LaTeX
réapprendre à utiliser un traitement de texte

Sébastien Crouzille, Archipel du Libre

1er septembre 2004

Archipel du Libre
Valoriser et développer le Logiciel Libre

Document créé avec LaTeX2ε, manipulation d'images sous The Gimp 2.0 [1].

Table des matières

1 Introduction

Bienvenue dans le monde merveilleux de LaTeX ! Ce monde où on ne voit rien de ce qu'on fait, où on consulte des kilomètres de documentation hétéroclite à chaque fois qu'on écrit un mot, où le texte est parsemé de symboles étranges qui ne se retrouvent même pas dans le document final !

Plus sérieusement, si nous abordons le sujet épineux de LaTeX dans le cadre d'une association « grand public »comme l'Archipel du Libre, c'est justement pour mettre en évidence une certaine façon de travailler sur un ordinateur. Nous mettrons un peu plus nos cellules grises à contribution certes, mais il s'agit de produire des documents clairs, lisibles, d'apparence professionnelle, bref de remplir nos objectifs, pas de faire du joli pour « voir comment ça fait ».

1.1 Ce qu'offre ce document

Nous trouverons donc dans ce document des séquences d'utilisation type de LaTeX, qui serviront un peu à comprendre sa syntaxe, beaucoup à découvrir les bases de fonctionnement de tout traitement de texte (WYSIWYG ou non). On verra par exemple ce qu'on doit attendre d'un traitement de texte, ce qu'il sait faire à la perfection et ce pour quoi il n'a pas du tout été conçu.

Surtout, ce document est la démonstration même de la puissance et de l'efficacité de LaTeX, puisqu'il a entièrement été composé avec ce traitement de texte, parfois sous Windows, parfois sous Linux. Il est donc possible de vous référer à ces pages pour avoir un bon aperçu de ce qu'on peut faire, même si vous n'avez pas LaTeX. Un simple lecteur de PDF est suffisant, et le fichier source est disponible [ici].

1.2 Ce que n'offre pas ce document

Il ne s'agit pas de fournir une documentation d'utilisation de LaTeX, il en existe de très bonnes sur le Réseau. De la même façon, nous n'expliquerons pas du tout comment installer LaTeX sous Linux, Windows, MacOS ou autre. Soit vous possédez une distribution GNU/Linux, et tous les packages nécessaires sont fournis, soit vous possédez Windows, et des CDROM comme TeX Live [3] vos fournissent une compilation de tout ce dont vous avez besoin (j'avoue ne pas m'être penché sur le cas des Mac, ça viendra je pense). De toute façon, ce document ne nécessite même pas d'expérimenter les exemples sous LaTeX, vous pouvez adapter ceux-ci à n'importe quel logiciel de traitement de texte.

Pour les utilisateurs de Windows, qui ont souvent moins l'habitude de manipuler des logiciels libres, je suggère d'aller faire un tour sur le site de Framasoft [4], qui fournit toutes les informations nécessaires sur LaTeX et les autres.

2 Définitions

2.1 La mise en forme de documents

Avant de se lancer, il faut impérativement centrer le débat, trop souvent faussé par des logiciels populaires. *Qu'est-ce qu'un traitement de texte* ? Une question aussi stupide doit certainement en faire sourire plus d'un, mais c'est peut-être le problème central de l'affaire.

En informatique, on rencontre une multitude de fichiers contenant du texte. Les plus simples contiennent seulement la suite de caractères composant le texte, sans aucune information concernant l'apparence de celui-ci. On appelle un tel format du *texte brut* ; sous Windows on en rencontre souvent avec un suffixe `.txt`. Ce format est le plus léger pouvant contenir du texte, et on l'utilise notamment pour la programmation (scripts, sources,...).

Tous les autres formats de fichiers texte contiennent des informations de *mise en forme*, ce qui permet, grâce au(x) logiciel(s) adapté(s), de visualiser le texte sous une forme plus élaborée : couleurs, polices, taille de texte, disposition en paragraphes, en tableaux, insertion de documents divers. On trouve des formats assez limités, comme le `.rtf` (*Rich Text Format*), ou d'autres bien plus élaborés, comme le format `.doc` de Microsoft Word, ou le format `.sxw` d'Open Office.

Il existe enfin des formats graphiques, mais qui peuvent afficher du texte. On citera le `.pdf` (*Portable Document Format*), le `.ps` (*PostScript*). Ces types de fichiers sont destinés essentiellement à la publication (impression), une fois le texte achevé et mis en forme.

L'objectif d'un logiciel de traitement de texte (dont LaTeX) est donc, à partir d'un texte brut (« au kilomètre ») de produire un document correctement mis en page, lisible et imprimable sur tout type de support informatique. Un tel document peut être une simple lettre, un *curriculum vitæ*, un article, une documentation (comme celle-ci), un roman, une thèse,... L'essentiel est que l'on puisse respecter les règles de typographie (lisibilité pour les personnes), et que l'on puisse ouvrir (correctement) le document final sur n'importe quelle plateforme, dans n'importe quelle imprimerie (lisibilité pour les machines).

2.2 WYSIWYG

Pour mener à bien ce travail, il existe deux principaux types de logiciel :
- les logiciels graphiques interactifs ;
- les logiciels de balisage.

Les logiciels de traitement de texte graphiques, comme Microsoft Word ou Open Office Writer, sont dits de type *WYSIWYG*, ce qui signifie What You See Is What You Get (« Ce que vous voyez est ce que vous obtenez »). En clair, le logiciel vous fournit une interface graphique qui tente de représenter au maximum les effets que vous donnez au texte. En général les opérations de mise en page se font à la souris, de manière intuitive, et le programme s'arrange pour comprendre tant bien que mal ce que vous voulez faire. La plupart d'entre nous, sinon la totalité, a déjà utilisé un programme de ce type, sans même y prêter attention, pour taper sa correspondance, des notes pour un ami, des cartes de vœux, etc.

Cette méthode semble la meilleure (sinon la seule) concernant la mise en forme de texte. En fait c'est la pire. Elle semble idéale car elle donne un semblant de résultat facilement et rapidement. Mais elle cache de graves inconvénients.

En premier lieu, elle engendre un amalgame entre deux choses distinctes : le texte lui-même et sa mise en forme à des fins de publication. Notamment il est évident qu'une personne qui écrit un texte, qui le pense, n'a pas forcément en tête de mise en page *a priori*. En outre il n'a pas nécessairement les connaissances pour mettre un texte en page, chose qu'il confiera éventuellement à une autre personne. Cette dernière, réciproquement, n'est pas forcée de comprendre le sens du texte pour le mettre en forme correctement. Tout ce qui l'intéresse, c'est la séparation en chapitres, le format du document, l'agencement des illustrations, etc. La conséquence de cet amalgame se voit régulièrement dans les documents Word : l'utilisateur, en entamant une rédaction sur un tel logiciel, commence déjà à mélanger les séquences de mise en page avec l'écriture pure. Il s'ensuit une perte incessante du fil de la rédaction, ainsi qu'une mise en forme « de survie », paragraphe par paragraphe, voire mot par mot, qui engendre souvent des résultats désastreux à la relecture sur un autre ordinateur.

Ensuite, par son réalisme à l'affichage, le WYSIWYG fait croire à l'utilisateur que ce qu'il fait reflète exactement la mise en forme du document. Souvent on a de grosses surprises entre l'écran et l'imprimante, et surtout d'un ordinateur à l'autre. La mise en forme de documents est un travail en soi, il serait illusoire de croire qu'un texte Word sera robuste en l'ayant formaté à coups de tabulations, d'espaces ou de « glisser/déplacer » à la souris. Le principe du WYSIWYG a pour conséquence de masquer un bon nombre d'informations de formatage, simplement pour faire comme sur le papier. L'interprétation de ce qu'on voit à l'écran est donc souvent mauvaise.

