction de ton besoin !

Il est aussi possible de mettre du texte dans une `minipage`, pour insérer une **courte** explication à côté de l'image. **Attention toutefois si le texte est trop grand :** ta `minipage` va prendre trop de hauteur, le rendu ne sera plus aussi esthétique et la place risque de manquer.

Dans ces cas-là, il faut soit être synthétique, soit utiliser le package `wrapfig`, soit revoir le rendu souhaité.

minipage & texte

```
% Ajout dans le préambule
%\usepackage{graphicx, float}


\begin{figure}[H]
\begin{minipage}{0.55\linewidth}
J'aime le chocolat !
\end{minipage}
\hfill
\begin{minipage}{0.4\linewidth}
\centering
\includegraphics[width =
    0.86\linewidth]{fond.jpg}
\caption{Légende}
\end{minipage}
\end{figure}
```

J'aime le chocolat !

Figure 8.6 – Légende

Pour terminer, si tu veux t'amuser un peu, voici un premier aperçu d'une autre option sympathique intégrée avec le package `graphicx` :

Yolo !

```
% Ajout dans le préambule
%\usepackage{graphicx, float}

\begin{center}
\includegraphics[width =
    0.5\linewidth, angle =
    13]{fond.jpg}
\end{center}
```

Il existe pas mal d'autres options, ainsi que la très pratique commande `\resizebox`, que nous aurons la chance de recroiser plus tard dans ce guide. Mais je te laisse aller lire la documentation officielle[7]. Les explications de base sont sur ce guide et c'est ce qui m'importe.

Et voilà, tu approches de la fin de ce guide. Tu peux clairement t'arrêter après le chapitre 9 Traitement des erreurs et revenir à ce guide beaucoup plus tard selon tes besoins.

7. Disponible sur `https://ctan.org/pkg/graphicx`.

Chapitre 9

Traitement des erreurs

Les erreurs peuvent être nombreuses sous LaTeX et pas toujours évidentes à corriger. Tout d'abord, nous appelons « erreur » en LaTeX tout bout de code qui nuit à la compilation du document et l'empêche de se poursuivre. Une erreur ne permet donc pas au compilateur de produire le fichier PDF espéré.

Ensuite, il est important de savoir que toutes les erreurs qui vont être abordées sont retournées par LaTeX, suite à la compilation. Ces erreurs sont affichées par `Texmaker`, dans une fenêtre spécifique tout en bas (bouton « `Messages/Log` » en bas à gauche pour faire apparaître la fenêtre « Informations du compilateur » si inexistante).

