

Introduction à L^AT_EX sous MacOS X



Fabien Conus

20 février 2004

Table des matières

1	Introduction	3
1.1	Késako ?	3
1.2	Installation	4
1.3	Compilation	5
1.4	Editeurs	6
1.5	Le premier document	7
1.6	Les erreurs	8
2	Le premier code \LaTeX	9
2.1	Les bases de \LaTeX	9
2.2	Quelques fonctions de texte	13
2.3	Mise en forme des paragraphes	16
3	La structure du document	18
3.1	La structure du document en \LaTeX	18
3.2	Les références	21
3.3	Les notes de bas de page	22
4	Insérer des objets	23
4.1	Créer une liste	23
4.2	Insérer une image	23
4.3	Insérer un tableau	28
5	Les mathématiques	33
6	Compléments et conclusion	40
6.1	Les oublis et les compléments	40
6.1.1	Les césures	40
6.1.2	L'orthographe	40
6.1.3	La grammaire	41
6.1.4	La fragmentation	41
6.1.5	Les images EPS	42
6.1.6	Les macros	42
6.2	Compléments propres à <code>pdflatex</code>	43
6.2.1	<code>hyperref</code>	43
6.2.2	<code>pdfsync</code>	45
6.3	Conclusion	46
6.4	Quelques liens	48

1 Introduction

Vous avez certainement souvent entendu parler de \LaTeX . Sous cet acronyme mystérieux, un peu effrayant, se cache un programme rassemblant autant de fidèles que de détracteurs. A la demande de plusieurs personnes, et suite aux nombreux débats sur l'avenir de FrameMaker, j'ai décidé d'aborder ce vaste sujet, en partie afin de le démythifier.

La tâche est rendue difficile par le fait qu'il existe une multitude de documents et de sites web vous proposant de découvrir \LaTeX , et je tenais à tout prix à éviter de faire du plagiat ou à tomber dans la redondance. J'ai donc pris le parti de faire une approche de \LaTeX en même temps que le test d'un éditeur dédié à ce programme et tournant sous MacOS X.

Ce texte sera donc exclusivement orienté MacOS X, même s'il pourra aisément être appliqué à d'autres plate-formes.

Car c'est bien là une des forces de \LaTeX : il tourne sur toutes les plate-formes connues, de l'Amiga à Linux en passant par Solaris, BSD, darwin, Windows ou DOS. Bienvenue dans le monde merveilleux du logiciel libre!

Voyons comment ça se passe sur notre machine.

1.1 Késako ?

En 1978, Donald E. Knuth désire créer un programme permettant d'obtenir sans peine de beaux documents. Il crée alors le formateur de texte \TeX , qui se prononce "tek" car il était également destiné à écrire des mathématiques (c'est donc un amalgame de "texte" et "tech"). La syntaxe de \TeX est toutefois assez complexe, et en 1982, L. Lamport crée \LaTeX , une couche de macros au-dessus de \TeX facilitant son utilisation. Pour faire une comparaison, on pourrait dire que \LaTeX est à \TeX , ce que l'HTML est au [SGML](#).

Pour une histoire détaillée de \TeX et \LaTeX , vous pouvez consulter [ce site](#).

De nombreuses personnes pensent que \LaTeX est un concurrent de Word, un traitement de texte parmi d'autres. Et bien c'est faux. \LaTeX n'est pas un *traitement* de texte, c'est un *formateur* de texte.

Ce que je veux dire par là, c'est que la différence fondamentale entre Word et \LaTeX c'est que ce dernier vous demande de vous concentrer uniquement sur le fond et pas du tout sur la forme. Tandis que le programme de Microsoft vous demande de vous occuper des deux.

Avec \LaTeX , vous entrez votre texte et il s'occupe de faire la mise en page (vous devrez quand même lui donner quelques indications). La conséquence de cela est que vous ne travaillez pas en WYSIWYG (what you see is what you get, ce que vous voyez est ce que vous obtenez) comme c'est le cas avec Word et les traitements de texte. Avec ces derniers il n'y a (en principe) pas de différence entre ce que vous voyez à l'écran et ce qui sortira de votre imprimante.

L'ennui, ce sont les deux petites parenthèses insignifiantes. Combien de temps avez-vous perdu avec Word à replacer une image qui s'était fait la malle lors d'un saut de page, à refaire un tableau qui pour une raison X ou Y était déformé à l'impression, etc.

Combien de temps avez-vous perdu à renuméroter vos références et vos chapitres suite à l'ajout d'un paragraphe au milieu de votre texte. Combien de temps pour placer cette fichue image qui coupait votre chapitre et ce tableau qui disparaissait lors de l'ajout d'une phrase. Et combien de temps pour numéroter et placer correctement ces notes de bas de page.

Bref, les exemples sont légion, un traitement de texte est censé vous faciliter la vie et en fait il la pourrit !

Grâce à \LaTeX tous ces soucis s'envolent, vous ne vous concentrez que sur le contenu.

A l'instar de l'HTML, vous allez créer un "code" \LaTeX contenant votre texte et quelques commandes de mise en page. Ce code sera alors interprété par le programme \LaTeX , tout comme l'HTML est interprété par votre navigateur, nous appellerons ça la "compilation".

1.2 Installation

Pour installer \LaTeX sur notre machine, la meilleure solution est de passer par [i-Installer](#) que j'avais testé il y a quelque temps. Les paquets à installer sont les suivants :

- TeX
- CM Super for TeX

Ce dernier n'est pas à proprement parlé nécessaire au fonctionnement de \LaTeX mais permet la création de documents beaucoup plus jolis.

Je n'entre pas dans les détails de l'installation et je vous renvoie à mon test pour plus d'informations.

L'installation terminée, vous ne verrez aucun nouveau programme dans votre dossier Applications ou Utilitaires. \LaTeX s'installe dans la couche Unix de votre OS.

A ce stade \LaTeX est complètement utilisable via la ligne de commande du Terminal. Si vous voulez en être sûr, ouvrez une fenêtre Terminal et entrez :

```
latex -v
```

Vous verrez apparaître le numéro de version de \LaTeX ainsi que quelques informations.

1.3 Compilation

Je ne vais pas m'attarder sur la compilation "à la main" d'un code \LaTeX , mais pour des raisons historiques je tiens à mentionner quelques petites choses.

La compilation d'un code \LaTeX se fait simplement en appelant la commande

```
latex mondocument.tex
```

S'en suivra une série de lignes obscures indiquant la compilation du document. Puis vous obtiendrez un fichier portant l'extension **.dvi** (device indépendant). Ceci est un format propre à TeX et ne peut être visionné que par un programme dédié (xdvi par exemple). On peut ensuite convertir ce dvi en PostScript pour l'envoyer sur une imprimante.

Ceci n'est pas très intéressant pour nous.

Heureusement, il existe depuis quelques années une nouvelle commande : `pdflatex` !

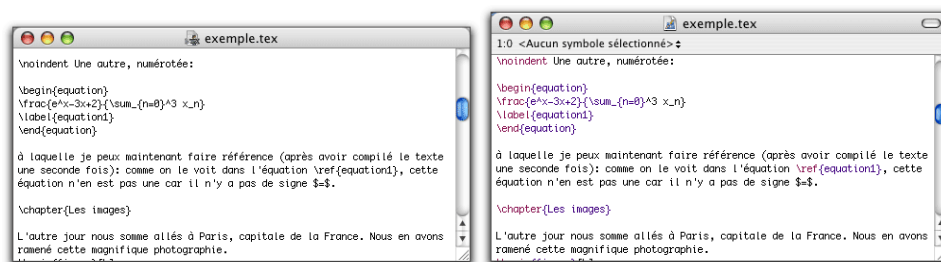
Grâce à cette commande, votre code \LaTeX sera directement compilé en un magnifique document PDF, donc totalement compatible avec tous les différents acteurs du monde informatique. Et comme le PDF fait partie intégrante de MacOS X son intérêt n'en est que plus grand.

Mais je vous connais, vous ne voulez pas mettre les mains dans le Terminal, et vous avez raison. C'est là qu'interviennent les éditeurs de texte destinés à \LaTeX .

1.4 Editeurs

Tout comme l'HTML, un code \LaTeX consiste simplement en un texte au format ASCII. N'importe quel éditeur de texte suffira donc à créer son document : pico, vi, SubEthaEdit, BBEEdit, XCode ou même TextEdit !

Bien sûr, certains sont mieux adaptés que d'autres. Comme pour l'HTML, un bon éditeur sera capable de coloriser les commandes (les balises en HTML). BBEEdit le fait, SubEthaEdit également, XCode aussi si on lui ajoute les [fichiers nécessaires](#). La colorisation rend le code beaucoup plus agréable :



Avec TextEdit il n'y a pas de colorisation, tandis qu'avec SubEthaEdit, oui.

Mais l'éditeur idéal ne se contentera pas de coloriser les commandes, il faudrait aussi qu'il nous évite l'utilisation du Terminal ! Et là, le choix se restreint.

Il y a à ma connaissance deux éditeurs \LaTeX idéaux pour la distribution de \LaTeX que nous avons utilisée¹ :

- [TeXShop](#) de Richard Koch
- [iTeXMac](#) de Jérôme Laurens

Pour ma part, j'ai une préférence pour le premier. Son interface est plus simple et l'auteur a toujours répondu à mes messages et a toujours implémenté ce que je lui demandais. Mais iTeXMac est également très populaire et paraît-il plus complet.

Je vais donc orienter cet article non seulement sur l'usage et la syntaxe de \LaTeX , mais également sur la façon de l'utiliser facilement avec TeXShop.

Téléchargez donc TeXShop sur le site ci-dessus, ou directement [là](#).

¹la distribution de \LaTeX installée par i-Installer s'appelle teTeX. Il existe d'autres distributions, comme [OzTeX](#).

L'installation est très simple, il vous suffit de copier l'icône de TeXShop dans votre dossier Applications (ou ailleurs si vous préférez). Une fois l'application lancée, vous vous retrouvez devant une fenêtre vide semblable à celle de TextEdit. C'est là que nous allons entrer notre code.

Nous voilà maintenant fin prêts, tous les outils sont installés et nous allons pouvoir commencer à créer nos documents.

1.5 Le premier document

Nous allons faire très simple pour ce premier document. Entrez donc dans l'éditeur le code suivant :

```
\documentclass[a4paper]{book}
\begin{document}
Bonjour le monde !\\
\end{document}
```

L'antislash (\) s'obtient par la combinaison alt-shift-7 sur un clavier suisse-romand. Sur un clavier français, c'est alt-shift- :.

Sauvegardez ce fichier (je vous conseille de créer un dossier spécial qui contiendra ce fichier) puis cliquez sur le bouton "Composer" de TeXShop. Une fenêtre va s'ouvrir et des lignes vont défiler, enfin, une nouvelle fenêtre va s'ouvrir contenant votre document PDF².

Bravo, vous venez de compiler votre code L^AT_EX !

Bon évidemment il est simplissime, mais nous allons très vite nous lancer dans des choses plus évoluées.

