Introduction à LaTeX sous Mac OS X (mise à jour)

Fabien Conus Franck Pastor

17 mars 2009

Table des matières

Introduction

Il y a quatre ans et demi environ, Fabien Conus a rédigé et publié sur le site `cuk.ch` (http://www.cuk.ch), alors dans sa première version, une remarquable série d'articles sur LaTeX, qui a permis à bien des gens de s'initier à ce formidable « formateur de texte ». Ces cours sont bien sûr toujours accessibles aux liens suivants :

- http://www.cuk.ch/articles/2059 (partie 1) ;
- http://www.cuk.ch/articles/2060 (partie 2) ;
- http://www.cuk.ch/articles/2061(partie 3) ;
- http://www.cuk.ch/articles/2062 (partie 4) ;
- http://www.cuk.ch/articles/2063 (partie 5) ;
- http://www.cuk.ch/articles/2064 (partie 6).

Malheureusement, les exemples de code LaTeX de ces cours ont très mal supporté le passage de `cuk.ch` à la version actuelle (disparition de certains signes comme la très importante contre-oblique « \ », apparition d'espaces et sauts de ligne intempestifs...), si bien qu'il est préférable maintenant de se référer à la version PDF complète de ce cours, à l'adresse suivante : http://redac.cuk.ch/fabien/IntroLaTeX.pdf.

Or ce n'est pas forcément vers cette version complète en PDF que pointent les divers moteurs de recherche. Ils peuvent tout aussi bien pointer vers l'un ou l'autre des articles séparés. Ce qui justifie déjà un toilettage de ces articles, qui rectifie les problèmes apparus lors de la migration de `cuk.ch` vers sa version 3 (cf. http://www.cuk.ch/articles/3143).

Mais, même en dehors de cela, il y a aussi le fait que ces articles ont maintenant quatre ans et demi, et qu'entre-temps le paysage LaTeX a pas mal changé. Pas tant le cœur du programme lui-même, bien qu'il ait connu une ou deux évolutions non négligeables, mais plutôt les « paquets » qui permettent de l'étendre, et surtout son implémentation sur Mac OS X.

Une mise à jour de ces articles était donc devenue nécessaire afin de leur conserver tout leur attrait et toute leur qualité. Fabien n'ayant guère plus le temps de s'y consacrer, je me suis proposé pour le faire (ce qui ne surprendra guère ceux qui me connaissent déjà sur `cuk.ch` comme grand fan de LaTeX) et il a gentiment accepté.

Vous trouverez donc ci-dessous une version actualisée par mes soins de ce cours LaTeX. Il s'agit par endroits d'un « copier-coller » de celui-ci, si bien que j'ai conservé tels quels les passages où Fabien dit « je ». Ainsi, dans la première partie, seul le passage concernant l'installation a été entièrement réécrit, les autres modifications n'y ont été que des corrections de détails. Les parties suivantes présentent plus de modifications.

Je tiens à remercier Noé Cuneo pour avoir accepté de relire les articles avant leur parution sur `cuk.ch`, et René Fritz pour en avoir fait autant avec cette version en PDF.

Laissons maintenant la parole à Fabien, et bonne (re)lecture !

Franck Pastor

1 Présentation et installation

Plusieurs fois, je vous ai parlé de LaTeX, que ce soit dans des tests, dans des commentaires ou dans le forum. Sous cet acronyme mystérieux, un peu effrayant, se cache un programme rassemblant autant de fidèles que de détracteurs. À la demande de plusieurs personnes, et suite à de nombreux débats au sujet de FrameMaker (http://www.adobe.com/fr/products/framemaker), dont le développement sur Mac a été arbitrairement stoppé en 2004 par Adobe, 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, TeXShop, tournant sous Mac OS X.

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

Car c'est bien là une des forces de LaTeX : il tourne sur toutes les plates-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 Kézako ?

En 1978, Donald E. Knuth désire créer un programme permettant d'obtenir des documents de qualité typographique irréprochable. Il crée alors le formateur de texte *TEX*, qui se prononce « tek » car inspiré du mot grec Τέχνη ayant donné « technique » en anglais et français, et qui signifie également « art », « métier ».

La syntaxe de TeX est toutefois assez lourde et complexe. Parfaitement conscient de cela, Knuth crée simultanément une première couche de macros au-dessus de TeX appelée Plain TeX (souvent confondue avec TeX lui-même) facilitant son utilisation.

Puis, en 1982, Leslie Lamport crée *LATEX*, une autre couche de macros sur TeX avec le même objectif (simplifier l'emploi de TeX), qui s'est immédiatement imposée dans le monde des « TeXniciens ». Pour faire une comparaison, on pourrait dire que LaTeX est à TeX ce que l'HTML est au SGML (http://wwwasdoc.web.cern.ch/wwwasdoc/WWW/publications/sgmlen/sgmlen.html).

Pour une histoire détaillée de TeX et LaTeX, vous pouvez consulter ce site : http://www.grappa.univ-lille3.fr/FAQ-LaTeX/1.1.html.

De nombreuses personnes pensent que LaTeX est un concurrent de Word, un traitement de texte parmi d'autres. Hé 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.

À la manière de l'HTML, vous allez créer un « code » LaTeX contenant votre texte et quelques commandes de mise en page. Ce code sera alors traité par le programme LaTeX pour finalement résulter en un document prêt à être imprimé. Dans le langage TeXnicien on appelle cela, de façon un peu raccourcie, la *compilation*.

1.2 Installation

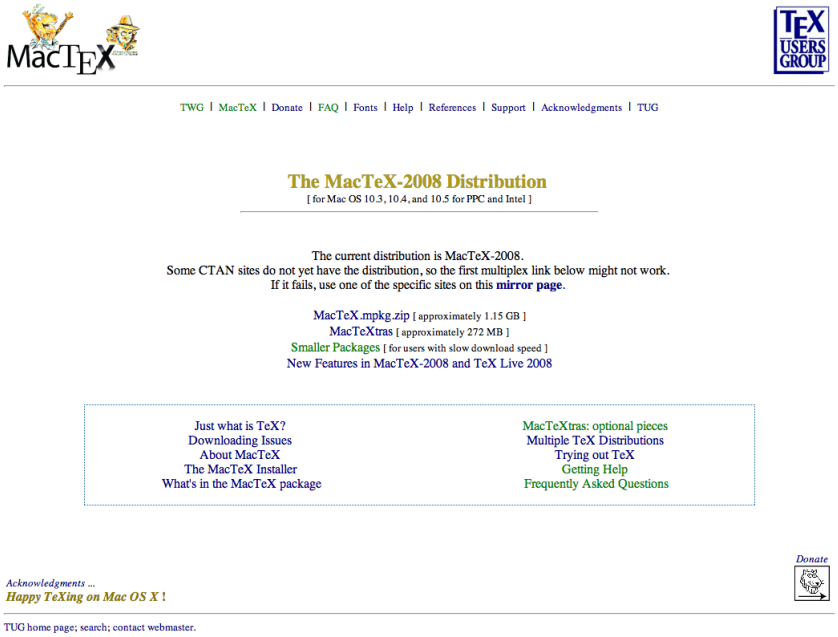
Pour commencer, une précision : contrairement à ce que beaucoup pensent, il n'est absolument pas nécessaire d'avoir X11 ou les outils développeurs installés sur sa machine pour profiter de LaTeX.

Pour disposer de LaTeX, il faut également installer TeX, puisque LaTeX est basé dessus, ainsi que d'autres programmes dont on ne parlera pas pour le moment. Pour simplifier, on parlera de *distribution TeX* pour désigner une distribution contenant TeX, LaTeX et leurs programmes connexes.

À l'époque de la première parution de cet article, la meilleure solution pour installer TeX et consorts était de le faire en utilisant un programme appellé *i-installer* (http://ii2.sourceforge.net). L'i-installer existe toujours, mais n'est plus « supporté » par son auteur, Gerben Wierda, ainsi que le précise l'en-tête de son site pointé par le lien précédent.

Il n'est cependant pas abandonné, et la distribution TeX qu'il permet d'installer, *gwTeX*, est toujours disponible, mais G. Wierda n'entretient son outil et ne le met à jour que selon ses besoins personnels, et ne répond pas aux rapports de bugs, même s'il les reçoit volontiers et s'efforce d'en tenir compte.

Mais cela s'est avéré finalement un mal pour un bien puisque, suite à cette « retraite » de l'i-installer, *MacTeX* (http://www.tug.org/mactex) a été élaboré par tout une équipe de passionnés, principalement Richard Koch, avec l'aide de G. Wierda d'ailleurs.



La page d'accueil de MacTEX.

Qu'est-ce que ce MacTEX ? Hé bien, il s'agit d'un ensemble de programmes comprenant tout d'abord une distribution TEX très complète, appelée *TEX Live*, très populaire sur les systèmes Unix en général. MacTEX contient également plusieurs outils permettant d'exploiter TEX Live au mieux, dont je parlerai plus loin. Il s'installe en quelques clics de souris, fonctionne sur Panther, Tiger et Leopard et tourne sur PowerPC comme sur Intel. La version actuelle de MacTEX date de septembre 2008, et est communément appelée MacTEX-2008 car elle permet d'installer TEX Live dans sa version de 2008, la plus récente à l'heure actuelle. La version précédente de MacTEX, MacTEX-2007, qui n'est maintenant plus disponible au téléchargement, proposait elle d'installer la version 2007 de TEX Live.

TEX Live est également disponible sur Mac via le projet MacPorts (http://www.macports.org), dans sa version de 2007 à l'heure où j'écris ces lignes. La distribution gwTEX de l'i-installer, dont j'ai parlé plus haut, est en fait une version allégée de TEX Live 2007, personnalisée par G. Wierda.

Mais MacTEX, en plus de fournir la version la plus récente de TEX Live, permet de l'installer de façon incomparablement plus simple qu'avec MacPorts ou l'i-installer, et c'est donc lui que nous allons vous proposer d'utiliser.

Mentionnons quand même au passage qu'il y a encore d'autres distributions TEX disponibles sur Mac, mais elles sont nettement plus anciennes que TEX Live 2008 ou même TEX Live 2007. Par exemple, teTEX [1], disponible via le projet Fink [2], ou encore les *sharewares* OzTEX [3] et CMacTEX [4].

1. http://www.tug.org/teTeX
2. http://www.finkproject.org
3. http://www.trevorrow.com/oztex
4. http://www.kiffe.com/cmactex.html

Téléchargez donc l'image-disque de MacTEX en cliquant ici : http://mirror.ctan.org/systems/mac/mactex/MacTeX.mpkg.zip. Ou bien, sur la page d'accueil de MacTEX (http://www.tug.org/mactex), cliquez sur le lien labellisé `MacTeX.mpkg.zip`.

Cela pourra mettre un certain temps à atterrir sur votre bureau, puisqu'il y a quand même 1,2 GB au total à transférer. Vous pouvez donc prendre un café, si l'humeur vous en dit ! Vous pouvez aussi en profiter pour visiter plus en détail le site de MacTEX qui regorge d'informations (en anglais). Lorsque l'installation sera terminée, l'espace occupé par l'ensemble de la distribution sera d'environ 1,8 GB.

Le chargement terminé, on procède à l'installation. Ouvrez l'image-disque : dans le disque virtuel qui est apparu, se trouve un *installer package*, `MacTeX.mpkg`, tel que vous en avez vu sûrement par dizaines sur Mac OS X. Ouvrez-le, et suivez comme d'habitude les instructions pas à pas, en cliquant sur Continuer après les informations de rigueur et la présentation des licences (toutes libres). Puis choisissez votre disque dur de destination (le disque dur sur lequel vous allez travailler avec LATEX évidemment !!), et installez finalement MacTEX tel qu'il est prévu par défaut ; inutile de personnaliser.

Ça y est, LATEX est installé sur votre machine !

Tout d'abord, vous avez deux nouveaux dossiers dans votre arborescence :

- dans votre dossier `Applications`, le dossier `TeX`. Attention, ce dossier ne contient pas TEX ou LATEX proprement dit, mais des applications Mac OS X qui permettent de les exploiter au mieux sur notre plate-forme :
  - *BibDesk*, un gestionnaire de base de données bibliographiques associé à LATEX ; il a été testé sur `cuk.ch` (http://www.cuk.ch/articles/2808),
  - *Excalibur*, un correcteur orthographique pour programmes LATEX,
  - *LATEXit*, un éditeur d'équations utilisant LATEX en sous-main,
  - *TEXShop*, une application dont on va faire très vite la connaissance,
  - un dossier Utilities, contenant tout ce qui est nécessaire pour l'installation du *TEX Live Manager*, dont on parle en détail dans un autre article de cuk.ch (http://www.cuk.ch/articles/3962) ;
- dans le dossier `Bibliothèque` de votre ordinateur, un autre dossier TeX contient justement des « alias » menant vers l'installation TEX proprement dite, qui se trouve dans la sous-couche Unix « invisible » de votre système. Mais le moment n'est pas non plus venu d'en parler.

De toutes les applications citées, seule TEXShop nous sera immédiatement utile, mais après la fin de ce cours, rien ne vous empêchera de jeter un œil aux autres.

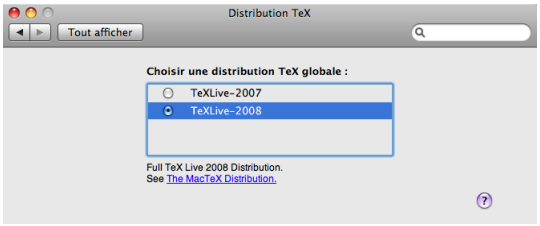
Dans le dossier `/Applications/Utilitaires` se trouve en outre l'i-installer lui-même ; une version ancienne avait été testée sur cuk en son temps (http://www.cuk.ch/articles/2058) ; il y a eu pas mal de chamboulement dans les *packages* Unix qu'il gère, mais ses principes n'ont pas changé. C'était par son intermédiaire que certains utilitaires Unix très précieux à LATEX étaient installés par MacTEX, comme *Ghostscript*. Mais depuis MacTEX-2008, Ghostscript est installé indépendamment de l'i-installer, en version 8.62.

Vous avez également une nouvelle Préférence système sur votre machine. Dans le champ Autre des Préférences système de Mac OS X doit en effet se trouver maintenant une icône

intitulée Distribution TeX :

Distribution TeX *est tout en bas, à gauche.*

Cette préférence liste comme son nom l'indique les différentes distributions TEX présentes sur votre machine. Dans votre cas, si c'est votre première installation, il ne devrait y avoir qu'un élément dans la liste, TEX Live 2008, installée par MacTEX. Sinon, si vous avez déjà une ou plusieurs installation(s) LATEX sur votre machine, elle sera également répertoriée et vous obtiendrez quelque chose comme ça :



Comme on le voit, il y a une autre distribution TEX sur cette machine : TeX Live 2007, qui venait avec la version précédente de MacTEX (MacTEX-2007).

En cliquant sur l'un ou l'autre bouton de la liste, on peut passer d'une installation à l'autre, de façon très fiable. Ce qui peut s'avérer pratique à chaque fois qu'une version de MacTEX sort (comme au moment de la parution de TEX Live 2008), et si on souhaite passer par une période de test avant de l'adopter définitivement. Ou bien lorsqu'on veut installer une autre distribution (celle de gwTEX, par exemple) et la tester sans prendre de risque d'interférer avec une autre distribution déjà présente.

Mais passons maintenant aux choses sérieuses. Vérifiez déjà que LATEX lui-même est utilisable, en faisant un très cours test via la ligne de commande du Terminal. Ouvrez une fenêtre du 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 au Terminal d'un code LaTeX, mais pour des raisons historiques je tiens à mentionner quelques petites choses.

La compilation d'un code LaTeX peut se faire simplement au Terminal en appelant la commande

```
latex mondocument.tex
```

Il s'ensuivra une série de lignes obscures indiquant la compilation du document. Puis vous obtiendrez un fichier portant l'extension `.dvi`. Le DVI *(DeVice Independant)* 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, car il existe depuis quelques années un nouveau programme : pdfLaTeX !

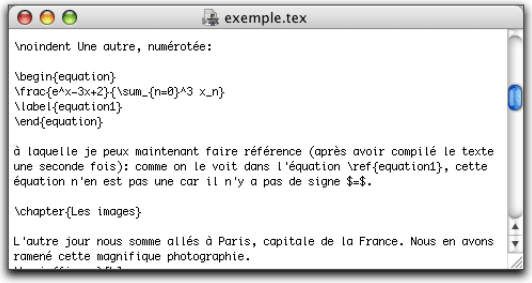
Si vous utilisez au Terminal la commande `pdflatex` au lieu de la commande `latex`, votre code LaTeX sera directement compilé en un magnifique document PDF, donc totalement compatible avec tous les différents acteurs du monde informatique. Et comme le PDF fait partie intégrante de Mac OS X, son intérêt 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.

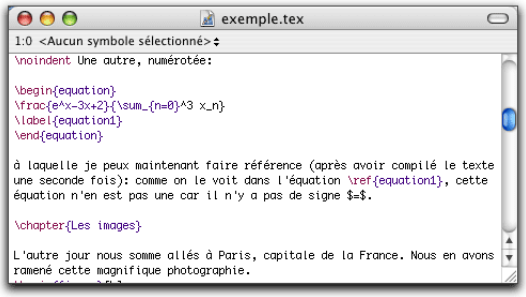
1.4 Éditeurs

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` ou `emacs` dans le Terminal, SubEthaEdit, BBEdit, Smultron, TextMate, XCode ou même TextEdit sur l'interface graphique de Mac OS X.

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). BBEdit le fait, `emacs`, TextMate, SubEthaEdit et Smultron 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...



. . . avec SubEthaEdit, si !

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.

Certains éditeurs polyvalents, comme `emacs` ou TextMate, proposent, outre la colorisation, des environnements de travail très performants pour LaTeX. Mais personnellement j'estime préférable, notamment pour le débutant, d'utiliser un éditeur spécifiquement dédié à LaTeX, à l'exclusion d'autres programmes sauf éventuellement ceux qui lui sont reliés (TeX lui-même par exemple).

Il y a à ma connaissance trois éditeurs sur Mac OS X qui permettent cela :

- *TeXShop* de Richard Koch ;
- *iTeXMac* de Jérôme Laurens ;
- *Texmaker* de Pascal Brachet.

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 populaire et se veut plus complet (le développement de la version actuelle n'est cependant pas achevé). Texmaker est tout récent, a le grand avantage d'être multi-plates-formes, mais n'a pas de visualiseur de PDF intégré, contrairement à TeXShop et iTeXMac, ce qui représente un handicap important ainsi qu'on va le voir.

Je vais donc orienter cette série d'articles non seulement sur l'usage et la syntaxe de LaTeX, mais également sur la façon de l'utiliser facilement avec TeXShop.

Comme on l'a vu, TeXShop a déjà été installé sur votre machine, dans le dossier

`/Applications/TeX`.

Son site dédié se trouve ici : http://darkwing.uoregon.edu/~koch/texshop/texshop.html. C'est l'endroit qu'il faut visiter si vous souhaitez suivre les mises à jours de ce programme, qui a une évolution indépendante de MacTeX. Au cas où vous en souhaitez une version plus récente, téléchargez-la donc sur ce site et installez-la par simple glisser-déposer, à la place de l'ancienne.

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 TeXShop le code suivant :

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

La contre-oblique \ (*backslash* ou *antislash* en anglais) s'obtient par la combinaison alt-shift-7 sur un clavier suisse-romand, et par alt-shift-: sur un clavier français.

Sauvegardez ce fichier (je vous conseille de créer un dossier spécial pour le contenir) sous un nom de votre choix, sans espace ni caractère accentué, et avec l'extension `.tex`. Par exemple, vous pouvez l'appeler `premier-document.tex`. Puis cliquez sur le bouton Composer de TeXShop. Une première fenêtre s'ouvrira où des lignes vont défiler, puis une deuxième qui contiendra cette fois votre document PDF [5], dans lequel sera fièrement écrit, en haut à gauche,

Hello le monde !

Bravo, vous venez de compiler votre code LaTeX !

