

Unité de Recherche en Enseignement des Mathématiques  
UREM-U.L.B.

---

**Ateliers L<sup>A</sup>T<sub>E</sub>X**  
**Animés par**  
**Yves Delhaye et**  
**Hugues Vermeiren**

---

**25 août 2010**

---



# Ateliers L<sup>A</sup>T<sub>E</sub>X

Jean-Côme CHARPENTIER, Yves DELHAYE

25 août 2010



# Sommaire

<b>I Premier atelier : prise en main</b>	<b>1</b>
1 Présentation de $\LaTeX$	3
2 Distributions $\LaTeX$ et installation	11
3 Premiers pas	15
4 Commandes de Base : environnements	39
5 Modes Mathématiques	51
6 Commandes de Base : structurer un document	61
7 Commandes de Base : Fontes, espaces	67
<b>II Deuxième atelier : tableaux et mises en page avancés</b>	<b>77</b>
8 Tableaux	79
9 Extensions	87
10 Mathématiques avancées	95
<b>III Troisième atelier : programmation, macros, compteurs, graphisme, programmes externes</b>	<b>111</b>
11 Graphisme	113
12 Graphisme avec Tikz	129
13 Macros et compteurs	145
<b>IV Annexes</b>	<b>153</b>
Bibliographie	159

**Préface à la quatrième édition** Cette version du cours est destinée au congrès de la SBPMef d’août 2010 à Dinant.

Yves Delhaye et Hugues Vermeiren ont prévu d’y tenir trois ateliers  $\LaTeX$  orientés “graphisme” : inclusion d’images, Tikz “de base” et utilisation avancée de Tikz’.

Ceci a servi de “prétexte” à un remaniement et une réécriture des parties de cet ouvrage concernant le graphisme : inclusion d’images externes et motivation du choix des formats d’images, référencements des images dans le document, primitives de base de Tikz et production de graphes de fonctions sous Tikz.

Le succès de la formation IFC du 19 janvier 2010 (29 participants sur 30 enchantés et prêts à suivre une “suite”) a motivé ce travail sur le graphisme.

Nouveauté, également, dans cette édition : US $\TeX$  son installation sous Windows et l’utilisation de  $\LaTeX$  via une clé usb sur “n’importe quel PC”. Ce sujet sera aussi présenté au congrès de la SBPM.

Nous tenons à exprimer ici notre gratitude à René Fritz de l’association GUTenberg qui nous a autorisé à reproduire son travail *Stratégies pour inclure des graphiques dans des documents en  $\LaTeX$*  en le plaçant sous une licence libre. Fort gentiment, sa seule demande a été que l’origine de l’information soit mentionnée, ce que nous faisons ici. L’article original peut être trouvé à l’adresse suivante : <http://www.gutenberg.eu.org/IMG/pdf/Graphiques.pdf>.

**Préface à la troisième édition** Cette version du cours a été spécialement modifiée pour la formation IFC du 19 janvier 2010.

L’IFC a donné son accord pour ces ateliers soient présentés comme formation inter-réseau le 19 janvier 2010. L’ouvrage sera gracieusement remis aux participants.

**Préface à la deuxième édition** Ce cours a été rédigé à l’origine par Jean-Côme CHARPENTIER. Le document original peut être trouvé à l’adresse suivante :

<http://melusine.eu.org/syracuse/texpng/jcc/camille.tex>

L’équipe  $\LaTeX$  de l’Unité de Recherche en Enseignement des Mathématiques de L’Université Libre de Bruxelles (UREM-ULB) œuvre à promouvoir l’utilisation de  $\LaTeX$  parmi les enseignants de Mathématique.

Yves DELHAYE s’est proposé d’organiser des ateliers  $\LaTeX$  pour les membres de l’UREM.

Le présent ouvrage dans sa version “UREM” n’a pas la prétention d’être un cours définitif sur  $\LaTeX$ , mais se veut surtout un support de cours.

Des transparents produits en “Beamer” accompagnent les ateliers et peuvent être téléchargés sur le site de l’UREM.

La contribution d’ Yves Delhaye a été de restructurer le document existant.

Voici les différents points modifiés :

- Le code source du document de départ était dans un seul fichier. Le document a été “éclaté” en différents fichiers répartis dans des dossiers séparés.
- Cette séparation en différents fichiers a permis de restructurer l’ordre des chapitres et sections dans un ordre qui convenait mieux à nos desseins.<sup>1</sup>
- Le code source a été modifié pour produire aussi bien un livre ou un article que des présentations Beamer.
- La volonté d’obtenir des présentations Beamer séparées pour chaque atelier justifiait le découpage en différents fichiers.

---

1. Ainsi, les commandes de modification de taille des polices de caractère ont été volontairement postposées. Nous avons voulu décourager les “mauvais réflexes” dûs aux traitements de texte classiques. Ces mauvaises habitudes encouragent trop à se concentrer sur la forme et pas assez sur le contenu.

- La classe Beamer n'aimant pas les `\paragraph{}`, ceux-ci sont dans un premier temps remplacés par des `\subsubsection{}` puis par des `\frame{}`<sup>2</sup>.
- Dans le même ordre d'idée, l'environnement "*SideBySide*" de la librairie "*fvr-b-ex*" a été remplacé par l'environnement "*LTExample*" de la librairie "*showexpl*" que "beamer" supporte beaucoup mieux.
- L'encodage du document de départ était en "iso-8859-1" (aussi connu sous le doux nom de "iso-latin"). les ordinateurs modernes (post 2007) utilisant plutôt l' "UTF-8". la compilation posait beaucoup de problèmes : Les accents étaient modifiés en signe "cabalistiques" divers. Tous les caractères accentués furent transformés selon les macros  $\LaTeX$ .
- Les installations de MikTeX et de TeXnicCenter ont été ajoutées.
- De nombreuses copies d'écran de ces installations sont jointes aux documents.
- Un chapitre sur les IDE a été ajouté.
- Nous avons créé des exercices sur les erreurs de compilation. Ces problèmes pouvant être très décourageants pour les débutants.
- Nous nous sommes, en général, efforcé de joindre plus d'illustrations et d'exemples au document d'origine.
- Nous avons développé l'introduction pour plus expliquer les motivations de ce cours.

Ceci décrit l'état des lieux fin janvier 2009.

En projet, pour les prochaines versions :

- Des exemples de graphisme autres que "pstricks" sont introduits et plus expliqués,
- des explications sur les codes d'erreurs,
- d'autres extensions utiles,
- du texte pour "lier la sauce" des illustrations d'installation,
- des exercices,
- ...

Tout ceci existe déjà mais n'a pas encore été ajouté au livre car encore trop "brut de décoffrage!"

---

2. Les frames créent des paragraphes dans les livres et articles





# **Première partie**

## **Premier atelier : prise en main**



# Chapitre 1

## Présentation de $\LaTeX$

### Sommaire

---

<b>1.1</b>	<b><math>\TeX</math>, <math>\LaTeX</math> et logiciel libre</b>	<b>4</b>
1.1.1	$\LaTeX$ pour le néophyte	4
1.1.2	Historique	6
1.1.3	Que signifie logiciel libre ?	8
<b>1.2</b>	<b>Avantages et inconvénients de <math>\LaTeX</math></b>	<b>9</b>
1.2.1	Inconvénients ou dix bonnes raisons de ne pas utiliser $\LaTeX$ .	9
1.2.2	Avantages ou dix bonnes raisons d'utiliser $\LaTeX$ .	9

---

## 1.1 $\TeX$ , $\LaTeX$ et logiciel libre

### 1.1.1 $\LaTeX$ pour le néophyte

#### Présentation pour néophyte

Dans l'historique nous expliquerons plus en détails, l'origine de  $\LaTeX$ . Ici, nous expliquerons rapidement, ce qu'est  $\LaTeX$ .

$\LaTeX$  est un "traitement de texte" destiné préférentiellement à la production de documents mathématiques et scientifiques. "Traitement de texte" est mis entre guillemets car il s'agit plutôt de tout un environnement de travail exigeant mais qui permet, une fois maîtrisé, la production de documents de grande qualité typographique tout en permettant de se concentrer sur la structure et le contenu du document.

#### Pourquoi $\LaTeX$ ?

$\LaTeX$  présente une série d'avantages qui le rende incontournable par rapport aux traitements de texte classiques :

- la production de formules mathématiques de grande qualité,

$$\tilde{f}(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-x^2} e^{-ikx} dx = \frac{1}{2\sqrt{\pi}} e^{-\frac{1}{4}k^2}$$

- une gestion irréprochable de toute la structuration du document :
  - de la table des matières,
  - des numérotations,
  - de la pagination,
  - des références croisées,
  - des bibliographies,
- des milliers d'extensions qui en étendent les possibilités.<sup>1</sup>

Le grand " plus " de  $\LaTeX$ , c'est que TOUT est programmable et donc " automatisable ". Mais cet avantage immense fait que  $\LaTeX$  n'est pas facile à aborder sans aide.

#### Motivation

$\LaTeX$  est un standard de fait dans la production académique de textes scientifiques : Les grandes revues scientifiques de mathématiques et de physique n'acceptent quasiment pas de document dans d'autres formats! Mais  $\LaTeX$  n'est pas limité au seul monde académique. En France, les questionnaires de bacs de mathématiques et de physique sont de plus en plus rédigés en  $\LaTeX$ . La SBPMef<sup>2</sup> produit désormais sa revue "SBPM-Infor" en  $\LaTeX$ . Les documents et sites pédagogiques en  $\LaTeX$  se multiplient. Et si, au premier abord, l'emploi de  $\LaTeX$  peut sembler difficile, la puissance et la souplesse qu'il octroie en font un outil dont on ne veut plus se passer. Bref,  $\LaTeX$ , c'est la perfection dans le traitement de texte.

#### $\LaTeX$ ?

Le nom  $\LaTeX$  fait référence simultanément à :

- un ensemble de programmes<sup>3</sup> :

---

1. pour écrire des formules de chimie organique, pour les partitions musicales, pour les tableaux de variations, pour les diagrammes de jeux (échecs, bridge, ...), pour faire des QCMs, des solutionnaires d'examen, ...

2. Société Belge des Professeurs de Mathématique

3. Ces logiciels sont des compilateurs.

- L<sup>A</sup>T<sub>E</sub>X,
- PDFL<sup>A</sup>T<sub>E</sub>X,
- XeT<sub>E</sub>X,
- ...
- un langage de programmation <sup>4</sup>,
- d'autres logiciels qui forment une galaxie d'utilitaires autour du couple "compilateur-code source" :
  - dvips,
  - ps2pdf,
  - L<sup>A</sup>T<sub>E</sub>X2html,
  - ...

### Code source, compilation ...

Le "cycle de production" d'un document L<sup>A</sup>T<sub>E</sub>X est le suivant :

- rédaction d'un fichier en code source,
- compilation,
- déboguage,
- soupir de soulagement!
- Et on recommence!

Sans rire, produire un document L<sup>A</sup>T<sub>E</sub>X implique toujours au minimum les deux premiers points et presque toujours les deux suivants. Et c'est une bonne pratique, d'avancer à petits pas et donc de "boucler" régulièrement via le cinquième point.

### Code ... quoi???

Code source <sup>5</sup>, en informatique, signifie un fichier texte dans le quel on écrit des commandes dans un langage de programmation.

Quand on "fait du L<sup>A</sup>T<sub>E</sub>X", l'activité créative consiste à écrire de tels fichiers.

### Compile à qui???

Le compilateur est le programme qui lit le code source et qui, en suivant des règles propres au langage de programmation, le transforme en ... autre chose. Le plus souvent, c'est un fichier exécutable <sup>6</sup> qui est produit.

Quand on "fait du L<sup>A</sup>T<sub>E</sub>X", au vingt et unième siècle, cet "autre chose" est le plus souvent du PDF, mais aussi du PostScript, voire de l'HTML. Parfois on passe par plusieurs étapes intermédiaires, parfois pas.

### Le cycle de production L<sup>A</sup>T<sub>E</sub>X

En résumé, le cycle de production L<sup>A</sup>T<sub>E</sub>X consiste, à partie d'un fichier dont l'extension est ".tex", à produire du pdf.

---

4. L<sup>A</sup>T<sub>E</sub>X est un langage compilé!

5. ou fichier source ou fichier de code, ou encore parfois "le code" ou "le source"

6. un logiciel

Ce qui peut se résumer via le schéma suivant :

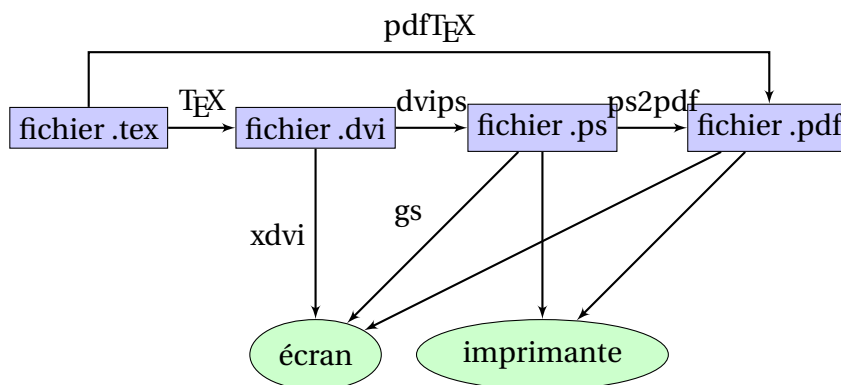


FIGURE 1.1 – Schéma de production de documents en  $\LaTeX$

Il est donc possible de produire directement du pdf via “pdf $\TeX$ ” ou “pdf $\LaTeX$ ” mais il est possible et parfois nécessaire de passer par la production de fichiers intermédiaires en “dvi” via  $\LaTeX$  puis de passer au pdf via du Postscript. Tout ceci pouvant être envoyé à une imprimante ou visualisé à l’écran.

## 1.1.2 Historique

### Pourquoi faire l’historique de $\LaTeX$ ?

Parce qu’ainsi nous pourrions mieux appréhender les avantages de  $\LaTeX$ .

Aussi parce qu’avoir du contexte sur un sujet, aide à mieux le comprendre.



FIGURE 1.2 – Richard Stallman

Dans les années 1970, Richard STALLMAN a commencé à travailler au laboratoire d’intelligence artificielle du MIT. À l’époque, le laboratoire pouvait être qualifié de communauté dans laquelle le logiciel était partagé sans se poser de question (partage qui avait lieu également avec l’extérieur). Cette équipe avait confectionné le système d’exploitation des gros ordinateurs de l’université. Au début des années 1980, l’université a renouvelé son parc informatique et a acquis des ordinateurs disposant d’un système propriétaire. La communauté n’a pas survécu à cela et les personnes sont parties chacune de leur côté. Richard STALLMAN était profondément attaché à la notion de liberté et de partage et a moins bien supporté cet état de fait que les autres. Il a alors décidé de reconstruire tout un système d’exploitation multi-plate-forme (c’est-à-dire pouvant fonctionner sur de multiples ordinateurs différents) en reprenant les idées de liberté : ce projet s’est appelé GNU car il devait se fonder sur le système Unix (GNU signifie GNU is Not Unix, STALLMAN aime beaucoup les acronymes récursifs !).

Cette décision est lourde de conséquences : afin de mener son projet à bien, il va devoir démissionner du MIT en 1984 pour éviter que son travail ne tombe automatiquement dans le domaine du logiciel propriétaire. En fait, c’est un de ces premiers programmes — Emacs, un traitement de texte — qui va le faire connaître et attirer vers lui une nouvelle communauté de développeurs. Le projet va prendre beaucoup d’ampleur. À côté du développement proprement dit, Richard STALLMAN va devoir mettre sur pied une structure juridique permettant de protéger les logiciels libres. En 1991, avec l’arrivée de Linux, le couple GNU/Linux peut enfin proposer un système complet entièrement libre.



FIGURE 1.3 – Don Knuth

### Knuth et T<sub>E</sub>X

Donald KNUTH, un informaticien de renommée mondiale, a entrepris la rédaction d'une véritable bible de la programmation : *The Art of Computer Programming*. Au début, l'ouvrage est composé de façon traditionnelle avec des caractères en plomb et tout allait très bien.

Puis, au cours des années 70, les premières photocomposeuses apparaissent et la qualité devient trop médiocre aux yeux de KNUTH.

Il décide alors de concevoir un logiciel permettant de composer des textes avec une qualité professionnelle. Dès le départ, il se fixe comme but d'arriver à un produit qui devra être parfait (ou presque) et qui devra le rester au cours du temps !

Ce travail va prendre un certain temps et, après la première version sortie à la fin des années 1970, la correction de quelques erreurs amène à la version finale au tout début des années 1980. Cette dernière version a été définitivement gelée et fonctionne ainsi depuis plus de vingt ans sans que le besoin impérieux d'ajouter des fonctionnalités se fasse sentir.

Lorsque T<sub>E</sub>X a été rendu public, de nombreux utilisateurs l'ont enrichi et, au cours du temps, il est devenu de plus en plus facile de réaliser de multiples types de compositions avec cet outil. Citons par exemple la réalisation de formules ou molécules chimiques, de diagrammes physiques, de partitions musicales, de textes arabes, etc. Une autre voie a été de doter les distributions de programmes annexes permettant de gérer le PostScript, le PDF, le langage HTML, etc.

### Lamport et L<sup>A</sup>T<sub>E</sub>X

Dès le début des années 1980, un de ces développeurs, Leslie LAMPORT, a créé une surcouche à T<sub>E</sub>X – L<sup>A</sup>T<sub>E</sub>X –

permettant une approche plus intuitive du langage et qui en a standardisé l'utilisation.

Le principe qui a guidé l'élaboration de L<sup>A</sup>T<sub>E</sub>X était de faire en sorte que l'utilisateur se concentre plus sur la structure du document que sur les détails de composition.

L<sup>A</sup>T<sub>E</sub>X propose pour cela toute une gamme de commandes de haut niveau construites à partir de T<sub>E</sub>X mais dont les détails n'ont pas obligatoirement besoin d'être connus de l'utilisateur. Actuellement, il y a beaucoup plus d'utilisateurs de L<sup>A</sup>T<sub>E</sub>X que de T<sub>E</sub>X, cela a pour conséquence que de nombreuses extensions du langage ont été développées en L<sup>A</sup>T<sub>E</sub>X et qu'elles ne fonctionnent pas toutes sous T<sub>E</sub>X.