La petite fenêtre qui s'ouvre lorsque vous compilez votre document s'appelle la console. C'est ce que vous verriez si vous lanciez la compilation depuis le *Terminal*. C'est également là que L^AT_EX vous fera part des éventuelles erreurs. Lors d'une compilation réussie, les deux dernières lignes seront toujours "Output written on..." et "Transcript written on...".

Vous remarquerez que plusieurs fichiers ont été créés dans le dossier qui contient votre document. Un fichier .log, un .out et un .aux, utilisés par L^AT_EX pour stocker quelques informations. Et bien sûr un fichier .pdf qui est votre document. C'est à cause de ces multiples fichiers que je vous conseille

²si a ce stade vous obtenez une erreur et que le PDF ne s'affiche pas c'est que L^AT_EX s'est mal installé.

de créer un dossier pour chacun de vos documents.

1.6 Les erreurs

Si vous avez fait une faute de frappe dans un nom de commande, \LaTeX ne sera pas en mesure de compiler votre document. Dans la fenêtre de console vous verrez apparaître la ligne :

```
!Undefined control sequence
```

suivie de la commande qui pose problème et d'un point d'interrogation.

TeXShop offre cette excellente option de vous placer directement à l'endroit du texte d'où provient l'erreur. Pour cela, il vous suffit de cliquer sur le bouton "Erreur" situé en haut à gauche de la console.

2 Le premier code L^AT_EX

Dans le chapitre 1, nous avons examiné comment installer les outils nécessaires au bon fonctionnement de L^AT_EX. Nous avons terminé par un exemple très simple de document.

Dans ce chapitre, nous allons examiner en détail ce petit exemple et aller plus loin dans les fonctionnalités de L^AT_EX.

2.1 Les bases de L^AT_EX

Je vous rappelle notre petit code L^AT_EX :

```
\documentclass[a4paper]{book}
\begin{document}
Bonjour le monde !\
\end{document}
```

Un document L^AT_EX commence toujours par `\documentclass`, suivi des options entre crochets et du type de document (aussi appelé *class*) entre accolades. Dans notre cas, on indique dans les options que l'on travaille au format A4 et dans le type de document que l'on rédige un livre (book).

Les types de document définissent la mise en pages. Les plus courants sont :

- article
- book (livre)
- report (rapport)
- letter (lettre)

mais il en existe beaucoup d'autres (exam, slides, etc.)

Les différences entre ces *class* ne seront pas flagrantes dans notre petit exemple, elles prendront toute leur importance plus tard.

Une autre option communément utilisée avec `\documentclass` est destinée à fixer la taille de la police. Par exemple :

```
\documentclass[a4paper, 11pt]{book}
```

définira une taille de police de 11 points dans le document. Vous avez le choix entre 10, 11 et 12 points.

La commande suivante de notre petit exemple est également obligatoire, `\begin{document}` déclare en effet le début du contenu du document, elle indique que la configuration du document est terminée. A chaque `\begin{document}`

correspond un `\end{document}`, également obligatoire, définissant la fin du document.

Les commandes ayant cette forme (`\begin` suivi de `\end`) s'appellent des environnements.

Vous l'aurez compris, toutes les commandes (enfin, presque toutes) de L^AT_EX sont de la forme :

```
\commande[options]{argument}
```

les options et l'argument étant optionnels (selon les commandes). Par exemple, la commande `\begin` ne prend aucune options mais l'argument est obligatoire.

La dernière commande à examiner dans notre petit exemple est : `\\`. Connue aussi sous le nom de `\newline`. Cette commande ne prend ni options, ni argument et définit passage à la ligne. En fait, elle n'est pas nécessaire dans notre exemple, puisque nous n'avons de toute façon qu'une seule ligne. Cette commande suivie d'une ligne vide définit un nouveau paragraphe suivit d'un saut de ligne, essayez donc ceci :

```
\documentclass[a4paper, 11pt]{book}
\begin{document}
Bonjour le monde !\\

Ceci est un nouveau paragraphe avec un saut de
ligne.
Ici, nous n'aurons pas de retour à la ligne.
\end{document}
```

La première phrase se termine par `\\` et est suivie par ligne vide, ce qui créé un saut de ligne suivit d'un nouveau paragraphe.

Un simple retour de chariot, sans l'utilisation du `\\` n'aura aucun effet, comme cela est démontré par la deuxième et la troisième phrase de l'exemple.

En effet, comme avec l'HTML, L^AT_EX ne "voit pas" les retours à la ligne (sauf si il y en a deux consécutifs comme je le mentionnais ci-dessus), de même il ne "voit pas" les espaces consécutifs. Que vous écriviez :

```
Ceci est un nouveau paragraphe.
```

ou

```
Ceci      est un nouveau paragraphe.
```

Ce sera pour lui exactement la même chose.

Une ligne ne se terminant pas par un `\\`, mais étant suivie par une ligne vide (donc un double retour de chariot) provoquera la création d'un nouveau paragraphe sans saut de ligne. Essayez par exemple ceci :

```
\documentclass[a4paper, 11pt]{book}
\begin{document}
Bonjour le monde !

Ceci est un nouveau paragraphe sans saut de
ligne.\\
Ceci est une nouvelle ligne, mais pas un nouveau
paragraphe.
\end{document}
```

La création d'un nouveau paragraphe (avec `\\` suivi d'une ligne vide, ou avec un double retour de chariot) impliquera une indentation de la première ligne de ce nouveau paragraphe. Cela ne se voit pas dans mon petit exemple, car mes phrases ne sont pas assez longues. Si vous voulez supprimer l'indentation d'une phrase, utilisez la commande `\noindent` :

```
\documentclass[a4paper, 11pt]{book}

\begin{document}
Bonjour le monde !\\

Ceci est un nouveau paragraphe et cette phrase
aura une indentation.\\

\noindent Par contre, celle-ci n'en aura pas.

\end{document}
```

Similairement à la commande `\newline` (ou `\\`), il existe également `\newpage` qui provoque un saut de page.

Si vous le désirez, vous pouvez aisément ajouter une page de titre à votre document. Pour ce faire, vous pouvez définir trois champs : le titre du document, l'auteur et la date. Ceci se fait grâce aux commandes `\title`, `\author` et `\date`. Puis il faut encore décider où vous désirez créer la page de titre (habituellement en début de document) avec la commande `\maketitle`. Notre code aura donc l'allure suivante :

```

\documentclass[a4paper, 11pt]{book}

\begin{document}

\title{Un petit exemple simple}
\author{Fabien Conus}
\date{2004}
\maketitle
Bonjour le monde !\\

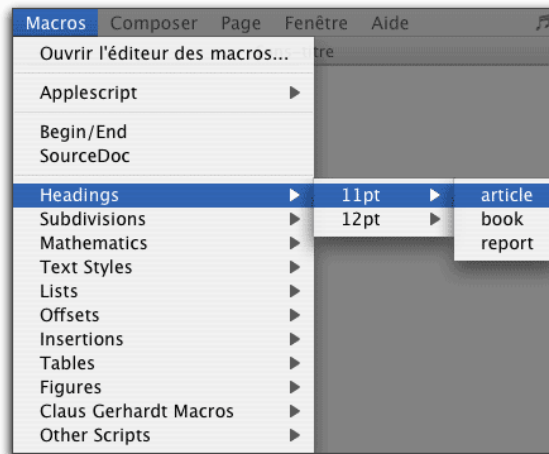
Ceci est un nouveau paragraphe et cette phrase
aura une indentation.\\

\noindent Par contre, celle-ci n'en aura pas.

\end{document}

```

Bien sûr, il n'est pas très agréable d'entrer ces signes barbares à la main. C'est pourquoi *TeXShop* vous propose une série de Macros. En effet, vous avez à disposition, via le menu "Macros" toute une série d'environnements pré-enregistrés. Par exemple, pour obtenir l'entête d'un document de classe "article" ayant des caractères de 11 points, il vous suffit de vous rendre là :



Et *TeXShop* vous insérera automatiquement tout ce qu'il vous faut. Il ne vous reste plus qu'à remplir les champs destinés à la page de titre puis de taper votre texte entre `\begin{document}` et `\end{document}`.

Je reviendrai plus tard sur les autres Macros, au fur et à mesure de notre découverte de L^AT_EX.

Comme pour les langages de programmation, il est possible d'ajouter des commentaires dans le code L^AT_EX. Ceci se fait grâce au signe % :

```
\documentclass[a4paper, 11pt]{book}

\begin{document}
% ceci est un commentaire
Bonjour le monde!\

Ceci est un nouveau paragraphe et cette phrase
aura une indentation.% ceci en est un également.

\end{document}
```

Tout ce qui suit le signe % sera ignoré à la compilation. Pour écrire un signe % vous devrez utiliser la commande \%.

Passons à présent à des choses plus sérieuses.

2.2 Quelques fonctions de texte

Vous l'aurez remarqué, j'ai pris soin pour l'instant de ne pas utiliser de caractères accentués. C'est fait exprès (oh le vil personnage!) En effet, comme toujours en informatique, les accents sont une difficulté. Essayez donc d'écrire des accents dans notre petit document d'exemple. Vous constaterez après compilation que les lettres accentuées ont tout simplement été omises.

Pendant longtemps, la seule solution pour écrire des lettres accentuées était de passer par des commandes spéciales. Pour l'accent aigu il fallait entrer : \', pour l'accent grave \', pour le circonflexe ^ et pour le tréma \" (guillemet). Par exemple :

```
Cet \'et\'e, dans la for^et, j'ai ou\"i un tr\'es grand cri.
```

Vous avouerez que ce n'est pas très agréable. Mais heureusement, la solution existe.

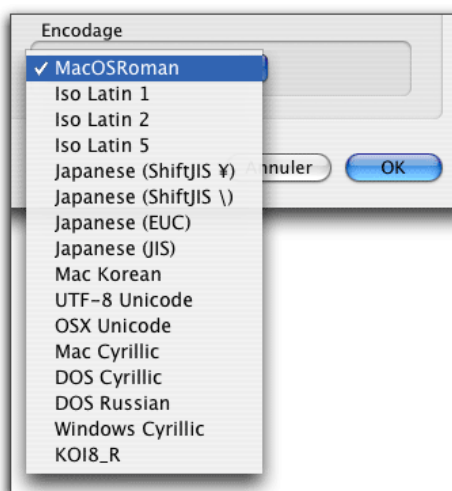
En informatique, qui dit accents dit encodage du texte. Dans le cas de l'HTML, on déclare l'encodage du texte dans les balises "meta" en spécifiant le "charset". Avec L^AT_EX il faut également déclarer l'encodage. Pour cela il faut ajouter un "paquet" de commandes. Ceci se fait avec la commande :

```
\usepackage[options]{nom du paquet}
```

Dans notre cas précis, nous voulons lui indiquer de charger le "paquet" de commandes relatif à l'encodage du texte, en lui spécifiant que nous travaillons avec un Mac :

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

En effet, par défaut, l'encodage du texte utilisé par TeXShop est "MacOS-Roman". Mais si vous voulez avoir plus de compatibilité avec nos voisins de chez Windows, vous allez certainement préférer l'encodage "iso-latin-1". Pour ce faire, rendez-vous dans les préférences de TeXShop. Dans l'onglet "Document" vous y trouverez une zone "Encodage" :



Choisissez alors "Iso Latin 1". Vos prochains documents auront cet encodage. Pour changer l'encodage de votre document en cours, vous devez en sauvegarder un nouvel exemplaire en passant par le menu "Fichier, Enregistrer sous...". Vous pourrez alors choisir quel encodage utiliser pour la sauvegarde.