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 dans cette console que LaTeX 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` et un fichier `.aux`, utilisés par LaTeX pour stocker quelques informations, et bien sûr un fichier `.pdf` qui est votre document. Également un fichier `.synctex.gz` dont la signification sera expliquée en section 6.7. C'est à cause de ces multiples fichiers que je vous conseille de créer un dossier pour chacun de vos documents.

Une dernière petite expérience pour aujourd'hui : dans votre document LaTeX, remplacez maintenant le texte `Hello le monde !` par `Bonjour le monde !` et appuyez à nouveau sur le bouton Composer (notez que TeXShop sauvegarde automatiquement lui-même le fichier à chaque compilation, s'il a préalablement été enregistré sous un nom précis). Constatez alors que TeXShop a automatiquement mis à jour la fenêtre montrant le PDF produit, pour afficher cette fois

Bonjour le monde !

Quoi de plus normal, me direz-vous ?

Hé bien, figurez-vous que les gens qui travaillent avec LaTeX sur Windows n'ont pas cette chance avec les PDF qu'ils génèrent. Ils n'ont en général qu'Adobe Reader (ou Adobe Acrobat) comme lecteur de PDF, donc une application complètement externe à leur éditeur. En conséquence, lorsqu'ils font une modification de leur code LaTeX,

5. Si à ce stade vous obtenez une erreur et que le PDF ne s'affiche pas, c'est que LaTeX s'est mal installé.

ils doivent avant de recompiler fermer eux-mêmes la fenêtre d'Adobe Acrobat/Adobe Reader qui résultait de leur précédente compilation, puis la rouvrir. Car sinon Adobe Acrobat/Adobe Reader ne montrerait pas la modification ! Et ainsi de suite à chaque modif., ce qui devient vite lassant...

C'est tout l'avantage de disposer sur Mac OS X d'un éditeur comme TEXShop (ou iTEXMac) qui a un lecteur de PDF intégré : ainsi, à chaque compilation, il peut refléter sur le fichier PDF les changements intervenus dans le code [6].

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. Si vous entrez la lettre `h` (comme l'anglais *help*) au clavier dans la console, LATEX tentera de vous fournir plus de détails sur l'erreur commise.

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. Vous pouvez également entrer la lettre `e` au clavier dans cette console, pour le même résultat. Si au lieu de cela vous appuyez sur Return sans rien d'autre, LATEX essaiera de passer outre cette erreur, jusqu'à la fin de la compilation... ou la prochaine erreur. Vous pouvez tout aussi bien choisir d'abandonner carrément ce processus de compilation, en entrant `x`.

Dans la prochaine section, nous examinerons ce petit document que nous venons de créer et nous explorerons plus en détail les fonctions de TEXShop.

6. Pour pouvoir bénéficier de cette agréable possibilité, les adeptes de LATEX sur Windows se contentent souvent, au moins dans un premier temps, de produire et visualiser des fichiers au format DVI, dont leurs éditeurs de texte peuvent piloter un lecteur dédié (`Xdvi`, `Yap`...). Le format DVI est pourtant sévèrement limité par rapport au PDF : pas d'affichage des figures ou des rotations de texte, par exemple.

2 Les principes fondamentaux de LaTeX

Dans la section précédente, nous avions examiné comment installer les outils nécessaires au bon fonctionnement de LaTeX. Nous avions terminé par un exemple très simple de document. Dans cette deuxième partie, nous allons examiner en détail ce petit exemple et aller plus loin dans les fonctionnalités de LaTeX.

2.1 Classes, commandes et environnements

Je vous rappelle notre petit code LaTeX :

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

Vous y repérez facilement les *commandes* LaTeX : par convention, celles-ci commencent en effet (presque) toutes par la contre-oblique « \ ».

Un document LaTeX doit toujours commencer par la commande `\documentclass`, suivie de la ou des *options* entre crochets et du type de document, aussi appelé *classe*, entre accolades.

Dans notre exemple, on indique dans les options qu'on travaille au format A4 et dans la classe de document qu'on rédige un livre *(book)*.

Les classes de document définissent la mise en page. Les plus courantes sont :

- `article` ;
- `book` ;
- `report` (rapport) ;
- `letter` (lettre).

Mais il en existe beaucoup d'autres : `memoir`, `exam`, `beamer`, etc.

Les différences entre ces classes 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 base des polices utilisées dans le document PDF. Par exemple :

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

définira une taille de base de 11 points, au lieu de 10 points qui est la taille de base proposée par défaut dans les classes `article`, `book` et `report`. Vous avez le choix entre 10, 11 et 12 points comme taille de base. On verra plus loin dans cette section comment faire varier localement les tailles des polices autour de cette taille par défaut.

La commande suivante de notre petit exemple, `\begin{document}`, est également obligatoire : elle déclare en effet le début du contenu du document et indique que la configuration du document est terminée. À chaque `\begin{document}` correspond un `\end{document}`, tout aussi obligatoire, définissant la fin du document. Rien de ce qui sera éventuellement écrit après le `\end{document}` ne sera pris en compte par LaTeX.

Un couple de commandes ayant cette forme (\begin suivi de \end) définit ce qu'on appelle un *environnement*. Nous venons donc juste de voir l'environnement `document`. Nous en verrons par la suite bien d'autres.

Vous l'aurez compris, toutes les commandes de LaTeX (enfin, presque toutes) sont de la forme

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

les options et l'argument pouvant être présents ou non, selon les commandes. Par exemple, la commande \begin ne prend aucune option mais l'argument est obligatoire.

2.2 Entrer son texte avec LaTeX

La commande que nous allons examiner dans le petit exemple qui suit est \\ qui est connue aussi sous le nom plus explicite de \newline. Cette commande ne prend pas d'argument et force un retour à la ligne dans le document de sortie. Vérifiez en compilant l'exemple suivant :

```
\documentclass[a4paper,11pt]{book}
\begin{document}
Bonjour le monde !\\
Ceci est sur une nouvelle ligne.
\end{document}
```

Entrez maintenant ceci dans TeXShop :

```
\documentclass[a4paper,11pt]{book}
\begin{document}
Bonjour le monde !\\
Ceci est sur une nouvelle ligne.

Ceci commence un nouveau paragraphe.
Ici, ni nouveau paragraphe, ni nouvelle ligne.
\end{document}
```

La deuxième phrase de ce programme est suivie par une ligne vide, car on a appuyé deux fois sur la touche « Entrée », donc sauté une ligne. Cela créera un nouveau *paragraphe* dans le document de sortie. On peut également employer la commande \par au lieu du saut de ligne pour définir un changement de paragraphe.

Après avoir compilé cet exemple, vous constaterez que le changement de paragraphe est marqué dans le document de sortie par un simple retour à la ligne, sans qu'il y ait eu de saut de ligne ou d'espace vertical ajouté. Mais en revanche il y a eu une *indentation* de la première ligne du nouveau paragraphe. C'est la convention par défaut adoptée par LaTeX pour distinguer les paragraphes entre eux.

La commande \\ placée après la première phrase a forcé dans le document de sortie un simple retour à la ligne, *sans indentation*. On était donc resté en fait dans le même paragraphe.

Un seul retour de chariot dans le document LaTeX n'entraînera pas de retour à la ligne dans le document de sortie, comme cela est démontré par la troisième et la quatrième phrase de notre exemple. Dans ce document de sortie, vous constaterez qu'elles se suivent sur la même ligne, séparées par une simple espace mot, et c'est LaTeX qui a décidé tout seul comme un grand à quel endroit se faisait la coupure de ligne.

Comme avec l'HTML, LaTeX ne « voit pas » les espaces consécutifs. Que vous écriviez :

```
Ceci est un paragraphe.
```

ou

```
Ceci      est    un      paragraphe.
```

ce sera pour lui exactement la même chose.

De façon analogue, LaTeX ne « voit pas » plusieurs sauts de ligne consécutifs. Que vous écriviez dans votre document LaTeX :

```
Ceci est un paragraphe.

Ceci est un autre paragraphe.
```

ou

```
Ceci est un paragraphe.




Ceci est un autre paragraphe.
```

cela revient au même au final : un nouveau paragraphe a été créé, marqué dans le document de sortie par un retour à la ligne et une indentation, mais sans saut de ligne.

Si on veut créer un nouveau paragraphe et simultanément sauter une ligne dans le document de sortie, il faudra placer une commande `\bigskip` avant ou après avoir sauté une ligne dans le document LaTeX.

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

Ceci est un nouveau paragraphe, qui suit un saut de ligne.

\bigskip Ceci est un nouveau paragraphe, suivant un saut de ligne.
\end{document}
```

Comme alternatives à `\bigskip`, les commandes `\medskip` et `\smallskip` permettent de sauter respectivement une moitié de ligne et un quart de ligne. À noter que ces trois espacements verticaux sont « élastiques », à la façon de ressorts : LaTeX peut décider de les comprimer ou de les étirer légèrement afin d'obtenir une meilleure mise en page.

La création d'un nouveau paragraphe impliquera, on l'a vu, une indentation de la première ligne de ce nouveau paragraphe. Si vous voulez supprimer l'indentation de cette ligne, utilisez la commande \noindent :

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

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

\bigskip\noindent Par contre, celle-ci n'en aura pas.
\end{document}
```

Si vous voulez au contraire forcer une indentation, utiliser \indent.

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

Un conseil pour conclure cette section : n'utilisez les commandes \\ (ou \newline) et \newpage qu'avec grande parcimonie, seulement si c'est absolument nécessaire, une fois votre document vraiment terminé. LaTeX se charge tout seul de la gestion des coupures de lignes et de la mise en pages d'un document, et il le fait de manière très « pro », de sorte que l'auteur ne devrait pas s'en soucier lui-même. Il doit seulement s'occuper de découper son texte en unités logiques cohérentes.

Et l'unité logique la plus petite d'un texte bien construit doit être représentée par un paragraphe, qu'on délimite dans le document LaTeX en utilisant simplement le saut de ligne, comme on l'a vu.

Nous verrons au cours suivant comment découper le texte en chapitres, sections, sous-sections, etc.

La commande \\ a néanmoins une grande utilité dans d'autres contextes que le texte « normal ». Par exemple dans la création de tableaux, ce qu'on verra dans la section 4.5.

2.3 La page de titre

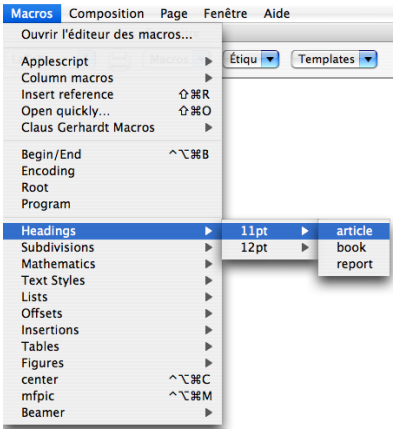
Si vous le désirez, vous pouvez aisément ajouter une page de titre à votre document. Pour ce faire, trois champs peuvent être remplis : le titre du document, l'auteur et la date. Ceci se fait grâce aux commandes \title, \author et \date. Puis il faut encore décider où vous désirez créer la page de titre (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, blablabla, etc.
\end{document}
```

Attention, si vous n'avez pas inséré de commande `\date`, la commande `\maketitle` fera en sorte que la page de titre inclue automatiquement la date courante de composition du document. Si vous désirez ne pas dater du tout votre document, insérez-y la commande `\date{}`, avec des accolades ne contenant rien entre elles.

2.4 Les macros de TEXShop

Bien sûr, il n'est pas très agréable d'entrer tous les signes barbares qu'on vient de voir 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 de commandes et d'environnements pré-enregistrés. Par exemple, pour obtenir le canevas d'un document de classe `article` ayant des caractères de taille 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 LATEX.

2.5 Les commentaires

Comme pour les langages de programmation, il est possible d'ajouter des *commentaires* dans le code LATEX. Ceci se fait grâce au signe pourcentage `%` :

```
\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 est également un commentaire
\end{document}
```

Tout ce qui suit le signe % sur la même ligne sera ignoré à la compilation. Pour écrire un signe % dans le document de sortie vous devrez utiliser la commande \%.

Si vous souhaitez rédiger plusieurs lignes successives de commentaires dans votre fichier LaTeX, il sera vite fastidieux de placer « à la main » un signe % devant chaque ligne. Mais TeXShop fournit une solution. Rédigez vos lignes de futurs commentaires dans votre fichier, puis sélectionnez toutes ces lignes à la souris et choisissez dans le menu Source l'option Ajouter un commentaire. Vous verrez que tous les signes % nécessaires seront placés. Si vous souhaitez en enlever, au contraire, l'option Enlever le commentaire est là pour ça (ah bon ?), par le même procédé.

Passons à présent à des choses plus sérieuses.

2.6 Les accents

Vous l'avez peut-être remarqué, j'ai pris soin jusqu'à présent 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 lettres accentuées dans un de nos petits documents d'exemple. Vous constaterez après compilation qu'elles auront été tout simplement omises.

LaTeX, en effet, considère par défaut que votre texte est encodé selon le standard ASCII *(American Standard Code for Information Interchange)*, élaboré aux États-Unis pour les échanges informatiques en anglais. Et cet encodage ne connait pas les caractères accentués, puisque la langue anglaise n'en a pas.

Pendant longtemps, la seule solution pour écrire avec LaTeX des lettres accentuées et à cédille a été de passer par des commandes spéciales. Pour l'accent aigu il fallait entrer \', pour l'accent grave \`, pour le circonflexe \^, pour le tréma \" (le guillemet double-quote). Le c cédille s'obtenait lui par \c{c} ou \c c. Par exemple :

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

Vous conviendrez que ce n'est pas très agréable. Certes, ça peut être pratique pour obtenir des lettres accentuées exotiques que notre clavier ne propose pas, mais pour nos lettres accentuées francophones, disponibles là sous nos doigts, devoir taper tout ça, berk ! Heureusement, la solution existe.

En informatique, si on veut des caractères accentués, on doit déclarer l'*encodage du texte* utilisé par l'éditeur. Dans le cas de l'HTML, on déclare l'encodage du texte dans les balises « meta » en spécifiant le « charset ». Avec LaTeX, il faut également déclarer l'encodage. Pour cela il faut ajouter un *paquet* de commandes au LaTeX initial (en anglais, *package*). 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, qui s'appelle `inputenc` (abréviation de *input encoding*, encodage d'entrée), en lui spécifiant que nous travaillons avec un Mac :

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

En effet, par défaut, l'encodage du texte utilisé par TEXShop est *Mac OS Roman*, ce qui correspond à l'option `applemac` du paquet `inputenc`.

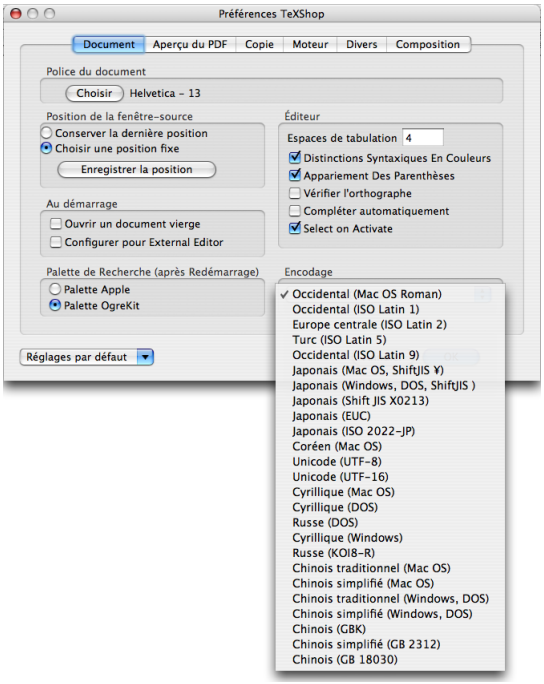
L'ajout de paquet doit se faire dans le *préambule* du document, c'est-à-dire après le `\documentclass` et avant le `\begin{document}`. Voyez l'exemple ci-dessous, et testez-le :

```
\documentclass[a4paper,11pt]{book}
\usepackage[applemac]{inputenc}
\begin{document}
Cet été, dans une forêt française, j'ai ouï un très grand cri.
\end{document}
```

Après compilation, vous verrez que tous les accents et la cédille sont bien présents. Ouf, problème réglé !

Cependant, si vous voulez avoir plus de compatibilité avec nos voisins de chez Windows, vous allez certainement préférer l'encodage *ISO Latin 1*. Sinon, lorsque vous transmettrez vos fichiers LATEX à quelqu'un travaillant sur Windows, il verra plein de caractères cabalistiques à la place des caractères accentués lorsqu'il ouvrira le document dans son propre éditeur de texte. Car un éditeur de texte sur Windows, neuf fois sur dix, utilise cet encodage ISO Latin 1 ou une extension de celui-ci. C'est également souvent le cas des éditeurs sur Linux. Évidemment, si vous vous contentez de transmettre les fichiers PDF, il n'y aura aucun problème.

Pour passer en ISO Latin 1, rendez-vous dans les Préférences de TEXShop. Dans l'onglet Document vous y trouverez une zone Encodage :



Choisissez alors Occidental (ISO Latin 1). Vos prochains documents auront cet encodage.

Si vous avez un document en cours dont vous voulez changer l'encodage, vous devez en sauvegarder un nouvel exemplaire en passant par le menu Fichier, option Enregistrer sous... Vous pourrez alors choisir quel encodage utiliser pour la sauvegarde.

Pour un fichier LaTeX donné, vous pouvez procéder d'une autre manière que d'aller dans les Préférences de TeXShop : insérez sur une ligne précédant le `\documentclass` du fichier le « commentaire » suivant :

```
%!TEX encoding = IsoLatin
```

Il s'agit d'un commentaire très particulier, car lors de la sauvegarde TeXShop en tiendra compte et encodera le fichier en ISO Latin 1, *quel que soit l'encodage mentionné dans ses Préférences*. Il vaut mieux donc insérer cette ligne dans le fichier avant d'y rédiger quoi que ce soit [7].

Attention, cette astuce est spécifique à TeXShop. Tout autre éditeur que lui verra ladite ligne comme un commentaire classique.

Vous pouvez vous aider du menu Macros, option Encoding, pour entrer la première partie de ce commentaire spécial, `%!TEX encoding =`.

Que vous imposiez l'encodage du document à TeXShop par la méthode « Préférences » ou la méthode « commentaire spécial », vous devez avertir aussi LaTeX lui-même de cet encodage. Pour ce faire, mentionnez cette fois-ci `latin1` comme option du paquet `inputenc` :

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

Ça y est, vous pouvez rédiger votre texte dans un fichier LaTeX qui sera transmissible sans problème au monde Windows.

Ces dernières années, un autre encodage, incroyablement riche en symboles de toutes sortes et couvrant quasi toutes les langues terrestres, est de plus en plus populaire : *UTF-8 Unicode*. Il est destiné à terme à remplacer l'ASCII, et est déjà l'encodage par défaut de certains systèmes d'exploitation Linux.

Si donc vous transmettez des fichiers à des collègues travaillant sous cet encodage, procédez comme précédemment, mais choisissez cette fois l'option Unicode (UTF-8) dans l'onglet Encodage des préférences de TeXShop. Ou bien insérez, avant le `\documentclass` de votre fichier LaTeX et avant de le rédiger, la ligne suivante :

```
%!TEX encoding = UTF-8 Unicode
```

si vous préférez la méthode « commentaire ». Enfin, dans votre fichier LaTeX, choisissez l'option `utf8` pour le paquet `inputenc` :

```
\usepackage[utf8]{inputenc}
```

Mentionnons au passage qu'une version de LaTeX appelée *XeLaTeX* a adopté Unicode (UTF-8 ou UTF-16) comme unique encodage possible de ses fichiers, anticipant par là sur l'avenir. J'en dirai un peu plus à la section 6.6. Jusque-là, nous resterons avec (pdf)LaTeX, le standard actuel.

7. Si vous aviez souhaité par ce type de commentaire forcer un fichier à être encodé par TeXShop en « Mac OS Roman », l'option correspondante à inscrire aurait été alors tout bonnement `MacOSRoman`.

2.7 Un petit problème tué dans l'œuf

Si vous adoptez l'encodage ISO Latin 1, sachez que ce dernier souffre d'une petite lacune : il ne reconnaît pas le caractère œ, le « e dans l'o », qui apparaît par exemple dans les mots œuf , bœuf ou œuvre. Pour obtenir ce caractère (ou plutôt cette ligature de caractères) avec cet encodage, vous devez utiliser la commande LaTeX `\oe`, et `\OE` pour obtenir sa majuscule. Par exemple,