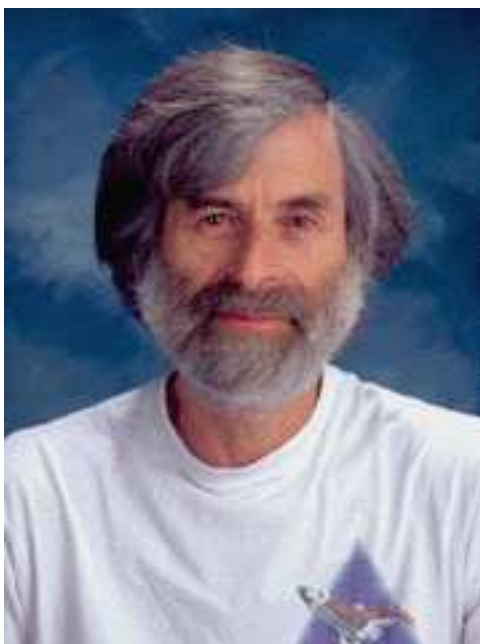


FIGURE 1.4 – Leslie Lamport

### 1.1.3 Que signifie logiciel libre ?

Lorsque  $\TeX$  fut au point, KNUTH en confia la garde à l’American Mathematical Society et livra le source du programme au monde du logiciel libre. Il est peut-être utile de faire ici une petite mise au point sur ce que signifie la notion de logiciel libre à propos de laquelle beaucoup de bêtises ont été dites ; la floraison de termes plus ou moins synonymes et/ou homonymes n’a certainement pas contribué à rendre les choses facilement intelligibles.

En premier lieu, il faut absolument distinguer la notion de logiciel gratuit et de logiciel libre. Ce n’est pas parce qu’un logiciel est gratuit qu’il doit être considéré comme un logiciel libre et un logiciel libre n’est pas nécessairement gratuit.

#### Logiciel libre

Pour clarifier un peu les choses dans une jungle relativement inextricable, j’appellerai “logiciel libre” un logiciel distribué sous la licence GPL (General Public Licence). En résumé, voici les principaux points de cette licence :

- tout le monde peut utiliser le logiciel ;
- le source du programme doit obligatoirement être fourni avec la distribution contenant le programme ;
- le point précédent est là pour permettre à quiconque de modifier le logiciel, soit pour l’adapter à des besoins particuliers, soit pour l’améliorer ;
- on a le droit de redistribuer les copies dans la mesure où on fournit également le source du programme.

Si un logiciel répond à ces exigences, il peut être qualifié de logiciel libre sinon, ce n’est pas un logiciel libre. Par exemple, une montagne de logiciels freeware ne sont pas des logiciels libres (source non disponible, interdiction de modifier le programme sans l’accord du propriétaire, ...).

On remarquera qu’il n’est jamais fait mention explicite d’argent. S’il est précisé que tout le monde peut redistribuer des copies, il n’est absolument pas interdit de les faire payer. En pratique, c’est d’ailleurs ce qui se passe pour les distributions occupant beaucoup de place mémoire et qui seraient très pénibles de rapatrier via internet.



Le projet GNU a inventé une licence couvrant le logiciel libre qu'il a appelée "copyleft". Elle protège le logiciel en empêchant, par exemple, une entreprise de modifier le programme et de déclarer alors que ce programme modifié n'est plus libre. En effet, il y a deux obligations pour le logiciel libre. La première est qu'une version modifiée d'un logiciel libre est ipso facto un logiciel libre, couverte par la même licence. La seconde est que la prise de cette licence est irrévocable : le développeur d'un programme donne irrémédiablement son produit à la communauté.

## 1.2 Avantages et inconvénients de $\LaTeX$

### 1.2.1 Inconvénients ou dix bonnes raisons de ne pas utiliser $\LaTeX$ .

Je vais avoir quelques difficultés à être impartial car j'ai du mal à voir des inconvénients à  $\LaTeX$  ! En fait, je reprendrai les arguments avancés contre  $\LaTeX$  et qui sont tout à fait légitimes même si je ne partage pas ces idées :

#### Les moins

1. Il s'agit d'un langage de programmation. Pour ceux qui sont habitués à travailler sous un traitement de texte plus classique, le mieux à faire est d'oublier à peu près tout ce qu'ils ont appris : la suite sera plus facile !

Ceci implique de se plonger dans une logique de programmeur. Il faudra savoir ce qu'est un code source, apprendre à compiler ce code, gérer les erreurs de compilation. Ce qui implique d'apprendre la syntaxe du langage et les outils associés.

2.  $\LaTeX$  n'est pas "Wysiwyg". À ma connaissance, il n'y a qu'une seule distribution (payante et qui ne fonctionne que sous Macintosh) permettant de voir le résultat au fur et à mesure de la frappe.

Il y a également le logiciel Lyx qui n'est pas vraiment Wysiwyg mais qui s'en rapproche, on peut aussi citer le logiciel Scientific Workplace.

Je déconseille fortement d'utiliser ces logiciels qui génèrent un code très lourd et difficilement manipulable.

Avec une distribution classique, il faut taper le source (décrire le document), le compiler puis visualiser ou imprimer le fichier de sortie.

3. La prise en main du logiciel demande un effort non négligeable en raison du nombre de commandes à connaître. Les non anglophones seront particulièrement désavantagés !
4. La multiplicité des options est parfois déroutante.
5. Certaines mises en forme sont assez lourdes à gérer. En particulier, on pourra citer les tableaux.
6. La documentation n'est pas centralisée, et donc parfois difficile à trouver.
7. La gestion des erreurs peut-être très décourageante, surtout pour un débutant.
8. Je n'arrive pas vraiment à en trouver d'autre!!

### 1.2.2 Avantages ou dix bonnes raisons d'utiliser $\LaTeX$ .

Comment arrive-t-on à concevoir un logiciel qui est utilisé depuis vingt ans sans que des mises à jour ne soient nécessaires ? Je ne peux pas expliquer toutes les idées de l'auteur de  $\TeX$  dans ce manuel et je ne les connais d'ailleurs pas toutes. Pour en avoir une petite idée, voici quelques caractéristiques de  $\LaTeX$  qui font sa force :

## Les plus

1. Il s'agit d'un langage de programmation. Le contrôle ainsi donné sur le document est très grand. On peut programmer des macros, faire appel à des programmes externes (graphique, algèbre,...).
2.  $\LaTeX$  n'est pas "Wysiwyg". Le Wysiwyg est plus un défaut qu'une qualité : elle oblige l'utilisateur à faire plus attention à la forme physique qu'à la forme logique d'un document. C'est également la porte ouverte à tout un tas de mauvaises habitudes typographiques. Pour clore le sujet, je pense que pour une lettre à un ami ou une feuille d'exercices destinée à des élèves, ce n'est pas excessivement important, en revanche, dès que le travail commence à être un peu plus conséquent, la supériorité de  $\LaTeX$  (ou de  $\TeX$ ) est manifeste. Dernier reproche fait à  $\LaTeX$  : les documents demandent plus de temps à être tapés. C'est évident lorsqu'on ne connaît pas encore bien le logiciel ; c'est complètement faux dès qu'on commence à en avoir une certaine habitude. Ainsi, la rédaction de formules mathématiques complexes devient étonnement rapide.
3. La qualité typographique est impressionnante. Il faut faire des efforts pour obtenir un document peu esthétique.
4. Le source étant un fichier texte, il n'y a aucun problème de format propriétaire. D'autre part, la taille des fichiers est très petite. (1 000 fois plus qu'un texte contenant beaucoup de formules mathématiques fait sous Word) Ce qui constitue un avantage énorme en cas de travail via internet ou d'échange de fichiers sur d'autres supports.
5.  $\LaTeX$  a été porté sur toutes les plates-formes possibles et imaginables. En fait, je ne connais aucune machine avec laquelle il soit impossible de travailler avec  $\LaTeX$ .
6.  $\LaTeX$  est libre.
7.  $\TeX$  a été très rapidement gelé. Ceci a pour effet que tous les documents sont lisibles sur n'importe quelle installation. Par exemple, le manuel de  $\TeX$  que KNUTH a écrit en 1982 est toujours lisible à l'heure actuelle. Inversement, un fichier écrit sur la dernière distribution pourra être compilé correctement sur une machine disposant d'une très vieille distribution.
8. Le fichier de sortie créé par  $\LaTeX$  est indépendant du périphérique. Cela fait que, hormis les problèmes de qualité, le résultat sera strictement identique quel que soit l'imprimante utilisée, que se soit une jet d'encre à 180 dpi datant de 10 ans ou la dernière laser couleur à 2 400 dpi.
9. Le fichier construit avec  $\LaTeX$  place les différents éléments avec une unité de longueur appelée le point d'échelle qui correspond exactement à  $2^{-16}$  point typographique, ce qui nous donne  $5,36 \times 10^{-6}$  mm. À titre indicatif, la plus petite longueur d'onde visible (violet) vaut environ  $400 \times 10^{-6}$  mm. Autrement dit, même en concevant des périphériques d'impression de plus en plus perfectionnés,  $\LaTeX$  sera toujours capable de gérer des différences de tailles très largement invisibles à l'œil ; tout au plus, pourrait-on regretter que  $\TeX$  devienne insuffisant pour fabriquer des microfilms lisibles au microscope électronique !
10. Il y a encore au moins 10 000 bonnes raisons... exercice : inscrivez les dans un document  $\LaTeX$ !

# Chapitre 2

## Distributions $\text{\LaTeX}$ et installation

### Sommaire

---

<b>2.1</b>	<b>Qu'est ce qu'une "Distribution"?</b>	<b>12</b>
2.1.1	Distribution Mik $\text{\TeX}$	12
2.1.2	Distribution $\text{\TeX}$ live	12
<b>2.2</b>	<b>EDI : Environnement de Développement Intégré</b>	<b>13</b>
2.2.1	Kile	13
2.2.2	$\text{\TeX}$ Maker	13
2.2.3	$\text{\TeX}$ nicenter	13
2.2.4	$\text{\TeX}$ Shell	13
2.2.5	$\text{\TeX}$ Shop	13

---

## 2.1 Qu'est ce qu'une "Distribution" ?

### Distribution : $\TeX$ + Metafont + ...

Une distribution ne contient pas que le seul programme  $\TeX$ . Tout seul,  $\TeX$  ne servirait pas à grand chose car, dans les caractéristiques originales du logiciel, KNUTH a séparé le placement des différents éléments dans la page et la forme de ces différents éléments. Par exemple, le présent paragraphe commence par la lettre " U ". Cette lettre doit être placée à un endroit très précis sur la page. C'est le rôle de  $\TeX$  de le déterminer en se servant de quelques renseignements succincts sur cette lettre (hauteur, profondeur, largeur et divers renseignements un peu plus exotiques). En revanche,  $\TeX$  ne sait absolument pas que cette lettre se dessine avec un trait courbe plus épais à gauche qu'à droite, surmonté d'empattements variés. Le dessin proprement dit de la lettre est confié à un autre programme (METAFONT), également créé par KNUTH en même temps que  $\TeX$ . Ces deux programmes constituent le tandem de base de la distribution.

Que trouvons-nous de plus dans une distribution ?

### $\TeX$ + Metafont + ... et un raton laveur

En fait, tout plein de choses :

- des fichiers décrivant les fontes<sup>1</sup> de caractères,
- des programmes permettant d'organiser
  - les index,
  - les bibliographies,
- des bibliothèques de commandes préprogrammées,
- des formats (fichiers décrivant les grandes lignes de la présentation d'un document),
- différents utilitaires,
- d'autres choses plus ou moins exotiques.

Le tout fait plusieurs centaines de fichiers et une distribution classique peut très bien atteindre 100, 200 ou 300 Mo.

L'installation complète de la distribution  $\TeX$  live occupe 1,5 Go du disque dur.

### 2.1.1 Distribution Mik $\TeX$

#### Mik $\TeX$ : un choix pragmatique

Durant ce stage, nous utiliserons la distribution Mik $\TeX$ .

Mik $\TeX$  n'est plus activement développée, pour certaines extensions "pointues" ceci peut poser problème.

Windows étant malheureusement le système d'exploitation le plus répandu et pour faciliter la vie de ceux qui voudraient l'installer sur leur ordinateur, nous utiliserons cette distribution prévue pour Windows et relativement facile à installer.

Les membres de l'UREM peuvent demander de l'aide à Gérald Houart qui se chargera d'ailleurs d'aider ceux qui le désirent à l'installer sur leur ordinateur lors de l'atelier *3et $\frac{1}{2}$* .

### 2.1.2 Distribution $\TeX$ live

#### $\TeX$ live : Il y a moins bien mais c'est plus cher!

---

1. ou police de caractères

Personnellement, j'utilise la distribution  $\TeX$ live distribuée, entre autres, par l'association GUTenberg.

Il s'agit d'une distribution classique du système  $\TeX$  permettant une installation sous Unix, Windows ou Macintosh.

La procédure d'installation sous Windows est décrite plus loin.

Une installation sous Linux est beaucoup plus simple : pour ceux qui dispose d'un environnement Linux, en général, l'installation d'un système  $\TeX$  se fait lors de l'installation du système d'exploitation lui-même.

## 2.2 EDI : Environnement de Développement Intégré

### EDI : motivation

Une distribution  $\LaTeX$  est donc une collection ou une fédération d'outils divers. On peut faire fonctionner ces outils isolément mais il est souvent pratique d'avoir une interface centralisant les multiples possibilités :

- édition du code,
- messages de compilation,
- gestion des documents répartis sur plusieurs fichiers en projet,
- raccourcis pour les commandes  $\LaTeX$ ,
- et pour visualiser le résultat...

Accompagnant souvent une distribution, vous trouverez donc un programme remplissant ces fonctions : un EDI ou Environnement de Développement Intégré.

Sur le CDROM ou le DVD d'une distribution, vous avez même parfois le choix entre plusieurs EDI. A vous de les essayer et de trouver celui qui vous convient le mieux.

En voici quelques uns, tous excellents et gratuits.

### 2.2.1 Kile

N'existant malheureusement que sous Linux, mon EDI préféré se nomme "Kile"!

### 2.2.2 $\TeX$ Maker

$\TeX$ Maker existe sous Windows et sous Linux. Pas mal du tout.

### 2.2.3 $\TeX$ nicenter

$\TeX$ nicenter est associé à Mik $\TeX$ .

### 2.2.4 $\TeX$ Shell

L'EDI  $\TeX$ Shell se trouve sur la distribution  $\TeX$ Live.

### 2.2.5 $\TeX$ Shop

L'EDI  $\TeX$ Shop est l'EDI de référence sur Macintosh.



# Chapitre 3

## Premiers pas

### Sommaire

---

<b>3.1 Premiers documents</b> . . . . .	<b>16</b>
3.1.1 Structure d'un fichier $\LaTeX$ . . . . .	16
3.1.2 Etude du résultat . . . . .	22
3.1.3 Deuxième exemple . . . . .	26
3.1.4 Troisième exemple . . . . .	27
<b>3.2 EDI : prise en main de TexnicCenter et premières compilations</b> . . . . .	<b>28</b>
3.2.1 Les différentes sous fenêtres de $\TeX$ nicCenter . . . . .	28
3.2.2 Compilation de l'exemple . . . . .	32
3.2.3 Autres EDI . . . . .	35
<b>3.3 Erreurs de compilation</b> . . . . .	<b>37</b>

---

## 3.1 Premiers documents

Un des aspects les plus déroutant de  $\LaTeX$ , lorsqu'on est habitué à des traitements de texte de type Word, est qu'il ne s'agit pas d'un environnement Wysiwyg (What You See Is What You Get : ce qu'on voit est ce qu'on obtient). Comme c'est le premier reproche fait à  $\LaTeX$ , une petite mise au point s'impose. Tout d'abord, le Wysiwyg n'est qu'à peu près Wysiwyg : l'affichage écran ne peut de toute façon pas rendre toutes les nuances possibles au niveau d'une impression, même sur une imprimante actuelle très bas de gamme. D'autre part, il faut se persuader que la prétendue supériorité du Wysiwyg n'est en fait qu'une histoire d'habitude. Passer de Word à  $\LaTeX$  demande donc de perdre certaines habitudes, ce qui n'est jamais agréable au départ.

Produire un document en utilisant  $\LaTeX$  se fait en deux étapes. La première consiste à taper le source du document et la seconde consiste à le compiler.

On retrouve donc les deux étapes essentielles de beaucoup de langages de programmation.

En théorie, il faut donc un éditeur de texte pour produire le source puis un appel à un programme qui va se charger de traduire le source en un fichier décrivant le document d'une façon facilement compréhensible pour l'ordinateur (donc totalement incompréhensible pour l'humain).

Avec un EDI comme Kile ou  $\TeX$ Nicenter, les choses sont un peu plus simples car cet éditeur de texte propose, via des menus et boutons, d'automatiser les différentes étapes de compilation.

Je ne rentrerai donc pas dans les détails techniques en renvoyant le stagiaire curieux vers les livres cités en bibliographie.

En fait, lorsqu'il est indiqué qu'un document se fait en deux étapes : saisie du source puis compilation, on laisse de côté une phase quelquefois pénible, voire très irritante qui est celle de la correction d'erreurs. Avec l'expérience, le temps passé à traquer les erreurs diminue mais il faut se préparer, les premiers temps, à quelques ennuis, surtout lorsqu'on demandera des choses très spéciales à  $\LaTeX$ .