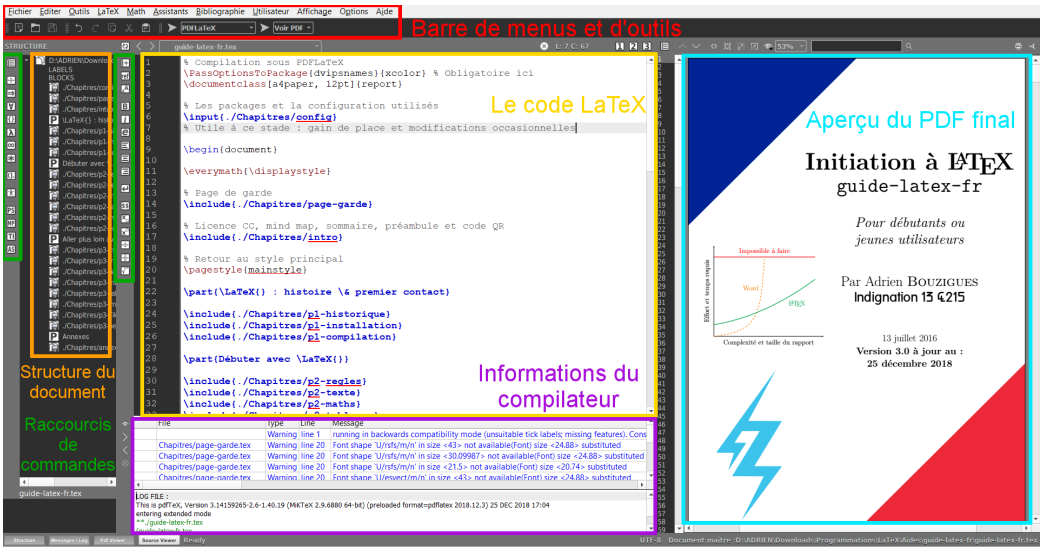


Figure 9.1 – Rappel de l'organisation de l'interface de `Texmaker`

Quand il y a une erreur, cette fenêtre t'indique aussi la ligne de code qui pose problème à LaTeX pour compiler (colonne `Line`). 90 % du temps, c'est sur cette ligne ou dans ses environs qu'il faut relire son code et chercher l'erreur.

Voyons maintenant la liste des erreurs couramment rencontrées quand tu débutes avec LaTeX, et mes conseils personnels pour les éviter, synthétisée sous forme d'un tableau.

Les erreurs courantes	Comment les corriger
`Missing $ inserted`	2 cas fréquents : → oubli de fermer un mode mathématiques ⇒ le fermer avec le symbole `$` manquant ; → emploi d'un symbole propre au mode mathématiques (`^` ou `_` par exemple) ⇒ supprimer le `^` inutile ou utiliser la commande `\_` pour afficher un *underscore* en mode texte.

`Missing } inserted`
ou
`I suspect you have forgotten a '}'`
	Très probablement, oubli de fermer une commande par une accolade `}` ⇒ Commencer par chercher les erreurs parmi les lignes de codes écrites ou modifiées depuis la dernière compilation.
	Au début, **compiler régulièrement son code** permet de simplifier la correction de cette erreur. [1]

`! Too many }'s.`	Plus rare : oubli d'une accolade ouvrante. Mêmes conseils que ci-dessus.

(suite sur la page suivante)

1. C'est plus pratique de corriger plein de petites erreurs que de s'arracher les cheveux sur un très grand nombre.

Les erreurs courantes	Comment les corriger
`There's no line to end here`	Saut de ligne incompris par LaTeX (après un environnement `center` par exemple). ⇒ Commencer par regarder le résultat sans saut de ligne : certains environnements laissent un peu de blanc avant et après (comme `center` justement). Autrement, utiliser la commande `\vspace`.
`undefined control sequence`	2 cas possibles : → oubli d'un élément à un endroit, comme une virgule lors d'un espace insécable (`\13` *vs* `\,13`) ; → appel d'une commande inexistante ou appel d'une nouvelle commande bien définie mais faute de frappe lors de son écriture. ⇒ Vérifier le code et le corriger.

(suite sur la page suivante)

Les erreurs courantes	Comment les corriger
`Package inputenc Error: Unicode char`, suivi éventuellement d'un caractère et de son code UTF-8	Utilisation d'un caractère du clavier interdit avec ce moteur de compilation. L'exemple le plus courant : symbole ° (commande `\degres{}` sous PDFLaTeX ; appel "normal" au clavier sous XeLaTeX). [2] Erreur fréquente si texte copier-collé d'un autre document (Word, PDF, page Internet) ⇒ Dans un 1er temps, reprendre tous les accents et les apostrophes. [3]
`Option clash for package <nom_package>`	Conflit entre certains packages. ⇒ Charger les packages dans un ordre bien précis. (exemple : package `xcolor` avanr `wallpaper`).
`Extra alignment tab has been changed to \cr`	Erreur dans un tableau : oubli hautement probable d'indiquer un changement de ligne (\\). *Rightarrow* Ajouter le \\ manquant.

`! [...] \begin{document} ended by \end{<env>}`
ou
`! [...] \begin{<env>} [...] ended by \end{document}`
	Environnement mal ouvert ou fermé. Très fréquent à cause de l'auto-complétion.
	⇒ Aller à la ligne indiquée par l'erreur, regarder l'environnement concerné, corriger selon le besoin.