Enfin, si le WYSIWYG peut satisfaire l'utilisateur écrivant de temps en temps une lettre en se contentant de ce qui sort de l'imprimante, il peut devenir un handicap dès qu'il s'agit de traiter une grande quantité de texte. Pour un roman par exemple, avec ses innombrables pages qui devront partir à l'imprimerie, pour une thèse, qui peut contenir des centaines de références bibliographiques et d'illustrations à gérer, pour du postage de masse comme du courrier d'entreprise, du publipostage, des courriers à entête,... le WYSIWYG force l'usager (qui souvent dans ce cas est une secrétaire, pas un informaticien) à manipuler des fonctions typographiques et graphiques, alors qu'il n'y a que du texte brut à changer.

Attention, nous ne blâmons pas ici les logiciels comme Word, Open Office, Star Writer ou autres, ce sont des logiciels puissants qui en général ont des fonctionnalités de mise en page et de typographie qui n'ont rien à envier à LaTeX et fonctionnent d'ailleurs, en coulisses, de la même façon. Nous remettons en question le principe du WYSIWYG comme méthode efficace de traitement de texte.

2.3 Les langages de balisage

Mais y a-t-il dans ce cas une alternative au WYSIWYG ? Evidemment, sinon cette documentation n'existerait pas. Une méthode de travail alternative consiste à écrire d'abord le texte en tant que texte brut, à l'aide du bloc notes Microsoft par exemple, puis d'insérer des symboles dans ce texte afin d'en *baliser* les différentes parties dans le but de faire traiter le tout par un programme, en une fois, comme le ferait un imprimeur lorsque vous lui confiez votre manuscrit. On imagine par exemple que j'écrive le texte suivant :

```
Chapitre premier : comment procéder

Ceci est ma première phrase, elle doit mettre en évidence le principe suivant : blabla.
```

Très intéressant. A présent que j'ai entièrement écrit mon texte, je décide d'y ajouter des annotations, toujours en texte brut, pour un hypothétique traitement :

```
[à écrire comme un titre de chapitre] Chapitre premier : comment procéder [fin de titre de chapitre]

Ceci est ma première phrase, elle doit mettre en évidence le principe suivant :
[à mettre en valeur] blabla [fin du traitement].
```

On comprend tout à fait, on entoure juste les parties de texte à traiter de façon particulière avec des indications entre crochets. N'importe quel imprimeur, avec une telle notification, pourrait produire un document présentable :

Chapitre premier : comment procéder
Ceci est ma première phrase, elle doit mettre en évidence le principe suivant : *blabla.*

En fait nos venons simplement d'inventer et d'utiliser un *langage de balisage*. Les balises sont de parties de texte distinctes du texte initial, qui servent à donner des indications pour un traitement (de texte ?) futur, par un imprimeur, un programme, une machine, bref n'importe quoi qui puisse respecter les règles de typographie et les appliquer à notre document.

Il est fondamental de remarquer qu'on n'a donné aucune indication de police, de taille, de disposition de caractères, choses qu'on aurait effectuées immédiatement dans un programme WYSIWYG. Ici les indications concernent le découpage sémantique du texte, pas les effets purement graphiques. C'est dans la phase de traitement que nous déciderons par exemple si la balise `[à mettre en valeur]` devra se matérialiser par de l'italique, du gras, avec un type de police spécial, etc. Dans ce sens, on parle de traitements de texte WYSIWYM, What You See Is What You Mean, traduisez « ce que vous voyez est ce que vous signifiez ».

On peut citer en exemple l'un des langages les plus utilisés dans l'informatique actuelle, bien que vous ne le voyez jamais directement : le HTML, HyperText Markup Language, « langage de balises hypertexte ». C'est bel et bien un système de balises qui, incorporé à un texte, permet de présenter ce dernier de façon spectaculaire pour peu que le tout soit interprété dans un navigateur (comme Internet Explorer ou Mozilla). Voici un exemple de HTML :

```
<TITLE>Un document</TITLE>
<BODY>
  <H1>Mon premier document</H1>
  Bienvenue dans le WWW
  <P>C'est un paragraphe.</P>
  <P>Ceci en est un autre</P>
</BODY>
```

Voilà, c'est très simple, on peut remarquer par exemple les balises `<H1> ... </H1>` qui déterminent un titre de chapitre, ou encore les balises de séparation en paragraphes... Le résultat, uniquement visible à travers un navigateur, dépendra largement des habitudes de celui-ci, mais la disposition du texte sera respectée : titres, chapitres, paragraphes,... HTML n'a pas du tout la vocation d'être un traitement de texte, mais il montre bien les avantages de cette méthode de travail : le document est portable (c'est du texte brut !), de très petite taille, et malgré son potentiel interactif (qui donne des pages web complexes) le tout reste suffisamment lisible.

Il est à noter que les logiciels de traitement de texte WYSIWYG fonctionnent aussi avec des balises, mais vous donnent un aperçu en temps réel de ce que vous pouvez obtenir à l'impression. Vous n'aurez jamais accès aux balises, seulement à des paramétrages graphiques qui construisent indirectement les balises.

2.4 LaTeX

Lorsqu'on voit tourner LaTeX pour la première fois, on se demande quel esprit malade a pu concevoir une telle démarche pour mettre en forme du texte. A présent que nous avons entrevu ce qu'est un langage de balises, l'intérêt de LaTeX doit commencer à apparaître. Nous allons essayer de transformer l'essai.

Comme par hasard, LaTeX est donc basé sur un langage de balises. Et c'est pour cela qu'il est si puissant, portable et simple (malgré tout ce qu'on peut penser). Evidemment, comme on devra écrire des balises, des symboles, des paramètres, il est nécessaire de connaître quelque peu la syntaxe de base de LaTeX. Certains y voient là un obstacle, critiquent une complexité d'emploi qui le réserve aux génies de l'informatique et/ou aux masochistes. En fait quand on y réfléchit, une suite bureautique comme MS Office est très complexe, met en œuvre des principes avancés de mise en page, qui sont parfois difficiles à comprendre. Mais ils cachent cette complexité sous une apparence conviviale et intuitive. C'est dangereux, car de ce fait on ne cherche presque jamais à comprendre comment marche réellement une fonctionnalité, et on se cantonne à ce qu'on croit qu'elle fait. Avec LaTeX il n'y a pas de vernis, on écrit tout ce qu'on veut faire. Mais on n'en fait pas plus, et si un texte demande peu de mise en forme, le travail sous LaTeX se résumera à quelques symboles éparpillés dans le texte, voire rien du tout !

Mais trève de théorie, regardons concrètement comment se déroule un travail sur LaTeX.

3 Au travail !

3.1 Premier contact

Dans cette section nous allons voir dans de très grandes lignes la procédure à suivre lorsqu'on travaille sous LaTeX. Nous omettons énormément de détails dans la construction du code source, mais nous y reviendrons plus tard.

Commençons par étaler l'armement nécessaire à la production d'un document avec LaTeX. Il nous faut :

- un ordinateur pour lequel LaTeX est disponible (environ tous) ;
- un éditeur de texte pour écrire tous nos documents (du simple bloc-notes jusqu'à des pointures comme emacs [1]) ;
- la syntaxe des balises LaTeX en tête ;
- une suite LaTeX moderne comprenant les programmes de traitement, les polices,... ;
- un programme pour lire et imprimer le résultat (maintenant très souvent du PDF) ;
- pas mal de café.

La procédure à suivre est très simple, pour peu que l'on sache ce que l'on veut faire. On commence donc par... s'éloigner de l'ordinateur, se mettre à son bureau et écrire son texte au crayon, avec toutes les annotations que l'on veut, histoire d'avoir une vraie base de travail. Entre autres, les principaux problèmes de disposition, de sémantique, de mise en forme seront résolus instinctivement par cette méthode. Si votre texte est déjà écrit, essayez tout de même de passer par une phase d'annotation, le traitement n'en sera que plus facile.

Ensuite, tapez votre texte dans votre éditeur de texte favori. On verra à ce moment qu'un texte réellement tapé au kilomètre n'est pas nécessairement simple à lire. Ensuite on incorpore, suivant les annotations faites sur papier, les balises LaTeX correspondantes. Sauvegardez tout ça dans un fichier, mettons `essai.tex`.