### 3.1.1 Structure d'un fichier $\LaTeX$

Voici le code source d'un document relativement minimal. Les explications suivent :

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[frenchb]{babel}
\begin{document}
  Mes premiers pas avec  $\LaTeX$  sont un peu émouvants. Pour les premiers
  exemples, il faudra taper exactement ce qui est proposé, en faisant bien
  attention ! Une fois la compilation effectuée, on peut
  \begin{itemize}
    \item visualiser le document à l'écran ;
    \item imprimer le document ;
    \item convertir le document en fichier PostScript ;
    \item et plein d'autres choses.
  \end{itemize}
\end{document}
c'est fini.
```

Le résultat de la compilation est montré page ci-contre.



Mes premiers pas avec L<sup>A</sup>T<sub>E</sub>X sont un peu émouvants. Pour les premiers exemples, il faudra taper exactement ce qui est proposé, en faisant bien attention ! Une fois la compilation effectuée, on peut

- visualiser le document à l'écran ;
- imprimer le document ;
- convertir le document en fichier PostScript ;
- et plein d'autres choses.

Etudions en détail ce code :

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[frenchb]{babel}
\begin{document}
  Mes premiers pas avec \LaTeX{} sont un peu émouvants. Pour les premiers
  exemples, il faudra taper exactement ce qui est proposé, en faisant bien
  attention ! Une fois la compilation effectuée, on peut
  \begin{itemize}
    \item visualiser le document à l'écran ;
    \item imprimer le document ;
    \item convertir le document en fichier PostScript ;
    \item et plein d'autres choses.
  \end{itemize}
\end{document}
c'est fini.
```

### 3.1.1.1 Entête, préambule ou header

#### Entête : mettre le décor en place

Ceux d'entre vous qui ont déjà programmé de l'html ou un langage compilé comme le C ou le Pascal seront familiers avec cette notion. Avant de commencer le travail, il faut mettre des choses en place :

- Dans la déclaration prévenir  $\LaTeX$  que le document est un livre, un article, une lettre une présentation ou un poster.
- Les appels aux bibliothèques de commandes servent à étendre les possibilités de  $\LaTeX$  sans devoir réinveter la roue à tout moment.
- Nous verrons plus avant que c'est ici que nous pouvons définir nos propres nouvelles commandes : macros...
- C'est aussi ici que nous modifierons la mise en page globale du document si nous le désirons.

#### Déclaration ou classe : livre, rapport, article, ...

Un document  $\LaTeX$  commence obligatoirement par la **déclaration**

```
\documentclass{classe_document}
```

où *classe\_document* est un des mots-clés suivants :

- book ;
- report ;
- article ;
- slides ;
- beamer ;
- letter.

Il s'agit de la *classe* du document. Théoriquement, book est destiné à taper le source d'un livre, report d'un rapport (ouvrage moins important qu'un livre), article d'un ... article (document encore moins important qu'un rapport), slides ou beamer de transparents et letter de lettres. Les trois dernières classes sont un peu spéciales et nous n'en parlerons pas. En revanche, les trois premières sont d'un usage courant.

**bibliothèque de commandes, extension ou package**

Les deux lignes suivantes commencent par `\usepackage`.

Il s'agit d'un appel à une bibliothèque de commandes appelée *extension* ou *package* permettant d'automatiser certaines actions.

La syntaxe générale de cette commande est :

```
\usepackage [options] {nom_extension}
```

Dans notre exemple, on trouve :

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

qui permet de taper directement les lettres accentuées

( $\TeX$  est un logiciel américain fait à une époque où le codage ASCII ne prenait pas en compte les lettres dites européennes).

On trouve également :

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

qui permet de se plier aux règles de la typographie française<sup>1</sup>. Nous allons immédiatement voir en quoi cela consiste.

**3.1.1.2 Le document****Le document et ...**

La ligne suivante dit :

```
\begin{document}
```

ce qui indique que le document proprement dit va débiter après cette ligne. Le document prend fin avec la commande :

```
\end{document}
```

C'est pourquoi le texte " c'est fini " (dernière ligne du source) n'a pas été pris en compte dans le document final.

**... rien que le document**


---

1. En fait, il existe deux façons d'indiquer qu'on est français : celle utilisée ici et l'appel au package `french`. En ce qui concerne les points abordés dans ce manuel, cela ne fera aucune différence.

Mes premiers pas avec L<sup>A</sup>T<sub>E</sub>X sont un peu émouvants. Pour les premiers exemples, il faudra taper exactement ce qui est proposé, en faisant bien attention ! Une fois la compilation effectuée, on peut

- visualiser le document à l'écran ;
- imprimer le document ;
- convertir le document en fichier PostScript ;
- et plein d'autres choses.

### 3.1.1.3 En résumé

#### Rappel de vocabulaire

La partie située entre `\documentclass` et `\begin{document}` s'appelle le *préambule*.

Et la partie située entre le `\begin{document}` et le `\end{document}` s'appelle le ... *document*.

Jusqu'au chapitre extension

9 (page 87),

le préambule sera toujours :

```
\documentclass{article}
```

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

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

et nous ne le spécifierons donc plus, ainsi que le couple :

```
\begin{document}
```

```
\end{document}
```

### 3.1.2 Etude du résultat

#### TeX et comment le texte apparaît.

Regardons à nouveau ce qui a été produit.

#### TeX vs. Source

Mes premiers pas avec TeX sont un peu émouvants. Pour les premiers exemples, il faudra taper exactement ce qui est proposé, en faisant bien attention ! Une fois la compilation effectuée, on peut

- visualiser le document à l'écran ;
- imprimer le document ;
- convertir le document en fichier PostScript ;
- et plein d'autres choses.

```

1 Mes premiers pas avec \LaTeX{} sont
2 un peu \’emouvants. Pour les premiers
3 exemples, il faudra taper exactement
4 ce qui est propos\’e, en faisant bien
5 attention ! Une
6 fois la compilation effectu\’ee,
7   on peut
8   \begin{itemize}
9     \item visualiser le document
10    \’a l\’ecran ;
11    \item imprimer le document ;
12    \item convertir le document
13    en fichier PostScript ;
14    \item et plein d’autres choses.
15  \end{itemize}

```

#### 3.1.2.1 Césures

On voit que les coupures de lignes n’ont pas du tout lieu au même endroit que dans le source et que les lignes produites sont justifiées (alignement au niveau des marges gauche et droite).

TeX calcule automatiquement les endroits où doivent se situer les coupures et procède, le cas échéant, aux césures de mots. (comme “ émouvants ” dans l’illustration ci-dessus et “ attention ” notre exemple)

Dans la plupart des cas, ces coupures sont correctes ; on peut forcer les coupures de façon manuelle mais, sauf cas particulier, c’est généralement une très mauvaise idée.

D’autre part, une fin de ligne au niveau du source est comprise comme un espace. Ce qui fait que les mots “ bien ” et “ attention ” sont normalement séparés au niveau de la sortie. Dans l’exemple de départ, il s’agissait des mots “ premiers ” et “ exemples ”.

#### 3.1.2.2 Paragraphes et indentation

On notera également que la première ligne du paragraphe est indentée (retrait vers la droite de la première ligne) alors que le source ne demandait rien. Il s’agit de la présentation par défaut des paragraphes (en typographie française). On verra au chapitre “ commandes de base : Fontes, ... ” comment spécifier les formes que devront avoir les paragraphes.

#### 3.1.2.3 Macro commandes

##### MACRO

Le terme “ \LaTeX{} ”, dans le source, a été traduit par “ TeX ” dans le document. De façon générale, tout ce qui commence par une *contre-oblique* est une *macro* (macro-instruction, une sorte de commande en fait). Ici, le nom de cette commande est LaTeX (notez l’emploi des majuscules

et minuscules, par exemple, la commande `latex` n'existe pas). Il s'agit d'une suite d'instructions permettant d'obtenir le curieux logo avec ses lettres qui ne sont pas alignées verticalement.

### 3.1.2.4 Listes

#### Environnements

Le gros bloc :

```
\begin{itemize}
  \item visualiser le document à l'écran ;
  \item imprimer le document ;
  \item convertir le document en fichier PostScript ;
  \item et plein d'autres choses.
\end{itemize}
```

a été traduit par une liste d'items.

On notera la syntaxe qui permet de déclarer où commence la liste d'items (`\begin{itemize}`) et où elle se termine (`\end{itemize}`).

Ces blocs qui commencent par un `\begin{bidule}` et qui se terminent par un `\end{bidule}` s'appellent des *environnements*.

### 3.1.2.5 Typographie

#### Typographie US et française

Voilà pour la partie visible. Maintenant, regardez avec beaucoup d'attention le point d'exclamation et les points virgules. Avec de bons yeux, on remarque que l'espace située<sup>2</sup> avant la double ponctuation est plus étroite que celle mise après. Il s'agit d'une règle de typographie française, l'espace située avant étant en fait une espace fine insécable. C'est l'appel de `\usepackage[frenchb]{babel}` qui a déclenché automatiquement ce comportement.

#### Tirets

Dans le même registre, chaque item de la liste débute par un tiret long (un tiret demi cadratin pour être précis). Sans la précision `\usepackage[frenchb]{babel}`, le signe au début de chaque item aurait été un rond noir ( $\bullet$ )<sup>3</sup>.

### 3.1.2.6 Marges et format de papier

#### A4 et Letter

On peut remarquer également, et ce n'est pas très esthétique, que les marges sont très importantes et que le numéro de page est situé très haut dans la page. Ce comportement malheureux a deux origines. La taille des marges par défaut est de un pouce (environ 2,54 cm) ; c'est KNUTH qui en a décidé ainsi et KNUTH aime bien les marges importantes. D'autre part, T<sub>E</sub>X a été créé par un américain qui devait écrire des textes sur du papier américain. Or, le format standard outre-atlantique est " letter " alors qu'en France, il s'agit du format A4. Le document a été créé en supposant que le papier était du type letter ce qui explique la position bizarre du numéro de page (il suffit de couper la feuille au format letter et on retrouve quelque chose de plus propre !) Bien évidemment,

2. Les espaces horizontales sont des mots féminins en typographie.

3. Il s'agit d'une règle beaucoup plus controversée. L'auteur du package a cru bon de la mettre !

ce comportement peut être réglé. Pour le moment, on indiquera qu'on travaille sur du papier au format A4 en spécifiant un paramètre optionnel au niveau de la commande `\documentclass`. En fait, j'avais menti précédemment en disant que tous nos documents commenceraient par cette commande ! En réalité, ils commenceront par la commande :

```
\documentclass[a4paper]{classe_de_document}
```

### 3.1.2.7 Ligature

Petite cerise sur le gâteau : examinez avec attention le mot " effectuée " comme il apparaît sur le document et le mot " effectuée " tel qu'il apparaîtrait sur des traitements de texte peu scrupuleux. Dans le premier cas, les deux " f " ont été liés : il s'agit d'une ligature. Un texte correctement composé doit faire les ligatures ff, fi, fl pour ff, fi et fl afin d'éviter les chocs disgracieux entre ces lettres.  $\TeX$  gère également d'autres ligatures permettant d'obtenir les différents types de tirets :

" - " (trait d'union) obtenu avec " - " ;  
 " – " (tiret demi cadratin) obtenu avec " -- " ;  
 " — " (tiret cadratin) obtenu avec " --- " .

### 3.1.2.8 En résumé

Ce premier exemple a permis de présenter le processus général pour obtenir un document et de voir les notions de classe de document, d'extension (ou package), de préambule, de macro (ou commande) et d'environnement.

### 3.1.2.9 Caractères réservés

#### Surprises avec les macros

Un dernier mot pour spécifier ce qu'est exactement un nom de macro et pourquoi les accolades qui suivent  $\LaTeX$  n'ont pas été traduites par des accolades au niveau du document final.

$\TeX$  permet d'écrire n'importe quel caractère au niveau du source (les caractères accentués nécessitant quand même la présence d'un `\usepackage[T1]{fontenc}` au niveau du préambule) sauf les dix caractères suivants qui ont chacun un rôle très particulier.

Une utilisation erronée de ces caractères produira invariablement des erreurs lors de la compilation et des résultats surprenants au niveau de la sortie.

#### Caractères réservés

<code>\</code>	début de nom de macro
<code>%</code>	commentaire
<code>~</code>	espace insécable
<code>{</code>	début de groupe
<code>}</code>	fin de groupe
<code>\$</code>	mode mathématique
<code>_</code>	indice
<code>^</code>	exposant
<code>#</code>	paramètre de macro
<code>&amp;</code>	colonne de tableau



Les cinq symboles du bas ne seront vus que plus tard. Nous allons terminer cette présentation rapide en expliquant sommairement la signification des cinq symboles supérieurs.

### Règles des macros

`\` sert à indiquer une macro, les caractères qui suivent donnant le nom de cette macro.

$\TeX$  a des règles qui peuvent sembler bizarres pour déterminer ce qu'est un nom de macro.

Une série de lettres (majuscules et/ou minuscules) qui se suivent déterminent un nom de macro, un caractère autre qu'une lettre (y compris un des 10 symboles spéciaux) détermine *seul* un nom de macro.

### Exemples des macros

Voici quelques exemples pour bien comprendre :

`\bonjour` est la macro `bonjour` ;

`\\bonjour` est la macro `\` suivie du texte " `bonjour` " ;

`\\bonjour` est la macro `\` suivie de la macro `bonjour` ;

`\_` où `_` désigne un espace est la macro `_` (le nom est constitué par un espace : bizarre n'est-ce pas ?) ;

`\formatA4` est la macro `formatA` suivie du texte " `4` ".

On notera le piège du dernier exemple (les chiffres ne sont pas des lettres).

### Commentaire %

Le caractère `%` introduit un commentaire, c'est-à-dire un texte apparaissant dans le source mais n'ayant aucune influence au niveau du document<sup>4</sup>. Un commentaire se termine à la fin d'une ligne du source (voir l'exemple suivant pour mieux comprendre). Les commentaires sont particulièrement utiles pour mettre des notes explicatives au niveau du source sans que celles-ci n'influencent le résultat.

### Espace insécable

Le caractère `~` permet de placer une espace insécable. Les espaces insécables sont obligatoires entre certains mots. Par exemple entre un prénom et un nom de famille (Yves~Delhaye), entre certaines abréviations et le numéro qui va avec (page~4, Cf. ~6.4, etc.). Pour une liste des situations exigeant une espace insécable, on pourra se reporter à [LEX].

### Groupes

Les caractères `{` et `}` sont très importants sous  $\TeX$ . Ils délimitent des *groupes*. Expliquer en détails ce qu'est un groupe sortirait du cadre de ce stage. Pour simplifier à outrance, disons qu'un groupe délimite une zone à l'intérieur de laquelle on peut obtenir des comportements qui n'existeront plus à l'extérieur de ce groupe. Dans le source on a employé la construction `\LaTeX{}`, c'est-à-dire la macro `LaTeX` suivie d'un groupe vide. Sans ce groupe vide, le source :

Mes premiers pas avec `\LaTeX` sont un peu émouvants

aurait produit la sortie suivante :

Mes premiers pas avec  $\TeX$  sont un peu émouvants

---

4. Ce n'est pas tout à fait vrai mais c'est une bonne approximation de la vérité.

car l'espace qui suit la macro `LaTeX` ne sert qu'à indiquer que le nom de la macro se termine et cet espace disparaît une fois ce rôle terminé. C'est heureux, sinon il serait difficile d'écrire quelque chose comme " Le `LaTeX`book " qui a bien le droit d'exister après tout ! La première idée serait de mettre deux espaces à la suite : le premier pour stopper le nom de macro et le second pour obtenir une espace véritable au niveau de la sortie. Malheureusement cette idée brillante est inopérante à cause d'une règle de `TeX` qui remplace une séquence quelconque d'espaces au niveau du source par un seul espace. De même, les espaces situés en début de ligne dans le source ne sont pas pris en compte (l'exemple suivant utilise ces règles pour permettre une présentation lisible). Par conséquent, dans la syntaxe `\LaTeX{}`, les accolades ne produisent rien (il s'agit d'un groupe vide) mais elles ont quand même le rôle de stopper le nom de la macro, " { " n'étant pas une lettre. Ainsi, l'espace qui suite le groupe vide n'a plus ce rôle de fin de nom de macro et reprend son comportement normal pour désigner une espace au niveau de la sortie (suis-je assez clair ?). L'exemple suivant utilise deux autres façons d'obtenir une espace après une macro.

### 3.1.3 Deuxième exemple

Voici le second exemple qui va tenter d'illustrer tout ce qui vient d'être vu. À partir de maintenant, les sources ne présenteront pas le préambule mais il faudra absolument le spécifier. Le source sera présenté sur la droite de la page, le résultat correspondant étant présenté à gauche en vis-à-vis. Pour faciliter les explications, les lignes du source seront numérotées.

Le second exemple de source `LaTeX` va permettre d'illustrer les points que nous venons de voir. Dans les exemples, nous avons vu la macro dont le nom était constitué uniquement d'un espace. Ce source `LaTeX` en donne une utilisation possible. Cet exemple va présenter les caractéristiques telles que la gestion des espaces, les commentaires, les espaces insécables et une toute petite illustration de ce qu'il est possible de faire avec les groupes.

```

1  Le second exemple de
2  source {\LaTeX} va
3  permettre d'illustrer
4  les points que nous
5  venons de voir. Dans
6  les exemples, nous
7  avons vu la macro dont le
8  nom \’etait constitu\’e uniquement
9  d’un espace. Ce source \LaTeX\ en
10  donne une utilisation possible.
11
12  Cet exemple va pr\’esenter les
13  caract\’eristiques telles que
14  la  gestion  des  espaces,
15  les commentaires, % comme celui-
16  ci
17  les espaces ins\’ecables et
18  une toute petite illustration de
19  ce
    qu’il est possible de faire
    avec
    les groupes.
```

On notera les deux façons supplémentaires d'écrire le mot " `LaTeX` " en mettant une espace à la suite, le fait que le commentaire " comme celui-ci " (ligne 15) n'apparaît pas au niveau de la sortie, et les ribambelles d'espaces au niveau du source qui n'ont donné qu'une seule espace au niveau du document final (lignes 14 à 19).

Une ligne vide au niveau du source indique un saut de paragraphe. Ici, pour des raisons de commodités, les paragraphes ne présentent ni indentation, ni espaces verticaux entre eux : dans un exemple réel, ils seraient présents.