```
Qui vole un \oe uf vole un b\oe uf.
```

donnera

Qui vole un œuf vole un bœuf.

Avec les deux autres encodages présentés ici, Mac OS Roman et UTF-8 Unicode, vous pouvez entrer œ et Œ directement au clavier (par les combinaisons alt-o et alt-shift-o sur un clavier français, alt-q et alt-shift-q sur un clavier suisse).

2.8 Une subtilité d'espacement

Vous aurez noté dans notre exemple précédent l'espace entre la commande `\oe` et le reste du mot, ici les lettres `uf`. Sans cette espace, LaTeX lirait la commande `\oeuf` qu'il ne connait évidemment pas, et signalerait une erreur.

Mais vous avez pu également vérifier que cet espace n'apparaît pas dans la sortie après compilation, et que les mots « œuf » et « bœuf » y sont correctement formés, comme il se doit.

De façon générale, LaTeX « absorbe » tout espace entré au clavier juste après une commande constituée de lettres alphabétiques, comme la commande `\oe` (les commandes comme `\\` ou `\%` ne sont donc pas concernées). Ceci pour permettre la formation de certains mots, comme dans l'exemple précédent.

Mais cela a un effet secondaire surprenant. Voyons-le sur un exemple, avec la commande `\LaTeX`, elle aussi constituée uniquement de lettres, qui permet de composer le logo calligraphié officiel « LaTeX » (appréciez les différences avec le mot « LaTeX » entré tel quel). Le bout de code suivant :

```
\LaTeX est très puissant.
```

donnera après compilation

LaTeXest très puissant.

Catastrophe, il n'y a pas d'espace mot entre LaTeX et le mot suivant !! Pas de panique, pour l'introduire, séparez la commande de l'espace par une paire d'accolades :

```
\LaTeX{} est très puissant.
```

ou bien forcez l'introduction d'une espace-mot, avec la commande `\␣` (contre-oblique suivie d'un espace-clavier) :

```
\LaTeX\ est très puissant.
```

et tout rentre dans l'ordre :

LaTeX est très puissant.

2.9 Changer le style du texte

Un *style* de police en LaTeX est défini par trois caractéristiques : sa famille (*family*, en anglais), sa forme *(shape)* et sa série ou graisse *(series)*. La police par défaut de LaTeX est une police de famille romaine (en anglais, *roman*), de forme droite *(upright)* et de série/graisse normale *(medium)*. LaTeX dispose aussi des familles

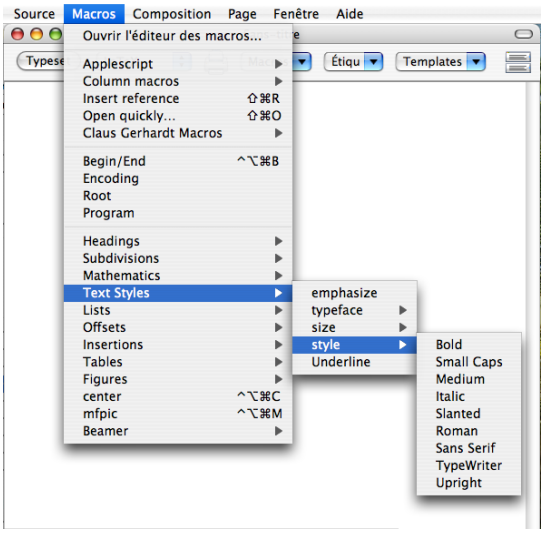
- sans serif (c'est-à-dire sans empattement, sans aucune fioriture) ;
- à chasse fixe (style « machine à écrire », *teletype* en anglais ; chaque caractère occupe la même largeur).

Il propose également les formes

- italique *(italic)* ;
- inclinée *(slanted)* ;
- petites capitales *(small caps)* ;

ainsi qu'une série grasse *(bold)*.

Les commandes LaTeX de changement de style correspondantes peuvent s'obtenir par le menu Macros de TeXShop, option Text Styles > style :



Voici ces commandes, regroupées selon les caractéristiques de style :

- les familles :

romaine (défaut)	`\textrm`
à chasse fixe	`\texttt`
sans serif	`\textsf`

– les formes :

droite (défaut)	`\textup`
italique	`\textit`
penchée	`\textsl`
petites capitales	`\textsc`

– les séries :

normale (défaut)	`\textmd`
grasse	`\textbf`

J'imagine que vous vous demandez à quoi ressemblent concrètement toutes ces « familles », « formes » ou « séries ». Alors voici quelques petits exemples et leurs résultats :

`La famille \texttt{à chasse fixe}.`	La famille `à chasse fixe`.
`La famille \textsf{sans serif}.`	La famille sans serif.
`La forme \textit{italique}.`	La forme *italique*.
`La forme \textsl{inclinée}.`	La forme *inclinée*.
`La forme \textsc{en petites capitales}.`	La forme EN PETITES CAPITALES.
`La série \textbf{grasse}.`	La série **grasse**.

Attention, si vous omettez les accolades encadrant l'argument d'une de ces commandes, seul le premier caractère qui suit la commande sera mis dans le style requis.

Avez-vous remarqué que la forme italique est différente de la forme inclinée ? Regardez attentivement les exemples correspondants et vous verrez que les caractères n'ont pas le même dessin. La forme inclinée est en fait la forme droite que l'on a « incliné » vers la droite, tandis que la forme italique a un dessin qui lui est tout à fait particulier.

On devrait en théorie pouvoir combiner chaque famille avec chaque forme et chaque graisse. Par exemple, la forme italique avec la série grasse (la famille est romaine puisque rien n'est imposé là-dessus) :

```
Un bout de texte \textit{\textbf{en italique gras}}.
```

On obtiendra

Un bout de texte ***en italique gras***.

Autre exemple, la famille sans serif combinée avec la forme inclinée (la graisse est normale, puisqu'on ne dit rien dessus) :

```
Un bout de texte \textsl{\textsf{en sans serif incliné}}.
```

Cela donne

Un bout de texte *en sans serif incliné*.

Il y a malheureusement pas mal d'exceptions. Par exemple, dans l'ensemble de polices par défaut de LaTeX, appelées *Computer Modern*, il n'y a pas de police sans serif à la fois penchée et grasse, ou de police à chasse fixe et grasse. On fera connaissance à la leçon suivante d'un ensemble de polices bien plus complet : les *Latin Modern*.

Je vous encourage à essayer par vous-même quelques autres combinaisons. Pour cela, n'hésitez pas à user et abuser des macros de TEXShop correspondantes.

Gardez quand même à l'esprit que pour un document « sérieux », il n'est pas recommandé de changer de style à tout bout de champ.

2.10 Mise en évidence du texte

Vous pouvez bien sûr souligner une partie du texte avec LaTeX. Ceci se fait grâce à la commande `\underline` avec comme argument ce que vous voulez 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 !
```

produira après compilation

Il est *très important* de lire ceci !

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

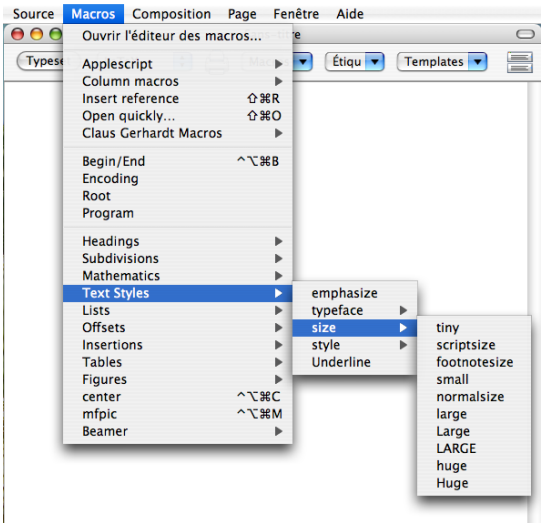
```
\textit{Il est \emph{très important} de lire ceci !}
```

donnera

Il est très important *de lire ceci !*

2.11 Changer la taille du texte

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



L'utilisation de ces tailles obéit à une autre syntaxe que celle vue précédemment sur les styles. Exemple :

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

Résultat :

Voici un tout petit texte et un très gros.

Vous pouvez constater dans le code de cet exemple que les accolades encadrent ici à la fois la commande et le texte où celle-ci s'applique.

Si vous omettez les accolades, tout le texte qui suivra la commande prendra la taille définie par cette commande, jusqu'à une éventuelle commande de même type, et sinon jusqu'à la fin du plus petit environnement qui la contient.

Dans ces circonstances, pour revenir à la taille normale, vous pouvez utiliser la commande `\normalsize`. Ainsi, une façon d'entrer l'exemple précédent sans accolades serait :

```
Voici un \tiny tout petit texte \normalsize et un \Huge très gros.
```

Vous comprendrez donc aisément pourquoi les commandes de ce type sont appelées *bascules (shifters)* !

2.12 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 : LaTeX s'arrange pour que chaque ligne prenne la même longueur (sauf la dernière ligne d'un paragraphe), quitte à utiliser s'il le faut la césure d'un mot en fin de ligne, comme ci-après :

> Bien sûr, il faut un texte de longueur suffisante pour remarquer sa justification ou son alignement, sinon vous n'obtiendrez pas de différence visible.

Pour aligner à gauche, il faut entrer notre texte dans un environnement `flushleft` :

```
\begin{flushleft}
Bien sûr, il faut un texte de longueur suffisante
pour remarquer sa justification ou son alignement, sinon vous
n'obtiendrez pas de différence visible.
\end{flushleft}
```

ce qui donne ici

> Bien sûr, il faut un texte de longueur suffisante pour remarquer sa justification ou son alignement, sinon vous n'obtiendrez pas de différence visible.

Naturellement, pour aligner à droite, il faudra placer le texte dans un environnement `flushright`. Ainsi le code suivant

```
\begin{flushright}
Bien sûr, il faut un texte de longueur suffisante
pour remarquer sa justification ou son alignement, sinon vous
n'obtiendrez pas de différence visible.
\end{flushright}
```

produira

> Bien sûr, il faut un texte de longueur suffisante pour remarquer sa justification ou son alignement, sinon vous n'obtiendrez pas de différence visible.

Pour centrer un bout de texte, vous utiliserez l'environnement `center`. Ainsi le code

```
\begin{center}
Bien sûr, il faut un texte de longueur suffisante
pour remarquer sa justification ou son alignement, sinon vous
n'obtiendrez pas de différence visible.
\end{center}
```

donnera

> Bien sûr, il faut un texte de longueur suffisante pour remarquer sa justification ou son alignement, sinon vous n'obtiendrez pas de différence visible.

Noter que lorsque le texte est aligné à gauche ou à droite, ou centré, le mécanisme de césure est désactivé.

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, option 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 les différentes commandes présentées dans ce cours et ainsi de faire vos propres expériences. Par exemple, compilez un document LaTeX contenant l'exemple précédent, puis remplacez l'environnement `quote` par `quotation` et voyez la différence.

Vous possédez maintenant déjà les bases nécessaires à la rédaction de documents simples sous LaTeX. Dans la prochaine section, nous nous attaquerons à la structure du document.

3 Structuration et francisation d'un document

Grâce aux deux sections précédentes, nous savons à présent comment créer un document, y entrer du texte et le mettre quelque peu en forme. Il est temps à présent de nous attaquer à la structure du document. Au passage, nous verrons comment obtenir un document correctement francisé.

3.1 La structuration du document en LaTeX

Un document se structure habituellement en chapitres, sections, sous-sections, paragraphes, etc. La création des paragraphes dans un document LaTeX est simple, comme je l'ai déjà expliqué dans la section précédente : il suffit de faire un saut de ligne. Les autres structures possibles ainsi que leur apparence dépendront de la classe du document. Je vous rappelle que, toujours dans la section précédente, 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'on place des chapitres et des sections dans une lettre !), je ne la traiterai donc pas 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}
```

Si vous faites quelques essais, vous remarquerez très vite que LaTeX 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. Il en est de même pour les sous-sections et les sous-sous-sections.

De plus, LaTeX prend entièrement en charge la taille et la graisse des titres ainsi que les espaces entres le titre et le texte autour, inutile de vous en occuper vous-même.

Une fois votre document structuré, vous pouvez aisément naviguer de chapitres en sections, grâce à l'élément Étiquettes de la barre d'outil de TeXShop.

3.2 Francisation avec babel

Durant vos essais, vous aurez remarqué que LaTeX nous gratifie d'un très désagréable *Chapter* dans chaque titre de chapitre, au lieu du doux « Chapitre » que nous attendons. En effet, par défaut LaTeX travaille en anglais. Mais rien de plus facile que de le faire parler la langue de Molière !

Il existe pour cela un paquet de commandes judicieusement intitulé `babel` (inspiré par la tour du même nom qui inspira à Dieu la création de langues différentes). Il s'agit donc d'indiquer à LaTeX que nous désirons utiliser ce paquet de commandes en lui précisant que nous écrivons en français. Dans la partie précédente nous avons vu comment ajouter un paquet de commandes grâce à la commande `\usepackage`, nous allons faire la même chose ici :

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

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

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

Vous pouvez bien entendu appeler les commandes `\usepackage` dans l'ordre que vous voulez, cela n'a pas d'importance *a priori*, sauf dans quelques rares cas particuliers, dont un qu'on verra en section 6.

Vous vous demandez peut-être pourquoi cette option `frenchb` s'appelle ainsi, et non tout simplement `french` ? En fait, sur les distributions TeX récentes, comme TeX Live version 2007 ou 2008, vous pouvez choisir d'utiliser les options `french` ou `francais` (sans la cédille !) à la place de `frenchb`, sans aucun problème : ce sont maintenant de parfaits synonymes de `frenchb`, qui est le nom original.

Par contre, si vous utilisiez une distribution plus ancienne, utiliser l'option `french` pouvait faire appel en sous-main à des options de francisation différentes de celle prévue par le paquet `babel`. D'où des risques de commandes incompatibles, et une présentation du document peut-être différente de celle que vous attendiez. Ceci explique ce nom de `frenchb`, avec `b` comme `babel`, pour éviter toute ambiguïté : avec ce nom-là, on charge bien la francisation prévue par le paquet `babel`.

Notez au passage une commande anodine, mais très utile de LaTeX, c'est la commande `\today`. Celle-ci sera remplacée lors de la compilation par la date du jour, qui sera en français grâce à l'option `frenchb` du paquet `babel`.

Le paquet de commandes `babel`, option `frenchb`, ne fait pas que traduire *Chapter* en « Chapitre » et donner la date courante en français, il indique également à LaTeX d'utiliser les conventions typographiques françaises et vous offre également quelques commandes propres à notre langue. Nous allons maintenant nous pencher là-dessus.

3.3 Les césures

On l'a vu, le texte est par défaut justifié à droite et à gauche avec LaTeX, ce qui pose le problème des césures éventuelles de mots en fin de ligne.

Les césures sont automatiques avec LaTeX et sont faites de façon particulièrement fiable. Cependant, par défaut, elles sont faites selon les règles anglophones. Charger `babel` avec l'option `frenchb` force heureusement LaTeX à faire ses césures de mots en respectant les conventions francophones.

Le problème est que LaTeX ne parviendra pas, même alors, à couper les mots contenant des accents (ou des cédilles) au-delà du premier caractère accentué. Du moins si on reste avec les polices de caractères qu'il utilise par défaut, les *Computer Modern*.

Cela vient du fait que ces polices ne contiennent pas vraiment de caractères accentués. Elles connaissent les accents et les caractères alphabétiques de base, mais pour elles un caractère accentué est la superposition de deux caractères distincts : un caractère « accent » est superposé à un caractère de base. Et cette superposition bloque le mécanisme de césure du mot.

La parade ? Utiliser des polices contenant les caractères accentués, chacun formé « d'un seul bloc ».

C'est là qu'intervient la notion d'*encodage de police*, à ne pas confondre avec l'encodage du texte, exercé par votre éditeur de texte (cf. section 2.6). Un encodage de police est la liste, dans un ordre bien défini, de tous les caractères proposés par une police. Différentes polices proposant la même liste de caractères sont dites avoir le même encodage. Or l'encodage commun des polices *Computer Modern*, appelé « OT1 », ne contient malheureusement pas nos caractères accentués.

À la place, on fera utiliser par LaTeX des polices d'encodage « T1 », lequel contient tous les caractères, accentués ou non, de la plupart des langues européennes à alphabet latin, dont la nôtre.

Cela se fait en chargeant le paquet de commandes `fontenc` (de l'anglais *font encoding*, « encodage de police ») avec l'option `T1` :

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

Une fois ce paquet chargé, LaTeX utilisera par défaut les polices *cm-super (Computer Modern super)*, d'encodage T1, et avec elles les césures seront correctes.

Donc, problème réglé ? Pour les pinailleurs, pas tout à fait. Et le monde de LaTeX est un monde de pinailleurs, où ces polices *cm-super* sont contestées, bien qu'encore très utilisées. Elles ressemblent beaucoup aux *Computer Modern*, mais elles ne reproduisent pas toujours fidèlement leur apparence, notamment dans les titres, ce qui peut gêner les adeptes du *look* des *Computer Modern*. De plus, la qualité des polices *cm-super* serait douteuse aux yeux des spécialistes avertis (moi je n'ai rien vu, car je n'en suis pas).

La meilleure alternative est d'utiliser les polices *Latin Modern*, elles aussi disponibles à l'encodage T1. Elles sont destinées d'ailleurs, à terme, à remplacer les *Computer Modern* comme polices par défaut de LaTeX. Ces polices *Latin Modern* ont l'avantage d'avoir exactement la même apparence extérieure que les *Computer Modern*, et d'être reconnues comme étant de très bonne qualité, même par les experts en polices les plus tatillons.

Les polices *Latin Modern* s'obtiennent en chargeant le paquet `lmodern` en plus du paquet `fontenc` option `T1` :

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

S'il arrivait (mais d'expérience, avec toute police encodée T1 [8], c'est très rare) 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.

3.4 Le symbole euro

C'est une des questions les plus fréquemment posées en LaTeX : « Mais comment qu'on fait pour obtenir ce *%£$& de symbole euro ??? ».

Pour obtenir le symbole dollar, c'est simple, c'est la commande `\$`, la contre-oblique étant nécessaire puisque LaTeX utilise le caractère `$` lui-même pour délimiter une formule mathématique dans le cours du texte, ainsi qu'on le verra en section 5. Mais pour le symbole euro, c'est plus problématique. Bon, c'est sans doute moins important en Suisse, mais ça vaut quand même la peine d'être abordé ici.

La réponse standard est d'utiliser le paquet de commandes `textcomp` :

```
\usepackage{textcomp}
```

qui fournit (entre autres) la commande `\texteuro`. Celle-ci composera automatiquement le symbole euro contenu dans la police couramment utilisée, si celle-ci en contient un, ce qui est le cas la plupart du temps. Un exemple :

```
Ce logiciel m'a coûté 500~\texteuro, une paille !
```

Notez la commande `~` insérée entre `500` et `\texteuro`. Elle introduit une *espace insécable* à cet endroit. Je vais y revenir en section 3.7.

À la compilation, on obtiendra ceci avec les polices par défaut, *Computer Modern*, et les polices *Latin Modern* (qui ont la même apparence, je le rappelle) :

Ce logiciel m'a coûté 500 €, une paille !

8. Si vous souhaitez une police encodée T1 avec un autre look que celui des *Computer Modern*, il y a beaucoup de possibilités. Par exemple, on peut utiliser les polices *mathptmx*, basées sur les polices de texte Times et les complétant par des polices mathématiques. Il suffira de charger le paquet `mathptmx` à la place de `lmodern`. Si vous avez envie d'en savoir plus, vous trouverez un large panorama des polices disponibles avec LaTeX sur le site de Benoît Rivet : http://benoit.rivet.free.fr/tex/tex_polices.htm.

Si on utilise l'encodage d'entrée UTF-8 Unicode, on peut tout simplement entrer le symbole euro directement au clavier, avec la combinaison de touches alt-$ sur clavier français, alt-e sur clavier suisse. Mais même alors il faut avoir chargé le paquet `textcomp`.

Dans le cas où la police courante ne contient pas de symbole euro, `textcomp` crée alors un « euro du pauvre », en plaçant deux barres horizontales au milieu du C majuscule de la police. Le résultat n'est la plupart du temps pas très heureux esthétiquement parlant.

Il vaut mieux alors utiliser un paquet tel qu'`eurosym` qui fournit une commande appelée tout bêtement `\euro`. Cette commande compose le logo de la monnaie européenne commune dans sa forme « officielle », laquelle est invariable quelle que soit la famille de polices utilisée. Exemple :