Fin du tableau

Voilà dans les grandes lignes les principales erreurs que j'ai recensées

2. Les moteurs de compilation sont abordés dans la partie suivante si tu es intéressé.
3. Dans ce cas, la fonction `Remplacer` de `Texmaker` peut se révéler très utile.

jusqu'à présent. Avec l'expérience, tu verras que tu en feras de moins en moins ou que tu les corrigeras très rapidement.

Sache aussi que tu peux te rendre sur `http://fr.wikibooks.org/wiki/LaTeX/%C3%80_l%27aide_!` si tu veux des informations complémentaires.

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. Le nombre d'erreurs devrait diminuer.

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

Par ailleurs, je souhaite revenir sur l'erreur `Option clash for package`. Si jamais tu veux tester un nouveau package pour ton rapport ou adapter un code trouvé sur Internet, **ne jamais le faire sur ton document final !** C'est le meilleur moyen de perdre du temps (compilation et adaptation du code). Il vaut mieux procéder par étapes :

1) Copie du code à adapter sur un nouveau fichier `.tex` de test, avec juste les packages absolument nécessaires.

2) 1ère compilation pour s'assurer que le code copié fonctionne. Suppression des éléments inutiles et/ou gênants pour la compilation (commandes définies par l'utilisateur et non fournies par exemple).

3) Adaptation du code jusqu'à obtention du résultat souhaité.

4) Copie du code final dans ton rapport, ajout du/des package(s) requis, compilation et gestions des dernières potentielles erreurs.

Tu verras que tu perdras moins de temps à compiler, à étudier le résultat dans l'affichage `Texmaker` et tu travailles sur un fichier de test, sans pollué ton rapport.

La règle absolue avec les packages

Par défaut, *toujours* charger le package `hyperref` *en dernier* ! (sauf indication contraire : cf. la documentation du package `menukeys` par exemple).

Et voilà, la première partie de ce guide est (enfin) 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 débuter en LaTeX. Et si je commence à avoir pas mal de repères et d'expériences, la route est encore longue avant de pouvoir maîtriser les innombrables facettes de ce langage.

Tu trouveras dans la partie suivante mes notes personnelles sur du code LaTeX plus poussé, pour arriver à produire des résultats de plus en plus complexes. 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 lecteurs.

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

LaTeX, c'est la vie !

Troisième partie

Aller plus loin avec LaTeX

[Mise à jour prévue pour le 1er semestre 2019]

Annexes

Table A.1 – Les différentes possibilités de mise en forme du texte

Texte	Rendu	Environnement
`\textbf{gras}`	**gras**	`bfseries`
`\textit{italique}`	*italique*	`itshape`
`\emph{emphase}`	*emphase*	`em`
`\textsl{penché}`	*penché*	`slshape`
`\textsc{Petites Capitales}`	Petites Capitales	`scshape`
`\textsf{sans empattement}`	sans empattement	`sffamily`
`\texttt{machine}`	`machine (à écrire)`	`ttfamily`

TABLE A.2 – Liste non exhaustive des symboles disponibles sous LaTeX

Code	**Rendu**	**Description**
`\&`	&	Esperluette
`\oe` et `\OE`	œ et Œ	Ligature œ
`\ae` et `\AE`	æ et Æ	Ligature æ
`\ss`	ß	Eszett
`\no`	n°	Numéro
`-`	-	Tiret court
`--`	–	Tiret moyen
`---`	—	Tiret long
`\dots`	…	Points de suspension
`\og` et `fg`	« et »	Guillemets français ouvrants
` `` ` (accents graves)	“	Guillemets anglais ouvrants
`''` (apostrophes)	”	Guillemets anglais fermants
`\%`	%	Pourcent
`\euro`	€	Euro (package `marvosym`)
`\$`	$	Dollar
`\textcopyright`	©	Copyright
`\textregistered`	®	Marque déposée
`\texttrademark`	™	Trademark
`\#`	#	Dièse
`\{`	{	Accolade ouvrante
`\}`	}	Accolade fermante
`\_`	_	*Underscore*
`\textbackslash`	\	*Backslash*
`\textasciitilde`	~	Tilde