[1] emacs, l'éditeur vedette du projet GNU, offre notamment le module AucTeX, un mode d'édition LaTeX complet qui permet d'insérer des commandes, de reformater le texte, et de lancer les compilations en quelques clics.

Le code source est maintenant prêt, sans erreurs (miracle), il reste à le traiter. Comme pour envoyer un manuscrit à l'imprimeur, on va fournir le code source au **compilateur LaTeX**. Ce programme, contenu dans toute distibution LaTeX — en fait *c'est* LaTeX —, va analyser vos balises et mettre en forme votre document suivant vos instructions. Pour faire cela, il suffit de lancer le programme nommé `latex` (`latex.exe` sous DOS ou Windows), en lui disant simplement le nom de votre fichier source :

```
latex essai.tex
```

Immédiatement après (et comme vous n'avez fait aucune erreur à l'édition !), vous obtenez un fichier graphique vectoriel, soit en PostScript, soit en DVI, soit en PDF, prêt à l'impression.

Et on dit que LaTeX est compliqué !!!

Bon, évidemment on a omis ici bon nombre d'explications importantes, mais c'est vraiment pour percevoir la démarche. Regardons à présent de plus près comment écrire un document avec une syntaxe LaTeX.

3.2 Parlons LaTeX

3.2.1 Principes basiques de l'écriture LaTeX

LaTeX paraît très compliqué à utiliser du fait de ses nombreuses possibilités et combinaisons, mais en fait il repose sur assez peu de conventions syntaxiques. Le premier point important est « comment indiquer à LaTeX une instruction, afin qu'il fasse la distinction entre le texte lui-même et des directives de mise en page ? ». Il faut pour cela des *caractères spéciaux*, qui indiqueront que ce qui les suit sera une instruction LaTeX. En l'occurrence il en existe deux principaux : le *signe pourcentage* % et le *backslash* \.

Le premier, « % », sert à débuter un *commentaire*, c'est-à-dire du texte lisible dans le fichier source, mais qui ne se verra pas du tout dans le document final. Le principe des commentaires vient du passé « informaticien » de LaTeX ; cela permet d'annoter le fichier source pour expliquer en bon Français ce que l'on fait, pour indiquer des noms d'auteur, des dates, des instructions de compilation, etc. C'est un énorme avantage par rapport à un logiciel WYSIWYG, qui comme son nom l'indique n'affiche que ce qu'on imprimera !

Exemple :

```
% Cette phrase ne fera pas partie du document final.
Cette phrase fera partie du document final.
```

Cela donne, une fois compilé :

> Cette phrase fera partie du document final.

Le second, « \ », est le caractère d'échappement proprement dit, il doit être systématiquement présent avant toute instruction LaTeX. Il existe plusieurs façons d'invoquer une instruction, mais toutes nécessitent ce backslash.

Profitons-en pour voir quelques fonctions de base. Pour des traitements relativement ponctuels, on entoure le texte d'accolades, et on fait précéder le tout par l'instruction LaTeX appropriée. Je veux par exemple mettre en évidence le mot « important » dans la phrase suivante :

> Ceci est très important.

Pour cela, on doit écrire ceci dans le fichier source :

```
Ceci est très \emph{important}.
```

pour obtenir ceci :

Ceci est très *important*.

La fonction `\emph`, qui signifie *style emphatique*, est traité à sa façon par LaTeX, mais ce dernier garantit quoi qu'il arrive que l'effet sera réussi. En l'occurrence il emploie ici un style italique.

Pour traiter du texte en plus grande quantité, certaines instructions doivent, comme en HTML s'écrire au début et à la fin du texte à traiter. On utilise pour cela les instructions `\begin` et `\end` comme ceci :

```
\begin{instruction}
  texte à traiter
\end{instruction}
```

Par exemple on peut écrire un texte centré :

```
\begin{center}
  Il était une fois une petite fille de village,\\ la plus jolie qu'on eût su voir : sa
  mère en était folle, et sa mère-grand plus folle encore. Cette bonne femme lui fit
  faire un petit chaperon rouge qui lui seyait si bien, que partout on l'appelait
  le Petit Chaperon rouge.
\end{center}
```

On obtient :

Il était une fois une petite fille de village,
la plus jolie qu'on eût su voir : sa mère en était folle, et sa mère-grand plus folle encore.
Cette bonne femme lui fit faire un petit chaperon rouge qui lui seyait si bien, que partout
on l'appelait le Petit Chaperon rouge.

Profitons de cet exemple pour remarquer un trait important du travail sur LaTeX : la façon de taper le texte dans le code source n'a aucune incidence sur le traitement. On peut passer des lignes, faire des indentations, multiplier les espaces entre les mots, tout ceci se traduit par un simple espace dans le texte final. Cela peut paraître très contrariant au début, mais en fait c'est totalement lié aux principes que nous exposons ici. Le texte est le contenu, représenté par des mots, et LaTeX se charge de *toute* la mise en page, y compris les espaces et les interlignes, afin de garantir l'homogénéité graphique du document. Ainsi il est impossible de séparer des lignes en tapant dix fois sur Entrée ; à la place, on doit écrire une instruction telle que `\paragraph{}`, qui assurera une séparation correcte, ou `\\` qui force le retour à la ligne dans un paragraphe (comme dans notre exemple). De la même manière, l'utilisateur français n'a pas besoin de donner les césures, LaTeX le fait automatiquement. Dans le cas contraire, on peut forcer une césure en ajoutant `\-` dans le mot à couper.

Il existe d'autres instructions qui utilisent des caractères spéciaux pour délimiter les parties à traiter. Par exemple, pour écrire des formules mathématiques (exercice dans lequel LaTeX excelle, sans jeu de mots), on les encadre par le caractère $ au début et à la fin. Si vous êtes quelque peu perspicaces, vous commencez à vous dire : « puisque des caractères spéciaux servent à délimiter les instructions, comment fait-il pour nous les écrire dans ce document » ? C'est vrai : comment peut-on dire à LaTeX d'écrire par exemple le mot `\begin{center}`, alors qu'il est sensé l'interpréter et pas l'écrire ? En fait LaTeX a prévu le coup, et met en place un système d'*échappement*, afin de pouvoir écrire tout ce qu'on veut, y compris les signes spéciaux %, {, }, $, et même \. Le fameux backslash est encore utilisé, mais pour prévenir que l'on va écrire un caractère spécial.

Voici les façons d'écrire tout ça :

```
Caractère pourcentage : \% ;\\
accolade ouvrante : \{ ;\\
accolade fermante : \} ;\\
signe Dollar : \$ ;\\
backslash : \textbackslash.
```

Cela donne :

Caractère pourcentage : % ;
accolade ouvrante : { ;
accolade fermante : } ;
signe Dollar : $;
backslash : \.

On voit que certains caractères s'obtiennent en leur collant un \, tandis que d'autres (comme le backslash) demandent une vraie description. La liste des caractères spéciaux est grande, et il est très facile d'écrire le symbole que l'on veut *via* n'importe quel didacticiel correctement fait.

3.2.2 Exemples de traitement

Voici pêle-mêle quelques exemples simples de mise en page LaTeX. Comme à chaque fois, il ne faut pas se formaliser sur l'exactitude du rendu graphique final, mais sur la mise en valeur de son texte. LaTeX assurera le reste. Nous évoquerons la partie purement graphique de la mise en page plus loin.

Pour **centrer un texte qui tient sur une seule ligne** (sans retours à la ligne à l'intérieur), on utilise `\centerline{mon texte}`.

Pour **pousser l'écriture à gauche**, on écrit son texte dans un environnement *flushleft* :

```
\begin{flushleft}
  Texte qui sera poussé à gauche.
\end{flushleft}
```

Texte qui sera poussé à gauche.

Pour **pousser l'écriture à droite**, on écrit son texte dans un environnement *flushright* :

```
\begin{flushright}
  Texte qui sera poussé à droite.
\end{flushright}
```

Texte qui sera poussé à droite.