### 3.1.4 Troisième exemple

Le troisième exemple de source  $\LaTeX$  va permettre d'illustrer les points que nous venons de voir. Dans les exemples, nous avons vu la macro dont le nom était constitué uniquement d'un espace. Ce source  $\LaTeX$  en donne une utilisation possible.

```

1 Le troisi\‘eme exemple de
2 source {\LaTeX} va
3 permettre d’illustrer les points
4 que nous venons de voir. Dans
5 les exemples, nous avons vu
6 la macro dont le nom \’etait
7 constitu\’e uniquement d’un espace.
8 Ce
9 source~\LaTeX\ en donne une
   utilisation
   possible.
```

Le troisième exemple montre la puissance de  $\TeX$ . Le premier paragraphe a été repris tel quel sauf qu'on a placé une espace insécable entre deux mots (`source~\LaTeX` en ligne 8).  $\TeX$  a alors repris tous les calculs de coupure de ligne afin de présenter un résultat correct. Si les mot " source  $\LaTeX$  " avaient été déplacés sur la quatrième ligne du paragraphe, les lignes précédentes auraient été trop serrées (elles auraient présenté des espaces trop étroites entre les mots).  $\TeX$  a alors complètement revu les coupures de lignes, dès la troisième ligne dans l'exemple, en élargissant un peu plus les lignes précédentes. Pour  $\TeX$ , il était préférable de rendre les lignes un peu plus lâches que de les serrer. Les calculs entrant en jeu sont très compliqués et assurent d'avoir un résultat proche de celui obtenu de la part d'un typographe professionnel. Ces deux derniers exemples montrent qu'un source  $\LaTeX$  n'a rien de vraiment compliqué lorsque la présentation voulue n'appelle pas de constructions spéciales. Pour obtenir des résultats non standard, il va falloir faire appel à des macros et des environnements. La suite de ce manuel se charge de cette présentation. Le logiciel  $\TeX$  connaît environ 300 commandes qu'on appelle des *primitives* car elles sont construites à l'intérieur même du logiciel. Le format Plain  $\TeX$  (une bibliothèque de macros incontournables) en déclare environ 300 autres : ces 600 commandes sont la base d'un travail sous  $\TeX$ .  $\LaTeX$ , en tant que surcouche de  $\TeX$ , en définit très approximativement autant que  $\TeX$  et Plain  $\TeX$  réunis ce qui fait qu'au final, un utilisateur dispose de plus de 1 000 commandes pour indiquer ce qu'il veut faire. Ce nombre est assez colossal et apprendre par cœur la syntaxe et le rôle de ce millier de commandes est une tâche aussi ardue qu'inutile (sauf si vous vous destinez à devenir gourou de  $\LaTeX$ , et encore ...). En premier lieu, un utilisateur lambda n'a pas besoin, loin s'en faut, de toutes ces commandes (ou macros). La plupart du temps, un document n'en nécessite qu'un petit nombre et, normalement, est constitué pour l'essentiel de texte brut. Il est clair que, pour l'essentiel, les macros et environnements utilisés sont très usuels et qu'ils sont connus après peu de temps de pratique. Il n'en reste pas moins que leur apprentissage nécessite un petit effort et qu'il faut avoir de la documentation en permanence lorsqu'on travaille sous  $\LaTeX$  afin de gérer les situations exceptionnelles. Nous allons voir que les macros ont des noms qui permettent souvent de bien s'en souvenir (pour peu que l'anglais ne soit pas complètement inconnu).

## 3.2 EDI : prise en main de TexnicCenter et premières compilations

### EDI : Un mot d'introduction

Nous allons écrire nos premiers fichiers  $\LaTeX$ , apprendre à les compiler et surtout apprendre à gérer les messages d'erreurs de  $\LaTeX$ !

C'est ici que l'utilité d'un EDI prend tout son sens.

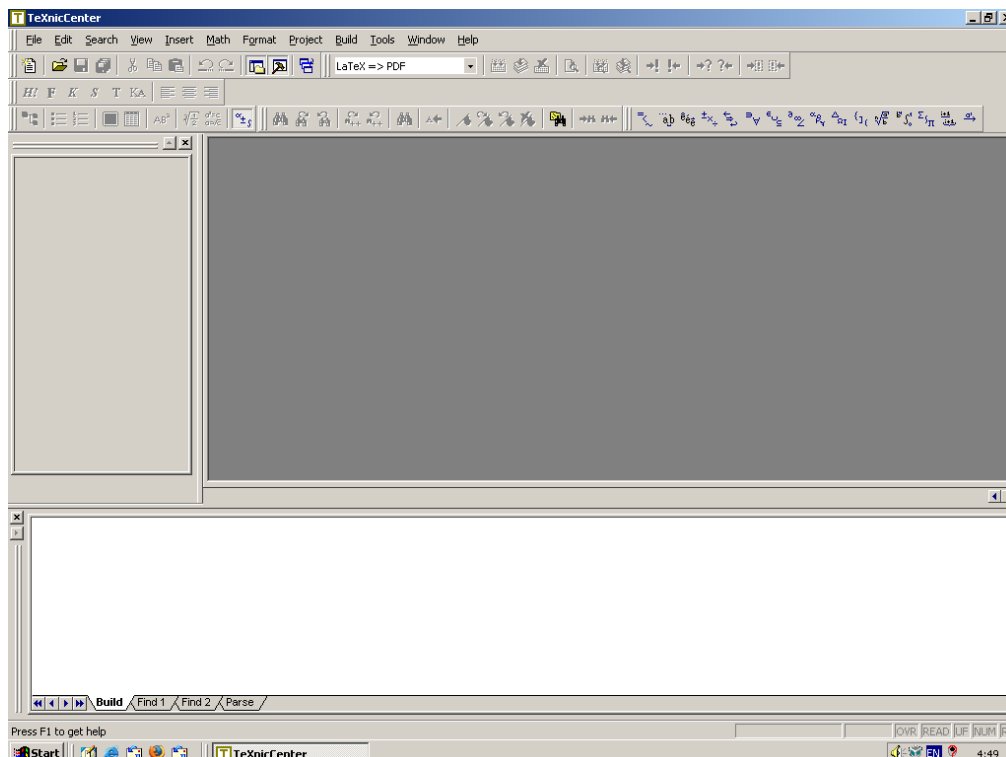
Comme, pour vous faciliter le démarrage, nous avons fait le choix de travailler sous Windows, le choix le plus logique était de choisir le couple "Mik $\TeX$ — $\TeX$ nicCenter". Je ne résisterai cependant pas à la tentation de vous présenter "Kile—Linux" comme autre couple infernal!

### 3.2.1 Les différentes sous fenêtres de $\TeX$ nicCenter

#### EDI : $\TeX$ nicCenter

Un EDI permet d'avoir accès à plusieurs outils en un seul coup d'œil.

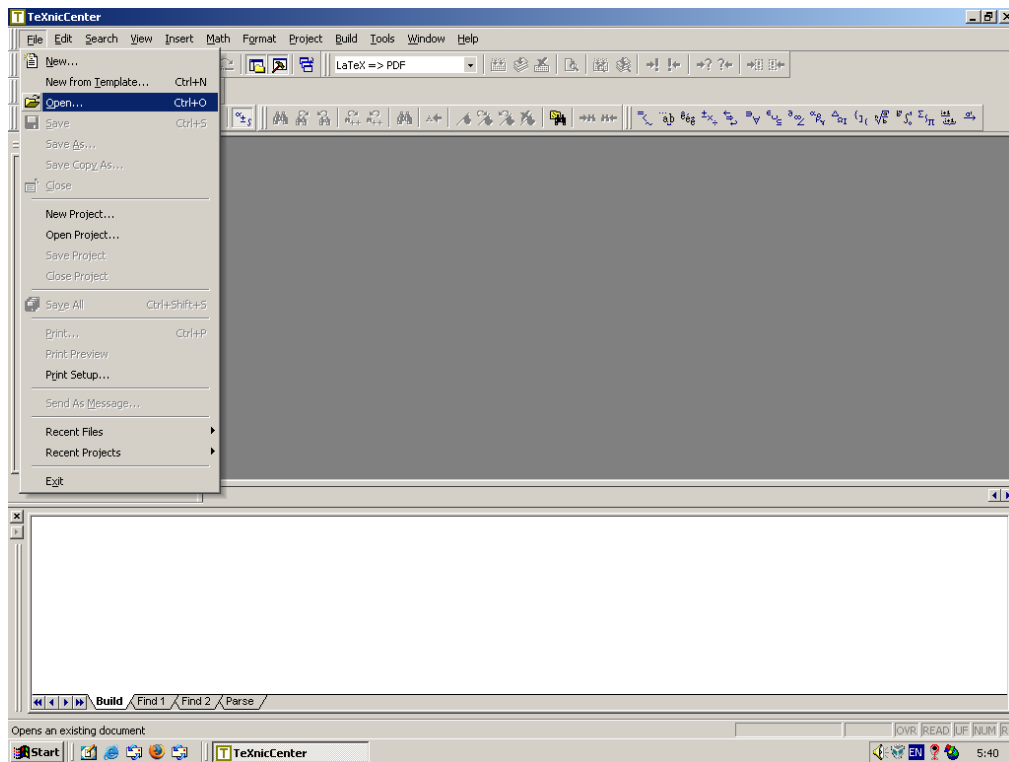
Afin d'identifier ces différents outils, démarrons  $\TeX$ nicCenter. Notre écran ressemble alors à ce qui suit.



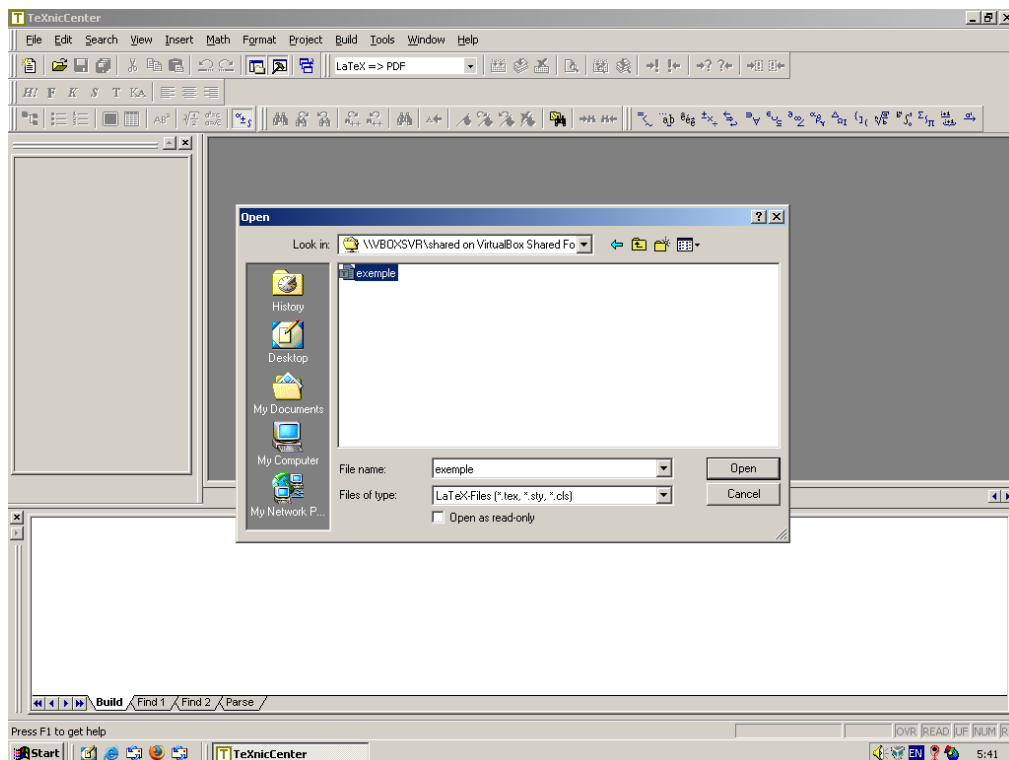
#### $\TeX$ nicCenter : ouverture du fichier "exemple.tex"

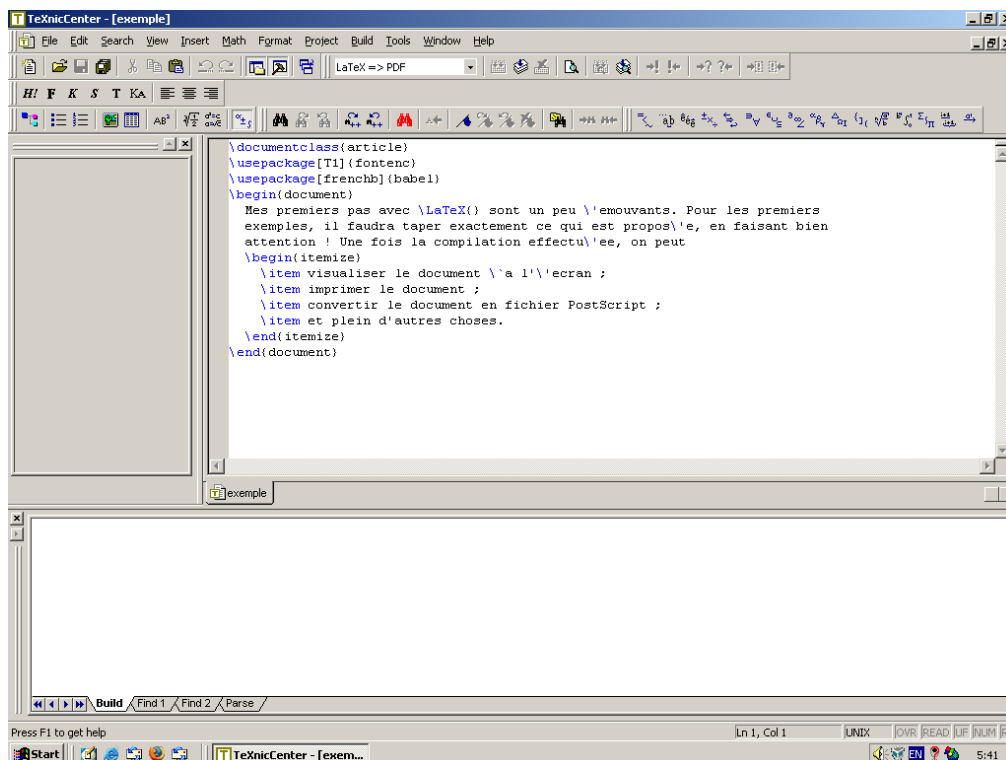
Nous n'allons pas faire le tour de toutes les options mais identifier des parties importantes de l'outil.

Pour commencer, ouvrons un fichier  $\LaTeX$  : le fichier exemple.tex



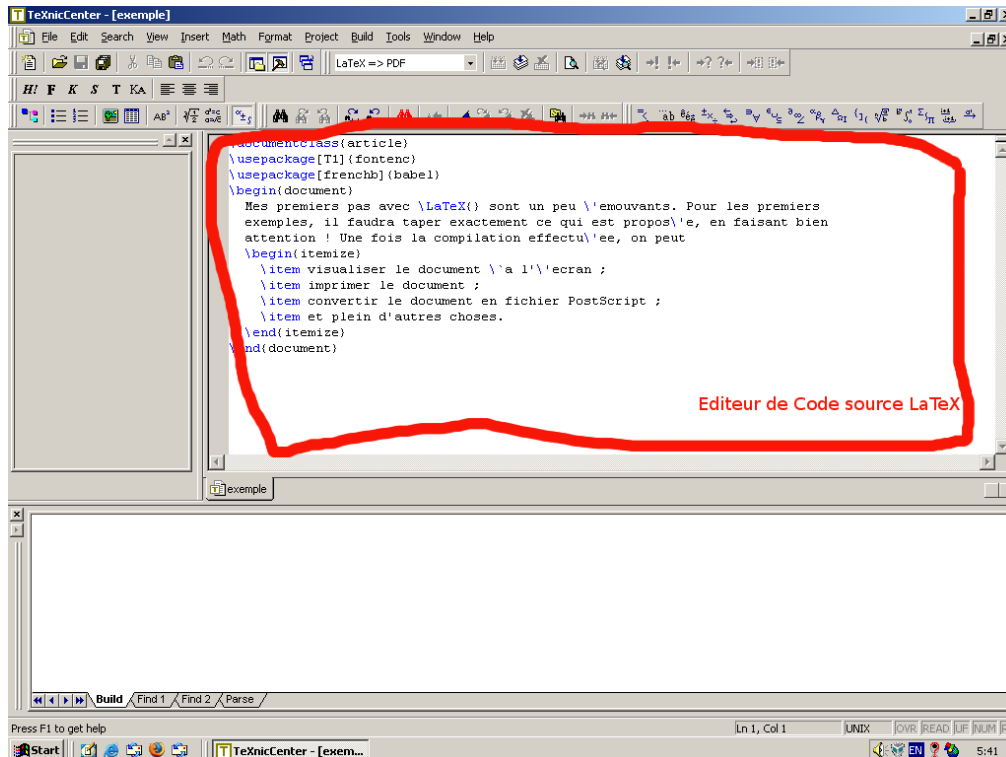
Classiquement, il faut aller dans le menu ouvrir et choisir le sous-menu “ fichier ”. Il faut aller dans le bon répertoire, choisir le fichier et cliquer sur ouvrir.





Le code source  $\LaTeX$  s'affiche alors. La coloration syntaxique nous permet de faire la distinction entre le texte et les commandes  $\LaTeX$ .

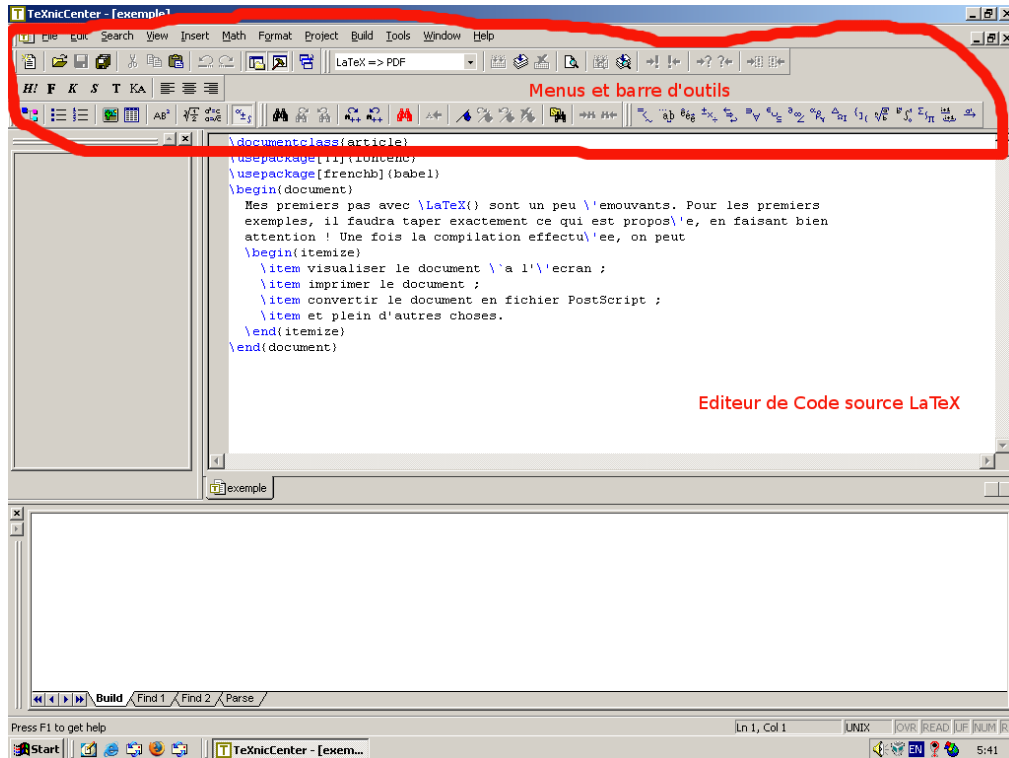
Identifions maintenant les parties essentielles de l'écran.