Si vous décidez de passer à l'encodage "Iso Latin 1", vous devez encore en avertir \LaTeX :

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

L'ajout de paquets doit se faire après la déclaration du type de document et avant le début du document :

```

\documentclass[a4paper, 11pt]{book}
\usepackage[applemac]{inputenc}

\begin{document}
Bonjour le monde!\\

Ceci est un nouveau paragraphe.

Et ceci est une nouvelle ligne.
Cet été, dans la forêt, j'ai ouï un très grand
cri.
\end{document}

```

La police par défaut de L^AT_EX est une police "roman". Il est évidemment possible de modifier le style du texte. On peut écrire *en italique*, **en gras**, mais aussi en "machine à écrire", en petites capitales, en "slanted" et enfin en "sans serif". Voici les commandes correspondantes, que vous retrouvez dans le menu Macros, sous "Text styles" -> "style" :

Italique	<code>\it</code>
Gras	<code>\bf</code>
Machine à écrire	<code>\tt</code>
Petites capitales	<code>\sc</code>
Slanted	<code>\sl</code>
Sans serif	<code>\sf</code>
Roman	<code>\rm</code>

Pour vous montrer l'utilisation de ces commandes, voici un petit exemple et son résultat :

Un bout de texte <code>{\it en italique}</code> .	Un bout de texte <i>en italique</i> .
Un bout de texte <code>{\bf en gras}</code> .	Un bout de texte en gras .
Un bout de texte <code>{\tt en "typewriter"}</code> .	Un bout de texte en "typewriter".
Un bout de texte <code>{\sc en petites capitales}</code> .	Un bout de texte EN PETITES CAPITALES.
Un bout de texte <code>{\sl en slanted}</code> .	Un bout de texte sl en slanted.
Un bout de texte <code>{\sf en Sans Serif}</code> .	Un bout de texte en Sans Serif.

Si vous omettez les accolades, tout le texte qui suivra la commande prendra le style défini par cette commande. Pour revenir au style normal, vous pouvez alors utiliser `\rm`.

Vous pouvez décider de souligner une partie du texte. Ceci se fait grâce à la commande `\underline{mon texte à souligner}`.

Vous pouvez également mettre l'accent sur une partie d'une phrase. Ceci se fait grâce à la commande `\emph` (emphasize, en anglais, signifie accentuer). Par exemple :

```
Il est \emph{très important}de lire ceci !
```

```
Il est très important de lire ceci !
```

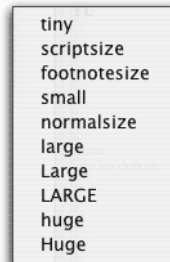
L'intérêt d'utiliser cette commande plutôt que `\it` est qu'elle s'adapte au style en cours, par exemple :

```
{\it Il est \emph{très important}de lire  
ceci !}
```

donnera

```
Il est très important de lire ceci !
```

Vous pouvez également agir sur la taille du texte. Là encore vous retrouvez la liste dans le menu Macros :



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

L'utilisation de ces tailles est identique à celle des styles, par exemple :

```
Voici un {\tiny tout petit texte}et  
un {\Huge très gros}
```

2.3 Mise en forme des paragraphes

Comme dans n'importe quel traitement de texte, les paragraphes peuvent être alignés, centrés ou justifiés.

Par défaut, le texte est justifié à gauche et à droite (\LaTeX s'arrange pour que chaque ligne prenne la même longueur. Pour aligner à gauche, il faut entrer notre texte dans un environnement "flushleft" :

```
\begin{flushleft}  
Ce bout de texte sera aligné à gauche, alors  
que par défaut le texte sera justifié des deux  
côtés. Bien sûr, il faut un texte assez long pour  
remarquer ceci.  
\end{flushleft}
```


Naturellement, pour justifier à droite, il faudra placer le texte dans un environnement "flushright".

Pour centrer un bout de texte, vous utiliserez l'environnement "center" :

```
\begin{center}  
Ce bout de texte sera centré.  
\end{center}
```

Vous pouvez également décaler un bout de texte vers la droite grâce aux environnements "quote" (cite) et "quotation" (citation) que vous trouverez dans le menu Macros sous "Insertions". La seule différence entre ces deux environnements est qu'avec "quotation" la première ligne est indentée. Voici un exemple :

```
Comme le disait Jean de la Fontaine dans sa  
fable :  
  
\begin{quote}  
Rien de sert de courir, il faut partir à point.  
\end{quote}
```

Bien sûr, le meilleur moyen de se rendre compte du résultat que l'on obtient avec ces différents environnements c'est de faire l'essai. Je vous invite donc à compiler un document utilisant ces différentes commandes et à faire vos essais.

3 La structure du document

Grâce aux deux chapitres précédents, nous savons à présent comment créer un document, y entrer du texte et les mettre quelque peu en forme. Il est temps à présent de nous attaquer à la structure du document.

3.1 La structure du document en \LaTeX

Un document se structure en chapitres, sections, paragraphes, etc.

La création des paragraphes est simple, je l'ai déjà expliqué au paragraphe 2.1, il suffit de faire un saut de ligne.

Dans un document \LaTeX , les structures possibles ainsi que leur apparence dépendront de la classe du document. Je vous rappelle que dans la deuxième partie de cette série d'articles, j'avais mentionné l'existence de plusieurs classes :

- article
- book (livre)
- report (rapport)
- letter (lettre)

La classe letter est un peu à part (il est rare qu'un place des chapitres et des sections dans une lettre) je ne la traiterai donc pas en détail ici. La classe article ne propose pas de chapitres tandis que les classes book et report le font.

Pour définir le début d'un nouveau chapitre, on entre la commande suivante :

```
\chapter{Titre de mon chapitre}
```

Pour définir le début d'une section à l'intérieur d'un chapitre :

```
\section{Titre de ma section}
```

On peut alors faire une sous-section :

```
\subsection{Titre de ma sous-section}
```

et même une sous-sous-section :

```
\subsubsection{Titre de ma sous-sous-section}
```

Attention, dans la classe "article", la commande `\chapter` n'est pas disponible.

Si vous faites quelques essais, vous remarquerez très vite que L^AT_EX se charge lui-même de la numérotation des chapitres ainsi que de la mise en page. En effet, il commencera automatiquement un nouveau chapitre sur une nouvelle page. Si une nouvelle section est commencée alors qu'il ne reste plus beaucoup de place sur la page, il fera automatiquement un saut de page, sinon il commencera la section sur la même page. De même pour les sous-sections et les sous-sous-sections.

Une fois votre document structuré, vous pouvez aisément naviguer de chapitres en sections, grâce à l'élément "Etiquettes" de la barre d'outil de *TeX-Shop*.

Mais vous aurez remarqué que L^AT_EX nous gratifie d'un très désagréable "Chapter" au lieu du doux "Chapitre" que nous attendons. En effet, par défaut L^AT_EX travaille en anglais. Mais rien de plus que de le faire parler la langue de Molière !

Il existe pour cela un paquet de commandes intitulé très justement "babel" (comme la tour du même nom qui inspira à Dieu la création de langues différentes). Il s'agit donc d'indiquer à L^AT_EX que nous désirons utiliser ce paquet de commandes en lui précisant que nous écrivons en français. Dans le paragraphe 2.2 nous avons vu comment ajouter un paquet de commandes grâce à `\usepackage`, nous allons faire la même chose ici :

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

Nous appelons donc le paquet de commandes "babel" avec l'option "french". Ceci ira s'ajouter en préambule du document :

```
\documentclass[a4paper, 12pt]{book}
\usepackage[applemac]{inputenc}
\usepackage[french]{babel}
\begin{document}
...
\end{document}
```

Vous pouvez bien entendu appeler les commandes `\usepackage` dans l'ordre que vous voulez, cela n'a aucune importance.

Ce paquet de commandes ne fait pas que traduire "Chapter" en "Chapitre", il indique également à L^AT_EX d'utiliser les conventions typographiques française. Il vous offre également quelques commande propres à notre langue

(voir [l'article de Nicolas Seriot](#) à ce sujet) et l'écriture de la date en français. En effet, il existe une commande anodine, mais très utile de L^AT_EX, c'est la commande `\today`. Celle-ci sera remplacée lors de la compilation par la date du jour. Ainsi, lors de la mise en place de la page de titre (que nous avons créé au paragraphe 2.1), il suffit de mettre :

```
\date{\today}
```

pour que la page de titre reflète toujours le jour de la compilation. Si vous travaillez sur votre texte pendant plusieurs jours et imprimez chaque jour une version, cela vous aidera à vous y retrouver.

Bien sûr, une fois votre texte rédigé et bien structuré en chapitres, sections, et autres sous-sections, vous allez vouloir créer une table des matières.

Avec L^AT_EX c'est un jeu d'enfant ! Il vous suffit de placer la commande `\tableofcontents` là où vous désirez voir apparaître la table des matières. Traditionnellement celle-ci sera placée avant le début du premier chapitre, vous aurez donc :

```
\documentclass[a4paper, 12pt]{book}
\usepackage[applemac]{inputenc}
\usepackage[french]{babel}

\begin{document}
\title{Un petit exemple simple}
\author{Fabien Conus}
\date{\today}
\maketitle
\tableofcontents
\chapter{Mon premier chapitre}
...
\end{document}
```

Lorsque vous compilerez votre document pour la première fois, vous vous retrouverez devant une table des matières vide. En effet, lors de la compilation, L^AT_EX rencontre le `\tableofcontents` avant les `\chapter` et les `\section`. Il ne peut donc pas deviner le contenu de votre texte. Il vous faut donc lancer la compilation une deuxième fois pour voir votre table des matières se remplir.

Si vous désirez ajouter un chapitre ou une section qui ne soit pas numérotée et qui n'apparaîtra donc pas dans la table des matières, vous pouvez utiliser les commandes :

```
\chapter*{Titre de mon chapitre non numéroté}
```

et

```
\section*{Titre de ma section non numérotée}
```

3.2 Les références

Il est parfois utile dans un texte de faire référence à un chapitre ou à une section. Imaginons que j'écrive un texte avec trois chapitres. Dans le troisième chapitre, je veux faire référence au deuxième chapitre. J'écrirais donc :

```
Dans le chapitre 2 nous avons découvert de très intéressants mystères.
```

Mais en reprenant mon texte le lendemain, je décide d'introduire un quatrième chapitre que je veux placer entre le premier et le deuxième. Comme nous l'avons vu, la numérotation des chapitres est automatique avec \LaTeX , je n'ai donc pas de soucis à me faire à ce niveau-là. Par contre, ma phrase ci-dessus est à présent fautive puisque le chapitre 2 est devenu le chapitre 3. Bien sûr, pour un petit document cela ne pose pas trop de problème de tout changer à la main, mais \LaTeX peut très bien se charger lui-même de vos références.