```
Ce logiciel m'a coûté 500~\euro, une paille !
```

Cela donnera :

Ce logiciel m'a coûté 500 €, une paille !

Évidemment, avec cette démarche on perd en adaptabilité à la police en cours. Un autre inconvénient est qu'on ne peut plus entrer ce symbole directement au clavier.

3.5 Francisation complète de LaTeX

Nous avons vu au cours de cette partie et de la précédente tout ce qui est nécessaire à la francisation de LaTeX. Il est temps maintenant d'en faire le bilan.

Pour toute la suite, afin d'assurer une francisation parfaite, nous supposerons que nos programmes LaTeX contiennent les quatre lignes suivantes dans leur préambule :

```
\usepackage[applemac]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern,textcomp}
\usepackage[frenchb]{babel}
```

Bien sûr, l'option du paquet `inputenc` pourra être différente selon l'encodage de texte — ne pas confondre, encore une fois, avec l'encodage des polices LaTeX ! — que vous avez choisi pour votre éditeur. Ici nous supposerons que TeXShop fonctionne avec l'encodage Mac OS Roman, d'où l'option `applemac`.

Nous verrons bien d'autres paquets bien utiles à ajouter au préambule, mais ce sera pour autre chose que la francisation.

3.6 Les modèles de TeXShop

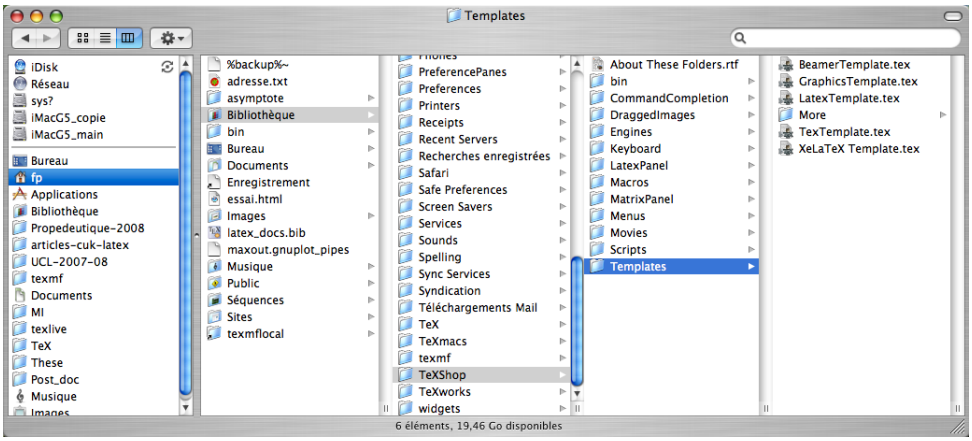
Je vous entends déjà râler : « pfff, tout ce code supplémentaire à entrer pour avoir une francisation convenable... »

C'est là que TeXShop intervient, encore une fois, pour simplifier tout ça. Grâce à une autre de ses fonctionnalités bien pratiques : les *modèles* (*templates* en anglais).

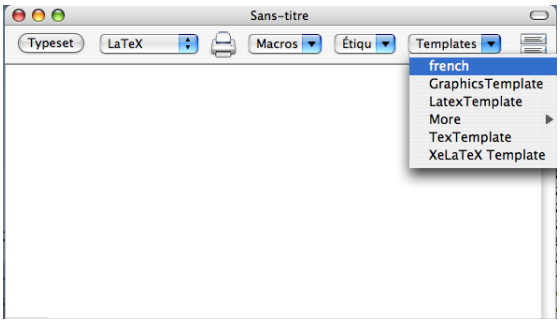
Voyons de quoi il s'agit sur un exemple : vous venez de voir qu'un document LaTeX parfaitement francisé devrait ressembler à cela (avec les éventuelles variantes d'encodage et de classe, bien sûr) :

```
\documentclass[a4paper,12pt]{book}
\usepackage[applemac]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern,textcomp}
\usepackage[frenchb]{babel}
\begin{document}
Mon texte à moi.
\end{document}
```

Nous allons faire de ce document un « modèle » pour les autres documents LATEX en français qui suivront. Pour cela, enregistrez-le, sous le nom de votre choix, par exemple `french.tex`, dans le sous-dossier `Templates` du dossier `TeXShop` qui se trouve dans votre `Bibliothèque` personnelle.



Une fois ceci fait, quittez et redémarrez TEXShop. Ouvrez un nouveau fichier (la sempiternelle opération command-n), et dans la barre d'outils de cette nouvelle fenêtre, à droite, cliquez sur le menu déroulant intitulé Templates. Vous y verrez apparaître, entre autres, votre fichier `french` (sans l'extension `.tex`) :



Sélectionnez-le, et le code du fichier `french.tex` sera entièrement transféré dans votre nouveau document ! Ne reste plus qu'à entrer votre texte proprement dit, entre les `\begin{document}` et `\end{document}`, à la place de « `Mon texte à moi` ».

Ainsi, à chaque fois que vous voulez entrer un texte français, vous pourrez aller dans ce menu déroulant et choisir le modèle french avant de commencer votre travail, sans avoir à rentrer vous-mêmes le code nécessaire à la francisation. C'est-y pas mieux comme ça ?

Vous avez pu constater que TeXShop avait d'autres modèles en réserve. Un modèle LaTeXTemplate, par exemple, mais qui ne contient aucune de ces francisations. Ce sera à vous de jouer pour créer votre propre modèle de fichier, celui le plus adapté à votre utilisation, peut-être en partant d'un de ces modèles déjà disponibles et en le modifiant.

3.7 Ponctuation et guillemets avec babel

En ce qui concerne la ponctuation, l'option frenchb de babel contraint LaTeX à respecter les règles en vigueur à l'Imprimerie nationale française, qui font référence en France, mais pas forcément en Suisse romande, si j'en crois quelques commentaires d'un article récent sur cuk.ch concernant Antidote (http://www.cuk.ch/articles/3819/#commentaires). À savoir l'insertion d'une *espace fine insécable* devant les points-virgule, points d'exclamation et d'interrogation, et d'une *espace-mot insécable* devant le double-point.

Rappelons qu'une espace insécable empêche la coupure de ligne à l'endroit où elle est insérée. L'espace-mot insécable a la longueur d'une espace normale entre deux mots. L'espace fine insécable est *grosso modo* deux fois plus petite.

Grâce à frenchb, que vous tapiez dans votre document LaTeX votre texte sans espace avant chaque ponctuation haute, « à la Suisse romande », comme ceci :

```
Vraiment? Alors là, je dis: bravo!
```

ou que vous insériez un espace-clavier avant chaque ponctuation haute :

```
Vraiment ? Alors là, je dis : bravo !
```

on obtiendra rigoureusement le même résultat dans le document de sortie :

Vraiment ? Alors là, je dis : bravo !

avec les espace insécables convenables insérées aux bons endroits.

Pour information, les instructions LaTeX insérant explicitement des espaces insécables sont respectivement

- « ~ », le « tilde » espagnol (qui s'obtient par la combinaison alt-n) pour l'espace-mot insécable ;
- « \, » pour l'espace fine insécable.

Attention, si vous utilisez frenchb, n'insérez pas vous-même ces espaces insécables avant les ponctuations hautes. On a vu que frenchb s'en chargeait déjà. Si vous le faisiez quand même, vous obtiendriez des espaces doublées en longueur.

Mais il y a bien d'autres circonstances où il faudra insérer soi-même une espace insécable. On en a déjà vu un exemple en section 3.4, et on en verra d'autres par la suite. Il faudra alors utiliser l'une au l'autre des instructions précédentes, selon le type d'espace insécable requis.

Frenchb fournit également deux commandes permettant d'obtenir les guillemets à doubles chevrons : \og et \fg. Exemple :

```
Les \og double chevrons \fg{} sont les vrais guillemets français.
```

Résultat :

Les « double chevrons » sont les vrais guillemets français.

(Vous rappelez-vous pourquoi a été insérée une paire d'accolades `{}` juste après la commande `\fg` ?)

Ces commandes ont l'avantage d'insérer automatiquement des espaces insécables de longueur adéquate. Pour ceux qui préfèrent introduire les guillemets directement au clavier, ce sont les combinaisons alt-è et alt-shift-è sur un clavier français qui permettent de les obtenir. Mais dans ce cas insérez vous-même les espaces insécables à l'aide du symbole `~`, ou bien modifiez le *setup* de `frenchb` en insérant en préambule la commande

```
\frenchbsetup{og=«,fg=»}
```

disponible dans les plus récentes versions de `frenchb`. Ceci rend les guillemets du clavier équivalents aux commandes `\og` et `\fg`, et les espaces insécables seront alors insérés automatiquement.

Si vous préférez les guillemets anglo-saxons, ils s'obtiennent en entrant « ``` » (deux accents graves) et « ´´ » (deux apostrophes), ou bien directement au clavier par les combinaisons alt-" et alt-shift-" sur clavier français, alt-2 et alt-shift-2 sur clavier suisse. Évidemment, il n'y a pas d'espace à insérer entre ces guillemets-là et le texte qu'ils encadrent.

Il est parfaitement possible de configurer `frenchb` afin de le rendre plus conforme à une autre conception de la typographie française. La documentation de `frenchb` (http://daniel.flipo.free.fr/frenchb/frenchb2-doc.pdf), très claire et instructive, vous renseignera sur les autres commandes bien pratiques qu'elle propose.

3.8 Table des matières

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 LaTeX c'est un jeu d'enfant ! Il 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}
...
\begin{document}
\title{Un petit exemple simple}
\author{Fabien Conus}
\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, LaTeX 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. Il aura utilisé pour cela, lors de cette deuxième compilation, les informations contenues dans le fichier à extension `.toc` qui a été créé lors de la première compilation.

Si vous désirez ajouter un chapitre ou une section qui ne soit pas numéroté 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}
```

3.9 Les références croisées

Il est parfois utile dans un texte de faire une 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.
```

Notez l'usage de l'espace-mot insécable entre le mot « chapitre » et sa numérotation.

Mais en reprenant mon texte le lendemain, je décide d'introduire un nouveau 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 fausse puisque le chapitre 2 est devenu le chapitre 3. Bien sûr, pour un petit document cela ne pose pas trop de problèmes 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 `mysteres` (il faut éviter les accents, ainsi que les espaces 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, sous-sections et autres sous-sous-sections.

Lorsque vous compilez votre document après avoir ajouté des références, celles-ci seront d'abord remplacées par deux points d'interrogation « `??` » et vous verrez dans la fenêtre de console 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. Recompilez
pour obtenir des références correctes.
```

En effet, pour la même raison qui expliquait la table des matières 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. Lors de cette deuxième compilation, LaTeX se servira des données qui ont été recueillies dans le fichier `.aux` créé lors de la première compilation.

LaTeX fournit également la commande `\pageref`, qui fonctionne selon le même principe que la commande `\ref` mais qui indique un numéro de page du document de sortie. Cette page est celle correspondant à l'endroit où se trouve la commande `\label` de même étiquette que celle de `\pageref`.

3.10 Notes de bas de page et notes marginales

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 à la suite du mot concerné par la note. Par exemple :

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

donnera

Les CFF [9] transportent chaque année des millions de passagers.

Comme vous le voyez, LaTeX a ajouté un petit numéro à la suite de « `CFF` » et a créé une note de bas de page correspondante contenant « `Chemins de Fer Fédéraux` ». Bien entendu, la numérotation des notes est automatique, ainsi que leur mise en page, et le compteur des notes de bas de page tient compte des notes produites précédemment.

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

9. Chemins de Fer Fédéraux

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

produira

Les CFF transportent chaque année des millions de passagers. à vérifier

Ces deux commande sont accessibles depuis le menu Macros, dans le sous-menu Offsets.

Vous avez à présent les clés en main pour faire de beaux documents. Dans la prochaine section nous verrons comment insérer des objets dans votre texte (images, liste et tableaux).

4 Listes, figures, tableaux et flottants

Grâce aux parties précédentes, vous avez tout le nécessaire 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 section 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`. Un exemple :

```
\begin{itemize}
\item Un élément ;
\item un autre élément ;
\item et encore un autre...
\end{itemize}
```

Après compilation, on obtient

- Un élément ;
- un autre élément ;
- et encore un autre...

Vous noterez le retour à la ligne automatique entre chaque élément (pas besoin de saut de ligne ou de commande `\\`). Vous aurez aussi noté au passage l'utilisation de la commande `\dots` qui correspond aux trois points de suspension « ... », qu'on peut aussi entrer directement au clavier si on dispose d'un encodage de texte contenant ce symbole : Mac OS Roman, UTF-8 Unicode par exemple, mais malheureusement pas ISO Latin 1.

Il existe aussi un environnement `enumerate` qui fonctionne comme `itemize` mais qui remplace les puces par des numéros. Exemple :

```
\begin{enumerate}
\item Premier élément.
\item Deuxième élément.
\item Troisième élément, etc.
\end{enumerate}
```

Résultat :

1. Premier élément.
2. Deuxième élément.
3. Troisième élément, etc.

On dispose également de l'environnement `description` permettant d'établir des listes de… descriptions (mais où va-t-il chercher tout ça ?). Chaque item doit être muni d'un argument optionnel, qui n'est donc plus si optionnel que ça et qui contient le nom à décrire.

```
\begin{description}
\item[Word] Traitement de texte \textsc{wysiwyg}
potentiellement nuisible à votre santé mentale.
\item[Open Office Writer] Traitement de texte
\textsc{wysiwyg} cherchant trop à ressembler au précédent.
\item[\LaTeX] Époustouflant formateur de texte non \textsc{wysiwyg}.
\end{description}
```

Résultat :

Word Traitement de texte WYSIWYG potentiellement nuisible à votre santé mentale.

Open Office Writer Traitement de texte WYSIWYG cherchant trop à ressembler au précédent.

LaTeX Époustouflant formateur de texte non WYSIWYG.

Il est bien sûr possible d'emboîter des listes entre elles. LaTeX adaptera alors de lui-même les numérotations éventuelles et introduira les indentations nécessaires. Voici un exemple de deux listes `enumerate` emboîtées :

```
\begin{enumerate}
\item Surtout rester zen.
\item Mais faire les courses quand même.
    \begin{enumerate}
    \item À la boulangerie d'abord ;
    \item À Carrefour ensuite.
    \end{enumerate}
\item Rentrer à la maison, râler quand même un peu, etc.
\end{enumerate}
```

On obtient dans ce cas

1. Surtout rester zen.
2. Mais faire les courses quand même.
    a) À la boulangerie d'abord ;
    b) À Carrefour ensuite.
3. Rentrer à la maison, râler quand même un peu, etc.

Vous pouvez faire quelques expériences de listes emboîtées, de même type ou non, et voir ce qui en résulte.

Remarquons que la présentation des listes est influencée par la langue choisie pour composer le document. Voyez-vous même ce qui se passe lorsque vous composez une liste `itemize` non plus avec le paquet `babel` option `frenchb`, mais sans `babel`, ou avec l'option `english`, ce qui revient au même.

Attention, si entre deux compilations vous changez la langue du document, c'est-à-dire l'option donnée à `babel`, il faut supprimer le fichier `.aux` précédemment créé avant de recompiler, sinon LaTeX coincera.

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 ceci à notre préambule :

```
\usepackage{graphicx}
```

Grâce à ce paquet, 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 entrant

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

Les formats d'image reconnus par pdfLaTeX (qui est le programme que TeXShop utilise par défaut) sont PDF, JPG et PNG (les extensions de fichier correspondantes 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 :

```
Voici une photo \includegraphics{medor.jpg} de mon chien Médor.
```

Comme en HTML, l'image fait alors partie intégrante du texte :

Voici une photo  de mon chien Médor.

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` :

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

Pour ajouter une légende, on la placera simplement à la ligne, par exemple en utilisant la commande `\\*`. Celle-ci fait la même chose que `\\`, mais en outre elle empêche le changement de page à l'endroit où elle est insérée (ce qui serait très malvenu entre une figure et sa légende) :

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

Résultat final :

Voici une photo

Photo prise en juin 02

de mon chien Médor.

(À tout hasard, je précise avoir trouvé la photo de ce bichon maltais « au pif » sur le Web, et que son nom n'y était pas mentionné.)

4.3 Les flottants

Il arrive également qu'on 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{figure}
\includegraphics{medor.jpg}
\end{figure}
```

L'image devient alors un *float element*, un *élément flottant*. Cela signifie que LaTeX réservera un espace quelque part pour cette seule image et s'occupera tout seul de son placement. S'il n'y a pas assez de place sur la page en cours, il la placera sur la page

suivante. S'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. Voici un exemple-type de code où on place un flottant :

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

Notez l'emploi de la commande-bascule `\centering` au lieu de l'environnement `center` pour centrer l'image. Ceci permet de ne pas rajouter d'espace vertical superflu entre la figure et le texte qui l'entoure, l'environnement `figure` en ayant déjà ajouté !

L'emploi d'un flottant ne garantira pas que l'image soit placée entre les deux paragraphes. Vous pouvez toutefois indiquer à LaTeX vos préférences en ajoutant une option :

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

`Option` est un (ou plusieurs) caractère(s) 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) :
- `p` : *page*, 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. De fait, si vous n'indiquez aucune option, l'environnement `figure` fera comme si vous aviez indiqué l'option de placement `tbp`. Si vous voulez essayer de forcer LaTeX à placer votre image là où vous le souhaitez, 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 placer votre image là où vous la voulez, il devient difficile de la mentionner. Dans le dernier exemple de la section 4.2, si l'image de mon chien Médor est rendue « flottante » et se trouve insérée 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 utiliser la commande `\caption` *(légende)*. Exemple :

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

FIGURE 1 – Voici la photo de Médor prise en juin 02.

Après compilation (comme je l'ai fait ici), LaTeX insère la figure là où il le juge adéquat, le plus proche possible cependant, et en cherchant à respecter vos options de placement. Dans ce cas, l'option `ht` incitera LaTeX à chercher à la placer à l'endroit même de l'insertion de l'environnement `figure` ou sinon en haut de page. Voyez sur cette page même où se trouve la figure et vérifiez si LaTeX a satisfait ou non à cette exigence. Voyez également que l'argument de la commande `\caption` est apparu en légende de la figure, avec devant une numérotation effectuée automatiquement.

L'argument de la commande `\caption` peut rester vide si vous ne désirez pas placer de légende, dans ce cas seule la numérotation apparaîtra.

Pour faire référence à la figure dans le cours du texte, on fait appel à l'étiquette introduite dans l'argument de la commande `\caption`, grâce à la commande `\ref`, comme ici :

```
Sur la figure~\ref{medor}, vous pouvez voir la photo de Médor.
```

Voilà ce que cela donnera (après deux compilations, comme de juste) :

Sur la figure 1, vous pouvez voir la photo de Médor.

Une particularité problématique à laquelle vous serez vite confronté : supposons que vous souhaitiez imposer un retour à la ligne dans la légende en utilisant les commandes `\\` ou `\newline` dans l'argument de `\caption`. Ou bien que vous vouliez y créer un nouveau paragraphe (en insérant une ligne blanche ou une commande `\par`). Hé bien, au pire vous obtiendrez des erreurs, au mieux ces commandes n'auront aucun effet...

La première étape pour résoudre ce problème est d'utiliser la commande `\caption` avec une option entre crochets contenant une version « courte » de votre légende. Cette version courte sera alors celle qui paraîtra dans la liste des figures, à la place de la légende complète. Exemple :