Pour **écrire une liste avec des puces** on utilise l'environnement *itemize* :

```
\begin{itemize}
\item un premier point~;
\item un deuxième point~;
\item un troisième point.
\end{itemize}
```

- un premier point ;
- un deuxième point ;
- un troisième point.

Pour **écrire une liste avec des numéros** on utilise l'environnement *enumerate* :

```
\begin{enumerate}
\item un premier point en mode \emph{enumerate}~;
\item un deuxième point~;
\item un troisième point.
\end{enumerate}
```

1. un premier point en mode *enumerate* ;
2. un deuxième point ;
3. un troisième point.

4 Structuration des documents

4.1 Disposition d'un fichier source LaTeX

Nous avons constaté qu'il est possible d'écrire très simplement du texte et laisser à LaTeX le soin de le traiter. Cependant, il faut respecter un format particulier pour l'écriture du fichier source, afin que le compilateur puisse s'y retrouver.

Typiquement, un fichier `.tex` possède deux structures majeures :

l'entête de document dans lequel on énumère, au besoin, des paramètres pour la totalité du document ;

le corps de texte entouré des lignes `\begin{document}` et `\end{document}`, dans lequel on écrit le texte balisé comme nous l'avons vu auparavant.

On peut donc schématiser un fichier source ainsi :

```
\paramétrage_1
\paramétrage_2
...
\paramétrage_n

\begin{document}

  blablabla...

\end{document}
```

Un fichier source peut devenir énorme au besoin, ou rester minuscule, mais il conserve cette forme, et il faut toujours s'en rappeler lorsqu'on travaille sur un document, aussi complexe soit-il.

Nous allons donc voir ce qu'on met dans ces fameux paramétrages, et comment réutiliser les fichiers LaTeX de document en document.

4.2 Familles de documents

Bien que tout document soit composé en grande partie de texte, leur structure peut beaucoup varier selon leur usage. On ne présentera pas du tout de la même façon une lettre, un journal, une série de transparents, un article scientifique ou un roman ; leurs dimensions, leur agencement (pages de garde, table des matières, bibliographie), leurs ornementations (entêtes, pieds de page, annotations, lignes de massicotage), leur traitement typographique sont très différents.

Comme à son habitude, LaTeX ne vous perdra pas dans cette jungle de techniques d'imprimerie, et vous laissera vous concentrer au maximum sur votre travail. Ainsi, au lieu de vous faire régler des tonnes de paramètres, il vous proposera des familles de documents, qui prédéfinissent ces paramètres pour quelques documents typiques.

En standard, LaTeX propose cinq classes de document :
- *article*, originaire des milieux scientifiques, permet de faire une mise en page légère mais complète (titre, sections, table des matières, bibliographie,...). Pratique pour des documents de base, comme celui que vous lisez actuellement ;
- *report*, ressemblant à *article*, mais plus formel (titre sur une seule page,...) ;
- *book*, pour écrire un livre complet avec pages de garde,... ;
- *letter*, pour écrire du courrier ;
- *slides*, pour faire des transparents.

Cela semble très peu, mais en fait on peut modeler à l'infini ces classes pour les adapter à n'importe quelle mise en page.

Quoiqu'il en soit, il faut absolument déterminer la classe de document utilisée au début d'un fichier source LaTeX :

```
\documentclass[a4paper,10pt,oneside]{article}

\begin{document}

  blablabla...

\end{document}
```

On en profite pour voir la façon d'ajouter des options à une instruction : entre les crochets (optionnels) de l'instruction `\documentclass{article}`, on indique par exemple le format de papier (`a4paper`), la taille de police par défaut (`10pt`), la pagination recto ou recto-verso (`oneside`).

Par la suite nous utiliserons en priorité la classe *article* pour nos exemples, c'est la plus utilisée et la plus parlante.

4.3 Françisation

Histoire de rassurer tout le monde assez rapidement, il faut signaler que LaTeX gère beaucoup de langues. Non seulement pour les caractères spéciaux (comme le œ ou les accents en Français), mais également pour les règles de typographie spécifiques à un pays, ce qui est au moins aussi important. Ainsi, en Français, on utilise des espacements de part et d'autre des point-virgule et des deux-points, nous utilisons des tirets longs (— au lieu de -) pour une *incise* — équivalent d'un texte entre parenthèses. D'ailleurs, si LaTeX peut écrire à la perfection en typographie française, il est impératif de connaître celle-ci un minimum, afin d'utiliser la bonne typographie aux bons endroits. Cela implique un travail supplémentaire, mais on obtient des documents largement plus lisibles, aérés, compréhensibles, ce qu'on rate souvent sur des traitements de texte WYSIWYG.

Par exemple, on peut citer quelques directives typographiques à signaler manuellement dans notre fichier source [5] :

- on doit mettre un *espace insécable* (représenté par ~ dans le fichier source) avant toute ponctuation double (caractères ! ? : ;) ;
- un espace normal après tous les signes de ponctuation (caractères ! ? : ; , .) ;
- un *espace insécable* avant les unités de mesure (k, min, kg, km/s,...) et les unités monétaires, ainsi que le symbole % ;
- un *espace insécable* après les guillements français ouvrants («) et avant les guillements fermants (»).

La liste est plus longue. Entre autres il convient également d'écrire les locutions latines en italiques (par exemple avec l'instruction `\emph{}`), et certaines abréviations ont leur typographie particulière (ex. : c.-à-d., *i.e.*, cf., etc.,...).

Il existe plusieurs packages pour invoquer l'utilisation de la langue française. Nous vous donnons celui utilisé dans cet article : `\usepackage[francais]{babel}`. Il faut ajouter à cela l'appel au bon jeu de caractères, soit *latin-1*, par la commande `\usepackage[latin1]{inputenc}`.

On obtient :

```
\documentclass[a4paper,10pt,oneside]{article}

\usepackage[latin1]{inputenc}
\usepackage[francais]{babel}

\begin{document}

  blablabla...

\end{document}
```

4.4 Le découpage des documents

Un texte est en général divisé en sections. Pour un bouquin par exemple, on distingue le titre, l'auteur, la table des matières, les chapitres qui peuvent être divisés en sous-chapitres de plusieurs niveaux, la table des illustrations, la table de bibliographie, les annexes,... Tout ceci est évidemment géré dans LaTeX, de façon parfois déroutante mais efficace.

Concernant le découpage en chapitres, sections, sous-sections, encore une fois on ne s'intéresse pas du tout au rendu graphique, mais on indique à LaTeX qu'une phrase est un titre de section, grâce à des instructions.

Par exemple :

```
...
\begin{document}
\section{Ma première section}
\subsection{Ma première sous-section}
blablabla...
\subsection{Ma deuxième sous-section}
```

```
\subsubsection{Ma première sous-sous-section}
blablabla...
\subsubsection{Ma deuxième sous-sous-section}
blablabla...
\section{Ma deuxième section}
\end{document}
```

On obtient le document de la figure 1.

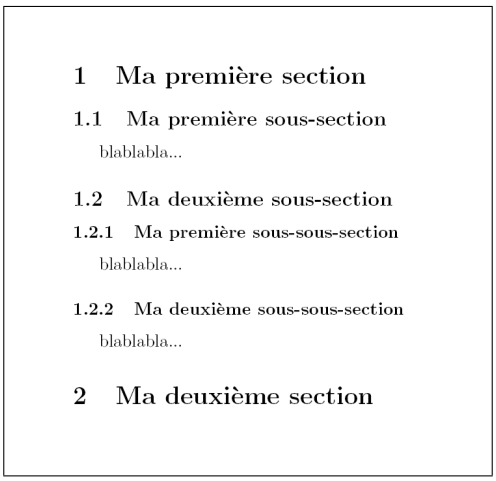

Fig. 1 – Utilisation des sections dans un document.