Pour cela, nous allons donner une étiquette à notre chapitre grâce à la commande `\label` :

```
\chapter{De très intéressants mystères}
\label{mysteres}
```

J'étiquette donc mon chapitre avec le mot "mystere" (il faut éviter les accents et les caractères spéciaux lors de l'étiquetage). Lorsque dans mon texte je désire faire référence à ce chapitre je le ferai avec la commande `\ref` :

```
Dans le chapitre\ref{mysteres} nous avons découvert de très intéressants mystères
```

Ainsi, nous n'aurons plus à nous préoccuper du numéro que porte ce chapitre. Bien entendu, cela fonctionne également avec les sections.

Lorsque vous compilez votre document après avoir ajouté des références, celles-ci seront d'abord remplacées par ?? et vous verrez dans la fenêtre de console (celle où toutes ces lignes défilent) apparaître ces lignes (éventuellement une seule des deux) :

```
LaTeX Warning : There were undefined references.
```

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

Traduction :

Avertissement : Certaines références étaient indéfinies.

Avertissement : Les étiquettes ont peut-être été modifiées. Re-compilez pour obtenir des références correctes

En effet, pour la même raison pour laquelle la table des matières était vide après la première compilation, \LaTeX n'est pas en mesure de trouver toutes les références lors du premier passage. Il vous faut donc recourir à une deuxième compilation pour voir apparaître toutes vos références.

3.3 Les notes de bas de page

Les notes de bas de page sont un autre type de références. Leur utilisation et leur mise en place est très simple avec \LaTeX . Pour ajouter une note de bas de page, il suffit d'ajouter la commande `\footnote` à la suite du mot concerné par la note : Par exemple :

```
Les CFF\footnote{Chemin de Fer Fédéraux}transportent  
chaque année des millions de passagers.
```

Dans cet exemple, \LaTeX ajoutera un petit numéro à la suite de "CFF" et créera une note de bas de page correspondante contenant "Chemin de Fer Fédéraux". Bien entendu, la numérotation des notes est automatique ainsi que leur mise en page.

De la même manière, vous pouvez créer facilement des notes dans la marge du document avec la commande `\marginpar`. Par exemple :

```
Les CFF\footnote{Chemin de Fer Fédéraux}transportent  
chaque année des millions de passagers\marginpar{à  
vérifier}.
```

Ces deux commandes sont accessibles depuis le menu Macros, sous "Offsets".

4 Insérer des objets

Grâce aux parties précédentes, vous avez tous les clefs en main pour écrire un document ne contenant que du texte. Mais il est forcément utile de pouvoir insérer des objets comme des listes, des tableaux ou des images. Nous allons voir dans cette partie comme procéder.

4.1 Créer une liste

La création d'une liste se fait grâce à l'environnement "itemize". Une fois dans l'environnement, chaque élément de la liste est défini par la commande `\item` :

```
\begin{itemize}
\item premier élément
\item deuxième élément
\item etc\ldots
\end{itemize}
```

donnera

- premier élément
- deuxième élément
- etc...

Vous noterez qu'il ne faut pas placer de `\\` entre chaque élément, le retour à la ligne est automatique. Vous aurez aussi noté au passage l'utilisation de la commande `\ldots` qui correspond aux trois points de suspension...

Il existe également un environnement "enumerate", qui fonctionne comme "itemize" mais qui remplace les puces par des numéros.

4.2 Insérer une image

Pour insérer une image, nous devons faire appel à un nouveau paquet de commandes intitulé "graphicx" (il existe également un paquet "graphics" mais qui est moins complet). Nous devons donc ajouter à notre préambule :

```
\usepackage{graphicx}
```

Ceci étant fait, nous pouvons ajouter une image au moyen de la commande :

```
\includegraphics{monimage.jpg}
```

Ceci suppose que le fichier "monimage.jpg" se trouve dans le même répertoire que le fichier ".tex". Si vos images sont dans un dossier "images" se trouvant dans le dossier contenant le fichier ".tex", vous pouvez accéder à vos images en faisant :

```
\includegraphics{./images/monimage.jpg}
```

Les formats d'image reconnus par pdfL^AT_EX (qui est le moteur utilisé par *TeXShop*) sont : .jpg, .png et .pdf. Le format PDF est particulièrement utilisé pour des images vectorielles, mais est également tout à fait adapté aux images bitmap. Il est également possible d'introduire des images eps en changeant le mode de compilation de *TeXShop*, je reviendrai là-dessus plus tard.

Voyons donc un exemple concret :

```
\begin{document}
Voici une photo : \includegraphics{medor.jpg} de
mon chien Médor.
\end{document}
```

Comme en HTML, l'image fait alors partie intégrante du texte. Mais dans la plupart des cas, ce n'est pas ça que l'on veut. On veut placer l'image entre deux parties du texte. Il suffit alors de la placer dans un environnement "center" :

```
\begin{document}
Voici une photo :
\begin{center}
\includegraphics{medor.jpg}
\end{center}
de mon chien Médor.
\end{document}
```

Pour ajouter une légende, on la placera simplement à la ligne :

```
\begin{document}
Voici une photo :
\begin{center}
\includegraphics{medor.jpg} \\
Photo prise en juin 02
\end{center}
de mon chien Médor.
\end{document}
```

Mais il arrive également qu'on ne s'intéresse peu au placement précis de l'image, mais qu'on veuille par contre la numéroter pour pouvoir y faire référence (c'est le cas le plus fréquent lors de publications scientifiques). Pour cela nous allons placer cette image dans un environnement "figure" :


```
\begin{document}
\begin{figure}
\includegraphics{medor.jpg}
\end{figure}
\end{document}
```

L'image devient alors un "float element", un élément flottant. Cela signifie que \LaTeX s'occupera tout seul du placement de l'image. Si il n'y a pas assez de place sur la page en cours il la placera sur la page suivante. Si il y a un titre de section à proximité, il s'arrangera pour placer l'image le plus judicieusement possible. La position de l'image ne dépend donc plus uniquement de l'endroit du texte où vous la déclarez. Par exemple :

```
\begin{document}
Ceci est un premier paragraphe, bla bla bla
bla bla bla, etc...\\
\begin{figure}
\includegraphics{medor.jpg}
\end{figure}
Ceci est un deuxième paragraphe, bla bla
bla bla bla bla, etc...
\end{document}
```

ne garantira pas que l'image soit placée entre ces deux paragraphes. Vous pouvez toutefois indiquer à \LaTeX vos préférences en ajoutant une option :

```
\begin{figure}[option]
```

Option est un (ou plusieurs caractères) indiquant vos préférences quant au placement de l'image, les valeurs possibles sont :

- t : "top", l'image sera placée en haut d'une page
- b : "bottom", l'image sera placée en bas d'une page
- h : "here", l'image sera placée là où elle est déclarée (dans la mesure du possible)
- f : "float", l'image sera placée sur une page séparée ne contenant que des éléments flottants

Vous pouvez bien entendu combiner ces quatre caractères dans l'ordre de vos préférences. Si vous voulez essayer de forcer \LaTeX à placer votre image là où vous le voulez, vous pouvez ajouter un point d'exclamation "!" à ces options. Mais cela n'ira jamais à l'encontre du "look" du document.

Si vous ne pouvez pas placer votre image là où vous la voulez, il devient difficile de la mentionner. Dans mon exemple ci-dessus, si l'image de mon chien Médor se trouve à la page suivante, ma phrase n'a plus beaucoup de sens.

Il s'agit donc d'y faire référence de la même manière que nous l'avons fait pour des chapitres et des sections dans la partie précédente, en y ajoutant une étiquette.

Pour cela, il faut y ajouter une légende. Ceci se fait grâce à la commande `\caption` :

```
\begin{figure}[ht]
\includegraphics{medor.jpg}
\caption{Voici la photo de Médor prise en
juin 02}
\label{medor}
\end{figure}
Sur la figure \ref{medor}vous pouvez voir
la photo de Médor.
```

La numérotation sera effectuée automatiquement par \LaTeX . Le contenu de la commande "caption" peut prendre du texte de la longueur que vous voulez, elle peut rester vide si vous ne désirez pas placer de légende, seule la numérotation apparaîtra. Vous pouvez également y faire des sauts de ligne avec la commande `\newline` (étonnamment la commande `\\` génère des erreurs et je ne sais pas pourquoi).

Pour centrer la figure, vous pouvez bien évidemment placer l'environnement "figure" dans un environnement "center", mais vous pouvez également utiliser la commande `\centering` :

```
\begin{figure}[ht]
\centering
\includegraphics{medor.jpg}
\caption{Voici la photo de Médor prise en
juin 02}
\label{medor}
\end{figure}
Sur la figure \ref{medor}vous pouvez voir
la photo de Médor.
```

Bien entendu, il est souvent nécessaire de manipuler un peu l'image car elle est trop grande ou trop petite. Ceci se fait très simplement en passant des options à la commande `\includegraphics` :

```
\includegraphics[option=valeur]{monimage.jpg}
```

où "option" peut prendre les valeurs suivantes :

option	valeur
scale	facteur compris en 0 et 1
angle	angle en degrés (0 à 360)
draft	true ou false
width	dimension
height	dimension

Il y en a en fait d'autres, mais qui sont plus poussées et inutiles dans la plupart des cas (je ne les ai jamais utilisées).

L'option "scale" permet une mise à l'échelle de l'image ou facteur est le rapport avec l'image originale.

L'option "angle" permet une rotation de l'image et l'angle est donné en degrés (dans le sens inverse des aiguilles d'une montre).

L'option "draft" permet d'afficher un rectangle vide en lieu et place de l'image. Les dimensions seront toutefois celles de l'image.

Les options "width" et "height" (largeur et hauteur) prennent comme valeur une dimension. Cette dimension peut être donnée de plusieurs manières.. En centimètres bien sûr, par exemple "10cm", en pouces : "5in" ou en points "500pt", mais aussi en fonction de la taille de la page ou du texte grâce aux commandes `\paperwidth` et `\textwidth` (respectivement `\paperheight` et `\textheight`). La première commande donne la taille de la page (marges y comprises) tandis que la deuxième donne la taille effective du texte (sans les marges). Ceci est très utile pour avoir une image qui à la même largeur (ou longueur) que le texte, mais aussi une fraction de cette dimension. En effet, vous pouvez très bien ajouter un facteur devant ces commandes :

```
\includegraphics[width=0.75\textwidth]{monimage.jpg}
```

Ainsi, l'image aura une largeur égale aux trois quarts de la largeur du texte.