```
\caption[Médor, juin 2002]{Voici la photo de Médor\\
prise en juin 2002}
```

Ainsi votre légende ne risquera plus d'être trop longue pour paraître dans une éventuelle table des figures (l'analogue pour les figures de la table des matières traitée à la leçon

précédente, on en reparlera un peu plus loin) et les commandes `\\` ou `\par` n'entraîneront plus d'erreurs. Mais elles continueront à n'avoir aucun effet sur le texte de la légende... Un moindre mal, certes, mais quand même ! D'où la deuxième étape de la résolution du problème : utiliser le paquet `caption` (à ne pas confondre avec la commande `\caption` !). Insérez donc en préambule la ligne suivante :

```
\usepackage{caption}
```

et le problème est enfin complètement réglé.

Le paquet `caption` ne fait pas que cela, il permet une multitude de personnalisation des légendes de flottants, mais cela dépasse le cadre de ce cours.

4.4 Manipuler une image

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 une ou des options à la commande `\includegraphics`. Si vous utilisez plusieurs options, les affectations de valeur à chaque option doivent être séparées par une virgule :

```
\includegraphics[option1=valeur1,option2=valeur2,...]{monimage.jpg}
```

Voici un tableau listant les principales options et les valeurs qu'ils peuvent prendre :

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 a d'autres options, 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. Le facteur requis est le rapport de taille 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).

Ainsi, la commande suivante :

```
\includegraphics[scale=0.5,angle=120]{medor.jpg}
```

rapetisse ce pauvre Médor de moitié et le fait pivoter de 120 degrés :

L'option `draft` (« brouillon »), permet d'afficher un rectangle vide en lieu et place de l'image. Les dimensions de ce rectangle seront toutefois celles de l'image. Cette option peut être utile en cours d'élaboration de votre document (son « brouillon »), si la figure en question occupe une grande place mémoire, car dans ce cas son incorporation peut ralentir notablement la compilation. On enlèvera cette option au moment de la finalisation du document. Si vous avez de nombreuses figures de ce type, il peut être judicieux de donner cette option au paquet `graphicx` lui-même :

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

Dans ce cas, ce seront toutes les figures de votre document qui seront remplacées par des rectangles vides.

Les options `width` et `height` (« largeur » et « hauteur ») prennent comme valeur une dimension. Cette dimension peut être donnée de plusieurs manières. En millimètres ou centimètres bien sûr, par exemple `5cm`, ou en points (`pt`), 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 a 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 LaTeX, mais celles-ci sont beaucoup plus sophistiquées. Ce que je viens de vous décrire devrait suffire pour la plupart des cas (je n'ai jamais utilisé d'options plus poussées).

Comme je l'ai déjà mentionné, il existe un équivalent de la table des matières 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).

4.5 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 :

```
Voici un tableau :
\begin{tabular}{lcr}
Ligne 1 & Élément 1 & Élément 2\\
Ligne 2 & Élément 3 & Élément 4\\
Ligne 3 & Élément 5 & Élément 6
\end{tabular}
```

Étudions 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 puisqu'il y a 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 (`c` pour *center*) 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 `&` (l'esperluette). Pour terminer une ligne du tableau, on utilise la commande `\\` sauf pour la dernière ligne qui se termine par `\end`.

Si vous compilez l'exemple précédent, vous obtiendrez ceci :

	Ligne 1	Élément 1	Élément 2
Voici un tableau :	Ligne 2	Élément 3	Élément 4
	Ligne 3	Élément 5	Élément 6

On y remarque deux choses :

- la création d'un tableau n'implique pas de création de paragraphe ou même 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 pallier la première constatation et mettre le tableau sur une nouvelle ligne, vous pouvez certes utiliser la commande `\\`. Mais il y a beaucoup plus élégant, vous pouvez utiliser l'environnement `center` que nous avons défini dans la partie précédente. Le tableau sera alors placé sur un nouveau paragraphe et centré.

Pour ajouter des traits verticaux à notre tableau, nous allons modifier la chaîne de caractères définissant le tableau. Si nous voulons séparer la première colonne des deux autres par un trait vertical, nous allons transformer `lcr` en `l|cr`. Ce caractère vertical est obtenu sur un clavier suisse-romand par alt-7 et sur un clavier français, alt-shift-`l`. Si nous voulons séparer toutes les colonnes par un trait vertical, nous utiliserons `l|c|r`, et enfin pour que notre tableau soit délimité par deux traits verticaux nous utiliserons `|l|c|r|`. Logique. Bien sûr, vous pouvez utiliser plusieurs traits 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{center}
\begin{tabular}{|l||c|r|}
\hline
Colonne 1 & Colonne 2 & Colonne 3\\
\hline\hline
Ligne 1 & Élément 1 & Élément 2\\
\hline
Ligne 2 & Élément 3 & Élément 4\\
\hline
Ligne 3 & Élément 5 & Élément 6\\
\hline
\end{tabular}
\end{center}
```

Voici le résultat :

Colonne 1	Colonne 2	Colonne 3
Ligne 1	Élément 1	Élément 2
Ligne 2	Élément 3	Élément 4
Ligne 3	Élément 5	Élément 6

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

```
\begin{center}
\begin{tabular}{|l||c|r|}
\hline\\
Colonne 1 & Colonne 2 & Colonne 3\\
\hline\hline\\
Ligne 1 &
    \begin{tabular}{c|c}
    Tableau & dans\\
    \hline
    un & tableau
    \end{tabular}
& Élément 2\\
\hline
Ligne 2 & Élément 3 & Élément 4\\
\hline
Ligne 3 & Élément 5 & Élément 6\\
\hline
\end{tabular}
\end{center}
```

Le résultat est cette fois-ci :

Colonne 1	Colonne 2		Colonne 3
Ligne 1	Tableau dans un tableau		Élément 2
Ligne 2	Élément 3		Élément 4
Ligne 3	Élément 5		Élément 6

Pas mal, non ?

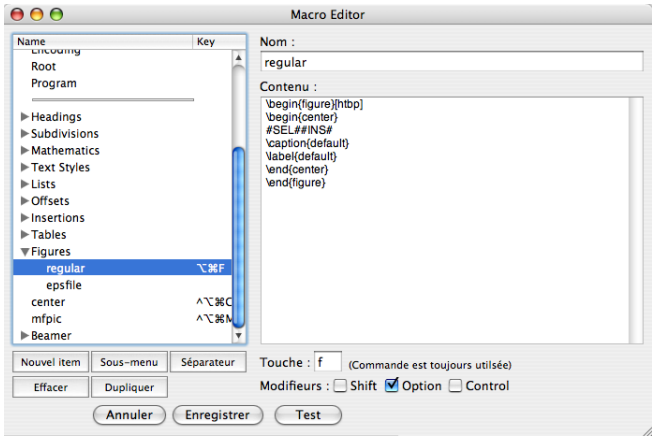
Comme pour les images, les tableaux peuvent être définis comme des éléments flottants auxquels on pourra alors faire référence. Pour les tableaux, on utilisera l'environnement `table` au lieu de l'environnement `figure`. La syntaxe reste la même. Par exemple :

```
\begin{table}[ht!]
\centering
\begin{tabular}{lcr}
Ligne 1 & Élément 1 & Élément 2\\
Ligne 2 & Élément 3 & Élément 4\\
Ligne 3 & Élément 5 & Élément 6
\end{tabular}
\caption{\label{beautableau}Mon beau tableau}
\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`.

4.6 Les macros, le retour

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 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-là 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 le lancement de la macro.

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 contre-obliques 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.

Je pense que c'est assez pour aujourd'hui, vous avez 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.

Dans la prochaine section nous attaquerons l'écriture des mathématiques avec LaTeX.

5 Les mathématiques avec LaTeX

Comme je l'ai écrit en commentaires (hélas aujourd'hui disparus) du test de Renan Fuhrimann sur MathMagic (http://www.cuk.ch/articles/2092), 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). À ce sujet, vous pouvez aller voir un article publié par le service informatique de l'EPFL à cette adresse : http://sawww.epfl.ch/SIC/SA/publications/FI99/fi-2-99/2-99-page1.html ; vous y trouverez entre autres une comparaison du rendu des équations avec Word, FrameMaker et LaTeX.

L'écriture des mathématiques dans LaTeX est très, très complète, et il me serait donc bien difficile d'être exhaustif. Je vous en propose donc un tour d'horizon plutôt qu'une visite complète, à l'image des autres articles de cette série.

Les mathématiques peuvent s'écrire soit dans le cours du texte, soit en dehors. Nous commencerons par examiner le premier cas.

5.1 Les formules dans le cours du texte

Pour écrire une équation dans le texte, il faut simplement l'encadrer de deux signes $. L'équation est alors traitée dans le *mode mathématique* de LaTeX. 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 « ^ ».

Après compilation, voici ce qu'on obtient :

Einstein a écrit $E = mc^2$.

Vous aurez noté que les noms de variables sont automatiquement mis en italique dans la formule sortie, ainsi qu'il est de règle dans toute publication mathématique.

Précisons tout de suite qu'en mode mathématique, les espaces entrés au clavier ne sont *pas du tout* pris en compte. LaTeX prend lui-même complètement en charge les espaces entre symboles. Que vous ayiez entré

```
$E=mc^2$
```

ou

```
$E  =   m c^2$
```

serait revenu exactement au même pour la formule finale. Profitez-en pour « aérer » les codages de vos formules dans votre document LaTeX, ce sera apprécié à la relecture dudit document !

Cependant, LaTeX fournit plusieurs commandes pour insérer en mode mathématique une espace horizontale plus ou moins grande. Les plus utiles, vous les connaissez déjà, elles fonctionnent aussi dans une formule : « `\,` »pour une espace fine,et « `\␣` » (contre-oblique + espace-clavier) pour une espace-mot. Vous ne devriez avoir à les utiliser que très rarement : LaTeX se débrouille en général très bien tout seul dans la gestion des espaces entre symboles mathématiques.

Autre particularité du mode mathématique de LaTeX : les sauts de ligne n'y sont pas admis. Si vous laissez une ligne blanche au milieu du code de votre formule, LaTeX signalera une erreur.

Essayons maintenant une formule plus compliquée :

```
Mais Einstein 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 « `_` ». Voyez le résultat :

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

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 en indice ou en exposant, il faut utiliser les accolades. Par exemple, le code

```
$2^2+6 = 256$
```

donnera

$$2^2+6 = 256$$

et donc une équation fausse, tandis que

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

donnera

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

ce qu'on voulait obtenir.

Pour écrire une fraction, on utilise la commande `\frac` :

```
$\frac{x}{y} = 2$.
```

Cette commande et quasi toutes celles que vous allez voir dans cette partie ne fonctionnent que lorsqu'elles sont entrées en mode mathématique. C'est-à-dire qu'elle doit se trouver entre deux signes `$` ou dans un des environnements que nous mentionnons ci-après. Il en va d'ailleurs de même des opérateurs de mise en exposant « `^` » et d'indice « `_` ».

5.2 Les formules hors-texte

Bien entendu, toutes les opérations que nous venons de voir peuvent s'emboîter les unes dans les autres, mais cela donne lieu à des équations de taille assez conséquente. Il ne sera donc pas conseillé de les placer dans le texte, mais plutôt sur une nouvelle ligne, et centrées pour bien les mettre en évidence.

Pour cela, nous allons placer l'équation dans un environnement `equation` (qui l'eût cru ?). Cet environnement fait également entrer LaTeX en mode mathématique, et toutes les règles et commandes que vous avez vues précédemment pour les formules dans le cours du texte s'appliquent aussi ici. Mais l'environnement `equation` se charge en outre de mettre en valeur la formule : il la place sur une nouvelle ligne légèrement décalée vers le bas, la centre, et fait en sorte que les fractions et racines, par exemple, se déploient plus que si elles étaient placées en cours de texte. Ainsi, ce code :

```
\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

$$\frac{x^{\left(\sqrt{p^2c^2+m_0^2c^4}-y_0^4\right)} - 2y + 8}{(y+x)^2} = 4K_1 + 3x. \tag{1}$$

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

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

```
2(\frac{x}{y}) = 8
```

vous obtiendrez

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

ce qui n'est pas très joli. En utilisant « `\left(` » et « `\left)` », LaTeX 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, il faut utiliser les commande `\{` et `\}`. En effet, l'accolade elle-même est réservée par LaTeX pour les arguments des commandes.

Revenons à notre environnement `equation`. Vous aurez remarqué que l'équation a été automatiquement numérotée, avec le numéro placé en vis-à-vis à droite de l'équation. 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 faire la référence elle-même :

```
\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 numéroter une équation, vous la placerez simplement dans un environnement `displaymath`, ou plus simplement entre les commandes `\[` et `\]`, comme ceci :

```
\[
2\left(\frac{x}{y}\right) = 8.
\]
```

Mais vous ne pourrez alors pas y faire référence.

Signalons au passage une application importante des signes « ^ » et « _ » : indiquer aux opérateurs leurs éventuelles bornes supérieures et inférieures. C'est notamment le cas pour les opérateurs de sommation, d'intégration et de limite.

```
\[
\lim_{x \to \infty} \frac{1}{x} = 0
\qquad
\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}
\qquad
\int_0^1 x \, dx = \frac{1}{2}
\]
```

Pour cet exemple, nous avons introduit les commandes `\int`, `\sum` et `\lim` (dont le sens est suffisamment transparent) pour les opérateurs, ainsi que les commandes `\to`, `\pi` et `\infty` pour respectivement la flèche à droite, la lettre grecque pi et le symbole infini. Ce qui ici donnera

$$\lim_{x\to\infty} \frac{1}{x} = 0 \qquad \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \qquad \int_0^1 x\,dx = \frac{1}{2}$$

Les équations ci-dessus sont restées sur la même ligne, séparées par une espace horizontale produite par la commande `\qquad`, de longueur égale à deux cadratins (le cadratin est la taille de la police courante, 10, 11 ou 12 points). Cette nouvelle commande d'espacement est également valable en mode texte, de même que la commande `\quad` produisant une espace d'un cadratin.

5.3 Alignement vertical des équations

Le principal handicap des environnements du type `equation` ou `displaymath` est précisément qu'ils ne permettent pas l'écriture sur plusieurs lignes. Pour pallier ceci, LaTeX fournit 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 l'esperluette `&` pour aligner les éléments. Dans cet exemple, on s'assure que les trois signes `=` seront alignés verticalement :

$$\begin{eqnarray}
2\left(\frac{x}{y}\right) &=& 8, \quad (2)\\
\frac{x}{y} &=& \frac{8}{2} \quad (3)\\
&=& 4. \quad (4)
\end{eqnarray}$$

Avec l'environnement `eqnarray`, comme vous avez pu le constater, chaque ligne est numérotée. Pour y faire référence, il s'agira de placer un `\label` à la fin de chaque ligne (avant la commande `\\`). Si vous ne désirez pas numéroter les équations, il faut utiliser l'environnement `eqnarray*`, avec une étoile ajoutée à la fin du nom de l'environnement, donc.

Si vous êtes pinailleur sur la qualité de rendu des équations, vous aurez remarqué que `eqnarray` a inséré des espaces particulièrement grandes autour des signes `=`. Pour éviter ce phénomène gênant, il est recommandé d'utiliser l'environnement `align` fourni par le paquet de commandes `amsmath`. Placez donc

```
\usepackage{amsmath}
```

en préambule, et remplacez le code précédent par celui-ci :

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

Le résultat est maintenant nettement meilleur :

$$\begin{aligned}
2\left(\frac{x}{y}\right) &= 8, \quad (5)\\
\frac{x}{y} &= \frac{8}{2} \quad (6)\\
&= 4. \quad (7)
\end{aligned}$$

Notez qu'une seule esperluette par équation a été nécessaire à l'environnement `align` au lieu de deux pour `eqnarray`. Les équations sont également numérotées, et bien sûr, on peut faire référence à chaque équation, de la même manière que précédemment. Il existe là aussi un environnement étoilé, `align*`, qui supprime cette numérotation.

Il vous faudra choisir entre l'environnement `eqnarray` et l'environnement `align` du paquet `amsmath`, car une fois ce dernier chargé, `eqnarray` risque de fonctionner de travers.

Pour ma part le choix est fait depuis longtemps, car le paquet `amsmath` fournit également quantité d'autres environnements et commandes, qui deviennent vite indispensables à tout scientifique rédigeant des articles à fort contenu mathématique. Si vous êtes concernés, voyez sa documentation, par exemple (en anglais) à l'adresse ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf, pour avoir une idée de ses possibilités.

5.4 Du texte dans les équations

Une autre commande très utile fournie par le paquet `amsmath` est `\text`. Elle permet d'insérer du texte à l'intérieur même d'une équation, tout en conservant la police et le style du texte utilisés hors équation. En fait, elle permet au texte qu'elle prend en argument d'être composé en mode texte et non en mode mathématique. Ainsi, voyez par vous-même la différence entre :

```
\[
E_{pot} = mgh
\]
```

$$E_{pot} = mgh$$

et

```
\[
E_{\text{pot}} = mgh
\]
```

$$E_{\text{pot}} = mgh$$

Si l'on souhaite un style particulier pour ce texte, qui ne soit pas celui environnant l'équation, vous pouvez utiliser soit une des commandes de changement de style déjà vues, du genre :

```
\textbf{texte}
```

soit utiliser les commandes analogues spécialement prévues par LaTeX pour le mode mathématique. Il y en a une pour chaque style (sauf les styles « incliné » et « petites capitales ») :

Italique	`\mathit{texte}`
Gras	`\mathbf{texte}`
Machine à écrire	`\mathtt{texte}`
Sans serif	`\mathsf{texte}`
Romain	`\mathrm{texte}`

Mais attention, contrairement à `\text`, le texte en argument de ces dernières commandes est composé en mode mathématique. En particulier, on l'a vu, les espaces entrés au clavier seront purement et simplement ignorés.

De plus, les caractères accentués ne doivent pas être entrés directement dans une formule mathématique (sous peine d'engendrer une floppée de *Warnings* dans la console). Les accents font l'objet de commandes spéciales en mode mathématique, et ces commandes sont différentes même de celles qu'on utilisait dans les temps anciens de LaTeX pour le mode texte (cf. section 2.6). Par exemple, pour obtenir un « é », on devra entrer `\acute{e}` pour l'accent aigu sur le e, au lieu de `\'e`. Ce qui devient vite très, très lourd à gérer.

Il est donc conseillé de réserver l'emploi des commandes `\math...` aux symboles et variables mathématiques qu'on voudrait dans un style donné, et d'utiliser les commandes `\text...` pour les véritables portions de texte. Par exemple, sachant que les conventions mathématiques exigent souvent qu'un vecteur soit noté en gras, pour obtenir en mode mathématique la propriété vectorielle suivante :

$$a\mathbf{x} = \mathbf{0} \qquad \text{équivaut à} \qquad a = 0 \quad \text{ou} \quad \mathbf{x} = \mathbf{0}$$

où a est un réel, $\mathbf{x}$ un vecteur et $\mathbf{0}$ le vecteur nul, j'ai entré ceci :

```
a \mathbf{x} = \mathbf{0}
\qquad \text{équivaut à} \qquad
a = 0 \quad \text{ou} \quad \mathbf{x} = \mathbf{0}
```

5.5 Les fonctions usuelles

Par convention, en mathématiques, les noms des fonctions usuelles (log, tan, sin, etc.) doivent être transcrits en caractères droits, contrairement aux symboles et variables qui, on l'a vu, doivent être en italique. Pour écrire les noms de ces fonctions, on dispose donc de commandes particulières, plus rapides d'emploi qu'en passant par `\text` ou `\mathrm` :

tangente	`\tan`
cosinus	`\cos`
sinus	`\sin`
log népérien	`\ln`
logarithme	`\log`

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

5.6 Matrices

Pour écrire une matrice (et donc un vecteur), on utilise l'environnement `array`. Il fonctionne exactement comme l'environnement `tabular` vu à la leçon précédente, à cette différence près qu'il ne fonctionne qu'en mode mathématique. Par exemple, le code

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

donnera

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

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

Le paquet `amsmath` fournit quelques autres environnements spéciaux pour créer des matrices, plus simples à coder, mais un peu moins souples. Voyez, là aussi, sa documentation.