On remarque que la numérotation est automatique, puisqu'elle n'est jamais mentionnée dans le fichier source. Ce principe est déterminant pour la fabrication du document, puisqu'il permet d'inclure d'autres parties de document dans le fichier, de fabriquer une table des matières, le tout sans aucun problème. Combien de fois devons-nous reprendre une numérotation dans un logiciel WYSIWYG si nous décidons de modifier l'ordre des sections ! Ici, comme la sortie graphique est formée lors de la compilation, en une fois, le problème ne se pose jamais.

On peut changer le style de numérotation (numérotation romaine, lettres,...), ou ne pas avoir de numérotation du tout (avec les instructions de type `\section*{}`), mais nous vous laissons le soin de découvrir tout cela par vous-mêmes [6]. Pour les parties spéciales du document (titre, table des matières,...), c'est aussi simple : pour insérer la table des matières, on se place où on la veut dans le document, et on écrit juste `\tableofcontents`. Si on a choisi un package français, on voit apparaître une belle table des matières, comme celle de cet article. Pour plus de clarté on peut forcer des sauts de page avant et après la table, avec l'instruction `\newpage{}` de part et d'autre.

Le titre et l'auteur d'un document sont renseignés en deux temps. D'abord on écrit au sein des paramètres (donc avant la ligne `\begin{document}`) les instructions `\title{mon titre}` et `\author{M. Dupont}` qui contiennent le titre et l'auteur. Attention, ces instructions n'écrivent rien, elles récupèrent juste du texte qui sera plus tard utilisé pour écrire le titre et le nom de l'auteur ! Ensuite, dans le document, on ordonne l'écriture réelle du titre, grâce à la simple commande `\maketitle`, qui va récupérer le contenu des instructions `\title{}` et `\author{}` et les mettre en forme.

Pour le présent article on obtient un fichier source qui ressemble à ça (à peu près) :

```
\documentclass[a4paper,10pt,oneside]{article}
\usepackage[latin1]{inputenc}
\usepackage[francais]{babel}

% titre du document
\title{\LaTeX \\ réapprendre à utiliser un traitement de texte}
% auteur du document
\author{Archipel du Libre}

\begin{document}
\maketitle
\newpage{}
\tableofcontents
```

```
\newpage{}

Blablabla...

\end{document}
```

Simple n'est-ce pas ? Le compilateur se charge de tout le reste, il suffit d'utiliser les instructions de découpage en sections.

Ce début de fichier source, qui ne contient pas encore son texte, est appelé squelette, et permet d'avoir sous la main une base de travail opérationnelle pour traiter les textes qu'on veut. Une bonne pratique de LaTeX veut qu'on fabrique des squelettes pour nos principaux types de documents, et au moment d'écrire un nouveau document on copie le squelette en le renommant. Cela évite non seulement du travail de frappe, mais surtout des erreurs de syntaxe dans les entêtes.

Le principe régissant le titre est assez courant dans LaTeX, on le retrouve notamment dans des documents très structurés comme des lettres. En effet il est assez compliqué de décrire correctement une lettre, avec toutes ses petites zones de texte éclatées sur la page. De plus il existe des règles différentes de mises en page selon le type de lettre, alors que le contenu (expéditeur, destinataire, signature, formule de politesse,...) sont toujours là. LaTeX, dans sa classe *letter*, propose de remplir les différents champs avant de début de document, puis il les place automatiquement dans la lettre selon le type de courrier (familier, officiel,...), et la langue utilisée (la disposition n'étant pas la même selon les pays).

Par exemple [5] (voir le résultat en figure 2) :

```
\documentclass[a4paper]{letter}
\usepackage[latin1]{inputenc}
\usepackage[francais]{babel}

% Expéditeur
\address{Mes nom et prénom \\ Mon adresse \\ Mon téléphone}
% signature
\signature{Mes nom et prénom pour la signature}

\begin{document}
\begin{letter}{Le destinataire \\ son adresse \\ son téléphone}

\opening{Cher destinataire,}

Corps de la lettre, blablabla...

\closing{mes jolies salutations}

\ps{PS~: un postScriptum}

\end{letter}
\end{document}
```

5 Ornementation des documents

Dans cette partie nous allons voir comment agrémenter le texte de parties plus spéciales, comme des formules mathématiques, des tableaux ou, comme vous l'attendez tous, des images.

5.1 Textes spéciaux

Certaines parties de texte demandent une mise en page particulière, soit en disposition, soit en formatage, soit les deux. La plupart de ces textes particuliers ont leur instruction LaTeX spécifique, toujours dans le but d'obtenir une sortie aussi efficace que possible.

On distingue notamment une **citation** dans un texte par l'instruction `quotation` :

```
...
\begin{document}

Voici une citation dont je voulais vous faire part~:
\begin{quotation}
 Patricia, mon petit... Je voudrais pas te paraître vieux jeu ni encore moins grossier.
```

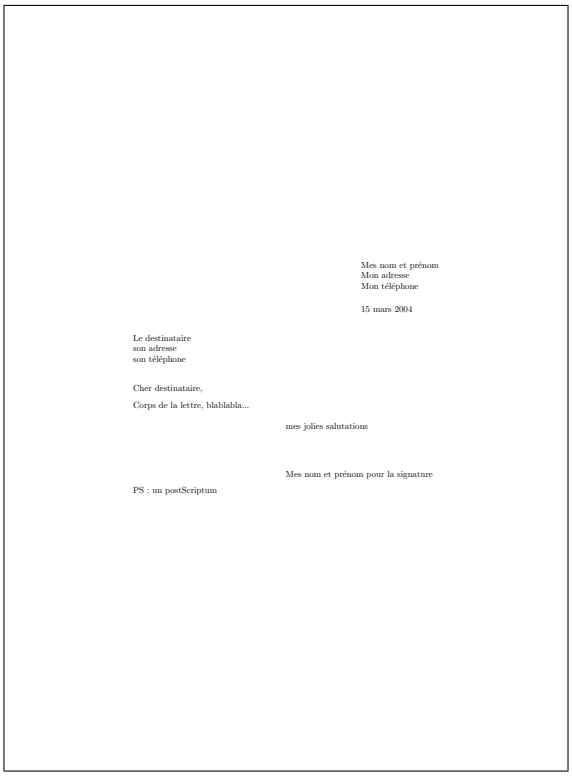

FIG. 2 – Résultat d'utilisation de la classe *letter*.

```
L'homme de la Pampa parfois rude reste toujours courtois mais la vérité m'oblige à te le dire~:
ton Antoine commence à me les briser menu~!
\end{quotation}

\end{document}
```

Ce qui donne :

> Voici une citation dont je voulais vous faire part :
>
> Patricia, mon petit... Je voudrais pas te paraître vieux jeu ni encore moins grossier. L'homme de la Pampa parfois rude reste toujours courtois mais la vérité m'oblige à te le dire : ton Antoine commence à me les briser menu !

De la même façon, si on veut citer de la poésie ou des textes de chanson, il existe l'instruction *verse*, qui prend en conte toutes les subtilités de l'exercice (césures, coupures de vers,...)

Exemple :

```
...
\begin{document}

Voici un poème inconnu de Joachim Du Bellay~:
\begin{verse}
Heureux qui, comme Ulysse, a fait un beau voyage,\\
Ou comme cestuy la qui conquit la toison,\\
Et puis est retourné, plein d'usage et raison,\\
Vivre entre ses parents le reste de son aage !
\paragraph{}
Quand revoiray-je, helas, de mon petit village\\
Fumer la cheminee, et en quelle saison,\\
Revoiray-je le clos de ma pauvre maison,\\
Qui m'est une province, et beaucoup d'avantage ?
\paragraph{}
Plus me plaist le sejour qu'ont basty mes ayeux,\\
Que des palais Romains le front audacieux,\\
Plus que le marbre dur me plaist l'ardoise fine :
\paragraph{}
Plus mon Loyre Gaulois, que le Tybre Latin,\\
Plus mon petit Lyré, que le mont Palatin,\\
```

```
Et plus que l'air marin la doulceur Angevine.
\end{verse}

\end{document}
```

Ce qui donne :

Voici un poème inconnu de Joachim Du Bellay :

Heureux qui, comme Ulysse, a fait un beau voyage,
Ou comme cestuy la qui conquit la toison,
Et puis est retourné, plein d'usage et raison,
Vivre entre ses parents le reste de son aage !