Il existe encore de nombreuses fonctions d'image dans différents paquets de commandes pour L^AT_EX, mais celles-ci sont beaucoup plus poussées. Ce que je viens de vous décrire devrait suffire pour la plupart des cas (je n'ai jamais utilisé d'options plus poussées).

Il existe un équivalent de la table des matières (que nous avons découvert la dernière fois) pour les figures. Pour créer une table des figures, il suffit d'entrer la commande `\listoffigures`, là où vous désirez voir apparaître cette liste (habituellement en début ou en fin de document).

4.3 Insérer un tableau

La création de tableaux est très puissante en \LaTeX . Pour créer un tableau il faut utiliser l'environnement "tabular". Voici un exemple simple :

```
\begin{document}
Voici un tableau :
\begin{tabular}{lcr}
Ligne 1 & Element 1 & Element 2 \\
Ligne 2 & Element 3 & Element 4 \\
Ligne 3 & Element 4 & Element 5 \end{tabular}
\end{document}
```

Etudions cet exemple en détail. Tout d'abord, on commence l'environnement "tabular" grâce à la commande `\begin`. Vous noterez que la syntaxe est un peu particulière puisqu'elle prend deux arguments (entre accolades). Tout d'abord le nom de l'environnement (tabular) puis une suite de caractères. Cette chaîne de caractères (ici "lcr") définit deux choses : le nombre de colonnes ainsi que l'alignement des éléments de chacune de ces colonnes.

Dans notre exemple, "lcr" signifie que notre tableau possède trois colonnes (trois caractères). Il signifie aussi que la première colonne aura des éléments alignés à gauche ("l" pour left), que les éléments de la deuxième colonne seront centrés et que ceux de la troisième colonne seront alignés à droite ("r" pour right).

On entre ensuite les éléments. Pour passer à la colonne suivante, on utilise le caractère "&", pour terminer une ligne du tableau, on utilise la commande `\\`, sauf pour la dernière ligne qui se termine par le `\end`.

Si vous compilez cet exemple, vous obtiendrez ceci :

Ligne 1	Element 1	Element 2
Voici un tableau :	Ligne 2	Element 3 Element 4
	Ligne 3	Element 4 Element 5

On y remarque deux choses :

- La création d'un tableau n'implique pas de saut de ligne. Notre tableau est placé à la suite de la phrase "Voici un tableau :"
- Le tableau n'est pas délimité par des barres horizontales et verticales.

Pour palier à la première constatation et mettre le tableau sur une nouvelle ligne vous pouvez simplement ajouter un `\\` à la fin de la phrase. Mais il y a encore mieux, vous pouvez faire d'une pierre deux coups en utilisant l'environnement "center" que nous avons défini dans la partie précédente. Le

tableau sera alors placé à la ligne et centré.

Pour ajouter des barres verticales à notre tableau nous allons modifier la chaîne de caractères définissant le tableau. Si nous voulons séparer la première colonne des deux autres par une barre verticale nous allons transformer "lcr" en "l|cr". Ce caractère vertical est obtenu sur un clavier suisse-romand par alt-7 (sur un clavier français il semble que ce soit alt-shift-l). Si nous voulons séparer toutes les colonnes par une barre verticales, nous utiliserons "l|c|r", et enfin pour que notre tableau soit délimité par deux barres verticales nous utiliserons "|l|c|r|". Logique. Bien sûr, vous pouvez utiliser plusieurs barres à la suite : "|l|c|r|".

Pour obtenir une barre horizontale, nous utilisons la commande `\hline`, que nous placerons entre deux lignes. Notre code deviendra par exemple :

```
\begin{document}
Voici un tableau :
\begin{center}
\begin{tabular}{|l|c|r|}
\hline
Colonne 1 & Colonne 2 & Colonne 3\\
\hline
\hline
Ligne 1 & Element 1 & Element 2\\
\hline
Ligne 2 & Element 3 & Element 4\\
\hline
Ligne 3 & Element 4 & Element 5\\
\hline
\end{tabular}
\end{center}
\end{document}
```

Voici son résultat :

Colonne 1	Colonne 2	Colonne 3
Ligne 1	Element 1	Element 2
Ligne 2	Element 3	Element 4
Ligne 3	Element 4	Element 5

Chaque élément du tableau peut contenir ce que vous voulez ! Du texte bien sûr, mais aussi un image (grâce à un `\includegraphics`) ou même un autre tableau ! Essayez donc le code suivant :

```

\begin{document}
Voici un tableau :
\begin{center}
\begin{tabular}{|l||c|r|}
\hline
Colonne 1 & Colonne 2 & Colonne 3 \\
\hline
\hline
Ligne 1 &
\begin{tabular}{|c|c|}
\hline
Tableau & dans \\
\hline
un & tableau \\
\hline
\end{tabular}
& Element 2 \\
\hline
Ligne 2 & Element 3 & Element 4 \\
\hline
Ligne 3 & Element 4 & Element 5 \\
\hline
\end{tabular}
\end{center}
\end{document}

```

Voici le résultat :

Colonne 1	Colonne 2		Colonne 3
Ligne 1	Tableau	dans	Element 2
	un	tableau	
Ligne 2	Element 3		Element 4
Ligne 3	Element 4		Element 5

Pas mal, non ?

Comme pour les images, les tableaux peuvent être définis comme des éléments flottants auxquels on pourra alors faire référence. Pour les tableaux, on utilisera l'environnement "table" au lieu de l'environnement "figure". La syntaxe reste la même. Par exemple :

```

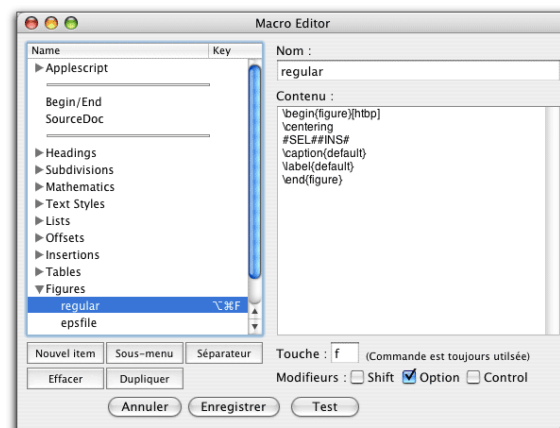
\begin{table}[ht]
\centering
\begin{tabular}{lcr}
Ligne 1 & Element 1 & Element 2 \\
Ligne 2 & Element 3 & Element 4 \\
Ligne 3 & Element 4 & Element 5
\end{tabular}
\caption{Mon beau tableau}
\label{beautableau}
\end{table}

```

On pourra alors faire référence à ce tableau grâce à un `\ref{beautableau}`.

Et comme pour les figures flottantes, il est possible de créer une liste des tableaux flottants qui se présentera comme une table des matières. Ceci se fait grâce à la commande `\listoftables`.

Bien sûr, tous les environnements que nous venons de voir sont accessibles depuis le menu Macros de *TeXShop*. J'en profite pour vous signaler que vous pouvez associer des raccourcis-clavier aux environnements que vous utilisez fréquemment. Pour cela, vous trouverez dans le menu Macros, l'élément "Ouvrir l'éditeur des macros...". Vous obtiendrez cette fenêtre.



Pour ajouter un raccourci à une macro, entrez un caractère dans le champ "Touche" et éventuellement un modifieur. Par exemple, j'ai ici associé commande-option-f à l'insertion de l'environnement "figure". Vous pouvez par ailleurs également dans cet éditeur modifier la macro ou en ajouter d'autres.

Je tiens à insister sur cet éditeur de macros car c'est lui qui vous rendra la vie plus facile! Bien entendu, il est très rébarbatif d'entrer ces antislashes et ces "begin" et "end". Mais les macros sont là pour le faire à votre place,

alors profitez-en. Pour ma part, tous les environnements que nous avons vus jusqu'à présent ont un raccourci-clavier, ce qui rend la rédaction du document beaucoup plus aisée.

Vous avez à présent de quoi vous amuser avec les tableaux et les images. Essayez donc de mettre des images dans un tableau, de placer plusieurs images et tableaux pour voir la numérotation utilisée par \LaTeX , de faire des tableaux dans des tableaux dans des tableaux, etc...

5 Les mathématiques

Une introduction à \LaTeX ne saurait être complète si je ne parlais pas des mathématiques, c'est en effet une des raisons pour lesquelles ce programme fut créé.

J'ai toutefois fait exprès d'aborder le sujet assez tard dans ce document, pour bien mettre l'accent sur le fait que \LaTeX ne sert pas uniquement à écrire des mathématiques.

Comme je l'ai écrit en [commentaires](#) du [test de Renan](#) sur MathMagic, je considère qu'il est totalement improductif de devoir écrire à la souris. Et pourtant, c'est ce que les éditeurs d'équations vous proposent (que ce soit celui de Word, MathMagic, ou un autre). Je pense pour ma part qu'écrire mes mathématiques sur mon clavier, sans le quitter est la situation optimale. De plus, \LaTeX offre le meilleur rendu d'équations qu'il soit possible d'obtenir (bien sûr, la beauté est toujours subjective). Vous pouvez aller voir un [article](#) publié par le service informatique de l'[EPFL](#) qui compare entre autres le rendu des équations avec Word, Framemaker et \LaTeX .

L'écriture des mathématiques dans \LaTeX est très très complète, et il me sera donc difficile d'être exhaustif.

Les mathématiques peuvent s'écrire soit dans le texte, soit dans des environnements à part. Pour écrire une équation dans le texte, il faut simplement l'entourer de deux $\$$. Par exemple :

Einstein a écrit $E=mc^2$

Vous le voyez, l'écriture d'une équation est assez naturelle. La mise en exposant se fait comme en programmation avec le signe $^$. Essayons quelque chose de plus compliqué :

Mais il a dit aussi $E=\sqrt{p^2c^2+m_0^2c^4}$

On y voit d'abord la commande \sqrt qui dessine une racine carrée (square root en anglais). Puis les mises en exposant, mais aussi une mise en indice qui se fait grâce au caractère $_$. Les mises en indice ou en exposant ne prennent *a priori* que le premier caractère qui les suit (sauf si c'est une commande). Pour mettre plusieurs caractères, il faut utiliser les accolades :

$2^2+6=256$

donnera $2^2 + 6 = 256$ et donc une équation fautive, tandis que

$$2^{2+6}=256$$

donnera $2^{2+6} = 256$.

Pour écrire une fraction, deux solutions s'offrent à vous :

Ecrire $\frac{x}{y}=2$ équivaut à écrire $x \over y=2$

Bien entendu, toutes ces opérations peuvent s'imbriquer les unes dans les autres, mais cela donne lieu à des équations assez conséquentes. Il ne sera donc pas conseillé de les placer dans le texte, mais plutôt sur une nouvelle ligne.

Pour cela, nous allons placer l'équation dans un environnement "equation" (qui l'eut cru ?). Par exemple :

```
\begin{equation}
\frac{x^{\left(\sqrt{p^2c^2+m_0^2c^4}-y_0^4\right)}-2y+8}{(y+x)^2}=4K_1+3x
\end{equation}
```

donnera

$$\frac{x^{\left(\sqrt{p^2c^2+m_0^2c^4}-y_0^4\right)}-2y+8}{(y+x)^2}=4K_1+3x$$

J'attire votre attention sur la qualité de l'équation produite.