5.7 Les palettes de TEXShop

Comme vous le savez sûrement, il est d'usage courant en mathématiques d'utiliser des lettres grecques pour désigner certaines variables du problème.

Avec LATEX, ce n'est pas plus compliqué que de connaître votre alphabet grec (vous en savez déjà au moins deux lettres : « alpha » et « beta » qui ont donné leur nom à l'alphabet). En effet, il suffit d'entrer une contre-oblique suivi du nom de la lettre. Si vous mettez une majuscule à l'initiale du nom de la lettre grecque, vous obtiendrez sa version majuscule, si vous mettez une minuscule vous obtiendrez sa version minuscule. Par exemple,

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

donnera

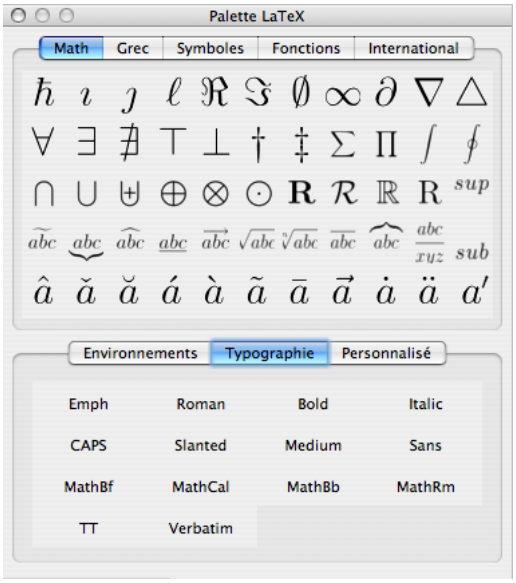
$$2\pi + \Omega = \xi \cos(\theta + \phi)$$

N'oubliez pas que les commandes donnant les lettres grecques, comme quasi toutes celles présentées dans cette section, ne fonctionnent qu'en mode mathématique. Par conséquent, même si vous souhaitez obtenir ces lettres en dehors d'une formule proprement dite, encadrez-les par deux symboles $, comme ci-dessous :

```
Les lettres $\alpha$ et $\beta$ sont très employées en mathématiques.
```

Si vous ne le faites pas, LATEX ne reconnaîtra tout simplement pas ces commandes et les signalera comme des erreurs.

Comme on n'attend pas de vous que vous sachiez votre alphabet grec sur le bout des doigts, TEXShop met à votre disposition la Palette LaTeX, qui permet d'accéder à une large gamme de symboles par le biais de la souris. Vous trouverez cette palette dans le menu Fenêtres :



Dans la partie supérieure, les quatre premiers onglets sont destinés aux mathématiques, avec tout d'abord les symboles (constante de Planck, opérateurs d'ensembles, l'ensemble des nombres réels, etc.), puis les lettres grecques, puis quelques autres symboles (opérateurs logiques, parenthèses, etc.), puis les fonctions trigonométriques dont je vous parlais à l'instant.

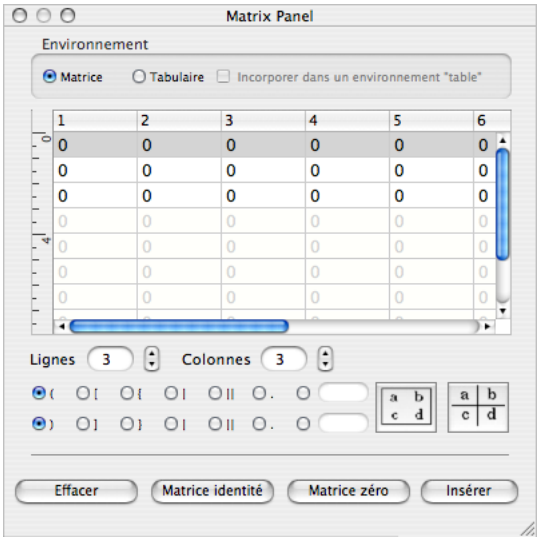
Le dernier onglet, intitulé International, concerne le mode texte. Il n'est pas très important pour nous puisqu'il permet d'écrire les caractères accentués « à l'ancienne », alors que nous avons appris en section 2.6 comment les entrer directement au clavier.

Bizarrement, TEXShop oublie de mentionner dans cette palette que les commandes donnant certains symboles importants (par exemple `\mathbb{R}` qui donne l'ensemble des nombres réels) ne sont définies que si on charge un nouveau paquet : `amssymb`. Comme pour `amsmath`, je vous conseille donc de le charger systématiquement en préambule de votre document :

```
\usepackage{amsmath,amssymb}
```

La partie inférieure de notre Palette LaTeX possède trois onglets. Le premier reprend les environnements les plus courants (`center`, `figure`, `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 (ou de tableaux, selon l'option que vous cochez dans le premier champ). Vous pouvez y définir la taille de la matrice, son contenu, ainsi que les contenants (parenthèses, crochets, norme, etc.)

Ces deux 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 en rester à un court aperçu et à la fin de l'article, je vous donnerai une série de références et de liens qui vous aideront à approfondir le sujet.

Je n'irai donc pas plus loin avec les mathématiques et je vous donne rendez-vous pour la prochaine et dernière partie de notre découverte de LATEX.

6 Compléments

Cette section correspond *grosso modo* à la sixième et dernière partie de l'ancienne version de l'*Introduction à LaTeX* de Fabien Conus. Dans cette mise à jour, j'ai choisi de la séparer de la conclusion générale et de la liste des références bibliographiques, que vous trouverez donc dans les deux sections suivantes.

Étant donné les circonstances LaTeXiennes actuelles, cette section-ci a été entièrement réécrite (à l'exception de la sous-section consacrée à `hyperref`), bien que l'idée générale soit restée la même : fournir des compléments très utiles sur TeXShop et LaTeX, et répondre à quelques demandes courantes de leurs utilisateurs. Elle se terminera par quelques considérations sur les récentes évolutions et le futur de LaTeX.

6.1 La fragmentation

C'est une des forces de LaTeX : il peut traiter des documents de très grande taille sans broncher le moins du monde. Alors que Word, dans la même situation, a tendance à exploser en route.

Cependant, même avec LaTeX, il peut être utile (mais pas indispensable) de scinder son fichier de travail en plusieurs parties. Par exemple, si on travaille sur un roman ou un rapport en plusieurs chapitres, et que l'on souhaite travailler sur un ou plusieurs chapitres particuliers, en laissant les autres inchangés. On souhaiterait alors éviter d'avoir à compiler tout le document, ce qui peut prendre un certain temps, alors que seule une partie de ce document était concernée par les modifications.

Pour résoudre ce problème, LaTeX fournit deux commandes très utiles : `\include` et `\includeonly`.

Voici la structure-type d'un programme utilisant la commande `\include` :

```
\documentclass[options]{une-classe}
...
\begin{document}
\include{partie1}
\include{partie2}
\include{partie3}
\include{complement}
\end{document}
```

Ceci constitue le *fichier principal*, ou *fichier racine*, qui doit contenir la déclaration de classe, le préambule et l'environnement `document`. C'est ce fichier qui doit être compilé. Appelons-le `fichier-principal.tex`.

Les fichiers `partie1.tex`, `partie2.tex`, `partie3.tex`, `complement.tex`, sont des fichiers que vous aurez créés et sauvegardés à part, dans le même répertoire que le fichier principal. Chacun d'entre eux pourra contenir une partie de votre texte : un chapitre, une section, ce que vous voulez. Ce sont les *fichiers annexes*. Ces fichiers annexes ne doivent contenir ni `\documentclass`, ni préambule, ni `\begin{document}` ou `\end{document}`,

qui sont l'affaire du seul fichier principal. Les contenus des fichiers annexes seront inclus dans le PDF produit après compilation du fichier principal, grâce aux commandes `\include`, et ce dans l'ordre d'introduction de ces commandes.

Attention, ne pas mettre l'extension `.tex` dans l'argument de la commande `\include`.

Pour travailler sur une ou plusieurs parties particulières, et ignorer les autres, on peut soit « commenter » les commandes `\include` correspondantes (en mettant un signe pourcentage `%` devant elles, ce qui les désactive), soit utiliser la commande `\includeonly` en préambule de `fichier-principal.tex`, avec en argument la liste des parties à inclure, séparées par des virgules. Par exemple :

```
\documentclass[options]{une-classe}
...
\includeonly{partie2,complement}
...
\begin{document}
\include{partie1}
\include{partie2}
\include{partie3}
\include{complement}
\end{document}
```

Ici, seuls les fichiers `partie2.tex` et `complement.tex` seront inclus dans le document de sortie.

L'intérêt de l'utilisation de `\includeonly` est que les diverses numérotations de chapitres, sections, sous-sections, etc., et les références croisées des parties incluses tiendront automatiquement compte des parties non incluses, à condition que le document ait été compilé au préalable avec toutes ses parties, avant d'entamer le travail sur les parties particulières.

A priori, c'est le fichier principal que l'on doit compiler, et il est impossible de compiler un des fichiers annexes. Mais TEXShop fournit un moyen de le faire : placer au tout début de chaque fichier annexe la ligne suivante :

```
%!TEX root = fichier-principal.tex
```

Ce commentaire-là sera reconnu par TEXShop (mais pas par un autre éditeur). Grâce à lui, si vous lancez la compilation sur un fichier annexe, TEXShop compilera en fait `fichier-principal.tex`.

TEXShop fournit aussi une macro vous aidant à rédiger cette ligne, si vous avez du mal à en retenir la syntaxe. Dans le menu Macros (toujours lui), choisissez l'option root *(racine)*, et celle-ci vous insérera `%!TEX root =` à l'endroit où se trouve le curseur. Il ne restera qu'à compléter avec le nom du fichier principal.

Notez que chaque nouvelle partie incluse grâce aux `\include` commencera sur une nouvelle page dans le document PDF de sortie. Si on trouve cela gênant, on peut utiliser la commande `\input` à la place de `\include` :

```
\input{partie}
```

L'inconvénient d'utiliser cette méthode est qu'on perd alors le bénéfice de la commande `\includeonly`.

6.2 KOMA-Script

Nous avons vu au cours de la section 3 comment franciser un document produit par LaTeX de façon convenable, avec l'utilisation des paquets `babel`, `fontenc`, `inputenc` et le choix des polices *Latin Modern*.

Cependant, LaTeX reste marqué par la mise en page et la présentation « à l'américaine ». Par exemple, le papier est par défaut supposé au format *letter*, et les marges des classes standards, même en option `a4paper`, sont légèrement trop grandes pour le goût européen. La titraille est en style romain alors que notre tradition les préfère plutôt en sans serif, la classe `letter` est seulement conforme aux standards américains, la taille par défaut de la police de base (10 points) est un peu trop petite pour nous, et d'ailleurs la mise en page ne s'adapte pas à la taille de la police en cours, etc.

Une des solutions les plus efficaces pour remédier à cela est d'utiliser les classes fournies par l'ensemble *KOMA-Script*. Cet ensemble a été créé dans ce but par des utilisateurs allemands de LaTeX, particulièrement Markus Kohm, qui supervise son développement actuel. De façon générale, KOMA-Script permet une personnalisation bien plus aisée du document que celle autorisée par les classes standards.

KOMA-Script, outre divers paquets très intéressants mais que je n'aborderai pas ici, propose quatre classes analogues aux classes standard de LaTeX. Le tableau ci-dessous montre ces classes en vis-à-vis de leur équivalent en LaTeX standard :

Classes standards	Classes KOMA-Script
`article`	`scrartcl`
`report`	`scrreprt`
`book`	`scrbook`
`letter`	`scrlttr2`

Si vous vous demandez pourquoi les noms des classes KOMA-Script sont si condensés, c'est qu'il semble que LaTeX ne permette pas pour ses classes l'emploi de noms de plus de huit lettres. En tout cas à une certaine époque. Les choses ont peut-être changé dernièrement, mais n'ayant pas d'expérience dans la création de classe, je ne peux pas en dire plus.

Chacune des classes KOMA-Script listées ci-dessus impose par défaut l'emploi du format A4, la création de marges moins importantes et une titraille en sans serif. Par défaut toujours, la taille de base de la police est de 11 points au lieu de 10, sauf pour la classe `scrlttr2`, où elle est de 12 points.

Qui plus est, KOMA-Script adapte la mise en page selon la taille de base de la police, et ce pour les trois tailles habituelles (10, 11 et 12 points). KOMA-Script permet en outre l'utilisation de tailles de base suppémentaires : 8, 9, 14, 17 et 20 points, à condition de charger aussi le paquet `extsizes`, qui se charge alors d'adapter la mise en page.

Quant à la classe `scrlttr2`, elle permet de composer des lettres d'allure très professionnelle selon les conventions allemandes et suisses. Cette dernière convention convient

également bien aux exigences françaises en ce domaine.

De plus, ces classes fournissent un grand nombre de macros et d'options supplémentaires qui permettent de personnaliser la mise en page de façon bien plus simple et efficace qu'avec les classes standards.

Pour vous en convaincre, un exemple : la délimitation des paragraphes. On a vu dès la deuxième leçon que LaTeX, dans sa configuration par défaut, distingue dans le document de sortie chaque paragraphe par une indentation de leur première ligne :

> Cette idole, yeux noirs et crin jaune, sans parent ni cour, plus noble que la fable, mexicaine et flamande ; son domaine, azur et verdure insolents, court sur des plages nommées, par des vagues sans vaisseaux, de noms férocement grecs, slaves, celtiques.
>
> À la lisière de la forêt — les fleurs de rêve tintent, éclatent, éclairent — la fille à lèvre d'orange, les genoux croisés dans le clair déluge qui sourd des prés, nudité qu'ombrent, traversent et habillent les arcs-en-ciel, la flore, la mer.

(Celui qui devine tout seul d'où est extrait ce texte a droit à toute ma considération !)

On peut vouloir que ce soit fait autrement : par exemple, qu'il y ait dans le document de sortie une ligne blanche entre chaque paragraphe. Et sans indentation, car imposer à la fois l'indentation et le saut de ligne serait redondant, selon les règles typographiques communément admises. Bref, on voudrait quelque chose comme ceci :

> Cette idole, yeux noirs et crin jaune, sans parent ni cour, plus noble que la fable, mexicaine et flamande ; son domaine, azur et verdure insolents, court sur des plages nommées, par des vagues sans vaisseaux, de noms férocement grecs, slaves, celtiques.
>
> À la lisière de la forêt — les fleurs de rêve tintent, éclatent, éclairent — la fille à lèvre d'orange, les genoux croisés dans le clair déluge qui sourd des prés, nudité qu'ombrent, traversent et habillent les arcs-en-ciel, la flore, la mer.

On a vu une façon de le faire en section 2 : insérer dans le document source entre chaque paragraphe une commande `\bigskip` suivie ou précédée d'une ligne blanche. Mais si le document comporte des dizaines et des dizaines de paragraphes, il est facile d'imaginer le temps qu'on perd à agir selon cette méthode ! On voudrait donc que notre nouveau type de délimitation de paragraphe soit celui produit par défaut par LaTeX pour tout le document, ce qui permettrait d'éviter d'avoir à entrer une commande entre chaque paragraphe.

Le moyen classique de le faire est de modifier les deux dimensions LaTeX concernées : `\parskip` qui définit l'espacement vertical entre paragraphe, et `\parindent`, qui en définit l'indentation de la première ligne. On entre alors les deux lignes suivantes en préambule :

```
\setlength{\parindent}{0cm}
\setlength{\parskip}{\baselineskip}
```

qui imposent respectivement une indentation nulle et un espace vertical supplémentaire entre deux paragraphes égal à la distance par défaut entre deux lignes.

Or, si la modification de \parindent ne pose pas de problème, celle de \parskip affecte malheureusement les éventuelles tables des matières, des figures et la liste des tableaux, ce qui n'était pas vraiment le but souhaité...

Les classes KOMA-Script fournissent une méthode beaucoup, beaucoup plus simple, et qui ne présente pas cette inconvénient. Il suffit d'appeler la classe KOMA-Script avec l'option `parskip` :

```
\documentclass[parskip]{scrartcl}
```

et voilà, vos paragraphes seront séparés par une ligne vide, sans indentation. KOMA-Script propose une autre option de même type, `halfparskip`, pour un espacement vertical entre paragraphes d'une moitié de ligne.

Pour avoir une idée des très nombreuses autres possibilités de KOMA-Script, je vous invite à lire sa documentation, `scrguien.pdf`, en anglais traduit de l'allemand, qui se trouve (si vous utilisez TEX Live 2008, installé par MacTEX-2008) dans le répertoire

```
/usr/local/texlive/2008/texmf-dist/doc/latex/koma-script
```

Si vous utilisez TEX Live 2007 (fournie par MacTEX-2007), remplacez simplement 2008 par 2007 dans la ligne ci-dessus.

Vous pouvez atteindre ce dossier en copiant la ligne ci-dessus dans la fenêtre ouverte via le menu Aller du Finder, option Aller au dossier...

6.3 La correction orthographique

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, deux applications sont disponibles : *Excalibur* et *CocoAspell*.

Excalibur est une application autonome déjà installée sur votre ordinateur (dans le dossier `/Applications/TeX`) par MacTEX. Je n'ai pas d'expérience avec ce logiciel, mais je compte faire un prochain test dessus, même si personnellement je ne suis pas fanatique des correcteurs orthographiques en général. Son utilisation semble assez simple, et il paraît performant. En particulier, outre les principales commandes LATEX, il est supposé reconnaître les ligatures. Il ne se limite d'ailleurs pas à corriger des document LATEX. Si vous voulez télécharger la version la plus récente et le(s) dictionnaire(s) approprié(s), cela se trouve ici : http://excalibur.sourceforge.net/downloads.html. Son manuel se trouve dans le dossier de l'application une fois téléchargée.

CocoAspell, non installé par MacTEX, diffère d'Excalibur dans sa conception, puisqu'il s'agit en fait d'une Préférence Système. Il a déjà été testé en profondeur sur Cuk (http://www.cuk.ch/articles/2053) par Fabien lui-même. Sa présentation a changé depuis, mais pas ses principes. Téléchargez donc sa version actuelle ici : http://people.ict.usc.edu/~leuski/cocoaspell, et suivez les conseils de Fabien pour l'utiliser.

6.4 La correction grammaticale

Malheureusement, à ma connaissance, la correction grammaticale n'est *a priori* pas disponible pour LaTeX. 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.

Tout d'abord, si l'essentiel du texte est « brut », sans commandes, il peut être copié pour être introduit dans un vérificateur. Le problème se posera lorsqu'il y aura beaucoup de commandes LaTeX disséminées de-ci, de-là.

Une astuce peut alors rendre service : que ce soit TeXShop lui-même, Aperçu ou Adobe Reader, ces applications vous offrent dans leurs versions récentes la possibilité de copier le texte contenu dans le document PDF qu'ils lisent. Vous pouvez donc avec une de ces applications ouvrir le PDF de votre document créé par LaTeX, tout sélectionner et l'envoyer au correcteur. Cela fonctionne même avec les Services !

Attention, si vous voulez que les caractères accentués survivent à ce copier-coller, votre programme LaTeX doit absolument contenir en préambule la ligne

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

que je vous avais conseillé d'utiliser systématiquement dans la section 3.3.

Un intervenant des forums de Cuk, Thierry, m'a fait remarquer qu'avec cette démarche apparaissait un autre problème : les mots contenant des ligatures et des césures n'étaient pas reconnus par le correcteur, Antidote en l'occurrence. Un contournement est alors de passer tout le document en style « machine à écrire » grâce à la commande-bascule `\ttfamily` (à placer juste après le `\begin{document}`) avant de l'envoyer au correcteur. En effet, le passage aux polices de style « machine à écrire » désactive les césures comme les ligatures. Une fois le texte corrigé, on supprime ou désactive cette commande (grâce à un signe `%` placé devant), et on recompile pour retrouver le « vrai » document.