Quand revoiray-je, helas, de mon petit village
Fumer la cheminee, et en quelle saison,
Revoiray-je le clos de ma pauvre maison,
Qui m'est une province, et beaucoup d'avantage ?

Plus me plaist le sejour qu'ont basty mes ayeux,
Que des palais Romains le front audacieux,
Plus que le marbre dur me plaist l'ardoise fine :

Plus mon Loyre Gaulois, que le Tybre Latin,
Plus mon petit Lyré, que le mont Palatin,
Et plus que l'air marin la doulceur Angevine.

5.2 Mathématiques

Alors là, on entre dans un domaine de prédilection de LaTeX. En effet, ce programme avait été mis au point dans les milieux universitaires, justement pour pouvoir écrire des formules complexes et les intégrer à loisir dans du texte standard. Evidemment il est peu probable que vous ayez des formules aussi compliquées à écrire que celles d'un mécanicien quantique, mais il est tout de même pratique de connaître les instructions mathématiques de LaTeX, notamment pace qu'elles permettent d'écrire énormément de symboles exotiques. Nous allons donc voir la base de l'insertion de mathématiques dans un document, tout en sachant que l'attirail disponible est beaucoup plus fourni grâce à la communauté d'utilisateurs qui fabriquent toujours plus d'outils (formules chimiques, dessins de courbes,...).

Voyons tout d'abord la fonction la plus simple et la plus utile, l'insertion de symboles mathématiques dans un texte standard. Comme les instructions mathématiques sont spécifiques (nous en verrons au fur et à mesure), il faut les séparer du texte, car leur traitement est différent pour LaTeX. On encadre donc ces instructions entre des signes Dollar (`$`).

Par exemple, nous voulons écrire une petite équation dans une phrase :

```
Aussi vrai que $\alpha + \beta = \gamma$, \LaTeX~est vraiment puissant.
```

Aussi vrai que $\alpha + \beta = \gamma$, LaTeX est vraiment puissant.

On peut remarquer le traitement particulier des caractères en mode mathématique, mais la formule s'intègre parfaitement à la phrase.

Evidemment on peut écrire des choses bien plus étoffées :

```
L'intégrale suivante~: $\int\limits_{-\infty}^{\infty}e^{-x^{2}}dx$ est très intéressante.
```

L'intégrale suivante : $\int\limits_{-\infty}^{\infty} e^{-x^2} dx$ est très intéressante.

Toujours aucun problème pour l'incorporation dans la ligne.

Souvent en maths on a besoin de séparer les formules du texte pour clarifier l'écriture, et tout simplement parce que l'écriture mathématique demande beaucoup de place. On a pour cela l'environnement mathématique en plusieurs lignes, qui se déclare en entourant les formules avec deux signes Dollar de chaque côté.

Un exemple :

```
Voici une belle formule~:
$$
\lim_{y \to \infty}\sqrt[3]{\frac{3x}{2y-3}}=0
$$
On comprend tout !
```

Voici une belle formule :

$$\lim_{y \to \infty} \sqrt[3]{\frac{3x}{2y-3}} = 0$$

On comprend tout !

Vous pouvez facilement voir l'action des différentes instructions. De plus on remarque que le code source est très petit, il prend autant de temps à écrire que la formule à la main.

Nous allons nous arrêter là en ce qui concerne les maths, mais il faut savoir que tout est possible à écrire, des matrices aux vecteurs, les opérateurs chapeau ou barre, les symboles typiques des mathématiques ou de la physique ($\in$, $\notin$, $\exists$, $\forall$, ∂, ∇, $\hbar$,...).

5.3 Insertion d'images

Oui, ne vous inquiétez pas, LaTeX sait parfaitement intégrer des images dans ses documents ! Il est néanmoins nécessaire de prendre quelques précautions, notamment selon le fichier graphique que l'on produit :

- lorsque l'on produit des fichiers PostScript (en lançant la compilation LaTeX classique), on ne peut qu'insérer du PostScript. C'est souvent assez embêtant surtout pour les Windowsiens qui ont peu ou pas d'outils pour manipuler ce format vectoriel ;
- lorsque l'on produit directement du PDF (ce qui est maintenant très courant grâce à la compilation par pdflatex), on peut à peu près tout insérer, du PNG, du JPEG, du TIFF, et même du... PDF ! Très pratique pour insérer un autre document produit par LaTeX.

Comme la deuxième solution est plus pratique, plus souple et plus employée, nous n'allons nous intéresser qu'au PDF. Pour le PostScript c'est à peu près la même histoire, à ceci près qu'on ne pourra manipuler que d'autres PostScript.

Afin de pouvoir manipuler des images, vous vous doutez bien qu'il va falloir le signaler à l'avance à LaTeX.

Nous devons donc ajouter, dans l'entête de notre fichier source, les lignes :

```
\usepackage[pdftex]{graphicx}
\usepackage{graphics}
```

Dans ce cas on peut insérer une image (mettons `monImage.jpg`) dans le document, en écrivant :

```
\begin{figure}[h]
  \centering
  \includegraphics[width=4cm]{monImage.jpg}
  \caption{Légende de l'image.}
\end{figure}
```

Et voilà :

Ici on utilise `\centering` pour centrer l'image sur la page, mais ce n'est pas obligatoire. On remarque également qu'on a contraint la largeur de l'image à 4 cm, en sachant que LaTeX respectera les proportions.

FIG. 3 – Légende de l'image.

Attention, le nom de l'image est comprise comme un nom de fichier. Par conséquent si elle n'est pas dans le répertoire du fichier source il faut l'accompagner de son chemin, relatif ou absolu. De même, comme LaTeX est un outil précis qui vous traite en adulte, il n'opérera aucune compression d'image, aucun changement de format. L'image que vous insérez dans le document a strictement le poids du fichier. Il faut donc faire attention au traitement d'image en amont[2] (redimensionnement, compression) afin de ne pas se retrouver avec un document PDF de plusieurs Megaoctets. Cela dit, cette remarque est également valable pour un logiciel WYSIWYG !

Revenons à notre exemple de la figure 3. La commande `\caption` permet d'écrire une légende pour l'image.

L'option `[h]` que nous rajoutons à l'environnement *figure* permet de contrôler le type d'ancrage :
- h : à l'endroit où l'image a été appelée ;
- t : en haut d'une page de texte ;
- b : en bas d'une page de texte ;
- p : sur une page séparée, avec d'autres figures.

On peut évidemment manipuler les images exactement comme on veut, mais avec cette simple commande vous pouvez déjà faire des documents très corrects et c'est le but de cet article. Se référer à la bibliographie pour plus de détails.

On peut quand même faire remarquer que dans l'environnement Linux par exemple, où on a quantité d'outils qui génèrent du PostScript, on peut facilement arriver à des documents très professionnels. Par exemple GNUPlot fait des graphes scientifiques en PostScript, Dia fait des diagrammes, etc. Et pour du PDF pas de problème, on a le filtre très pratique ps2pdf qui passe de l'un à l'autre.

5.4 Tableaux

Cet objet pseudo-graphique est très pratique pour représenter des listes, des statistiques, et même pour faire de la mise en page évoluée comme on le fait parfois en HTML. LaTeX propose un environnement très pratique pour faire des tableaux de toutes sortes, de toutes tailles, avec des séparateurs ou non. Il s'agit de l'environnement *tabular*.

Par exemple faisons un petit tableau :

```
\begin{tabular}{lll}
  \textbf{Nom} & \textbf{prénom} & \textbf{Métier} \\
  Dupont & Raymond & Charcutier \\
  Durand & Gérard & Ouvrier \\
  Untel & Gaston & Coiffeur
\end{tabular}
```

Nom	**prénom**	**Métier**
Dupont	Raymond	Charcutier
Durand	Gérard	Ouvrier
Untel	Gaston	Coiffeur

Les trois « l » dans l'accolade de *tabular* donnent l'alignement de chaque colonne. Ici, « l » signifie *left*, donc le contenu des trois colonnes est aligné à gauche. On aurait pu mettre « r » pour *right* ou « c » pour *center*. Evidemment on peut choisir indépendamment l'alignement de chaque colonne, par exemple `{rlc}`.