Table A.3 – Liste des symboles du package `pifont`

32		33	✁	34	✂	35	✃	36	✄	37	☎	38	✆	39	✇
40	✈	41	✉	42	☛	43	☞	44	✌	45	✍	46	✎	47	✏
48	✐	49	✑	50	✒	51	✓	52	✔	53	✕	54	✖	55	✗
56	✘	57	✙	58	✚	59	✛	60	✜	61	✝	62	✞	63	✟
64	✠	65	✡	66	✢	67	✣	68	✤	69	✥	70	✦	71	✧
72	★	73	☆	74	✪	75	✫	76	✬	77	✭	78	✮	79	✯
80	✰	81	✱	82	✲	83	✳	84	✴	85	✵	86	✶	87	✷
88	✸	89	✹	90	✺	91	✻	92	✼	93	✽	94	✾	95	✿
96	❀	97	❁	98	❂	99	❃	100	❄	101	❅	102	❆	103	❇
104	❈	105	❉	106	❊	107	❋	108	●	109	❍	110	■	111	❑
112	❒	113	❑	114	❒	115	▲	116	▼	117	◆	118	❖	119	◗
120	|	121	❙	122	❚	123	❛	124	❜	125	❝	126	❞		
		161	❡	162	❢	163	❣	164	❤	165	❥	166	❦	167	❧
168	♣	169	♦	170	♥	171	♠	172	①	173	②	174	③	175	④
176	⑤	177	⑥	178	⑦	179	⑧	180	⑨	181	⑩	182	❶	183	❷
184	❸	185	❹	186	❺	187	❻	188	❼	189	❽	190	❾	191	❿
192	①	193	②	194	③	195	④	196	⑤	197	⑥	198	⑦	199	⑧
200	⑨	201	⑩	202	❶	203	❷	204	❸	205	❹	206	❺	207	❻
208	❼	209	❽	210	❾	211	❿	212	➔	213	→	214	↔	215	↕
216	↘	217	➙	218	➚	219	➛	220	➜	221	→	222	⟶	223	➟
224	➠	225	➡	226	➢	227	➢	228	➤	229	➥	230	➦	231	➧
232	➨	233	➩	234	➪	235	➫	236	➬	237	➭	238	➮	239	➯
		241	➱	242	➲	243	➳	244	➴	245	➵	246	➶	247	➷
248	➸	249	➹	250	➺	251	➻	252	➼	253	➽	254	➾		

Table A.4 – Liste non exhaustive des symboles mathématiques disponibles sous LaTeX

Code	Rendu	Description
`$i_2$`	i_2	Indice
`$i^3$`	i^3	Exposant
`$\frac{a}{b}$`	$\frac{a}{b}$	Fraction
`$\cfrac{a}{b + \cfrac{c}{d}}$`	$\cfrac{a}{b + \cfrac{c}{d}}$	Fraction (étages)
`$\times$`	$\times$	Multiplication
`$\pm$`	$\pm$	Plus ou moins
`$\leq$` et `$\geq$`	$\leq$ et $\geq$	Inégalités larges
`$\leqslant$` et `$\geqslant$`	$\leqslant$ et $\geqslant$	Inégalités larges (bis)
`$\equiv$`	$\equiv$	Congruence
`$\neq$`	$\neq$	Non égal
`$\simeq$`	$\simeq$	Environ égal
`$\approx$`	$\approx$	Environ égal (bis)
`$\sim$`	$\sim$	Équivalence
`$\forall$`	$\forall$	Pour tout élément
`$\exists$`	$\exists$	Existence
`$\Rightarrow$`	$\Rightarrow$	Implication
`$\infty$`	∞	Infini
`$\int$`	$\int$	Intégrale simple
`$\iint$`	$\iint$	Intégrale double
`$\iiint$`	$\iiint$	Intégrale triple
`$\oint$`	$\oint$	Intégrale curviligne
`$\int_0^{+\infty} f(x)\,dx$`	$\int_0^{+\infty} f(x)\,dx$	Intégration
`$\sum$`	$\sum$	Somme
`$\partial$`	∂	Dérivée partielle

Table A.5 – La liste complète des lettres grecques sous LaTeX

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