Il est clair que ces astuces sont loin de fournir une solution idéale pour la correction grammaticale, mais c'est à ma connaissance la seule façon de faire à l'heure actuelle.

6.5 Les images EPS

Je vous l'ai dit, pour placer des images, les formats acceptés sont JPG, PNG et PDF. En fait, ceci est vrai car nous utilisons le programme pdfLaTeX, lequel se sert du moteur pdfTeX qui crée directement un document PDF.

Mais je vous ai également dit dans la première partie, que le programme LaTeX, qui lui se sert du moteur TeX, 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 Adobe Illustrator sauvait ses fichiers jusqu'à sa version 8.

Pour insérer des images EPS dans son document et quand même obtenir un PDF à l'arrivée, vous avez le choix entre deux méthodes.

1. Rester avec pdfTeX et pdfLaTeX et utiliser le paquet `epstopdf`. L'utilisation de ce paquet permet la conversion « à la volée » des images EPS en PDF lors de la

compilation, grâce à un utilitaire Unix appelé justement `epstopdf`. Cet utilitaire est fourni par l'application Unix *Ghostscript* (rappelez-vous : Ghostscript avait été installé par MacTEX en même temps que TEX, LATEX et pdfLATEX).

Cette méthode marche pour la plupart des fichiers EPS. Elle est néanmoins inefficace lorsque sont utilisés certains paquets de commandes LATEX destinés à produire des dessins scientifiques en PostScript. Le plus célèbre de ces paquets est `pstricks`. Si vous utilisez un de ces paquets, passez à la méthode suivante !

2. Compiler votre document avec TEX et LATEX puis convertir le fichier PostScript ainsi obtenu en PDF. Cela se fait là aussi grâce à Ghostscript, en utilisant cette fois-ci un autre de ses utilitaires, `ps2pdf`. Il y a plusieurs façons d'enclencher ce type de compilation.
   – Vous pouvez 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. La compilation de votre fichier courant se fera alors avec (La)TEX, puis le résultat en DVI sera automatiquement converti en PostScript, puis en PDF par Ghostscript.
   Si vous voulez que cette option soit définitive (pour tous vos fichiers à venir), vous pouvez aller dans les Préférences de TEXShop, onglet Composition, et là cocher l'option TeX + Ghostscript dans la section Script par défaut.
   – Pour un fichier donné, vous pouvez également forcer TEXShop à le compiler avec LATEX, par la voie TeX et Ghostscript, en insérant au début de ce fichier (avant le `\documentclass` et dans tous les cas dans les 20 premières lignes du document), la ligne suivante :

```
%!TEX TS-program = latex
```

   On peut s'aider du menu Macros, option Program, pour insérer la première partie de cette ligne.
   Une fois cette ligne insérée, TEX et Ghostscript seront automatiquement utilisés à chaque fois que vous cliquerez sur le bouton Composer ou son raccourci commande-t, et ce quel que soit le réglage des Préférences ou l'option choisie dans le menu Composer.
   Bien entendu, on peut également se servir de ce type de commentaire pour forcer la compilation d'un fichier avec pdfLATEX, au contraire. Remplacez simplement `latex` par `pdflatex` à la fin de la ligne !

L'option TeX+Ghostcript. porte assez mal son nom, car Ghostscript n'est pas à proprement parler nécessaire pour la conversion PS → PDF. En effet, depuis Panther, TEXShop peut utiliser le propre convertisseur PS → PDF de Mac OS X (`pstopdf`, celui-là même utilisé par Aperçu pour afficher les PS), à la place de celui fourni par GhostScript (`ps2pdf`).

Si vous voulez utiliser le convertisseur de Mac OS X, il vous faut aller avertir TEXShop que vous ne voulez pas utiliser Ghostscript, qui est son choix par défaut. Dans les Préférences de TEXShop, dans l'onglet Divers, sous Distiller vous devez cocher Apple Distiller.

Cependant, le convertisseur de Ghostscript a l'immense avantage sur son concurrent

de conserver les hyperliens éventuellement contenus dans le fichier PostScript. On verra plus loin comment procéder pour insérer ces hyperliens, aussi bien avec LaTeX qu'avec pdfLaTeX.

6.6 XeLaTeX

Un des problèmes majeurs de LaTeX comme de pdfLaTeX est qu'il n'ont pas directement accès aux polices incorporées à votre système d'exploitation. En effet, ils doivent utiliser leurs propres polices, installées dans la distribution TeX même, séparément des polices du système.

Mais depuis 2005, existe une nouvelle extension de TeX, qui pallie cette lacune : *XeTeX* (http://scripts.sil.org/xetex), créé par Jonathan Kew. Et très vite, XeLaTeX a été créé, pour permettre de disposer de la grande majorité des macros de LaTeX, mais cette fois-ci basées sur ce nouveau « moteur ».

XeTeX a tout d'abord la particularité d'imposer que tous les fichiers qu'il compile soient rédigés selon l'encodage de texte Unicode (UTF-8 ou UTF-16, au choix). Le plus souvent, on utilise l'UTF-8, très certainement le futur standard dans le domaine de l'encodage.

S'agissant de la compilation, XeTeX produit tout d'abord un fichier DVI « amélioré », d'extension `.xdv`, lequel est converti aussitôt en PDF.

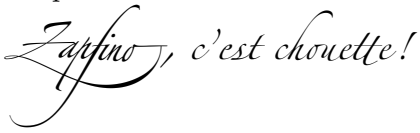
Mais la caractéristique majeure de XeTeX est qu'il permet d'utiliser, outre les polices embarquées dans votre distribution TeX, les polices du système d'exploitation de votre machine, qu'elles soient PostScript, TrueType ou OpenType. Et ce sur Mac OS X (sur lequel XeTeX a été développé à l'origine), mais maintenant aussi sur Linux et Windows.

Il faut savoir qu'à l'origine TeX n'exploitait que les polices METAFONT (langage de programmation développé par Knuth lui-même et dédié à la création de polices bitmap). Puis on a pu travailler sur des polices PostScript. PdfTeX permit d'aller plus loin en permettant, outre l'exploitation de polices METAFONT et PostScript, celle de polices TrueType et OpenType. Mais dans les deux cas, il devait s'agir de polices installées indépendamment de celles du système.

XeTeX, et XeLaTeX avec lui, permet de s'affranchir de cette limitation. Par exemple, imaginons que vous souhaitez exploiter la belle police cursive appelée *Zapfino*, qui est normalement déjà installée par défaut dans le répertoire Fonts du dossier Bibliothèque de la racine de votre ordinateur, sous la forme du fichier Zapfino.dfont. Voici un code XeLaTeX (minimaliste) permettant d'utiliser cette police :

```
%!TEX encoding = UTF-8 Unicode
%!TEX TS-program = xelatex
\documentclass{article}
\usepackage{xltxtra}
\usepackage[frenchb]{babel}
\setmainfont{Zapfino}
\begin{document}
Zapfino, c'est chouette !
\end{document}
```

Et voici le résultat de la compilation :



Pas mal, hein ?

Vous aurez noté dans ce programme les « commentaires » en première et deuxième ligne imposant respectivement à TEXShop l'encodage du texte en UTF-8 Unicode (obligatoire avec XeLaTeX comme avec XeTeX) et la compilation par XeLaTeX lorsqu'on utilise le bouton Typeset ou son raccourci commande-t. On peut également choisir l'option XeLaTeX dans le menu déroulant des programmes de compilation, dans la barre d'outils de la fenêtre de TEXShop, avant de lancer ladite compilation.

La deuxième particularité du programme est sa quatrième ligne. Le paquet `xltxtra` *(XeLaTeX extras)* permet de charger plusieurs macros très importantes pour XeLaTeX, sur lesquelles je ne m'étendrai pas ici, faute de place.

Mais surtout, il charge automatiquement un autre paquet, appelé `fontspec` (voyez son excellente documentation, à cette adresse : http://www.ctan.org/get/macros/xetex/latex/fontspec/fontspec.pdf. Ce dernier fournit lui les macros XeLaTeX permettant de charger facilement les polices du système. La plus importante étant la macro `\setmainfont`, utilisée en sixième ligne de notre programme, qui imposera à XeLaTeX d'utiliser comme police principale celle dont le nom est donné en argument. Ce devra être en fait un des noms (sans son extension) des fichiers se trouvant dans un des différents dossiers fonts des dossiers Bibliothèque de votre système.

`Fontspec` a également la très bonne idée d'imposer l'usage par défaut des polices *Latin Modern*, si aucune police du système n'est utilisée. Ainsi, avec l'emploi du paquet `frenchb` de `babel`, sont résolus d'entrée les problèmes de césures.

Bref, XeLaTeX permet un formidable bond en avant pour LaTeX dans le domaine des polices.

Mais XeLaTeX a ses limites : notamment, si on souhaite utiliser des polices adaptées aux mathématiques, il faut encore actuellement se rabattre sur les polices de votre distribution TEX. En effet votre système d'exploitation ne contient hélas pas de polices mathématiques dignes de ce nom. De plus, pdfTeX dispose de capacités typographiques très sophistiquées, rendues accessibles *via* un paquet appelé `microtype`, et ces facultés font (pour le moment) défaut à XeTeX.

Si je deviens un jour un peu plus versé dans le domaine des polices, je ferai un article sur XeLaTeX. Sans rien promettre, hein ?

6.7 La synchronisation source-PDF : SyncTEX

Il existe un problème auquel tous les utilisateurs de LaTeX sont fatalement confrontés à un moment ou à un autre : on bâtit un document LaTeX, on le compile, tout semble bien se passer, et puis, paf ! on s'aperçoit à la lecture de telle page du PDF produit qu'il y a une erreur, une faute de frappe ou un truc du même genre.

On doit alors retourner à la partie du code LaTeX concernée, dans le document source,

et rectifier. Seulement voilà, si le document est long, on risque de devoir faire défiler des lignes et des lignes de code avant de trouver le bon endroit.

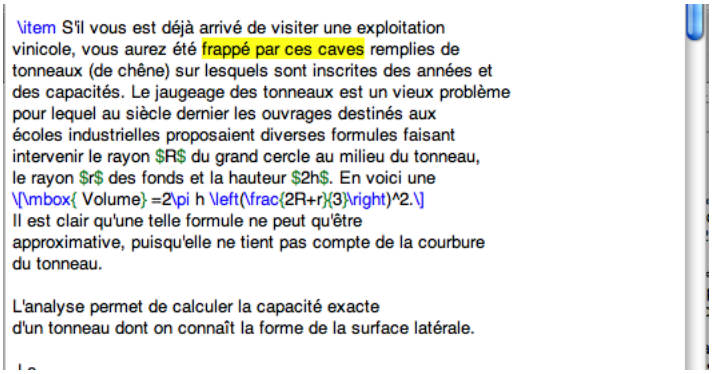
Mais TEXShop propose heureusement des solutions, qui diffèrent cependant selon ses versions et la distribution TEX utilisée.

Depuis la sortie de MacTEX-2008, la version de TEXShop livrée avec (la 2.18) intègre un mécanisme de synchronisation source-PDF appelé *SyncTEX*.

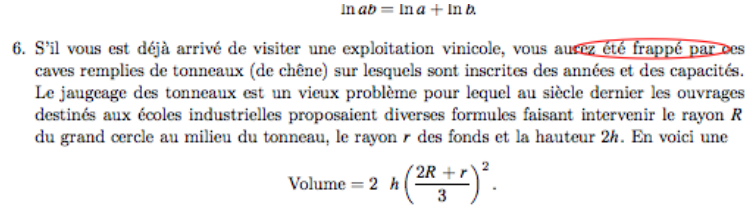
Attention, si vous utilisiez auparavant une version de TEXShop antérieure à la 2.18, il sera sans doute nécessaire de supprimer le fichier de préférences `TeXShop.plist` situé dans le dossier Préférences de votre dossier Bibliothèque personnel, puis de redémarrer TeXShop 2.18. Ce fichier `TeXShop.plist` sera alors recréé, avec cette fois-ci tout ce qu'il faut pour faire tourner SyncTEX.

Comment ça marche, ce SyncTEX ? Très simplement ! Il faut bien sûr avoir compilé votre document, et que la compilation se soit passée sans erreur (selon LATEX du moins). À cette occasion est créé un fichier du nom de votre document mais d'extension `.synctex.gz`. Ce fichier est la compression du fichier du même nom et d'extension `.synctex`, qui a recueilli toutes les données nécessaires à la synchronisation par SyncTEX, et qui avait été entre-temps automatiquement lu par TEXShop lors de la compilation.

Maintenant, il suffit de « commande-cliquer » à un endroit du document PDF que vous avez produit avec LATEX, et TEXShop vous amène (à peu de chose près) sur la partie concernée du document source LATEX. Les alentours immédiats de cette partie seront surlignés de jaune, comme ici :



Réciproquement, si vous « commande-cliquez » cette fois-ci à un endroit de votre document LATEX, TEXShop va repérer dans la fenêtre de lecture du document PDF, en l'entourant de rouge, la partie du texte (approximativement) correspondante :



Rassurez-vous, ces effets visuels ne sont que superficiels : ils disparaissent à la compilation suivante et n'affectent en aucune manière les documents à l'impression.

D'après ma propre expérience, le procédé SyncTEX marche parfaitement en toutes circonstances, sauf semble-t-il pour la classe de présentation `beamer`, où cependant il ne se comporte globalement pas plus mal que les précédentes options de synchronisation.

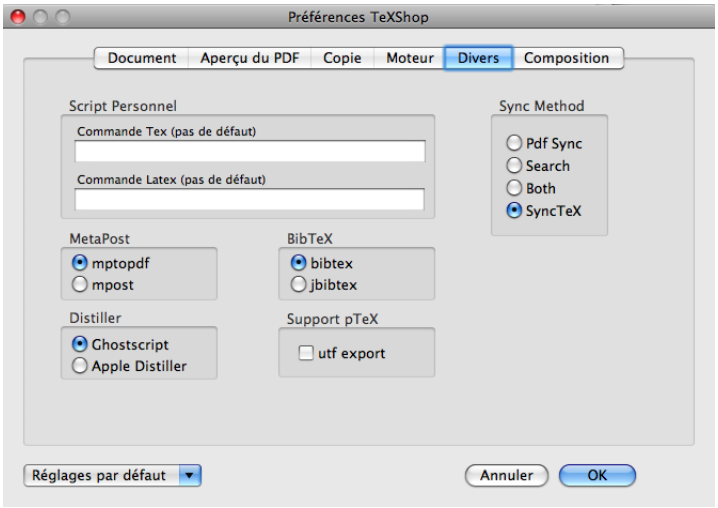
Du temps de MacTEX-2007 et autres distributions antérieures, la synchronisation source-PDF existait déjà, mais sous une autre forme. Elle utilisait un procédé appelé bizarrement Search qui lui-même faisait appel à des propriétés du « PDF Kit » intégré à Mac OS X. Mais Search coinçait dans un cas très gênant (pour moi du moins) : les formules mathématiques. Si on « commande-cliquait » sur un paragraphe de formules, que ce fut dans le document source ou dans le document PDF, la synchronisation échouait lamentablement.

Pour remédier à cela, il y avait heureusement le paquet `pdfsync`. Ce paquet de commandes, développé à l'origine par le créateur d'iTEXMac, Jérôme Laurens, permettait une synchronisation plus efficace du source vers le PDF, et *vice versa*, dans toutes les circonstances (y compris donc les formules mathématiques). Il fonctionnait suivant le même principe, le « commande-clic ».

Pour utiliser `pdfsync` dans son document LATEX, on devait le charger comme tout paquet de commande, avec

```
\usepackage{pdfsync}
```

Mais on devait également aller dans les préférences de TEXShop, onglet Divers, section Sync Method, et là cocher soit la case Pdfsync, soit la case Both qui activait à la fois Search et pdfsync. C'était alors le meilleur choix.



Vérifiez bien que vous avez la bonne option cochée. Choisir SyncTeX si vous utilisez MacTeX-2008 et TeXShop 2.18 ou plus récent, Both (pdfsync + Search) sinon. Dans les versions de TeXShop antérieures à 2.18, l'option SyncTeX n'apparaît pas.

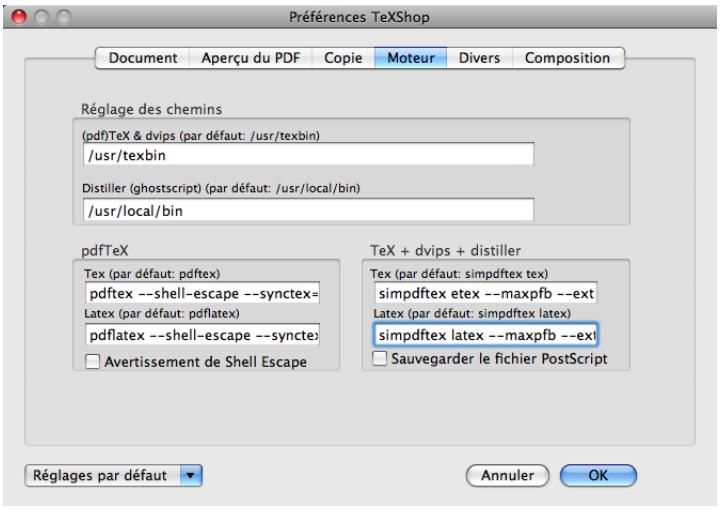
Seulement voilà, depuis MacTEX-2008 le paquet `pdfsync` est carrément incorporé à pdfTEX, le moteur de pdfLATEX, et c'est justement cette incorporation qui porte le nom

de SyncTEX. TEXShop intègre cette nouvelle fonctionnalité à partir de sa version 2.18. Ce qui fait que nous avons droit à une synchronisation quasi parfaite entre le source et le PDF, sans avoir à charger le moindre paquet de commandes. En particulier, plus aucun problème avec les formules mathématiques.

Cependant, si jamais il arrivait que SyncTEX cale, les options précédentes de synchronisation restent disponibles dans la fenêtre Sync Method des préférences de TEXShop, ainsi qu'on vient de le voir.

Les utilisateurs de LATEX par la voie de compilation TeX + Ghostscript (celle qui permet d'incorporer les figures en PostScript) et les adeptes de XeLATEX peuvent également bénéficier de SyncTEX. Mais il y a alors un brin de bidouille à faire.

Si vous utilisez TeX + Ghostscript, retournez dans les préférences de TEXShop, cette fois dans l'onglet Moteur, section TeX + dvips + distiller :

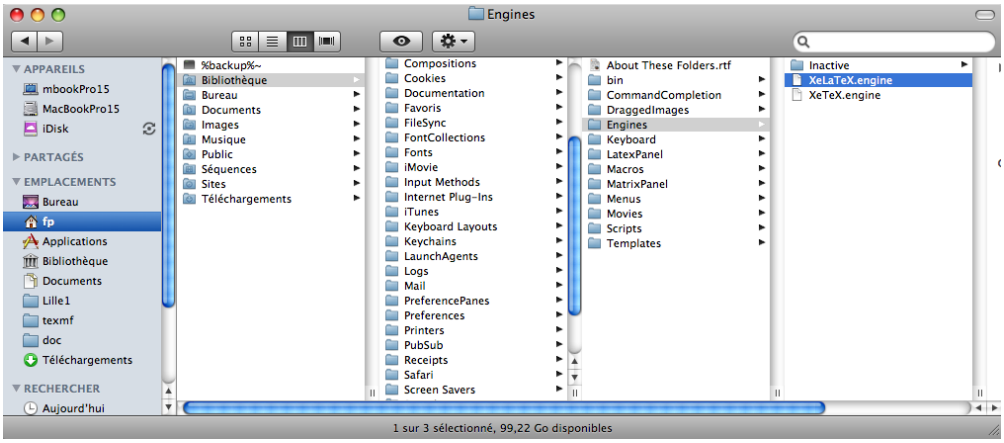


et remplacez la ligne qui se trouve sous l'intitulé « LaTeX (par défaut : simpdftex latex) » par cette ligne-là :

```
simpdftex latex --maxpfb --extratexopts --synctex=1
```

Cliquez alors sur OK, et SyncTEX devrait alors fonctionner aussi sans problème pour cette voie de compilation.