[2] On peut avantageusement utiliser THE GIMP [1] pour cela.

Profitons-en pour voir comment mettre des séparateurs :

```
\begin{tabular}{|r|l|c|}
  \hline
  \textbf{Nom} & \textbf{prénom} & \textbf{Métier} \\
  \hline
  Dupont & Raymond & Charcutier \\
  \hline
  Durand & Gérard & Ouvrier \\
  \hline
  Untel & Gaston & Coiffeur \\
  \hline
  Bidule & Marcel & Boulanger \\
  \hline
  Machin & Yvon & Menuisier \\
  \hline
  Lagadec & Jean-Kévin & Informaticien \\
  \hline
\end{tabular}
```

Nom	**prénom**	**Métier**
Dupont	Raymond	Charcutier
Durand	Gérard	Ouvrier
Untel	Gaston	Coiffeur
Bidule	Marcel	Boulanger
Machin	Yvon	Menuisier
Lagadec	Jean-Kévin	Informaticien

On peut difficilement faire plus simple... On voit que les lignes verticales sont données dans les accolades de l'alignement par |, les lignes horizontales étant dessinées par l'instruction `\hline`. On ne donne aucune directive de géométrie, le tableau se constitue toujours parfaitement par rapport à son contenu.

6 Référencement et liens

Nous entrons à présent dans une partie originale des traitements de texte, qui est rarement utilisée puisque la fonctionnalité disparaît lors du passage au papier. Le référencement et les liens sont des outils qui permettent de relier des parties de texte ou des objets entre eux. LaTeX nous a déjà montré sa capacité à calculer des références lorsqu'on a utilisé le chapitrage, ou l'environnement *enumerate*. En fait LaTeX dispose d'un arsenal bien plus important pour le référencement, afin d'agrémenter le texte d'une foule d'ajouts différents : notes de page (pied de page, marge,...), renvois à une bibliographie, à un numéro de figure, de tableau, d'équation,... Par contre dans le code source on ne numérote jamais rien explicitement. A la place on utilise des noms de référence, qui relieront grâce à une instruction un objet à un endroit dans le texte.

6.1 Notes

Il est très simple de faire une note de bas de page, numérotée automatiquement, grâce à l'instruction `\footnote{}`.

Exemple :

```
Voici une phrase sans intérêt\footnote{Voici une note.},
mais qui sert à démontrer les capacités de footnote\footnote{Voici une seconde note.}.
```

Voici une phrase sans intérêt[1], mais qui sert à démontrer les capacités de footnote[2].

[1] Voici une note.
[2] Voici une seconde note.

Les notes peuvent aussi se faire dans la marge :

```
Voici une phrase sans intérêt. \marginpar{\scriptsize Voici une note de marge.}
\marginpar{\scriptsize Voici une note de marge.}
Ceci sert à démontrer les capacités de marginpar.
```

Voici une note de marge. Voici une phrase sans intérêt. Ceci sert à démontrer les capacités de marginpar.

Ici l'exemple n'est pas très beau, parce que le document ne s'y prête pas...

6.2 Référencement des images

La commande `\caption{}`, qui ajoute une légende dans une figure, peut accueillir un marqueur servant à identifier l'image. On l'écrit avec la commande `\label{marqueur}`.

Exemple :

```
\begin{figure}[!h]
  \centering
    \includegraphics[width=4cm]{./exemple.pdf}
  \caption{\label{markexemple}J'ai déjà vu cette tête-là quelque part...}
\end{figure}
```

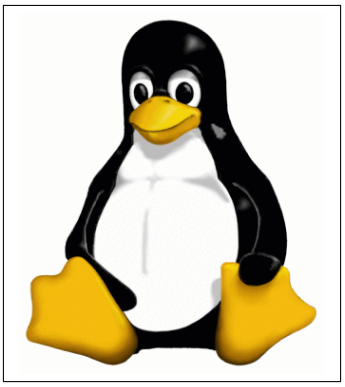

Fig. 4 – J'ai déjà vu cette tête-là quelque part...

On ne remarque rien à la compilation, mais on a dorénavant marqué l'image, ce qui nous permet à présent de la désigner ailleurs dans le texte, grâce à l'instruction `\ref{markexemple}`. Ainsi, lorsqu'on inclut cette commande dans le texte, LaTeX calcule automatiquement le numéro de l'image et l'écrit à cet endroit.

Exemple :

```
Comme nous le voyons dans la figure \ref{markexemple}, il est possible de désigner
une image où qu'elle soit dans le texte.
```

> Comme nous le voyons dans la figure 4, il est possible de désigner une image où qu'elle soit dans le texte.

Certains packages (*varioref*) permettent même de remplacer automatiquement le numéro par des textes comme « ci-dessous » ou « ci-contre » par exemple, s'il y a lieu.

6.3 Générer un index

Cet exercice est quasiment impossible à faire correctement en WYSIWYG. LaTeX, en employant le même principe que pour les numéros de figure (en marquant avec un mot plutôt qu'avec un numéro) peut fabriquer l'index de votre document très facilement. Il faut tout d'abord, dans l'entête, signaler qu'on va utiliser les possibilités d'indexation, par `\usepackage{makeidx}`, puis toujours dans l'entête écrire `\makeindex` (pour que LaTeX génère un fichier d'indexation). Enfin, à l'endroit où doit figurer l'index, écrire `\printindex`.

Evidemment, un ordinateur est toujours idiot par définition, il est donc incapable de savoir quels mots sont importants ou non. Il faut donc englober chaque mot susceptible de figurer dans l'index par la commande `\index{mot}`. LaTeX se chargera de les regrouper et de les classer par ordre alphabétique dans l'index, accompagnés de leur numéro de page. Cette commande est évidemment bien plus complète que cela, mais il vous appartient d'en découvrir les subtilités (renvois à d'autres mots, sous-index,...).

6.4 Gérer une bibliographie

Encore un outil extrêmement pratique, venant de l'utilisation scientifique de LaTeX : la possibilité de gérer une bibliographie complète, indépendante du texte.

Pour ce faire, on utilise en fait un compilateur annexe, BibTeX. Ce programme est chargé de fusionner le fichier source avec un autre fichier que l'on aura écrit en parallèle, avec un suffixe `.bib`. Ça semble très compliqué mais on s'y fait vite, et l'utilité de cette fonctionnalité est telle qu'on oublie vite les manipulations supplémentaires. Pensez que votre bibliographie est alors dans un fichier distinct, donc on peut s'en servir pour plusieurs documents, les références seront absolument identiques.

Il y a donc plusieurs étapes pour incorporer une bibliographie dans un document :
- on écrit un fichier `.bib` (nous allons en voir la syntaxe) ;
- on insère la bibliographie dans le fichier source LaTeX ;
- on ajoute les références des ouvrages aux endroits voulus dans le document ;
- après sauvegarde générale (`.tex` et `.bib`) on lance BibTeX avec en paramètre le fichier `.tex` ;
- on lance enfin LaTeX pour compiler le document final.

Voyons tout d'abord comment écrire un fichier de bibliographie (`.bib`). Il s'agit encore une fois d'un bête fichier texte, qui contiendra uniquement la liste des ouvrages qui nous intéressent. On pourrait imaginer une liste plate, mais LaTeX permet de faire des opérations élaborées de mise en page pour la bibliographie. Chaque ouvrage possède donc des paramètres comme l'auteur, le titre, l'éditeur, etc.