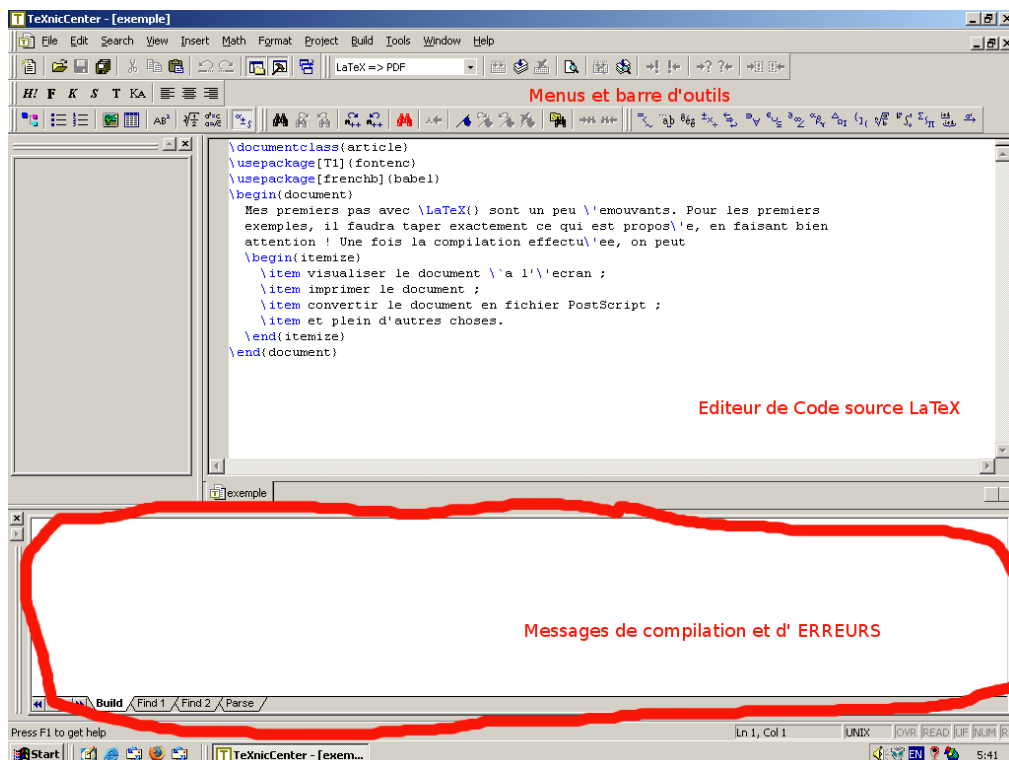
Le code source  $\LaTeX$  apparaît dans la partie la plus importante de la fenêtre. C'est normal, C'est là que notre création se trouve et c'est là que nous allons devoir le plus intervenir.

Cette partie de l'EDI, c'est l'éditeur de texte. La syntaxe  $\LaTeX$  est reconnue et les commandes sont mises en couleur. Ceci est déjà une aide importante : En cas de faute de frappe dans une commande,

la couleur ne sera pas présente.



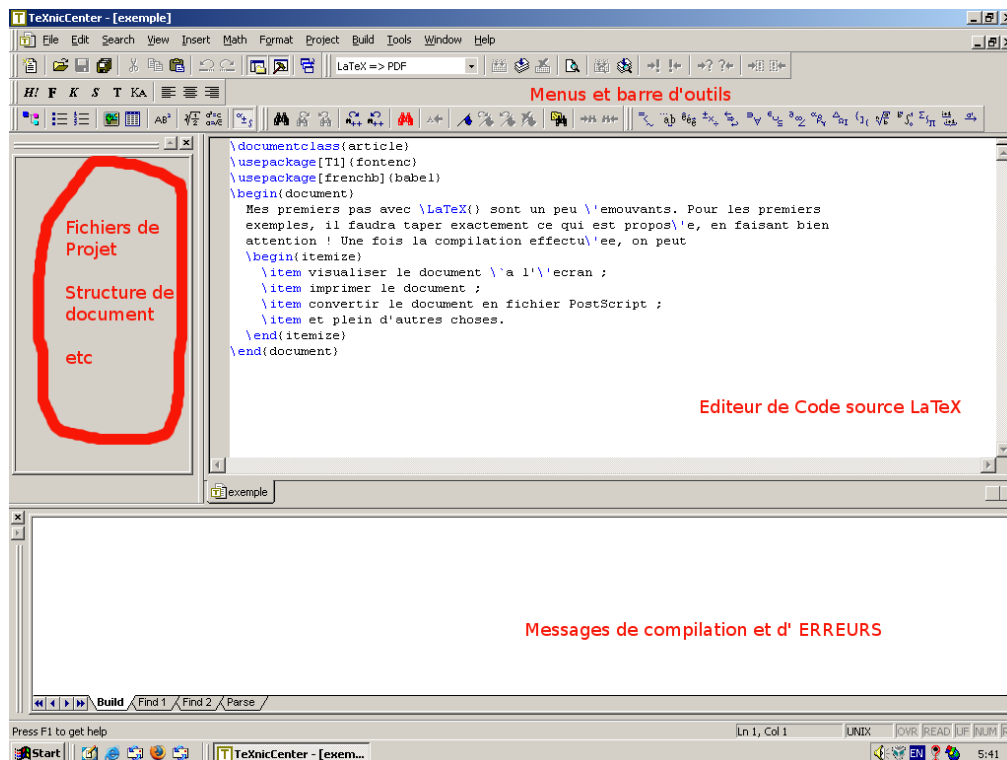
Des menus et des barres d'outils occupent la partie supérieure de l'écran Ceci nous rappelle un traitement de texte classique. Certains de ces outils auront une action dans le code comme de faire apparaître des commandes ; d'autres auront une action sur le code : le compiler par exemple. Nous y reviendrons.



Dans le bas de l'écran vont venir s'inscrire les messages de  $\text{\LaTeX}$  concernant le bon ou le mauvais déroulement des différentes actions et opérations que nous lui demanderons.

Quand tout va bien, on peut ignorer ce qui s'y passe<sup>5</sup>.

Quand il y a un problème, il faudra y trouver l'information utile. C'est une de ces choses auxquelles nous allons consacrer un peu de temps.



Toute une série d'autres outils peuvent venir se glisser dans la partie gauche de la fenêtre : Listes des fichiers ouverts, structure du document ouvert, caractères spéciaux divers... Je n'insisterai pas plus sur cet aspect pour l'instant.

## 3.2.2 Compilation de l'exemple

Nous allons nous lancer dans notre première compilation.

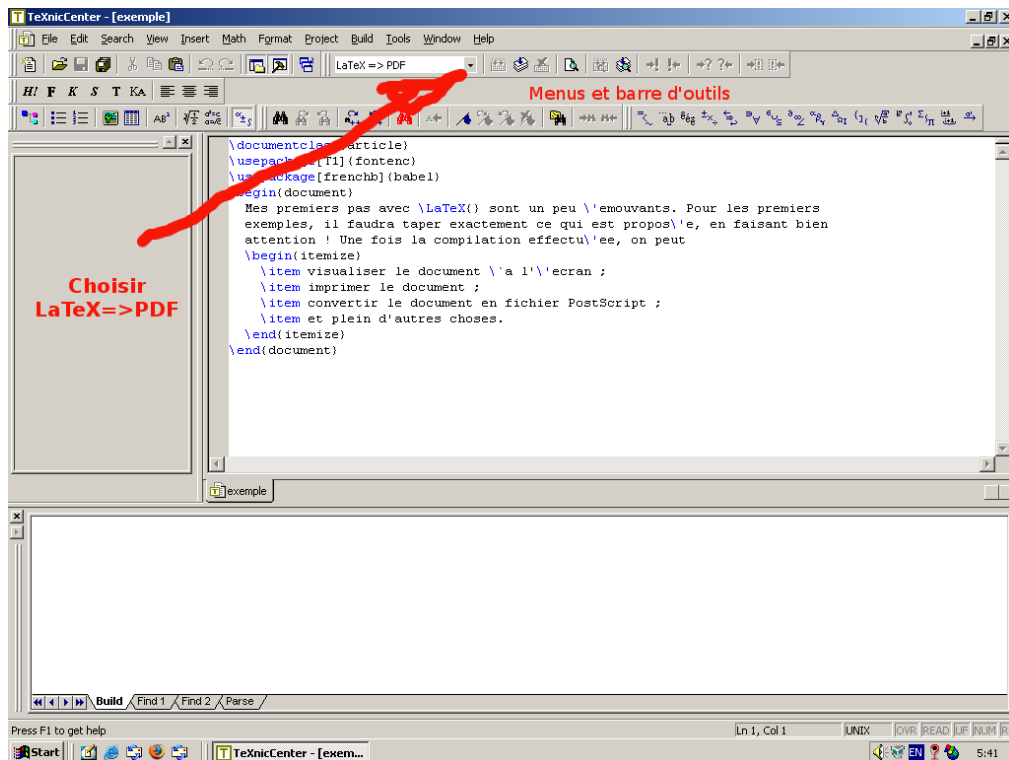
### 3.2.2.1 Choix du format de sortie

#### Votre toute première compilation

Un des avantages de  $\text{\LaTeX}$  est que le document peut être produit dans différents formats.  $\text{\LaTeX}$ , par défaut, produit du DVI, qui peut être transformé ensuite en postscript ou en pdf. Il y a aussi moyen d'obtenir du HTML et d'autres formats plus "exotiques". Pour "faire simple", nous allons créer des fichiers PDF.

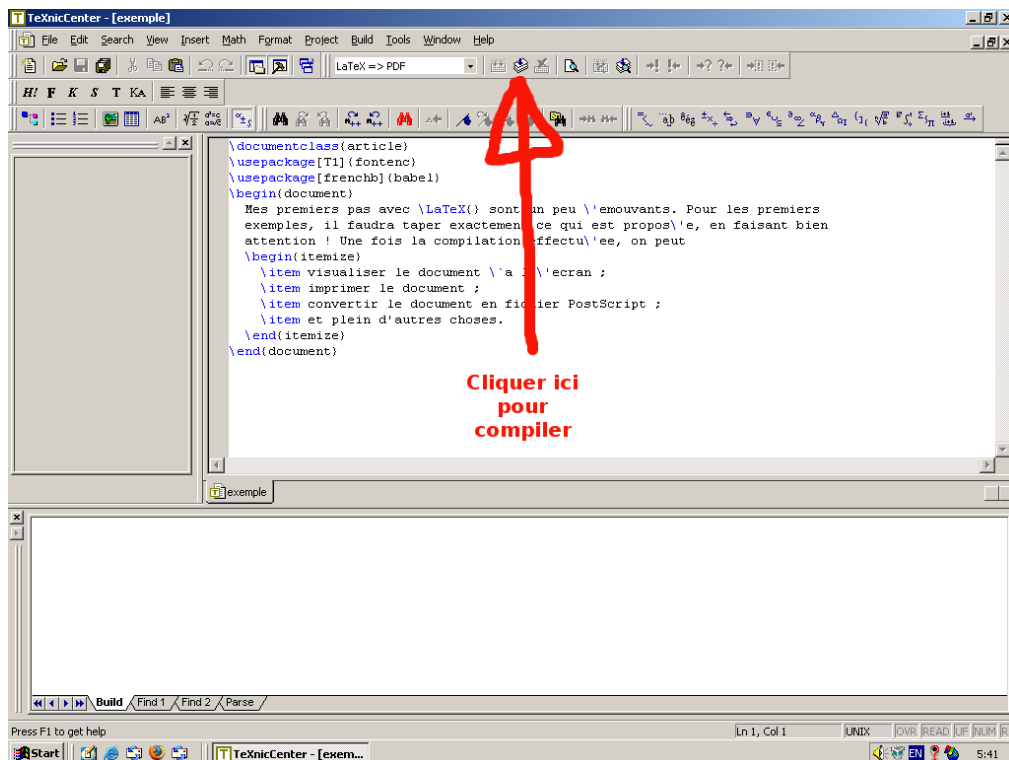
5. Les messages d'avertissement sont typiques du monde de la programmation.





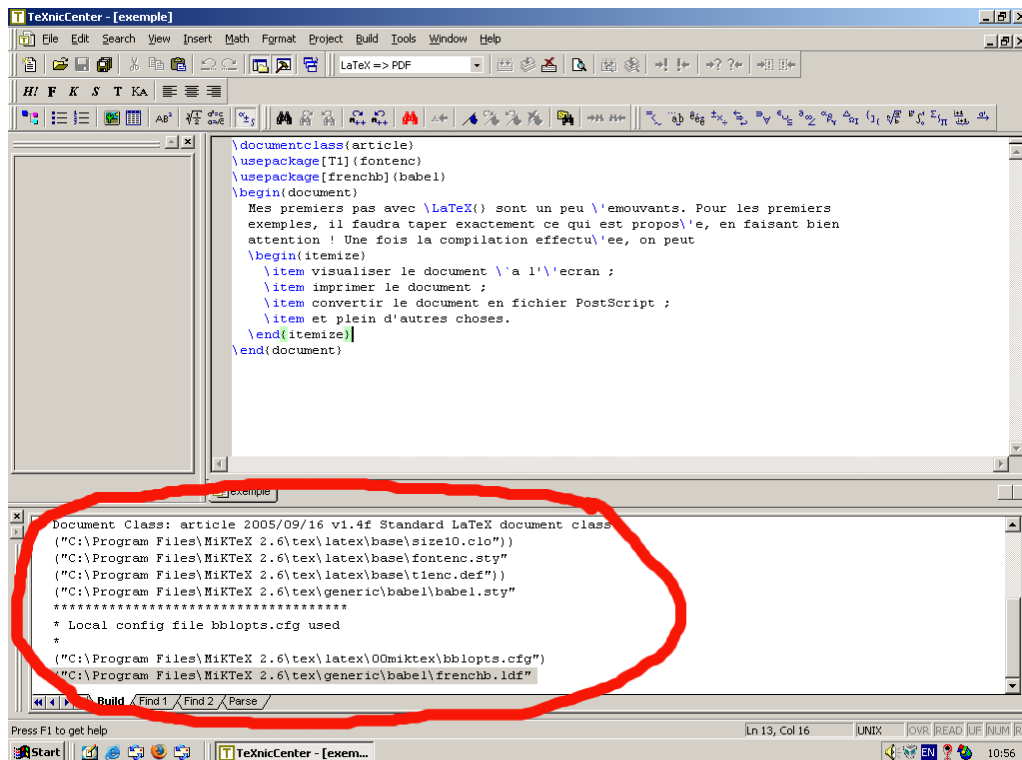
Dans la barre d'outils, choisissons la sortie PDF.

### 3.2.2.2 Compiler

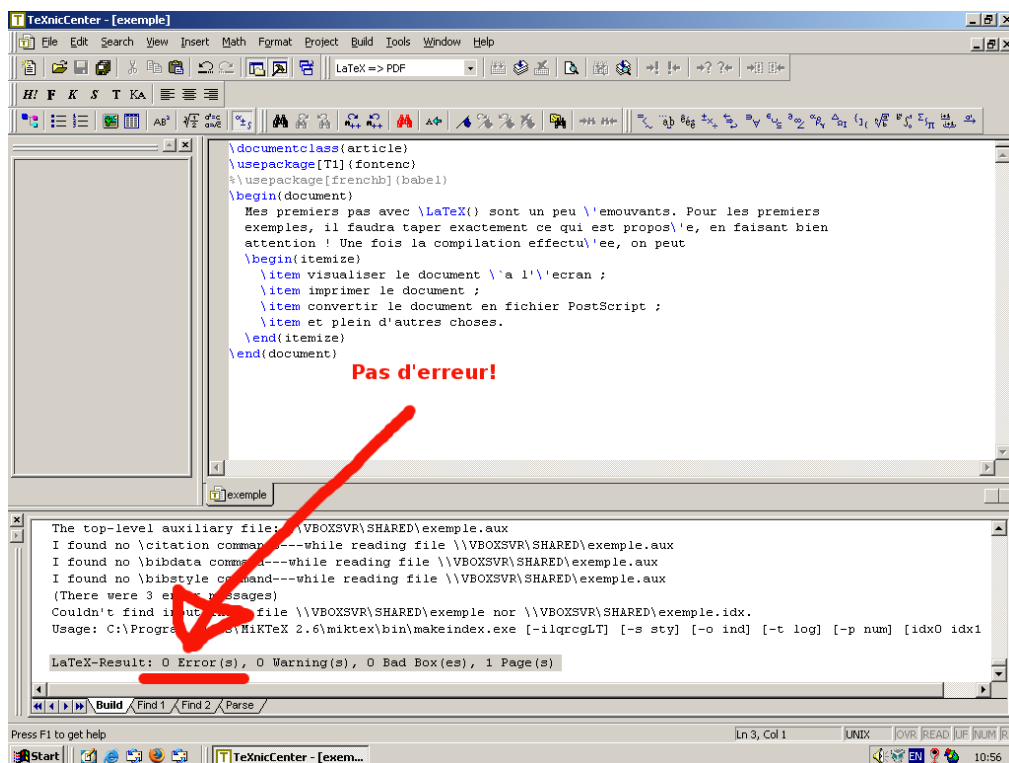


Lançons la compilation. Pour ce faire, il faut cliquer sur le bouton indiqué dans la copie d'écran. (ou appuyer sur les touches "CTRL F7")

De nombreux messages défilent alors dans la fenêtre de messages.