Vous aurez noté l'utilisation de `\left(` et de `\right)`. En effet, pour placer des parenthèses, il suffit d'écrire des parenthèses (sans blague ?). Mais si vous entrez ceci :

$$2\left(\frac{x}{y}\right)=8$$

vous obtiendrez ça :

$$2\left(\frac{x}{y}\right)=8$$

ce qui n'est pas très joli. En utilisant `\left(` et `\right)`, L^AT_EX adapte lui-même la taille des parenthèses au contenu :

$$2\left(\frac{x}{y}\right)=8$$

Ce qui est bien mieux. Bien entendu, ceci s'applique également aux crochets ([et]) et aux accolades. Par contre pour ces dernières, leur utilisation

"simple" se fera avec les commande `\{` et `\}`. En effet, l'accolade est réservée par \LaTeX pour les arguments des commandes.

Revenons à notre environnement "equation". Vous remarquerez que l'équation est automatiquement numérotée. Vous pouvez donc évidemment lui faire référence. Ceci se passe toujours de la même manière, à savoir utiliser un `\label` pour l'étiqueter et un `\ref` pour y faire référence :

```
\begin{equation}
2\left(\frac{x}{y}\right)=8\label{eq:xy}
\end{equation}
L'équation \ref{eq:xy} nous donne la
relation entre  $x$  et  $y$ .
```

Si vous désirez ne pas numéroté une équation, vous la placerez simplement dans l'environnement "displaymath". Mais vous ne pourrez alors pas y faire référence.

Pour abrégé, on peut remplacer les `\begin{displaymath}` et `\end{displaymath}` par `\[` et `\]` :

```
Voici une équation :
\[
2\left(\frac{x}{y}\right)=8
\]
```

L'environnement "equation" (et "displaymath") ne permet l'écriture que sur une seule ligne. Pour écrire plusieurs équations (sur plusieurs lignes) vous pouvez utiliser l'environnement "eqnarray". Ce dernier a l'avantage de vous permettre d'aligner vos équations. Il fonctionne en effet comme un tableau (que nous avons vu dans la partie précédente) à trois colonnes avec un alignement "rcl" :

```
\begin{eqnarray}
2 \left( \frac{x}{y} \right) & = & 8 \\
\frac{x}{y} & = & \frac{8}{2} \\
& = & 4
\end{eqnarray}
```

Vous remarquerez que comme pour un tableau, on utilise le caractère "&" pour aligner les éléments. Dans cet exemple, on s'assure que les trois signes "=" seront alignés verticalement :

$$2 \left(\frac{x}{y} \right) = 8$$

$$\frac{x}{y} = \frac{8}{2}$$

$$= 4$$

Dans l'environnement "eqnarray", chaque ligne est numérotée. Pour y faire référence, il s'agit de placer un `\label` à la fin de chaque ligne (avant le `\`). Si vous ne désirez pas numéroté les équations, il faut utiliser l'environnement "eqnarray*".

Pour entrer du texte dans une équation, vous devez utiliser des commandes spéciales. Il y en a une pour chaque style (les styles "slanted" et "petites capitales" ne sont pas disponibles en mode mathématique) :

Italique	<code>\mathit{texte}</code>
Gras	<code>\mathbf{texte}</code>
Machine à écrire	<code>\mathtt{texte}</code>
Sans serif	<code>\mathsf{texte}</code>
Roman	<code>\mathrm{texte}</code>

Par exemple, pour écrire l'énergie potentielle on entrera :

```
\begin{equation}
E_ \mathrm{pot.} = mgh
\end{equation}
```

Cas particulier, pour écrire les fonctions habituelles :

tangente	<code>\tan</code>
cosinus	<code>\cos</code>
sinus	<code>\sin</code>
log naturel	<code>\ln</code>
logarithme	<code>\log</code>

et j'en passe (fonctions hyperboliques, fonctions trigonométriques inverses, etc...).

Pour écrire une matrice (et donc un vecteur), on utilise l'environnement "array". Il fonctionne exactement comme l'environnement "tabular" :

```
\begin{equation}
A = \left( \begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array} \right)
\end{equation}
```

donnera :

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

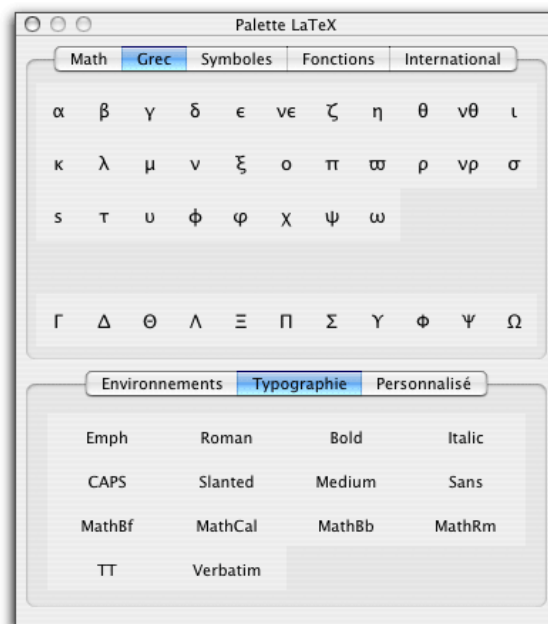
Vous pouvez évidemment placer n'importe quel élément dans cette matrice, que ce soit un chiffre, une équation ou une autre matrice !

Il est commun en mathématique d'utiliser des lettres grecques pour désigner certaines variables du problème. Avec \LaTeX , il vous suffit de connaître votre alphabet grec (vous en connaissez déjà deux lettres : alpha et beta qui ont donné leur nom à l'alphabet). En effet, il vous suffit d'entrer un antislash suivi du nom de la lettre. Si vous mettez une majuscule au nom de la lettre grecque vous obtiendrez sa version majuscule, si vous mettez une minuscule vous obtiendrez sa version minuscule :

```
\begin{equation}
2\pi+\Omega=\xi\cos(\theta+\phi)
\end{equation}
```

$$2\pi + \Omega = \xi \cos(\theta + \phi)$$

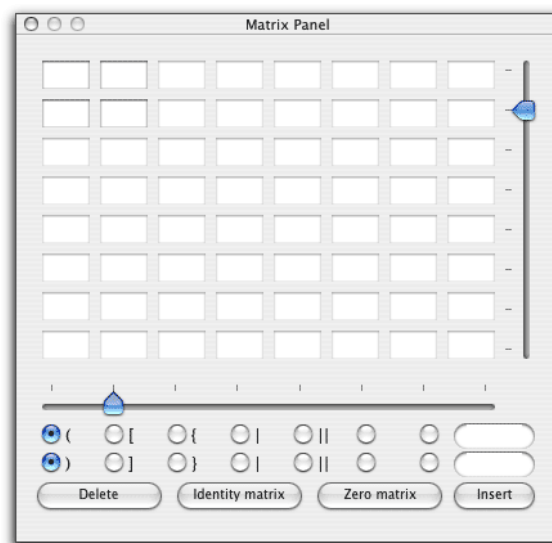
Mais comme on n'attend pas de vous que vous sachiez votre alphabet grec sur le bout des doigts, *TeXShop* vous met à disposition la "Palette \LaTeX ". Vous la trouverez dans le menu "Fenêtres" ou avec un pomme- \equiv :



Dans la partie supérieure de cette palette, vous trouvez cinq onglets destinés aux mathématiques avec tout d'abord les symboles spéciaux (constante de Planck, opérateurs d'ensembles, l'ensemble grand R, etc..), puis les lettres grecques, puis quelques symboles (opérateurs logiques, parenthèses, etc..), puis les fonctions trigonométriques dont je vous parlais à l'instant. Le dernier onglet est peu important puisqu'il vous permet d'écrire les accents "à l'ancienne", comme je le mentionnais dans la section 2.2.

La partie inférieure de cette "Palette LaTeX" possède trois onglets. Le premier reprend les environnements les plus courants (center, figure, table, tabular, itemize, etc...). Le deuxième vous donne les différents styles de texte, en mode texte et en mode mathématique. Enfin, le dernier onglet est une aide à la structure du document, dans lequel vous retrouvez les sections, le titre, etc...

Il existe également une deuxième palette, la palette "Matrix" (rien à voir avec le film) :



Cette palette est une aide à la création de matrices. Vous pouvez y définir la taille de la matrice, son contenu, ainsi que les contenants (parenthèses, crochets, norme, etc...)

Ces palettes sont donc une alternative ou un complément aux macros. C'est selon votre goût personnel. Pour ma part, comme je l'ai mentionné plus haut, je fais tout au clavier, les environnements que j'utilise le plus ont donc tous un raccourci-clavier.

Bien entendu, je pourrais encore écrire des kilomètres à propos des équations, tant les possibilités sont immenses. Mais je préfère vous en donner un petit aperçu et je vous donnerai au paragraphe 6.4 une série de liens vers des documents de référence qui vous aideront à approfondir le sujet.

6 Compléments et conclusion

Pour cette dernière partie de cette série d'articles consacrés à \LaTeX , je vais tout d'abord m'attarder sur quelques oublis qu'on m'a signalés en commentaires des articles parus sur [Cuk](#). Puis je vais aborder deux aspects propres à "pdflatex" et enfin je vous ferai part de mes conclusions et de mes réflexions. Enfin vous trouverez plusieurs liens vers des sites ou des documents qui vous permettront d'approfondir votre apprentissage de \LaTeX .

6.1 Les oublis et les compléments

6.1.1 Les césures

Tout d'abord, je remercie Franck et Mina (Nicolas) et tous ceux qui sont intervenus en commentaires de mes articles pour me signaler des oublis ou me mentionner des imprécisions ou des erreurs. En effet, j'ai oublié un point essentiel de \LaTeX : les césures.

Les césures sont automatiques avec \LaTeX . Il en connaît les règles françaises à partir du moment où on lui a indiqué d'utiliser le paquet de commandes "babel" avec l'option "french" (à ce sujet, je vous conseille de lire le commentaire de Franck, qui explique les différentes options de francisation).

Toutefois, avec les mots accentués, \LaTeX ne fera pas les césures correctes. Pour corriger cela, il faut utiliser un nouveau paquet de commandes :

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

Avec ce paquet de commandes, \LaTeX utilisera les polices CM-Super que je vous avais conseillé d'installer dans la première partie. Et avec ces polices, les césures seront correctes.

Mais s'il arrivait que \LaTeX ne parvienne pas à faire une césure dans un mot, ou qu'il en fasse une fausse, vous pouvez toujours le corriger en lui indiquant où il a le droit de couper un mot. Ceci se fait avec la commande `\-`. Par exemple, si \LaTeX a du mal à trouver où faire la césure dans le mot "internet", vous remplacerez ce mot par :

```
in\ -ter\ -net
```

\LaTeX saura ainsi quels sont les endroits autorisés pour faire la césure.