Si vous utilisez XeLATEX, ouvrez le dossier TeXShop de votre Bibliothèque maison, et là ouvrez le sous-dossier Engines. Vous y trouverez normalement le fichier `XeLaTeX.engine` :



Ouvrez ce fichier, et remplacez la ligne

```
xelatex "$1"
```

par la ligne

```
xelatex --synctex=1 "$1"
```

À partir du redémarrage suivant de TEXShop, la synchronisation fonctionnera alors également pour XeLATEX.

6.8 Le paquet `hyperref`

Le PDF est un format 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 de TEXShop et d'Aperçu, ce sont les signets d'Adobe Reader.

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, XeLATEX et même avec LATEX lorsque le PDF est produit par Ghostscript (mais pas par le distiller d'Apple) ! Il suffit d'ajouter le paquet de commandes `hyperref`

- avec l'option `pdftex` au cas où vous utilisez pdfLATEX :

```
\usepackage[pdftex]{hyperref}
```

- avec l'option `xetex` si vous vous servez de XeLATEX :

```
\usepackage[xetex]{hyperref}
```

- ou avec l'option `dvips` si c'est l'option TeX + Ghostscript qui est lancée :

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

Dvips est le nom de l'utilitaire Unix qui transforme le DVI produit par (La)TEX en PostScript, avant que ce dernier soit converti en PDF par Ghostscript.

Il est important que hyperref soit toujours le dernier paquet de commandes chargé afin qu'il puisse bien fonctionner.

Sans rien faire d'autre, tous les éléments de la table des matières deviendront automatiquement des hyperliens qui mèneront au chapitre ou à la section choisie, les références conduiront 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 soi-même des 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.

Ces hyperliens sont actifs dans la fenêtre de prévisualisation de TEXShop, dans ses versions récentes. Ils sont également présents dans Aperçu. Mais dans les deux cas, ils 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. Il suffit pour cela de configurer le paquet hyperref pour qu'il utilise des couleurs. Ajoutez ceci avant le \begin{document} :

```
\hypersetup{colorlink=true}
```

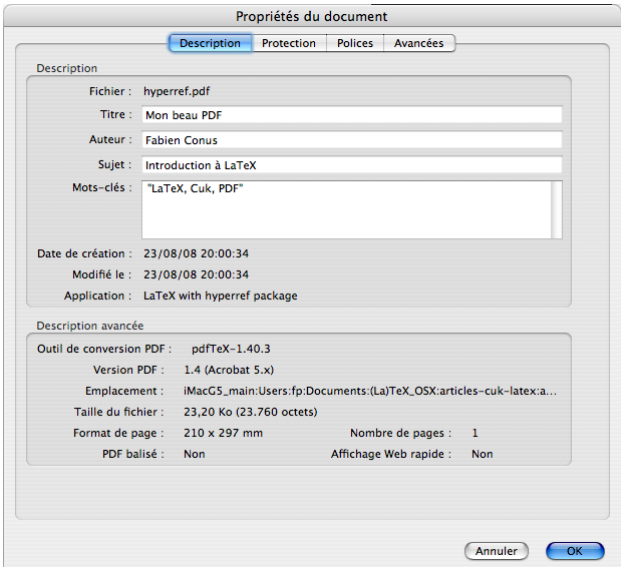
Il est possible de personnaliser les couleurs, mais je n'entrerai pas dans les détails ici.

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}
}
```

Attention à ne pas insérer d'espaces-clavier autour des signes =, hyperref risquerait de ne pas apprécier.

Si avec Adobe Reader vous lisez alors les informations du document (commande-d) vous obtiendrez ceci dans l'onglet Description :



Le paquet de commandes `hyperref` est très complet et très riche, vous en trouverez un manuel exhaustif ici : http://www.tug.org/applications/hyperref/ftp/doc/manual.pdf.

6.9 Le futur de LaTeX

LaTeX est en perpétuelle évolution, comme on vient de le dire. Et pourtant le code de base de TeX, son « moteur », est figé par son créateur, Donald Knuth, depuis 1989, et les macros de LaTeX lui-même, hormis les divers paquets de commandes, datent toutes de 1994, quand LaTeX version « 2e » est sorti. La prochaine version, LaTeX 3, se fait attendre depuis déjà pas mal de temps, au point que personne ne spécule encore sur une date de sortie.

Mais entre-temps sont sortis et continuent à sortir une multitude de paquets de commandes qui étendent LaTeX, dont on n'a eu qu'un tout petit aperçu dans ce cours, et ces extensions sont en perpétuelle évolution et rénovation.

Et surtout, si le créateur de TeX ne travaille plus lui-même sur son bébé, qu'il considère virtuellement libre de bugs, d'autres ont travaillé dessus et proposé des évolutions, ce que la licence libre de TeX permet. Il suffit de proposer ses évolutions sous d'autres noms.

Ainsi a été conçu pdfTeX vers 2000, une version de TeX produisant par défaut du PDF, tandis que TeX produit du DVI. Mais pdfTeX peut également produire du DVI, si on le lui demande « gentiment ». Et il s'est avéré tellement fiable et novateur par rapport au TeX originel que maintenant toutes les distributions TeX sont en fait basées sur pdfTeX.

Quand vous utilisez « LaTeX » pour produire du DVI, c'est en fait pdfTeX qui tourne en sous-main, en « mode » de production DVI, et non TeX à proprement parler, bien que par commodité on désigne encore par « TeX » ce qui est en fait pdfTeX en mode DVI. Et quand vous utilisez pdfLaTeX, il s'agit en fait *grosso modo* du même set de macros que LaTeX, mais qu'on fait tourner sur pdfTeX dans son mode de production PDF par défaut.

Le présent de LaTeX est donc pdfLaTeX. Mais quel est son futur ?

Hé bien, il est déjà bien engagé. Tout d'abord, on l'a vu, on dispose déjà de XeLaTeX, variante de LaTeX basée elle sur un moteur bien distinct, XeTeX, qui date de 2005. Comme je l'ai déjà mentionné, ce dernier fonctionne sous Unicode et permet d'exploiter aussi bien les polices de votre distribution TeX que celles du système d'exploitation.

PdfTeX, cependant, n'a pas dit son dernier mot : une équipe de développeurs très actifs travaille sur lui en permanence, si bien que la dernière évolution de pdfTeX est déjà (presque) prête. Elle est sortie en version « beta » dans TeX Live 2008, et elle s'appelle *LuaTeX*. Il s'agit d'une nouvelle version de pdfTeX donc, mais basé sur un langage de script appelé *Lua*.

Noé pourrait sans doute bien mieux que moi décrire les possibilité qu'ouvre l'incorporation de ce langage, mais il paraît qu'elles sont très appréciables. Comme XeTeX, LuaTeX fonctionne avec l'encodage UTF-8 Unicode et permet d'utiliser les polices du système. Mais contrairement à XeTeX, il peut produire directement du PDF et permet d'utiliser les propriétés améliorant encore la typographie du document PDF, propriétés déjà fournies par pdfTeX et dont XeTeX ne dispose à ce jour.

On peut déjà donc prévoir que l'implémentation de LaTeX basée sur LuaTeX, qui est appelée *LuaLaTeX* et qui n'est accessible pour le moment que par le Terminal, permettra d'exploiter ces belles propriétés. Ce qui n'est pas le cas actuellement, hélas. Pour TeX Live 2009 peut-être ? Ladite implémentation sera alors le successeur naturel de notre pdfLaTeX actuel.

6.10 ConTEXt

Il existe depuis quelques années un autre ensemble de macros concurrent de (pdf)LaTeX. Il était comme lui basé sur pdfTeX, mais est déjà pleinement basé sur LuaTeX dans TeX Live 2008, contrairement à lui : il s'agit de *ConTEXt*[10].

ConTEXt diffère de LaTeX sur plusieurs points essentiels : en premier lieu, sa conception est plus « monolithique », en ce sens qu'il incorpore déjà à l'origine l'équivalent des principaux paquets de commandes de LaTeX. De plus son développement est centralisé : c'est la compagnie néerlandaise *Pragma-ADE*[11] qui le gère, bien qu'elle ait rendu sa création libre et gratuite (noter que cette compagnie s'implique aussi activement dans LuaTeX). Cela diminuerait les risques de dispersion et d'incompatibilité entre paquets dont souffrirait LaTeX (je n'ai rien remarqué, moi !). Et ConTEXt a été conçu dès l'origine pour permettre des mises en page plus aisément personnalisables qu'avec LaTeX, ce qui le rendrait plus adapté pour la PAO en particulier.

Je ne peux guère en dire plus. Sachez cependant qu'un des convertis à ConTEXt s'appelle Noé Cuneo. Il pourrait peut-être nous faire un article dessus un de ces mois prochains !

10. http://wiki.contextgarden.net/Main_Page
11. http://pragma-ade.com

Conclusion

Ouf ! Nous voilà arrivés au terme de cette petite introduction à LaTeX sous Mac OS 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 et de références bibliographiques qui vous permettront d'approfondir les différents sujets et de devenir un pro de LaTeX, je vous donnerai quelques liens et titres plus loin.

J'aimerais profiter de cette conclusion pour (ré)engager un peu le débat : 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 page à l'allure très professionnelle.

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

Certains diront : « Mais moi je ne fais pas de gros documents, juste de la correspondance et quelques broutilles ». 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 €) [12]. 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. Évidemment 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. Hé 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 vos en-têtes et pieds de pages (par exemple avec le paquet de commandes `fancyhdr`). 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 déclaration de classe et le même préambule 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 LaTeX 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 command-alt-f m'insère automatiquement tout ce qu'il me faut pour inclure une figure, command-alt-e me permet d'insérer une équation, etc.

12. Ce sont les prix de 2004. Selon le site d'Adobe, FrameMaker en version standard sur Windows coûte aujourd'hui 2285 CHF en Suisse et 1379 € en France (prix hors taxe). Je ne suis pas parvenu sur Internet à mettre la main sur le prix de Word 2008 seul, ne tombant que sur les prix de la suite Office complète. Mais ça n'a pas dû tellement changer. (Note de Franck)

Si vous voulez créer des documents que vous souhaitez diffuser, ou si vous faites de l'écriture collaborative, LaTeX 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 !

LaTeX crée du PDF, format lisible sur toutes les plates-formes connues (sauf les consoles de jeux, et encore !). De plus le code LaTeX est un bête format texte en ASCII, il est donc également lisible sur toutes les plate-formes, et comme LaTeX est un logiciel libre il tourne également sur toutes les plate-formes. 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 il n'y a pas si longtemps avec la mésaventure de FrameMaker sous Mac OS 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. Évidemment, 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 un logiciel phare des débuts de la micro-informatique ? Un code LaTeX est en ASCII, le format le plus simpliste. Sa pérennité est donc assurée. De plus, le code source de LaTeX étant disponible pour 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 LaTeX est fait pour vous, c'est d'essayer. C'était le but de cette présentation simpliste : vous donner 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.

Quelques liens et références utiles

Ouvrages de référence

Sur le web, quatre articles, chacun complétant idéalement ce cours. Faites votre choix :

- *Présentation rapide de LaTeX 2ε*, par Florent Rougon, 2006, http://people.via.ecp.fr/~flo/2000/presentation_LaTeX. Comme son nom l'indique, une présentation de LaTeX, ici en HTML. « Rapide » mais détaillée.
- *The not so short introduction to LaTeX 2ε*, de Tobias Oetiker et pas mal d'autres contributeurs, http://tobi.oetiker.ch/lshort/lshort.pdf. La dernière version date de septembre 2008, c'est dire si elle est à jour. 141 pages, en anglais. C'est la référence web sur LaTeX la plus conseillée et consultée. À la fois claire, concise et précise. Une traduction française est disponible ici, mais elle date de 2001 et il semble ne plus y avoir de version française récente disponible sur la toile actuellement.
- *Stage LaTeX, niveau débutant*, donné aux enseignants du lycée Camille Guérin de Poitiers par Jean-Côme Charpentier au cours de l'année 2001-2002, et mis en ligne par la suite, http://melusine.eu.org/syracuse/texpng/jcc/camille.pdf. 62 pages. Déjà ancien, donc, notamment pour tout ce qui a trait aux distributions de LaTeX et ses éditeurs ou interfaces. Mais ce qui concerne le langage LaTeX lui-même reste d'actualité pour l'essentiel, et est très bien expliqué. Une curieuse lacune : ne mentionne pas le très important paquet `inputenc`.
- *Tout ce que vous avez voulu savoir sur LaTeX sans jamais oser le demander*, de Vincent Lozano, http://cours.enise.fr/info/latex. 237 pages. Dernière version : 17 octobre 2008, à jour donc. L'auteur est visiblement un joyeux drille et lance au moins une vanne par page ! S'adresse au débutant et souhaite l'accompagner jusqu'au niveau confirmé. Bonne nouvelle, ce guide est maintenant disponible sous forme de livre, à prix modique, dans la collection Framabook (http://www.framabook.org/latex.html).

Aussi dans les librairies, du niveau débutant jusqu'à la « geekitude » complète :

- *LaTeX pour l'impatient*, ouvrage collectif supervisé par Cécile Chevalier, collection MiniMax, éditions H&K, 2007, 160 pages. Honnêtement, c'est LE livre que je conseillerais de lire juste après ce cours. Toutes les plate-formes sont prises en compte. Remarquablement complet pour sa taille, truffé d'exemples, avec exercices d'application. Un *must*.
- *LaTeX, a document preparation system. User's guide and reference manual*, par Leslie Lamport, le créateur de LaTeX. En anglais, éditions Addison-Wesley, 1994, 272 pages. Si l'anglais ne vous fait pas peur, cela ne fait jamais du mal de lire les fondamentaux, d'autant que Lamport a une écriture particulièrement claire. Ne traite que de LaTeX lui-même, et pas de ses paquets d'extension, sauf quelques exceptions. Du fait que le langage LaTeX en tant que tel n'a guère changé depuis sa parution, contrairement à ses extensions, cet ouvrage reste d'actualité.
- *LaTeX, apprentissage, guide et référence*, par Bernard Desgraupes, éditions Vuibert, 2003, 762 pages. Ma référence personnelle la plus utilisée. Réussit à intéresser aussi

bien le débutant à son début (« apprentissage ») que l'utilisateur confirmé, en abordant un maximum de détails et de paquets ensuite (« guide et référence »). Commence malheureusement à prendre de l'âge.

- *Le LaTeX Companion, deuxième édition*, par Frank Mittelbach, Michel Gossens, et plusieurs autres auteurs. Traduit de l'anglais, éditions Pearsons Education, 2005 (éditon originale : 2004). 1008 pages (glourps), pour *tout* savoir sur LaTeX et ses principaux paquets. Une véritable encyclopédie. À ne pas lire comme un roman !
- *The LaTeX Graphics Companion, Second Edition*, par Frank Mittelbach, Michel Gossens, et al. En anglais, éditions Pearsons Education, 2007, 926 pages. Tout ou presque sur l'illustration graphique pour et par LaTeX. Pour le « mordu », c'est le complément obligatoire du livre précédent.
- *Le TeXbook*, de Donald Knuth, le créateur de TeX. Traduit de l'anglais, éditions Vuibert, 2003 (1986 pour l'édition anglaise), 555 pages. La bible des TeXniciens enragés. Si vous voulez tout savoir sur les entrailles de LaTeX, c'est cet ouvrage qu'il faut consulter. Il ne traite en effet pas de LaTeX lui-même, mais de TeX, sur lequel LaTeX est fondé. La lecture peut se faire à plusieurs niveaux, et l'écriture de Knuth est agréable, au point que je recommande plutôt la version originale, aux éditions Addison-Wesley, si vous maîtrisez l'anglais.

Quelques sites Internet utiles ou carrément indispensables

- *The TeX FAQ*, les questions les plus fréquemment posées sur TeX et surtout LaTeX. http://www.tex.ac.uk/cgi-bin/texfaq2html. À jour, en anglais.
- *La FAQ francophone sur LaTeX*, http://www.grappa.univ-lille3.fr/FAQ-LaTeX, est encore très utile et très consultée, mais n'est malheureusement plus mise à jour depuis 2001... Une autre FAQ plus récente se trouve à cette adresse : http://latex.developpez.com/faq, mais je ne sais pas trop ce qu'elle vaut.
- *Le site du* TeX Users Group *anglophone* (TUG), http://www.tug.org, fourmille de liens, renseignements et tuyaux, entre autres la FAQ citée plus haut. Son équivalent francophone s'appelle *GUTenberg* (Groupe des UTilisateurs francophones de TeX, http://www.gutenberg.eu.org). Ce dernier propose une liste de diffusion (*mailing list* en anglais), francophone bien sûr, sur TeX en général et LaTeX en particulier, fréquentés par des débutants comme des spécialistes qui pourront certainement résoudre vos problèmes les plus tarabiscotés. Voir la fin de cette section pour quelques conseils à ce sujet.
- *The Comprehensive TeX Archive Network* (CTAN), http://www.ctan.org. Si vous avez besoin d'un paquet LaTeX particulier qui ne se trouve pas déjà dans TeX Live (ce qui n'est pas fréquent), c'est ici qu'il faut le chercher !
- *The TeX on Mac OS X site*, http://mactex-wiki.tug.org/wiki, s'efforce de répertorier tout ce qui concerne TeX/LaTeX sur Mac OS X, comme son nom l'indique. Il est actuellement en pleine reconstruction. Il propose lui aussi sa propre liste de diffusion (http://mactex-wiki.tug.org/wiki/index.php?title=TeX_on_Mac_OS_X_mailing_list), en anglais, à laquelle je suis moi-même abonné. Les créateurs de

TEXShop et XeTEX, notamment, fréquentent cette liste. On y répond en priorité aux problèmes de LATEX liés à Mac OS X, mais aussi volontiers aux questions LATEX proprement dites. Voir également les conseils de la fin de cette section.

- *LATEX en entreprise*, http://www.ozone.ch/enrico/texlatex.htm. Enrico RIBONI y fournit des liens et des *tips* très utiles. Mériterait une petite mise à jour.
- *LATEX à portée de clic*, http://xavier.perseguers.ch/tutoriels/latex.html. Deux ex-étudiants Lausannois, Arnaud ZUFFEREY et Xavier PERSEGUERS, nous invitent à faire un petit tour de LATEX, de son installation, de son utilisation et de ses principales possibilités. Très bien fait !
- *Le site des tuteurs de l'École Normale Supérieure de Paris* (ENS) a toute une section consacrée à LATEX : http://www.tuteurs.ens.fr/logiciels/latex. Elle fourmille de précieux conseils. En particulier, une page très claire consacrée à la programmation de macros LATEX : http://www.tuteurs.ens.fr/logiciels/latex/macros.html. Celles-ci, comme celles de TEXShop, peuvent vous rendre d'immenses services une fois que vous les maîtrisez. En passant, je remercie Guillaume, lecteur de cuk.ch, qui m'a signalé cette page.

Enfin, il faut signaler également deux *newsgroups* (groupes de discussion) sur USENET dédiés à TEX et LATEX :

- `comp.text.tex`, en anglais. Les archives de ce groupe sont disponibles grâce à Google : http://groups.google.com/group/comp.text.tex/topics?hl=en.
- `fr.comp.text.tex`, en français. Archives également disponibles sur Google : http://groups.google.com/group/fr.comp.text.tex/topics?hl=en.

Ces deux groupes de discussion, ainsi que les listes de diffusion citées plus haut, sont fréquentés par les spécialistes les plus pointus de LATEX et consorts. Il y aura toujours quelqu'un à avoir réponse à vos problèmes les plus compliqués. Assurez-vous quand même d'avoir épuisé toutes vos autres ressources avant de leur soumettre votre problème, ils apprécieront. Ce sont en effet des bénévoles qui prennent sur leur temps libre pour vous aider. En particulier, fouinez dans les FAQ et dans les archives de ces groupes pour voir si quelqu'un n'a pas déjà eu le même problème. Prenez également soin de leur fournir un exemple de code LATEX, le plus court possible, qui met en évidence votre problème. Dans le jargon du groupe francophone, on appelle cela un ECM, un Exemple Complet Minimal.

Index