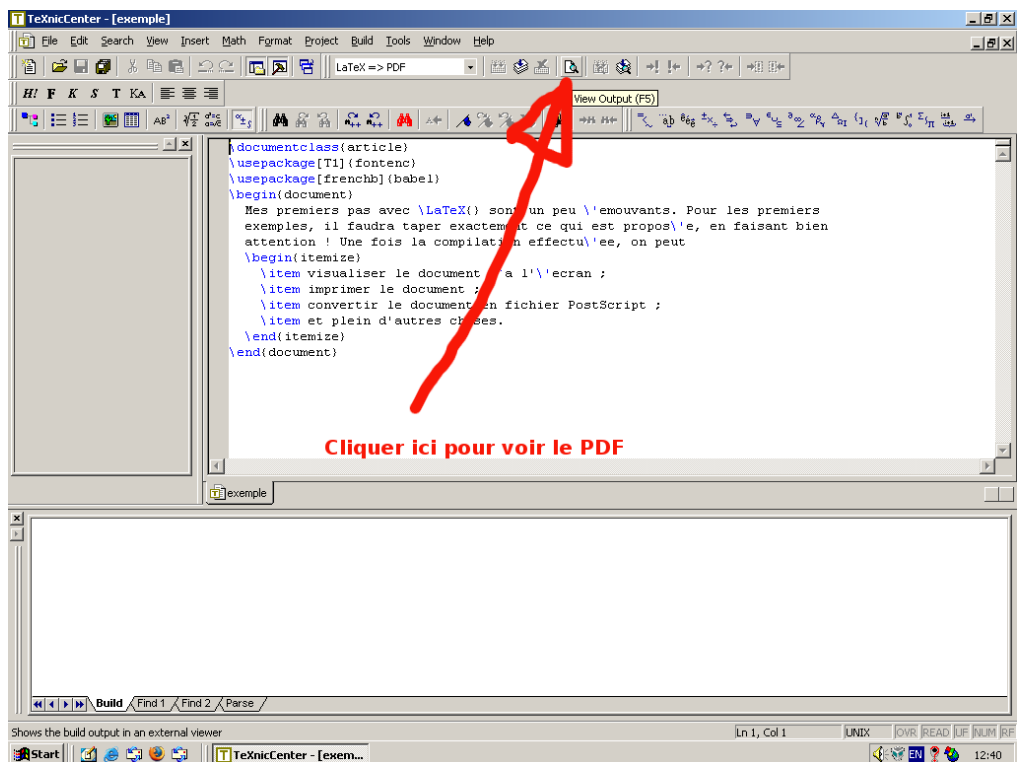
Jusqu'à l'annonce du succès.



Ce qui compte, c'est de ne pas voir d'erreur. Les autres messages (avertissement, "bad box", ...) peuvent, en général, être ignorés.

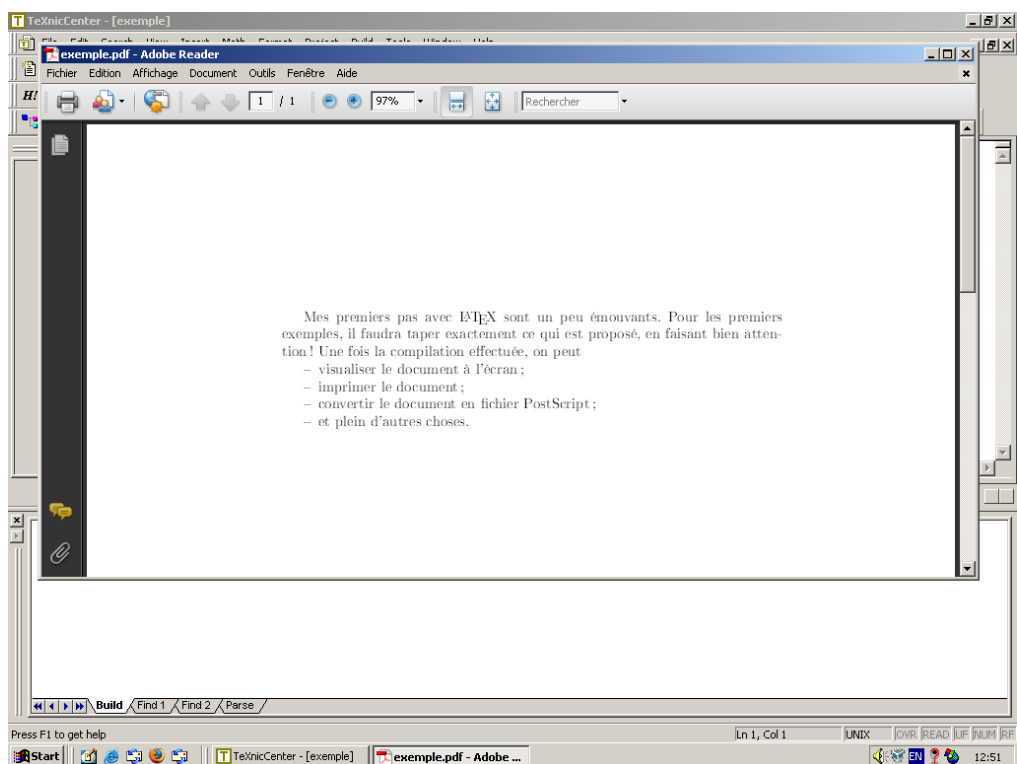
### 3.2.2.3 Voir le résultat

Mais notre but, c'est de produire de beaux documents (avec un contenu passionnant).



Cliquons donc sur le bouton indiqué ci-dessus. Comme on le voit sur la copie d'écran, le même résultat peut être obtenu en appuyant sur la touche "F5".

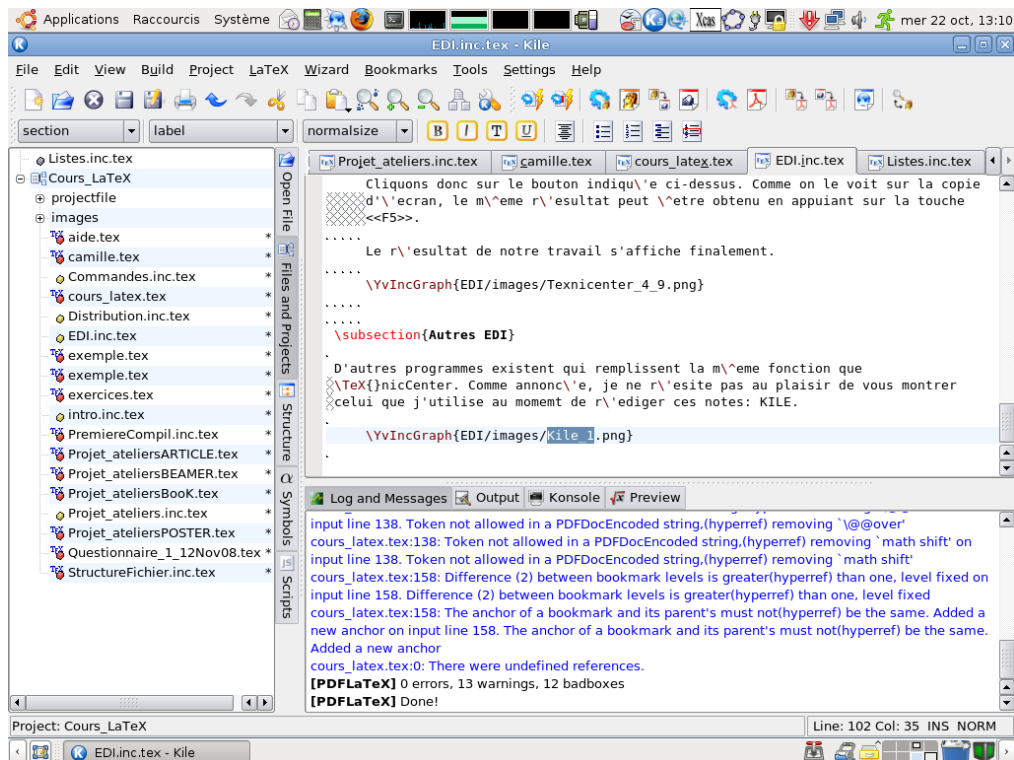
Le résultat de notre travail s'affiche finalement.



### 3.2.3 Autres EDI

Kile

D'autres programmes existent qui remplissent la même fonction que  $\text{\TeX}$ nicCenter. Comme annoncé, je ne résiste pas au plaisir de vous montrer celui que j'utilise au moment de rédiger ces notes : KILE.



Comme vous pouvez le constater, comme les fonctionnalités sont les mêmes, la structure des fenêtres sont fort proches : éditeur, messages d'erreur, barres d'outils, fichiers du projet, etc.

## 3.3 Erreurs de compilation

### Les erreurs de compilation $\LaTeX$

Comment gérer celles-ci!

Il est peu probable que votre première compilation se passe sans problème!

Nous n'allons pas encore analyser les messages d'erreur mais étudier quelques tactiques de base pour s'en sortir.

### Les erreurs les plus classiques

Il faut vérifier une série de points :

- L'orthographe exacte des commandes : *beguin*  $\neq$  *begin*
- La casse (majuscule, minuscule) : *Begin*  $\neq$  *begin*
- La parité des parenthèses, crochets et accolades : “ { ” solitaire, surnuméraire “ { { ... } ” ou mal apparié “ { ... ) ”
- des *environnements* non ou mal fermés ou encore imbriqués : *begin{une chose} ... end{une autre}*

### Décrypter les messages d'erreurs

Il faut procéder par ordre :

- Aller à la première erreur : Les erreurs suivantes sont souvent des *pseudo-erreurs*!
- Repérer le numéro de ligne concernée : Aller à cette ligne. Et essayer de comprendre!



# Chapitre 4

## Commandes de Base : environnements

### Sommaire

---

<b>4.1 Commandes de base</b> . . . . .	<b>40</b>
4.1.1 Listes . . . . .	41
<b>4.2 La gestion des erreurs de compilation</b> . . . . .	<b>43</b>
<b>4.3 Commandes de base : la suite</b> . . . . .	<b>47</b>
4.3.1 Caractères particuliers . . . . .	47

---

## 4.1 Commandes de base

### Commandes de base : introduction

Dans ce chapitre, nous allons voir les macros et les environnements qu'on doit absolument connaître.

Nous nous concentrerons sur les différents types de listes et la composition des formules mathématiques.

Nous considérerons alors la gestion de la mise en forme des caractères (caractères spéciaux, forme et taille des caractères) et des paragraphes, les espacements importants.

Délibérément, je traiterai ces problèmes de mises en forme dans un deuxième temps. La force de  $\text{\LaTeX}$ , c'est de nous faire nous concentrer sur le contenu et pas sur la forme!



### 4.1.1 Listes

#### Listes et gestion des erreurs

Les listes, outre leur intérêt propre, nous serviront de point de départ pour un apprentissage important lorsqu'on aborde  $\LaTeX$  (et qui en décourage plus d'un!) : la gestion des erreurs.

#### Les environnements de listes

$\LaTeX$  définit *cinq environnements* gérant les listes.

Deux d'entre eux sont très sommaires et servent en fait à créer des listes personnalisées ; ils sont un peu techniques et nous n'en parlerons pas dans ce stage.

Restent trois types de listes prédéfinies dont la syntaxe générale est :

```
\begin{type_de_liste}
  \item premier élément de la liste
  \item deuxième élément de la liste
  :
  \item dernier élément de la liste
\end{type_de_liste}
```

*type\_de\_liste* pouvant être *itemize*, *enumerate* ou *description*.

- Pour la liste *itemize*, chaque élément de la liste sera précédé d'un tiret demi-cadratin (avec une francisation par `\usepackage[frenchb]{babel}`, autrement le caractère utilisé par défaut par  $\LaTeX$  est un gros point noir).
- Pour la liste *enumerate*, chaque élément de la liste est précédé d'un nombre qui augmente de 1 à chaque fois qu'on passe à l'élément suivant.
- Enfin, pour la liste *description*, c'est l'utilisateur qui indique à chaque fois ce qui doit se trouver au début des éléments de la liste.

#### Trois exemples de listes

Voyons trois exemples simples pour commencer :

- en français, les premiers éléments d'une liste se terminent par un point virgule ;
- chaque élément commence par une minuscule ;
- le dernier élément a un point.

```
1 \begin{itemize}
2 \item en fran\c{c}ais,
3 les premiers \el'ements
4 d'une liste se terminent
5 par un point virgule ;
6 \item chaque \el'ement
7 commence par une minuscule ;
8 \item le dernier \el'ement
9 a un point.
10 \end{itemize}
```

1. C'est le premier point.
2. Voici le deuxième.
3. Et enfin le dernier.

```
1 \begin{enumerate}
2 \item C'est le premier point.
3 \item Voici le deuxi'eme.
4 \item Et enfin le dernier.
5 \end{enumerate}
```

**Un** Article indéfini singulier.

**Deux** Ce n'est plus un article. Parfois article introduisant le dual mais pas en français.

**Trois et etc.** Définitivement plus des articles et toujours pluriels.

```

1 \begin{description}
2 \item[Un] Article ind\'efini singulier
3
4 \item[Deux] Ce n'est plus un article.
5   Parfois article introduisant le dual
6     mais
7     pas en fran\c{c}ais.
8 \item[Trois et etc.] D\'efinitivement
9   plus des
10  articles et toujours pluriels.
11 \end{description}

```

### Changement de symbole

Si on veut modifier ponctuellement le symbole de la liste `itemize`, il suffit d'ajouter le nouveau symbole entre crochets à la suite de la macro `\item`

¶ une liste avec de jolis symboles ;  
& très esthétique ;  
? et illisible !

```

1 \begin{itemize}
2 \item[\P] une liste avec de jolis
3   symboles ;
4 \item[\&] tr\'es esth\'etique ;
5 \item[?] et illisible !
6 \end{itemize}

```

### Enumerations imbriquées

Les listes peuvent être imbriquées les unes dans les autres.

Pour la liste `enumerate`, chaque niveau dispose de son propre symbole de numérotation.  $\LaTeX$  autorise quatre niveaux d'imbrication ce qui est largement suffisant et déjà pas facile à suivre !

Les matières enseignées sont :

1. la physique à cause :
  - (a) de son élégance;
  - (b) de son ingéniosité;
  - (c) de sa clarté.
2. les mathématiques parce que :
  - (a) il faut que tout le monde vive :
    - les ingénieurs ;
    - les profs ;
    - les autres.
  - (b) il y en a quand même un peu besoin.

```

1 Les mati\`eres enseign\`ees sont :
2 \begin{enumerate}
3 \item la physique \`a cause :
4   \begin{enumerate}
5   \item de son \`el\`egance ;
6   \item de son ing\`eniosit\`e ;
7   \item de sa clart\`e.
8   \end{enumerate}
9 \item les math\`ematiques parce que :
10  \begin{enumerate}
11  \item il faut que tout le monde vive
12    :
13    \begin{itemize}
14    \item les ing\`enieurs ;
15    \item les profs ;
16    \item les autres.
17    \end{itemize}
18  \item il y en a quand m\`eme un peu
19    besoin.
20  \end{enumerate}
21 \end{enumerate}

```

## 4.2 La gestion des erreurs de compilation

Nous allons essayer d'acquérir quelques bonnes pratiques nécessaires pour ne pas se décourager lors des premiers pas avec  $\text{\LaTeX}$ . Ouvrons les fichiers des exemples de liste vu précédemment et essayons de les compiler.

Nous devrions obtenir ceci.

- en français, les premiers éléments d'une liste se terminent par un point virgule ;
- chaque élément commence par une minuscule ;
- le dernier élément a un point.

1. C'est le premier point.
2. Voici le deuxième.
3. Et enfin le dernier.

**Un** Article indéfini singulier.

**Deux** Ce n'est plus un article. Parfois article introduisant le dual mais pas en français.

**Trois et etc.** Définitivement plus des articles et toujours pluriels.

Ouvrons le fichier "ExercicesListes\_1.tex" et essayons de le compiler!

```

\documentclass[a4paper,12pt](article)
\usepackage[T1](fontenc)
%
%\usepackage(ucs)
%\usepackage[utf8x](inputenc)

\begin(document)

\begin(itemize)
\item en fran(c)cais, les premiers \el\ements
d'une liste se terminent par un point
virgule ;
\item chaque \el\ement commence par une
minuscule ;
\item le dernier \el\ement a un point.
\end(itemize)

\end(document)

```

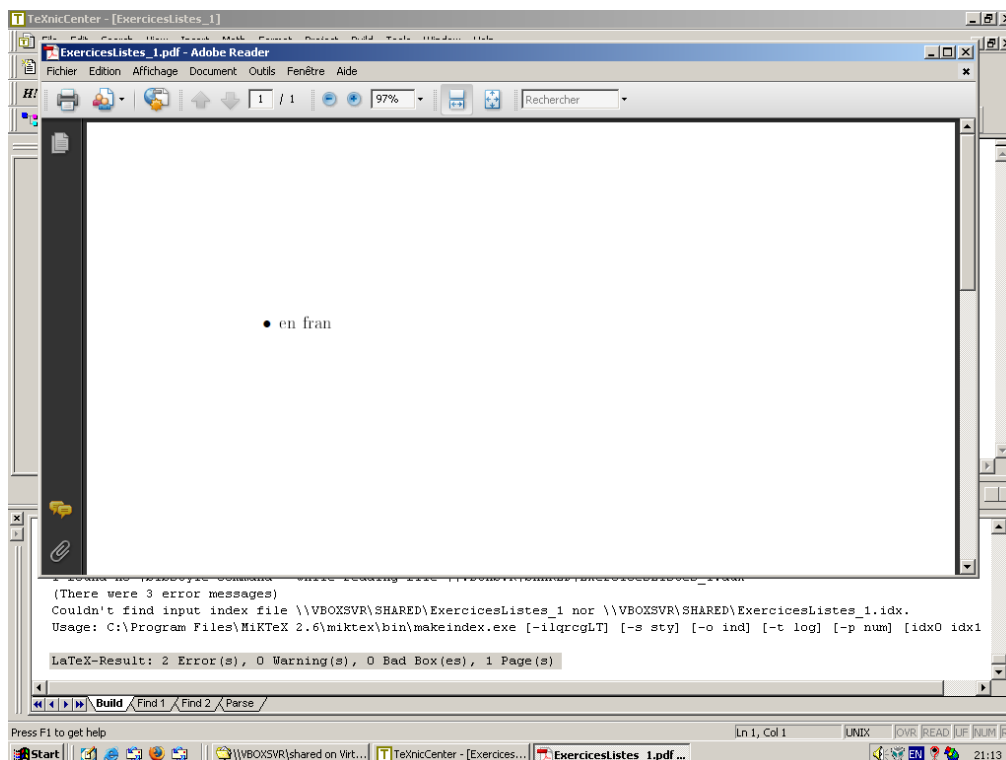
```

The top-level auxiliary file: \\VBOXSVR\SHARED\ExercicesListes_1.aux
I found no \citation command---while reading file \\VBOXSVR\SHARED\ExercicesListes_1.aux
I found no \bibdata command---while reading file \\VBOXSVR\SHARED\ExercicesListes_1.aux
I found no \bibstyle command---while reading file \\VBOXSVR\SHARED\ExercicesListes_1.aux
(There were 3 error messages)
Couldn't find input file \\VBOXSVR\SHARED\ExercicesListes_1 nor \\VBOXSVR\SHARED\ExercicesListes_1.idx.
Usage: C:\Program Files\MiKTeX 2.6\miktex\bin\makeindex.exe [-ilqrcgLT] [-s sty] [-o ind] [-t log] [-p num] [idx0 idx1]
LaTeX-Result: 2 Error(s), 0 Warning(s), 0 Bad Box(es), 1 Page(s)