6.1.2 L'orthographe

TeXShop étant une application Cocoa, elle profite pleinement de la correction orthographique mise à disposition par Apple. Toutefois, il est assez pénible

de voir tous les "section", "chapter", "begin", "end" et autres "usepackage" soulignés en rouge. Pour contourner cela vous pouvez installer les dictionnaires de [cocoASpell](#) qui reconnaît les commandes \LaTeX et sait ce qu'il faut vérifier et ce qu'il n'est pas nécessaire de vérifier.

6.1.3 La grammaire

François le demandait en commentaires. Non, la correction grammaticale n'est *a priori* pas disponible. En effet, l'existence de toutes ces commandes rend la correction difficile. Finalement, c'est le même problème pour l'HTML. Mais les solutions existent. Ou plutôt, les "trucs" existent.

Je vous propose deux solutions. Tout d'abord, la majeure partie du texte est "brut", elle peut donc être sans autre copiée pour être introduite dans un vérificateur. Le problème se posera lorsqu'il y aura des modificateurs de style (italiques, gras, etc.).

La deuxième solution que je vous propose utilise les capacités du nouvel *Aperçu* de Panther. En effet, ce dernier vous offre la possibilité de copier le texte contenu dans un document PDF. Vous pouvez donc ouvrir votre document créé par \LaTeX , tout sélectionner et l'envoyer au correcteur. Cela fonctionne même avec les services !

Toutefois, il est évident que ce n'est pas la solution idéale, mais c'est pour l'instant la seule que l'on a.

6.1.4 La fragmentation

Encore un sujet abordé dans les commentaires. Lorsqu'on travaille sur un document conséquent ou collaboratif (plusieurs auteurs), il est parfois nécessaire de le fragmenter. J'entends par là avoir par exemple un fichier distinct par chapitre.

Ceci se fait grâce à la commande `\include`. Par exemple :

```
\begin{document}
\chapter{Introduction}
bla bla bla...
\include{chapitre1}
\include{chapitre2}
\include{conclusion}
\end{document}
```

Vous placerez alors les fichiers "chapitre1.tex", "chapitre2.tex" et "conclusion.tex" dans le même dossier que le fichier principal.

Il va sans dire que la numérotation des chapitres, la table des matières, les références et tout le tintouin seront parfaitement gérés par \LaTeX .

6.1.5 Les images EPS

Je vous l'ai dit, pour placer des images, les formats acceptés sont jpeg, png et pdf. En fait, ceci est vrai car nous utilisons le compilateur pdflatex, qui crée un document PDF. Mais je vous ai également dit dans la première partie, que le compilateur \LaTeX crée en fait un fichier dvi qui est ensuite converti en PostScript. Et dans ce cas, le seul format d'image accepté est le format EPS (Encapsulated PostScript). C'est un format encore très utilisé, en particulier pour tout ce qui est vectoriel. C'est également le format dans lequel *Illustrator* sauve ses fichiers jusqu'à la version 8.

Pour insérer des images EPS dans son document et quand même obtenir un PDF à l'arrivée, *TeXShop* vous propose de compiler votre document avec \LaTeX puis de convertir le fichier PostScript ainsi obtenu en PDF.

Pour cela, vous devez vous rendre dans le menu "Composer". Dans la troisième partie de ce menu, vous verrez une coche devant "Pdftex", c'est le réglage par défaut. Sélectionnez l'élément "TeX et Ghostscript", pour que la compilation se fasse avec \LaTeX puis que le résultat soit converti en PDF.

Cette option porte mal son nom, car Ghostscript n'est plus nécessaire pour la conversion PS -> PDF. En effet, Panther inclut son propre convertisseur (qui est utilisé par *Aperçu* pour afficher les PS). Il vous faut juste aller avertir *TeXShop* que vous ne voulez pas utiliser Ghostscript. Dans les préférences, dans l'onglet "Misc", sous "Distiller" vous devez cocher "Panther Distiller". Bien entendu, cela ne fonctionne que sous Panther.

6.1.6 Les macros

Les macros sont votre meilleur allié. *TeXShop* vous met à disposition un certain nombre de macros prédéfinies, nous en avons vu plusieurs au court des différentes parties. Vous pouvez également modifier ces macros (par exemple ajouter l'utilisation du paquet "babel" dans la macro d'insertion des en-têtes) et en ajouter autant que vous voulez. Par exemple, il manque une macro pour centrer un bout de texte, ajoutez-la dans l'éditeur de macros grâce au bouton "Nouvel item". Donnez-lui un nom et entrez ceci dans "Contenu" :

```
\begin{center}  
#SEL##INS#  
\end{center}
```

Le `#SEL#` sera remplacé par la sélection. Si par exemple vous avez déjà écrit un bout de texte et que vous désirez le centrer, vous le sélectionnez et lancez la macro que vous avez créée, la sélection sera alors placée dans l'environnement "center".

Le `#INS#` définit l'emplacement du point d'insertion. C'est là que votre curseur se trouvera après que vous ayez lancé la macro.

6.2 Compléments propres à pdflatex

6.2.1 hyperref

Je vous l'ai dit, *TeXShop* ne compile pas avec \LaTeX mais avec pdflatex. La conséquence de cela est que le fichier final est un document PDF et non pas un fichier PS (après transformation du DVI).

Tout ce qui a été dit dans les parties précédentes était des généralités de \LaTeX (sauf pour les images qui doivent être au format eps lorsqu'on compile avec \LaTeX).

Mais le PDF est un format très très riche, capable de faire bien plus que bêtement afficher un texte et des images.

Tout d'abord, un document PDF peut contenir une table des matières, permettant la navigation dans le document. C'est le tiroir qui s'ouvre sur un des côtés de la fenêtre d'*Aperçu*.

Mais un document PDF peut également contenir des hyperliens ! Ne serait-il pas fantastique de pouvoir cliquer sur un élément de la table des matières et de s'y voir instantanément transporté ? Ne serait-il pas génial de pouvoir cliquer sur la référence d'une image et de la voir immédiatement apparaître ? Ne serait-il pas hypercool de pouvoir cliquer sur les mots "Cuk.ch" dans un document PDF et de voir s'ouvrir notre site préféré dans un navigateur ?

Tout cela est possible avec pdflatex ! Il suffit d'ajouter le paquet de commandes "hyperref" :

```
\usepackage{hyperref}
```

Sans rien faire d'autre, la table des matières deviendra automatiquement des hyperliens qui mèneront au chapitre ou à la section choisie, les références meneront directement au tableau, à la figure, au chapitre ou à l'équation correspondante, les notes de bas de page deviendront également des hyperliens, etc...

Mais il sera également possible de créer de véritables hyperliens dans le document, grâce à la commande `\href` :

```
Voici un lien vers le fantastique site \href{http://www.cuk.ch}{Cuk}
```

Le premier argument donne l'URL à atteindre et le deuxième argument donne le texte à afficher. Il est également possible de placer une image dans le deuxième argument au lieu d'un texte, voire même une équation.

On peut regretter que ces hyperliens ne soient pas actifs dans la prévisualisation de *TeXShop*. Ils sont présents dans *Aperçu*, mais ne ressortent pas, c'est uniquement le changement de pointeur qui vous indique qu'un lien est "cliquable". Ils apparaîtront par contre encadrés dans *Adobe Reader*.

Mais vous pouvez également mettre une couleur à ces liens pour les rendre plus apparents. Pour cela, il vous faut d'abord charger le paquet de commandes permettant de mettre un peu de couleur dans vos documents :

```
\usepackage{color}
```

Placez-le avant "hyperref", celui-ci doit toujours être le dernier paquet de commandes chargé !

Puis il s'agit de configurer le paquet "hyperref" pour qu'il utilise des couleurs. Ajoutez ceci avant le `\begin{document}` :

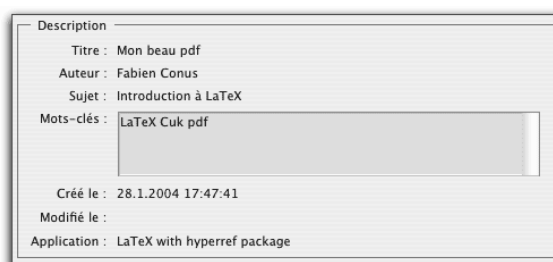
```
\hypersetup{colorlinks = true}
```

Il est possible de personnaliser les couleurs, mais je n'entrerai pas dans les détails ici.

Un document PDF contient également des informations sur l'auteur, le titre du document, etc. Vous pouvez entrer ces informations avec "hyperref" :

```
\hypersetup{
pdftitle = Mon beau pdf,
pdfauthor = Fabien Conus
pdfsubject = Introduction à LaTeX
pdfkeywords = LaTeX Cuk pdf
}
```

Si avec *Adobe Reader* vous lisez alors les informations du document (pomme-D) vous obtiendrez ceci :

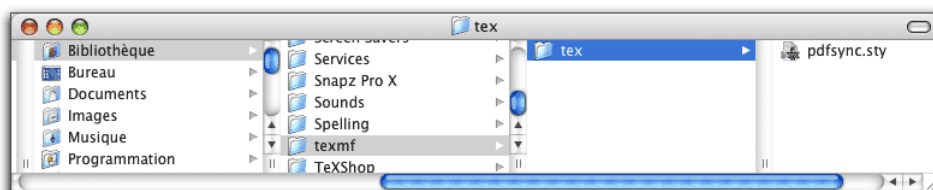


Le paquet de commandes "hyperref" est très complet et très riche, vous trouverez un manuel complet sur [ce site](#).

6.2.2 pdfsync

Il y a une chose ennuyante avec \LaTeX . Vous compilez votre document, tout se passe bien, mais en parcourant le fichier PDF vous remarquez une faute (de frappe, d'orthographe, peu importe). Selon la taille du document, il peut être pénible de retrouver l'erreur dans le code \LaTeX . Heureusement, il y a "pdfsync". Ce paquet de commandes est encore expérimental, mais il fonctionne déjà très bien. Le principe est simple : une fois que vous avez repéré l'erreur dans le fichier PDF, vous faites un pomme-clic dans le pdf à l'endroit de l'erreur et *TeXShop* vous amène directement à la ligne correspondante dans le code.

Pour mettre ce système en oeuvre, il faut d'abord l'installer, car il ne vient pas (encore) avec l'installation de \LaTeX . Lorsque vous montez l'image-disque de *TeXShop*, comme nous l'avons fait dans la première partie, vous trouvez un fichier intitulé "pdfsync.sty". Les fichiers ".sty" sont des paquets de commandes pour \LaTeX . Il faut donc placer ce fichier à un endroit que \LaTeX connaît et où il sait qu'il doit aller chercher les paquets de commandes. Allez dans votre dossier "Bibliothèque". Regardez si vous avez un dossier "texmf", si ce n'est pas le cas, créez-le. Dans ce dossier "texmf" créez un nouveau dossier "tex" et placez-y le fichier "pdfsync.sty". En résumé vous devez avoir ceci :



Ensuite, dans votre document \LaTeX vous devez encore indiquer que vous désirez utiliser ce paquet de commandes :

```
\usepackage{pdfsync}
```

(je vous rappelle que "hyperref" doit toujours être le dernier paquet de commandes chargé).