Un ouvrage dans un tel fichier ressemble à ça :

```
@Book{latexoreilly,
  author =       {Christian \textsc{Rolland}},
  title =        {\LaTeX~par la pratique},
  publisher =    {O'Reilly},
  year =         {1999}
}
```

Le type d'ouvrage est précédé d'une arobe (@), et l'ouvrage est entièrement décrit entre des accolades, les paramètres étant séparés par des virgules.
Le premier paramètre est le marqueur de l'ouvrage, le nom qui nous permettra de le désigner dans notre texte. Les autres paramètres sont du type paramètre=valeur ; pour cet exemple, qui représente un livre, on a respectivement les champs Auteur, Titre, Editeur et Année.

Il nous est proposé plusieurs types d'ouvrages, comme des articles, des revues,... et si aucune ne vous plaît vous pouvez toujours utiliser le type « Divers », donné par `@Misc{}`. Pour notre article nous nous en sommes servis pour les sites web de notre bibliographie.

Exemple :

```
@Misc{gnu,
  title =        {http://www.gnu.org/},
  note =         {Site officiel du projet GNU}
}
```

Une fois que nous avons sauvegardé notre fichier de biblio (par exemple avec le nom `maBiblio.bib`), nous pouvons le saisser de côté pour revenir à notre fichier `.tex`. On se dirige à la fin du document (enfin là où on veut voir figurer sa biblio), et on écrit :

```
\nocite{*}
\bibliographystyle{unsrt}
\bibliography{maBiblio}
```

La première ligne ordonne d'écrire tous les ouvrages mentionnés dans le document ; la deuxième ligne détermine l'ordre d'écriture (alphabétique, ou comme ici selon l'ordre d'apparition dans le document) ; la dernière ligne ordonne enfin l'écriture de la biblio à cet endroit. On remarque notamment que le fichier de biblio est en paramètre, mais sans son suffixe `.bib`.

Ensuite, il faut utiliser les références bibliographiques dans notre document. Tout comme un marqueur d'image, on écrit le marqueur de l'ouvrage concerné dans le texte avec la fonction `\cite{unOuvrage}`, ce qui se traduira dans le document final par le numéro de l'ouvrage entre crochets.

Exemple :

```
Je ne saurais que trop vous conseiller cet ouvrage \cite{latexoreilly}.
```

> Je ne saurais que trop vous conseiller cet ouvrage [5].

Les fichiers sont à présent prêts à être compilés. On lance donc BibTeX sur le fichier `.tex` (en tapant `bibtex monfichier.tex` en ligne de commande par exemple), en général deux fois pour bien avoir les relations croisées. Enfin on compile le document avec LaTeX deux fois également, et paf, on a une belle biblio, avec pour titre « Références », et les bons liens entre crochets aux bons endroits.

6.5 L'interactivité en PDF

Et en plus ça bouge ! Car tous les liens et référencements que vous avez formés dans ce chapitre sont bel et bien actifs. Lorsque l'on compile notre document directement en PDF, le fichier final est entièrement dynamique : un clic sur une ligne de la table des matières et on se retrouve dans le bon chapitre, un clic sur une référence d'ouvrage et on le retrouve dans la biblio, un clic sur un lien d'image et on la voit instantanément. C'est très utile pour faire des documents électroniques maniables, plus lisibles. Comme LaTeX fonctionne ainsi à la compilation avec ses systèmes de marqueurs, c'est un jeu d'enfant que de les transformer en liens croisés dans le PDF final... Voire dans le fichier HTML, puisqu'il existe un compilateur `latex2html` également. En une commande, votre bête bout de papier électronique devient une page web...

Si vous ne le croyez pas, essayez de cliquer sur ce document. Les liens ont été mis en noir pour des besoins d'impression, mais tout est fonctionnel.

Attention toutefois, il faut à nouveau prévenir LaTeX que vous voulez faire des liens dynamiques. Le package concerné est *hyperref*, et il faut l'ajouter dans les packages d'entête.

Voici les lignes de notre document pour faire des liens PDF actifs mais noirs :

```
% "couleurliens" va contenir la couleur noire
\definecolor{couleurliens}{rgb}{0.,0.,0.}

% activation des liens dans le PDF
\usepackage[pdftex,colorlinks=true,urlcolor=couleurliens,linkcolor=couleurliens,citecolor=couleurliens]{hyperref}
```

Voilà, rien de bien méchant, et tout fonctionne. Encore un dernier truc, grâce à ce package vous pouvez également écrire des adresses mail ou web et les rendre cliquables, avec la commande `\url{}`.

Exemple :

```
Ceci est notre page web~: \url{http://www.archipeldulibre.org/}.\\
Ceci est notre adresse mél~: \url{adl@infini.fr}.
```

> Ceci est notre page web : `http://www.archipeldulibre.org/`.
> Ceci est notre adresse mél : `adl@infini.fr`.

Si vous êtes sous Acrobat Reader et que vous passez la souris sur les adresses ci-dessus, vous pourrez constater qu'elles sont actives.

7 Et la mise en page ?

Vous avez sans doute remarqué que nous n'avons jamais parlé de la mise en page proprement dite dans cet article. Aucune donnée concernant les réglages graphiques. LaTeX vous permet bien entendu de modifier au millipoil chaque paramètre, la taille des marges, des pieds de page, les polices utilisées,... C'est une omission intentionnelle, pour recentrer le débat sur le véritable travail que l'on demande à un traitement de texte. Ce présent document a par exemple été écrit simplement en utilisant la famille de documents *article*, en donnant la taille générale de police (10 pt), et en élargissant un peu les marges :
`\usepackage[top=3cm, bottom=2cm, left=2cm, right=2cm]{geometry}`
Tout le reste a été contrôlé par LaTeX, et vous constatez que le résultat est parfait.

Avec l'habitude, le travail sous LaTeX se divise en deux temps, un temps pour écrire du texte et le mettre en forme, un temps pour composer des maquettes de documents personnelles. C'est une bonne méthode dans le sens où elle sépare bien les deux, et surtout parce que le résultat est toujours bon lorsqu'on arrive à la compilation.

Nous vous orientons vers l'excellent (et pour tout dire presque le seul) ouvrage en Français parlant de LaTeX : aux éditions O'Reilly, *LaTeX par la pratique* [5] vous apprendra en détails comment personnaliser vos documents.

Conclusion

Cet article est loin de couvrir les fonctionnalités de LaTeX, qui de par son architecture en packages grandit de jour en jour. L'intérêt était de recentrer le débat sur les traitements de texte, pas spécialement pour que tout le monde utilise LaTeX, mais pour que les personnes intéressées puissent mieux tirer parti de leur traitement de texte. Car au fil des exemples de cet ouvrage, il apparaît (enfin j'espère !) une méthodologie de travail radicalement différente de celle des logiciels WYSIWYG, et malgré tout on peut appliquer ces principes à Word ou OpenOffice.org : prendre l'habitude d'écrire son texte avant de le mettre en forme, voir le formatage comme des types structurés (les *styles* de Word ou Open office sont très pratiques pour ça) au lieu d'accumler les modifications graphiques (police 12, Times, vert à pois bleus, et on tire sur un paragraphe pour qu'il s'aligne avec un autre,...).

LaTeX, par son approche *séquentielle* (on écrit une séquence linéaire d'instructions), met en avant le travail sur des structures cohérentes, qui constituent la maquette du document. Il automatise au maximum la mise en page, et laisse l'utilisateur se recentrer sur ses vrais objectifs, présenter correctement un texte.

Il ne reste plus qu'à l'essayer, et se rendre compte qu'il s'agit là d'une révolution dans le monde des traitements de texte. Une révolution qui date quand même de 1978...

Références

[1] http://www.gimp.org/. Site du projet The GIMP, *Gnu Image Manipulation Program*.

[2] http://www.gnu.org/. Site officiel du projet GNU.

[3] http://texlive.sarovar.org/. TeX Live, un LaTeX clé en main pour toutes les plateformes (images ISO, documentation,...).

[4] http://www.framasoft.net/. Les aventures d'un peuple migrateur.

[5] Christian Rolland. *LaTeX par la pratique*. O'Reilly, 1999.

[6] http://www.giss.nasa.gov/latex/ltx-2.html. Liste alphabétique des commandes LaTeX.