```

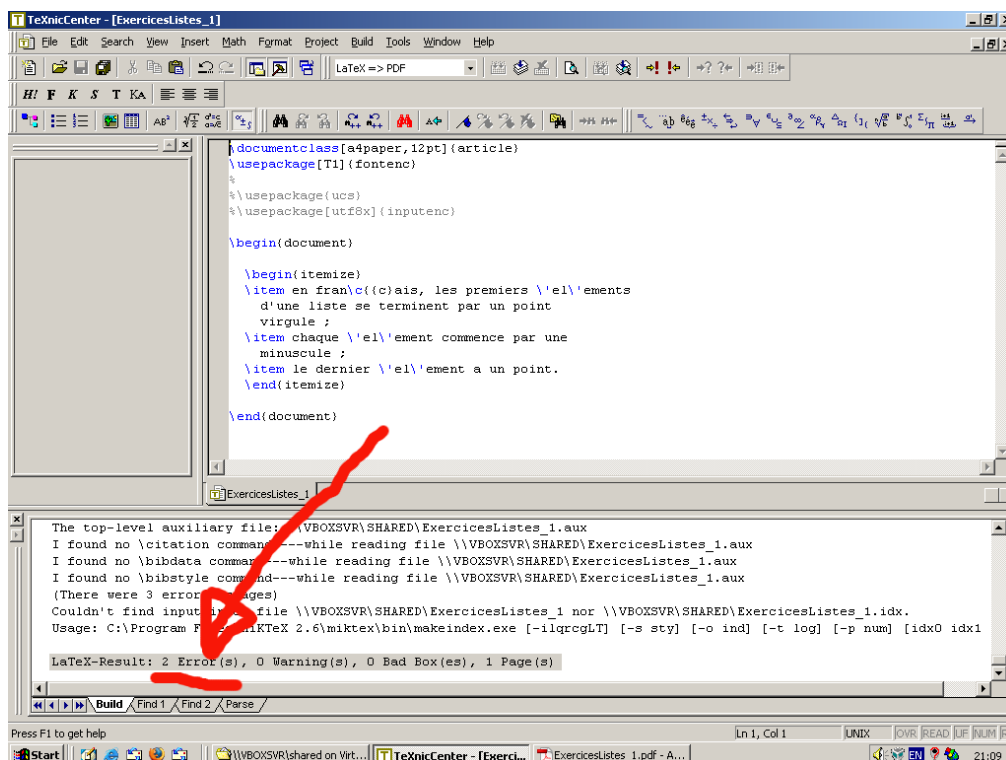
2 erreurs!

Et si nous essayons de voir le résultat.

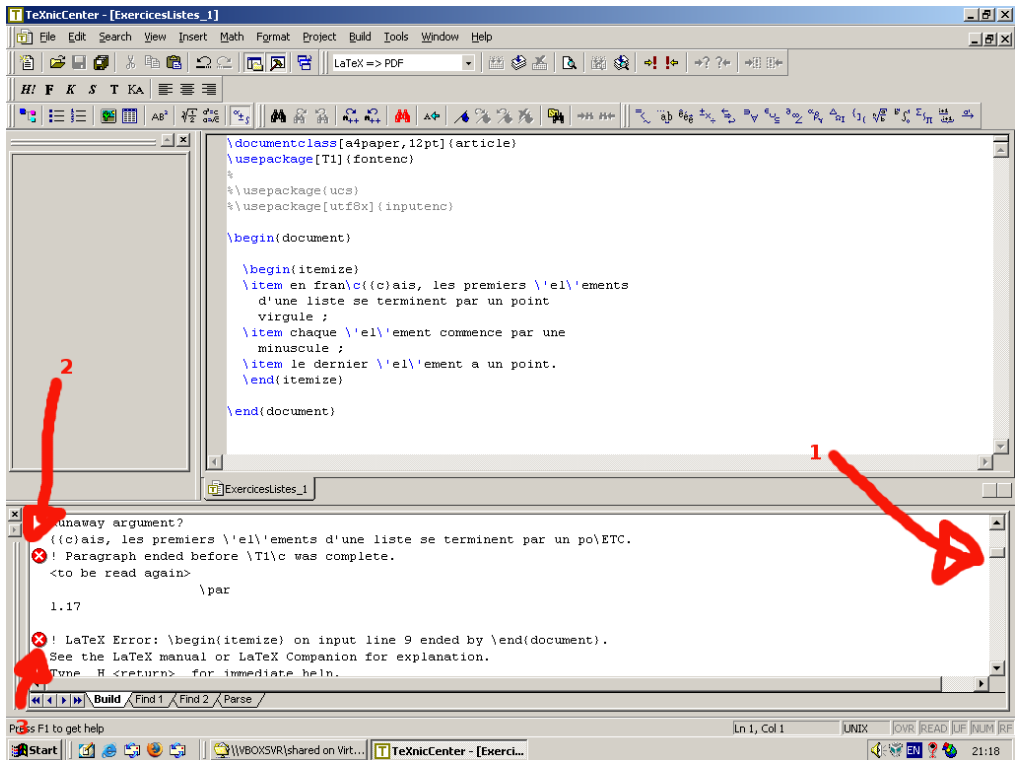


C'est l'horreur! C'est normal, c'est fait exprès!

Si nous retournons à la fenêtre de messages d'erreur... nous ne sommes pas vraiment aidés!



Utilisons l'ascenseur (en 1 sur la figure suivante) de la fenêtre de messages d'erreur et remontons dans les messages.



En 2 et 3, nous trouvons des informations plus utiles.

**Conseils**

- Compiler souvent!
  - Après chaque modification importante.
  - Quand on n'est pas trop sûr de soi!
  - Et même plus souvent que ça!
- Vérifier les *packages* chargés ou non.
- Attention aux
  - caractères spéciaux,
  - macros,
  - environnements,
  - accolades, crochets et pararenthèses diverses,
  - à prendre une pose avant de tout casser!

**Solutions**

- Compiler souvent!
- Mettre un "end document" juste avant ou juste après la zone où il y a une erreur.
- Utiliser l'EDI pour trouver une parenthèse, accolade, crochet excédentaire ou oublié.

## 4.3 Commandes de base : la suite

### 4.3.1 Caractères particuliers

#### Obtenir les 10 caractères réservés

On a vu précédemment que les 10 caractères `\ % ~ { } $ _ ^ #` et `&` avaient des rôles précis et ne pouvaient pas être employés pour obtenir les caractères correspondants au niveau de la sortie. Le tableau 4.1 montre les dix macros permettant de retrouver ces caractères.

caract.	macro
<code>\</code>	<code>\(\backslash\)</code>
<code>~</code>	<code>\~{}</code>
<code>^</code>	<code>\^{}</code>
<code>%</code>	<code>\%</code>
<code>{</code>	<code>\{</code>
<code>}</code>	<code>\}</code>
<code>\$</code>	<code>\\$</code>
<code>_</code>	<code>\_</code>
<code>#</code>	<code>\#</code>
<code>&amp;</code>	<code>\&amp;</code>

TABLE 4.1 – Obtention des caractères réservés

On voit que pour sept d'entre eux, la macro a le nom du caractère lui-même. Les exceptions concernent la barre oblique inverse (backslash en anglais), l'accent circonflexe et le niña.

Knuth a expliqué pourquoi il n'a pas voulu utiliser `\` comme macro permettant d'obtenir la barre oblique ; je n'entrerai pas dans les détails. Remarquons quand même que cette macro doit être entourée des signes `\(` et `\)` qui indiquent que l'intérieur est composé en mode mathématique (Cf. section 5.1.1).

#### 4.3.1.1 Les accents

L'accent circonflexe et le niña sont des exceptions car il ne s'agit pas de macros donnant un caractère mais un accent.

Ainsi, `\^` indique que le prochain caractère doit avoir un accent circonflexe.

Comme nous voulions l'accent seul, il a suffi de faire suivre cette macro d'un groupe vide.

`\^e` donne "ê", `\^a` donne "â" (qui n'existe pas mais  $\TeX$  s'en moque).

N'importe quel caractère peut être accentué.

Ainsi, `\^c` donne "ĉ", `\^b` donne "ĥ" (on remarquera que l'accent s'est automatiquement placé plus haut pour cette lettre présentant une hampe), `\^\$` donne "Ŷ".

En fait,  $\TeX$  offre la possibilité de placer 14 types d'accents couvrant les besoins de toutes les langues européennes (Cf. tableau 4.2).

#### Claviers QWERTY et autres

Certains utilisateurs refusent d'utiliser le `\usepackage[T1]{fontenc}` et sont donc obligés de taper toutes les lettres accentuées de cette façon. C'est une question d'habitude, chacun fait comme il l'entend !

caract.	code	nom
ó	\'o	aigu
ò	\'o	grave
ô	\^o	circonflexe
ö	\"o	tréma
õ	\~o	niña
ō	\=o	barre
ô	\.o	point
ö	\u o	brève
ř	\v o	tchèque
ő	\H o	tréma hongrois
ô	\t o	lien après
ç	\c o	cédille
ç	\d o	point-dessous
ç	\b o	barre-dessous

TABLE 4.2 – Accent en mode texte

En revanche, un grand nombre de claviers ne proposent pas les capitales accentuées.

J'en profite pour rappeler que les capitales doivent TOUJOURS être accentuées lorsque le mot comporte un accent.

Et ces macros deviennent alors indispensables.

## Locales

Attention cependant aux problèmes de *locale*. Les caractères non accentués étant les seuls sur lesquels tous les systèmes informatiques s'entendent, les accents peuvent poser problème si un texte à été écrit sur un "vieux" système et que l'on désire le compiler sur un nouveau système. La plupart des EDI permettent de décider quelle locale est utilisée.

## Macros à une lettre, accents et groupes

On notera la syntaxe un peu différente pour les macros dont le nom est une lettre (\u, \v, \H, etc.) car une écriture comme \vo ne produira certainement pas le caractère " ö " mais une erreur de compilation puisque la macro de nom vo n'existe pas.

On aurait pu écrire cela de différentes façons : \v{o}, \v{o} produisent le résultat voulu.

Si on ne veut pas réfléchir, on peut toujours écrire de façon systématique le caractère à accentuer dans un groupe.

Certains caractères européens ne sont pas souvent présents sur les claviers d'ordinateur, aussi,  $\TeX$  déclare les macros du tableau 4.3 pour pouvoir y accéder.

Les ponctuations espagnoles ¿ et ¡, appelées par les séquences ?' et !' ne sont en fait pas des macros mais des ligatures. Cette différence ne présente pas beaucoup d'intérêt pour l'heure. Normalement, un français ne devrait connaître que les quatre premières macros.

Quelques caractères d'usage courant et non forcément disponibles sur les claviers sont également définis (Cf. tableau 4.4).

Nous n'avons présenté que les caractères disponibles par défaut avec  $\TeX$ . En fait, grâce aux extensions



caract.	macro
œ	\oe
Œ	\OE
æ	\ae
Æ	\AE
ß	\ss
å	\aa
Å	\AA
ø	\o
Ø	\O
ł	\l
Ł	\L
¿	?‘
¡	!‘

TABLE 4.3 – Caractères européens spéciaux

caract.	macro
†	\dag
‡	\ddag
§	\S
¶	\P
©	\copyright
£	\pounds

TABLE 4.4 – Symboles spéciaux

(Cf. chapitre 9),

l'utilisateur a à sa disposition une bibliothèque prodigieuse de symboles. Les caractères particuliers vus ici ne sont que ceux des modes textes. Les modes mathématiques offrent une bien plus grande richesse en raison de la consommation immodérée de ces symboles par les mathématiciens et physiciens.



# Chapitre 5

## Modes Mathématiques

### Sommaire

---

<b>5.1</b>	<b>Mathématiques élémentaires . . . . .</b>	<b>52</b>
5.1.1	Différents modes mathématiques . . . . .	52
5.1.2	Opérations élémentaires . . . . .	53
5.1.3	Structures indispensables . . . . .	54
<b>5.2</b>	<b>Symboles en vrac . . . . .</b>	<b>56</b>

---

## 5.1 Mathématiques élémentaires

### 5.1.1 Différents modes mathématiques

#### Modes mathématiques

On s'en doute : les modes mathématiques sont destinés à composer des formules mathématiques mais en fait, on peut utiliser le mode mathématique pour des constructions qui n'ont rien à voir avec les math. Comme il s'agit d'un chapitre d'introduction, nous ne verrons cependant que l'aspect purement mathématique de la chose !

#### Deux modes

Il existe deux modes mathématiques :  
le mode en texte et le mode hors texte.

#### Mode texte

Le mode en texte permet de construire une formule destinée à être incorporée dans le texte d'où une étendue verticale réduite au minimum comme dans  $\frac{1}{2}x = \frac{x}{2}$ .

#### Mode hors texte

Le mode hors texte qui permet de composer des formules centrées (par défaut) et qui utilisent toute la ligne rien que pour elles ce qui permet des espacements verticaux plus généreux.

Voici la même formule en hors texte :

$$\frac{1}{2}x = \frac{x}{2}$$

#### displaystyle et interlignage

$\LaTeX$  permet des constructions telles que  $\frac{1}{2}x = \frac{x}{2}$ , c'est-à-dire composer une formule en texte comme elle apparaît en hors texte.

Mais, contrairement à ce que pensent beaucoup de personnes, c'est une erreur de vouloir le faire. L'interlignage est complètement détruit ce qui est une faute typographique de débutant !

#### Délimiteurs des modes mathématiques

Comment tape-t-on la formule qui a servi d'exemple ?

Tout d'abord, il faut indiquer à  $\LaTeX$  quand commence et se termine la formule.

Le tableau 5.1 suivant résume les possibilités :

#### Pour un \$ de plus!

Les délimiteurs utilisant le signe \$ (dollar) sont un codage  $\TeX$ .

Il est préférable d'utiliser un des deux codages purement  $\LaTeX$  pour éviter certains effets bizarres qui peuvent survenir dans des situations complexes. J'avoue que faisant la navette entre  $\TeX$  et  $\LaTeX$ , j'utilise souvent les dollars mais ce n'est pas bien !

délimiteurs	mode	exemple
$\$ \dots \$$	en texte	$\frac{1}{2}x = \frac{x}{2}$
$\langle \dots \rangle$	en texte	$\frac{1}{2}x = \frac{x}{2}$
$\backslash\text{begin}\{\text{math}\} \dots \backslash\text{end}\{\text{math}\}$	en texte	$\frac{1}{2}x = \frac{x}{2}$
$\$\$ \dots \$\$$	hors texte	$\frac{1}{2}x = \frac{x}{2}$
$\langle [ \dots ] \rangle$	hors texte	$\frac{1}{2}x = \frac{x}{2}$
$\backslash\text{begin}\{\text{displaymath}\} \dots \backslash\text{end}\{\text{displaymath}\}$	hors texte	$\frac{1}{2}x = \frac{x}{2}$
$\backslash\text{begin}\{\text{equation}\} \dots \backslash\text{end}\{\text{equation}\}$	+ $n^0$	$\frac{1}{2}x = \frac{x}{2}$

TABLE 5.1 – Délimiteurs des modes mathématiques

 **$\backslash\text{begin}\{\text{equation}\} \dots \backslash\text{end}\{\text{equation}\}$** 

$\backslash\text{begin}\{\text{equation}\} \dots \backslash\text{end}\{\text{equation}\}$  permet d'avoir des équations numérotées en mode hors texte. Cet environnement est très pratique si nous désirons référencer des équations dans un texte.

Si  $n \rightarrow +\infty$ , alors

$$\sum_{i=1}^n \left(\frac{1}{2}\right)^i = 2 \quad (5.1)$$

$$\sum_{i=1}^n \frac{1}{i} = +\infty \quad (5.2)$$

$$\sum_{i=1}^n (-1)^i \frac{1}{i} = \dots \quad (5.3)$$

```

1 Si  $(n \rightarrow +\infty)$ , alors
2 \begin{equation}
3 \sum_{i=1}^n \left(\frac{1}{2}\right)^i = 2
4 \end{equation}
5
6 \begin{equation}
7 \sum_{i=1}^n \frac{1}{i} = +\infty
8 \end{equation}
9
10 \begin{equation}
11 \sum_{i=1}^n (-1)^i \frac{1}{i} = \dots
12 \end{equation}

```

**5.1.2 Opérations élémentaires****Mathématiques élémentaires**

Maintenant que nous savons nous mettre en mode mathématique, il reste à voir ce qu'on peut mettre dedans ! Durant cette première approche, nous ne verrons que les bases, les compositions plus complexes seront traitées plus tard

à la section 10.1 page 96.

Dans cette section, nous allons voir un grand nombre de macros permettant des constructions diverses. À part une seule, ces macros sont interdites en dehors des modes mathématiques : leur utilisation dans du texte provoquera une erreur lors de la compilation.

**Les quatre opérations**

$\text{T}_\text{E}_\text{X}$  gère évidemment les lettres et les quatre opérations de base. Examinons la formule  $x+3 = y-5$  obtenue avec le code  $\backslash(x+3=y-5)$ .

On peut remarquer que des espaces de tailles variées ont été automatiquement ajoutées autour des signes plus, moins et égal. En fait, des espaces mis dans le source n'ont strictement aucune action en mode mathématique (hormis, bien sur, celui de stopper un nom de macro). Par exemple `\(x + 3 = y - 5\)` aurait donné strictement le même résultat.

On remarquera également que *les lettres sont en italique* ; c'est la règle pour les formules de math sauf qu'en France, la règle complète indique que les majuscules ne doivent pas être en italique ce que ne fait pas T<sub>E</sub>X par défaut (dans ce manuel, on transgressera cette règle typographique et même les majuscules seront en italique).

### Multiplier et diviser

Les signes plus, moins et égal sont présents sur le clavier mais pas ceux de la multiplication et de la division (celle de l'école primaire).