Et voilà, "pdfsync" est actif. Si, lors de la compilation, vous avez un message d'erreur indiquant qu'il ne trouve pas "pdfsync", essayez de relancer *TeXShop*.

6.3 Conclusion

Ouf! Nous voilà arrivés au terme de cette petite introduction à \LaTeX sous MacOS X. J'insiste sur le fait que c'est une courte introduction, il manque plein de choses et je ne suis pas allé au fond des sujets traités, ce n'est pas le but. Il existe beaucoup de ressources sur internet qui vous permettront d'approfondir les différents sujets et de devenir un pro de \LaTeX , je vous donnerai quelques liens plus bas.

J'aimerais profiter de cette conclusion pour reprendre un peu le débat qui à commencé dans les commentaires. Avez-vous besoin de \LaTeX ? \LaTeX est-il fait pour vous?

\LaTeX vous permet d'écrire sans peine des documents conséquents, avec figures, tableaux, références, et vous offre une mise en pages à l'allure très professionnelle.

Bien sûr, cela implique un apprentissage, mais pour faire des documents complexes sous *Word* ou *FrameMaker*, cela nécessite également un apprentissage.

Ceux qui me diront "Mais moi je ne fait pas de gros documents, juste de la correspondance et quelques brouilles". Alors ceux-là ont gaspillé leur argent! *Word* coûte 400 CHF (266 €), *FrameMaker* coûte 2430 CHF, soit 1620 € (1750 CHF en version anglaise, soit 1170 €). \LaTeX est complètement gratuit! Si c'est pour faire de la correspondance, vous avez meilleur temps d'utiliser *TextEdit*, en plus il sait lire et enregistrer les documents *Word*!

Je pense ne pas mentir en disant que \LaTeX est la solution de création de documents PDF la moins chère possible! De plus, il permet également la création de documents PDF dynamiques comme je viens de vous le montrer.

Evidemment \LaTeX n'est pas WYSIWYG (*rappel* : what you see is what you get, ce que vous voyez est ce que vous obtenez), mais c'est un mal pour un bien. C'est grâce à ça que vous obtenez cette très belle mise en pages sans vous fatiguer. Vous me direz que tous les documents \LaTeX se ressemblent. Et bien c'est faux. Oui, vos premiers documents auront tous la même allure. Mais vous apprendrez très vite à redéfinir la taille des marges, à changer

de police, à personnaliser votre mise en page (par exemple avec le paquet de commandes "fancy headers"). Ce qui est bien, c'est qu'une fois que vous êtes satisfaits de votre mise en page, vous n'avez qu'à garder la même entête pour tous vos documents, vous ne modifierez que ce qui se trouve entre `\begin{document}` et `\end{document}`. J'ai lu sur un site que L^AT_EX c'était du YAFIYGI, You asked for it, you got it : "tu l'as demandé, tu l'as eu".

Vous me direz que jamais vous ne parviendrez à vous souvenir de toutes ces commandes. Alors je vais encore une fois insister sur l'utilisation des macros. *TeXShop* est là pour vous simplifier la vie, il vous offre toute une série de macros auxquelles vous pouvez associer des raccourcis clavier. Chez moi pomme-alt-f m'insère automatiquement tout ce qu'il me faut pour inclure une figure, pomme-alt-e me permet d'insérer une équation, etc.

Si vous voulez créer des documents que vous souhaitez diffuser, ou si vous faites de l'écriture collaborative, L^AT_EX sera tout à fait approprié. J'ai passé une année à travailler avec quelques collègues pour un prof qui ne jurait que par *Word*. Et vous ne pouvez pas imaginer les heures de cauchemar pour parvenir à lire correctement les documents que mes collègues (sous Windows) m'envoyaient (et inversement). Et lorsqu'on parvenait à les lire, encore fallait-il pouvoir les imprimer !

L^AT_EX créé du PDF, format lisible sur toutes les plateformes connues (sauf les consoles de jeux, et encore !). De plus le code L^AT_EX est un bête format texte en ASCII, il est donc également lisible sur toutes les plateformes et comme L^AT_EX est un logiciel libre il tourne également sur toutes les plateformes. La compatibilité est donc totale, pour le résultat comme pour la source.

Ce qui soulève un autre sujet, celui de la pérennité des documents. Nous l'avons vu récemment avec la mésaventure de *FrameMaker* sous MacOS X, les logiciels commerciaux n'ont pas une survie assurée, et les documents sauvés dans les formats propriétaires associés à ces programmes ne seront peut-être plus lisibles dans quelques années. Evidemment, il est difficile de croire que *Word* n'existera plus dans, disons, 25 ans. Mais qui parvient aujourd'hui à lire le format *Visicalc*, qui était pourtant le logiciel phare des débuts de la micro-informatique ? Un code L^AT_EX est en ASCII, le format le plus simpliste. Sa pérennité est donc assurée. De plus, le code source de L^AT_EX étant disponible à tous, aucune firme ne pourra décider d'arrêter de le publier.

Finalement, pour répondre aux deux questions que je mentionnais plus haut, la meilleure manière de savoir si L^AT_EX est fait pour vous, c'est d'essayer. C'était le but de cette présentation simpliste, vous avez toutes les clefs en main pour faire de beaux documents. Essayez-le et faites-vous votre propre

opinion. Pour ma part, j'ai essayé *Word*, j'ai fait deux ou trois rapports avec et je me suis vite rendu compte que je n'allais pas vivre vieux si je continuais avec ce programme.

Ceci met un terme à cette présentation de \LaTeX . Il n'est pas impossible que je réécrive un ou deux articles ultérieurement pour aborder d'autres questions comme la gestion de la bibliographie, les en-têtes personnalisés, convertir un document \LaTeX en HTML, etc.

Je vous propose encore quelques liens qui vous permettront d'approfondir votre découverte de \LaTeX .

6.4 Quelques liens

En français :

- La référence pour apprendre \LaTeX : [Une courte \(?\) introduction à \$\text{\LaTeX}\$](#) .
- Paul Salort : [\$\text{\LaTeX}\$ pour les débutants sous MacOS X](#). Excellent chapitre sur les polices.
- Benoît Rivet : [teTeX sous MacOS X](#), installation de paquets, utilisation des polices.
- Nicolas Seriot : [trucs et astuces pour \$\text{\LaTeX}\$](#) . Au passage, essayez le thème Kuc sur son site.
- la [FAQ LaTeX de l'équipe Grappa](#)
- Enrico sur ozone.ch propose [une page sur \$\text{\LaTeX}\$](#) avec plein de liens intéressants.
- xrings : [Installer \$\text{\LaTeX}\$ sur son Mac](#) et [Utiliser \$\text{\LaTeX}\$ sur son Mac](#).
- [Apprendre \$\text{\LaTeX}\$ chez Loria](#). Attention, c'est "à la dure", au Terminal.
- [Raccourcis clavier avec TeXShop](#), par Daniel Martin, transmit par une lectrice de Cuk.

En anglais :

- [LA page de référence de \$\text{\LaTeX}\$ sous MacOS X](#).
- [MacOS X TeX/ \$\text{\LaTeX}\$ Web Site](#)
- [\$\text{\LaTeX}\$ as an Alternative to Conventional Word Processing Programs](#)
- [The not so short introduction to \$\text{\LaTeX}\$](#) : la version originale du document français ci-dessus.

Index

- .aux, 7
- .dvi, 4
- .log, 7
- .out, 7
- .sty, 48
- L^AT_EX, 2
- T_EX, 2

- accents, 13
- accolades, 37
- antislash, 6
- article, 8, 18, 19
- auteur, 11, 47

- babel, 19
- barres
 - horizontales, 30
 - verticales, 30
- begin, 9
- book, 8, 18

- césures, 42
- caption, 27
- centering, 27
- centrer, 16, 27, 30
- césures, 42
- chapitre, 18, 21
- citation, 17
- classes, 8, 18
- CM Super, 4, 42
- cocoAspell, 43
- commande, 9
- commandes, paquet de, 13
- commentaire, 12
- console, 7
- couleur, 46
- crochets, 37

- date, 11, 20
- dimension, 28
- displaymath, 37
- distiller, 44

- documentclass, 8
- éditeur des macros, 33, 45
- élément flottant, 26, 32
- emphasize, 15
- encodage, 13
- end, 9
- enumerate, 24
- environnements, 9
- eqnarray, 37
- equation, 36
- équation, 35
- étiquette, 21, 27, 37

- figure, 26
- float element, 26
- fonctions mathématiques, 38
- fontenc, 42
- footnote, 22
- formats d'images, 25, 44
- fraction, 36
- FrameMaker, 50
- french, 19

- ghostscript, 44
- graphicx, 24
- grecques, lettres, 39

- href, 46
- hyperliens, 46

- image, 24
- include, 43
- includegraphics, 24
- indentation, 10
- item, 24
- itemize, 24
- iTeXMac, 6

- justification, 16

- Knuth, Donald E., 2

- légende, 25, 27

- label, 21
- Lamport, L., 2
- ldots, 24
- left(, 36
- letter, 8, 18
- liste, 24
- listoffigures, 29
- listoftables, 33

- MacOS X, 2
- macros, 12, 23, 33, 45, 50
- maketitle, 11
- marginpar, 22
- matrice, 38
- mise en exposant, 35
- mise en indice, 35
- mise en page, 19

- newline, 9
- newpage, 11
- numérotation, 37
- numérotation, 19, 22, 27

- options
 - angle, 28
 - d'image, 28
 - draft, 28
 - height, 28
 - scale, 28
 - width, 28

- pérennité, 50
- palette LaTeX, 39
- palette matrix, 40
- paperheight, 29
- paperwidth, 29
- paragraphe, 9, 18
- parenthèses, 36
- PDF, 5
- pdfL^AT_EX, 4
- placement
 - de l'image, 26
- points de suspensions, 24
- police
 - style, 14
 - taille, 9, 16

- référence, 37
- raccourcis-clavier, 33, 50
- racine carrée, 35
- ref, 21
- référence, 21, 27, 33, 46
- report, 8, 18
- right), 36

- saut de ligne, 10
- saut de page, 11
- section, 18, 21
- souligner, 15
- sous-section, 18
- sous-sous-section, 18
- style, 14, 38

- table
 - alignement, 29
 - nombre de colonnes, 29
- table des figures, 29
- table des matières, 20
- table des tableaux, 33
- tableau, 24
- tableofcontents, 20
- tabular, 29
- texmf, 48
- TeXShop, 6
- textheight, 29
- textwidth, 29
- titre, 11, 47
- today, 20
- typographie
 - française, 20

- usepackage, 13

- vecteur, 38

- Word, 2, 50
- WYSIWYG, 3, 49

- YAFIYG, 49