On peut bien évidemment taper un `" / "` pour la division mais la macro `\div` donne le symbole  $\div$ . Le signe multiplié est obtenu avec la macro `\times` qui donne le symbole  $\times$ .

T<sub>E</sub>X est intelligent et la formule `x\times-3` est comprise comme il se doit, c'est-à-dire comme le produit de  $x$  et de  $-3$  ce qui fait que les espaces sont correctement gérées :  $x \times -3$ . En définitive, si vous tapez votre formule correctement, T<sub>E</sub>X la composera correctement d'un point de vue typographique.

### 5.1.3 Structures indispensables

Avec les variables et les quatre opérations, on ne fait quand même pas grand chose. Certaines structures reviennent très souvent dans les formules de mathématiques.

#### Exposants et indices

Les exposants et les indices utilisent deux caractères réservés de T<sub>E</sub>X, à savoir respectivement `^` et `_`.

Leur syntaxe est on ne peut plus simple : ce qui suit immédiatement `^` est placé en exposant (même chose avec `_` pour les indices).

Il est également possible de mélanger exposant et indice sans ce soucier de l'ordre de spécification.

Dans les exemples qui suivent, regardez avec attention la hauteur de ces éléments.

$x^2, l^2, x_2, y_2, x^2, x_i, x_i^2$ et $x_i^2$	1 <code>\(x^2\), \((1^2\), \((x_2\), \((y_2\)</code>
$x^2$	2 <code>\(x^2\), \((x_i\), \((x^2_i\)</code>
$x_i$	3 <code>et \((x_i^2\)</code>
$x_i^2$	4
$x^2 \times y^2, x^2 y^2$ et $10^{-3}$	5 <code>\[x^2\] \[x_i\] \[x^2_i\]</code>
	6
	7 <code>\(x^2\times y^2\), \((x^2y^2\)</code>
	8 <code>et \((10^{-3}\)</code>

Le dernier exemple " $10^{-3}$ " est certainement une faute ; on pensait sans doute à  $10^{-3}$  en tapant cette formule. T<sub>E</sub>X a fait ce qu'on lui a demandé, en l'occurrence écrire 10, suivi de `-` en exposant puis un 3 (qui n'est plus en exposant car ne suivant pas immédiatement le symbole `^`). Pour obtenir la forme désirée, il faut faire suivre l'accent circonflexe d'un groupe qui contiendra tout ce qui sera en exposant : la formule correcte est obtenue avec la syntaxe `10^{\{-3\}}`.

## Dérivée

Un exposant particulièrement utilisé en mathématique est le prime (dérivée par exemple).  $\TeX$  permet un raccourci sous la forme :

$$f'(x) = 2x$$

Si  $f'(x) = 2x$  alors  $f''(x) = 2$  et  $f'''(x) = 0$ .

```
1 \f'(x)=2x\
2
3 Si \f'(x)=2x\ alors \f''(x)=2\
4 et \f'''(x)=0\.
```

C'est-à-dire que la mise en exposant est automatique. En regardant à la loupe, on pourra voir que le prime dans la formule mathématique "  $f'$  " n'a pas la même forme que l'apostrophe dans le texte " Il f'afait fur le fable ".

## Fractions

On peut bien sûr employer le symbole / et c'est d'ailleurs mieux dans les formules en texte car les fractions à barre horizontale dans une formule en texte sont quelque peu ridicules (comme dans l'exemple  $\frac{1}{2}$ ). En revanche, dans les formules hors texte, il est souvent préférable de construire une fraction à barre horizontale.

Celles-ci se construisent grâce à la macro `\frac` suivi de deux groupes : le premier indiquant le numérateur et le second le dénominateur. Voici un exemple pour clarifier les idées :

$$\frac{x}{y} \div \frac{y}{x} = \frac{x}{y} \times \frac{x}{y} = \frac{x^2}{y^2}$$

```
1 \[ \frac{x}{y} \div \frac{y}{x}
2 = \frac{x}{y} \times \frac{x}{y}
3 = \frac{x^2}{y^2}
4 \]
```

En fait, on aurait pu écrire `\frac x y` à la place de `\frac{x}{y}` mais c'est une mauvaise habitude car les groupes étaient absolument obligatoires pour la dernière fraction. Il y a intérêt à automatiser les gestes lorsqu'on compose des formules mathématiques, les difficultés arrivant bien trop vite !

Le numérateur et le dénominateur sont composés avec un style plus tassé (équivalent au style en texte). Ainsi, un empilement de fraction peut sembler un peu bizarre :

$$\frac{\frac{x^2}{y^2}}{\frac{y^2}{x^2}} = \frac{x^4}{y^4}$$

```
1 \[\frac{\frac{x^2}{y^2}}{\frac{y^2}{x^2}}
2 = \frac{x^4}{y^4}\]
```

## "displaystyle"

Ce comportement est souvent préférable mais, si on veut absolument avoir la taille normale hors texte, on peut appeler la macro `\displaystyle` (*display* étant ici traduit par *hors texte*) et tout ce qui se trouve à la suite de cette macro à l'intérieur d'un groupe sera mis à la taille " display ".

$$\frac{\displaystyle\frac{x^2}{y^2}}{\displaystyle\frac{y^2}{x^2}} = \frac{x^4}{y^4}$$

```
1 \[\frac{\displaystyle\frac{x^2}{y^2}}
2 \displaystyle\frac{y^2}{x^2}}
3 = \frac{x^4}{y^4}\]
```

C'est de cette façon qu'on a obtenu l'exemple pas très joli du début de section :  $\frac{1}{2}x = \frac{x}{2}$  qui faisait peu de cas de l'interlignage.

## Racines

Peu de chose à dire. La macro `sqrt` place ce qui suit sous le signe racine. Comme pour les fractions, il vaut mieux ne pas trop réfléchir et utiliser systématiquement un groupe après cette macro.

$\sqrt{\sqrt{25}+4} = 3$	<code>1 \[\sqrt{\sqrt{25}+4} = 3\]</code> <code>2 en texte cela donne :</code> <code>3 \(\sqrt{\sqrt{25}+4} = 3\)</code>
en texte cela donne : $\sqrt{\sqrt{25}+4} = 3$	

## Racines cubiques et autres

La macro `\sqrt[n]{x}` permet des racines *n*ième en précisant le *n* entre crochets juste après la macro.

$f_n(x) = \sqrt[n]{x+1}$	<code>1 \[f_n(x)=\sqrt[n]{x+1}\]</code>
--------------------------	---

## Points de suspension

LaTeX offre quatre type de points de suspension. La macro `\ldots` spécifie les points de suspension " normaux ", c'est la seule des quatre macros à pouvoir être utilisée en dehors des modes mathématiques. On la place principalement entre deux virgules ou entre des lettres.

La macros `\cdots` place les points un peu plus haut de façon à ce qu'ils soient alignés avec les signes opératoires usuels (moins, plus, égal, ...).

La macros `\vdots` place trois points verticalement ( $\vdots$ ) et la macro `\ddots` les place en diagonal ( $\ddots$ ). On les utilise plutôt dans des matrices

(Cf. section 10.1).

$f(x_1, \dots, x_n) = x_1 + \dots + x_n$	<code>1 \[f(x_1, \ldots, x_n) = x_1 + \cdots + x_n\]</code>
--	---

## 5.2 Symboles en vrac

### La caverne d'Ali Baba aux symboles

Cette section est une véritable caverne d'Ali Baba. Les scientifiques ayant l'habitude d'utiliser un nombre important de symboles. Nous allons essayer de classifier un peu tout ceci !

### Lettres grecques

Elles sont obtenues en faisant précéder leur nom d'une contre-oblique.

Par exemple `\(\beta\)` donnera  $\beta$ .

Les majuscules sont obtenues en mettant la première lettre de leur nom en majuscule : `\(\Omega\)` donnera ainsi  $\Omega$ .

Seules les lettres n'existant pas dans l'alphabet romain sont spécifiées. Pour les minuscules, cela n'exclut que le omicron (`\omicron` est donc une macro inexistante) obtenu avec un " o ".

En revanche, c'est beaucoup plus fréquent pour les majuscules puisqu'il n'y a que 11 majuscules grecques n'ayant pas d'équivalent graphique dans l'alphabet romain : (`\Gamma`  $\Gamma$ , `\Delta`  $\Delta$ , `\Theta`  $\Theta$ , `\Lambda`  $\Lambda$ , `\Xi`  $\Xi$ , `\Pi`  $\Pi$ , `\Sigma`  $\Sigma$ , `\Upsilon`  $\Upsilon$ , `\Phi`  $\Phi$ , `\Psi`  $\Psi$  et `\Omega`  $\Omega$ ).



### Lettres grecques minuscules

D'autre part, 6 minuscules présentent deux graphies différentes, à chaque fois l'une d'elle sera obtenue en faisant précéder le nom de la lettre par var. On a :

<code>\epsilon</code>	$\epsilon$	<code>\varepsilon</code>	$\varepsilon$
<code>\theta</code>	$\theta$	<code>\vartheta</code>	$\vartheta$
<code>\pi</code>	$\pi$	<code>\varpi</code>	$\varpi$
<code>\rho</code>	$\rho$	<code>\varrho</code>	$\varrho$
<code>\sigma</code>	$\sigma$	<code>\varsigma</code>	$\varsigma$
<code>\phi</code>	$\phi$	<code>\varphi</code>	$\varphi$

### Lettres calligraphiques

La macro `\mathcal` permet de faire appel à la fonte calligraphique. Celle définie par défaut donne le résultat  $\mathcal{A}, \mathcal{B}, \dots, \mathcal{Z}$ .

Soit  $f$  une fonction appartenant à  $\mathcal{F}$ .

```
1 Soit \(\f\) une fonction appartenant \‘
  a
2 \(\mathcal{F}\).
```

Seules les majuscules sont accessibles.

Lorsque nous discuterons des mathématiques “ avancées ”, nous verrons que les extensions de l’ “ American Mathematical Society ” fournissent d’autres possibilités.

### Opérateurs binaires

En plus des signes + et – on trouve un grand nombre d’opérateurs binaires. Ces signes seront séparés de ce qui les entoure de la même façon que pour une addition ou une soustraction. La liste complète est donnée au tableau 5.2.

<code>\pm</code>	$\pm$	<code>\cap</code>	$\cap$	<code>\vee</code>	$\vee$
<code>\mp</code>	$\mp$	<code>\cup</code>	$\cup$	<code>\wedge</code>	$\wedge$
<code>\setminus</code>	$\setminus$	<code>\oplus</code>	$\oplus$	<code>\ominus</code>	$\ominus$
<code>\cdot</code>	$\cdot$	<code>\oslash</code>	$\oslash$	<code>\odot</code>	$\odot$
<code>\times</code>	$\times$	<code>\triangleleft</code>	$\triangleleft$	<code>\dagger</code>	$\dagger$
<code>\ast</code>	$\ast$	<code>\triangleright</code>	$\triangleright$	<code>\ddagger</code>	$\ddagger$
<code>\star</code>	$\star$	<code>\wr</code>	$\wr$	<code>\amalg</code>	$\amalg$
<code>\diamond</code>	$\diamond$	<code>\bigcirc</code>	$\bigcirc$		
<code>\circ</code>	$\circ$	<code>\bigtriangleup</code>	$\bigtriangleup$		
<code>\bullet</code>	$\bullet$	<code>\bigtriangledown</code>	$\bigtriangledown$		
<code>\div</code>	$\div$				

TABLE 5.2 – Opérateurs binaires

### Relations

En plus des signes =, > et <, on trouve également un grand nombre de relations. Ces signes seront séparés de ce qui les entoure par les mêmes espacements que ceux entourant le signe =. La liste complète est donnée dans le tableau 5.3

<code>\leq</code>	$\leq$	<code>\geq</code>	$\geq$	<code>\equiv</code>	$\equiv$
<code>\prec</code>	$\prec$	<code>\succ</code>	$\succ$	<code>\sim</code>	$\sim$
<code>\preceq</code>	$\preceq$	<code>\succeq</code>	$\succeq$	<code>\simeq</code>	$\simeq$
<code>\ll</code>	$\ll$	<code>\gg</code>	$\gg$	<code>\asymp</code>	$\asymp$
<code>\subset</code>	$\subset$	<code>\supset</code>	$\supset$	<code>\approx</code>	$\approx$
<code>\subseteq</code>	$\subseteq$	<code>\supseteq</code>	$\supseteq$	<code>\cong</code>	$\cong$
<code>\sqsubset</code>	$\sqsubset$	<code>\sqsupset</code>	$\sqsupset$	<code>\bowtie</code>	$\bowtie$
<code>\in</code>	$\in$	<code>\ni</code>	$\ni$	<code>\propto</code>	$\propto$
<code>\vdash</code>	$\vdash$	<code>\dashv</code>	$\dashv$	<code>\models</code>	$\models$
<code>\smile</code>	$\smile$	<code>\mid</code>	$\mid$	<code>\doteq</code>	$\doteq$
<code>\frown</code>	$\frown$	<code>\parallel</code>	$\parallel$	<code>\perp</code>	$\perp$

TABLE 5.3 – Relations

### Négation

Ces relations peuvent être négativisées en les faisant précéder de la macro `\not`. Par exemple, la séquence `\(\not\approx\)` produira le symbole  $\not\approx$ .

### Grands opérateurs

Ils sont indiqués par soucis d'exhaustivité. mais nous les étudierons à la section 10.1.

Indiquons toutefois que leur taille varie selon qu'on se trouve en mode en texte ou en mode hors texte. Le tableau 5.4 en dresse la liste complète :

<code>\sum</code>	$\Sigma$	$\sum$	<code>\bigcap</code>	$\cap$	$\bigcap$	<code>\bigodot</code>	$\odot$	$\bigodot$
<code>\prod</code>	$\Pi$	$\prod$	<code>\bigcup</code>	$\cup$	$\bigcup$	<code>\bigotimes</code>	$\otimes$	$\bigotimes$
<code>\coprod</code>	$\amalg$	$\coprod$	<code>\bigsqcup</code>	$\sqcup$	$\bigsqcup$	<code>\bigoplus</code>	$\oplus$	$\bigoplus$
<code>\int</code>	$\int$	$\int$	<code>\bigvee</code>	$\vee$	$\bigvee$	<code>\biguplus</code>	$\uplus$	$\biguplus$
<code>\oint</code>	$\oint$	$\oint$	<code>\bigwedge</code>	$\wedge$	$\bigwedge$			

TABLE 5.4 – Grands opérateurs

### Délimiteurs

Les délimiteurs sont des symboles qui encadrent une sous-formule. Ils sont présentés au tableau 5.5 et, comme pour les grands opérateurs, ils ne seront étudiés qu'à la section 10.1.

### Flèches

$\LaTeX$  propose une vaste panoplie de flèches. Le tableau 5.6 en dresse la liste complète.

### Symboles inclassables

Tous ces symboles sont des symboles ordinaires, c'est-à-dire qu'ils se comporteront comme des lettres au niveau des espacements. Ils sont présentés au niveau du tableau 5.7.

(	( )	)	\uparrow	↑
[	[ ]	]	\downarrow	↓
\{	{ }	}	\updownarrow	↕
\lfloor	[ \rfloor	]	\Uparrow	⇩
\lceil	[ \rceil	]	\Downarrow	⇨
\langle	\rangle	\rangle	\Updownarrow	↕
/	/ \backslash	\		
	\ \	\		

TABLE 5.5 – Délimiteurs

\leftarrow	←	\longleftarrow	←←	\uparrow	↑
\Lleftarrow	⇐	\Longleftarrow	⇐⇐	\Uparrow	⇩
\rightarrow	→	\longrightarrow	→→	\downarrow	↓
\Rrightarrow	⇒	\Longrightarrow	⇒⇒	\Downarrow	⇨
\leftrightharpoonup	↔	\longleftrightharpoonup	↔↔	\updownarrow	↕
\Leftrightharpoonup	⇔	\Longleftrightharpoonup	⇔⇔	\Updownarrow	↕
\mapsto	↦	\logmapsto	↦↦	\nearrow	↗
\hookrightarrow	↪	\hookrightarrow	↪	\searrow	↘
\leftharpoonup	↵	\rightharpoonup	↶	\swarrow	↙
\leftharpoondown	↽	\rightharpoondown	↷	\nwarrow	↖
\rightleftharpoons	⇌				

TABLE 5.6 – Flèches

\aleph	ℵ	\prime	′	\forall	∀	\infty	∞
\hbar	ℏ	\emptyset	∅	\exists	∃	\triangle	△
\imath	ι	\nabla	∇	\neg	¬	\clubsuit	♣
\jmath	ℵ	\surd	√	\flat	♭	\diamondsuit	♦
\ell	ℓ	\top	⊤	\natural	♮	\heartsuit	♥
\wp	℘	\bot	⊥	\sharp	♯	\spadesuit	♠
\Re	ℜ	\	∥	\backslash	\	\partial	∂
\Im	ℑ	\angle	∠				

TABLE 5.7 – Symboles mathématiques divers

### Noms de fonctions

Sous ce terme générique, on trouve tous les mots qui exigent d'être écrits en caractères romains (alors que par défaut, les lettres sont écrites en italique). Le tableau 5.8 indique toutes les possibilités.

### Modulo

Le tableau 5.8 indique toutes les possibilités mises à part les macros `\bmod` et `\pmod`. `\bmod` produit le texte " mod " mais le considère comme un opérateur binaire (donc avec des espaces de chaque côté). `\pmod` est une macro qui écrit " mod " avec ce qui suit entre parenthèses. Un petit exemple pour mieux comprendre :

$$\inf(a, b) = m \bmod n$$

$$a \equiv b \pmod{m+n}$$

```
1 \[\inf(a, b) = m \bmod n\]
2 \[a \equiv b \pmod{m+n}\]
```

<code>\arccos</code>	<code>arccos</code>	<code>\csc</code>	<code>csc</code>	<code>\ker</code>	<code>ker</code>	<code>\min</code>	<code>min</code>
<code>\arcsin</code>	<code>arcsin</code>	<code>\deg</code>	<code>deg</code>	<code>\lg</code>	<code>lg</code>	<code>\Pr</code>	<code>Pr</code>
<code>\arctan</code>	<code>arctan</code>	<code>\det</code>	<code>det</code>	<code>\lim</code>	<code>lim</code>	<code>\sec</code>	<code>sec</code>
<code>\arg</code>	<code>arg</code>	<code>\dim</code>	<code>dim</code>	<code>\liminf</code>	<code>liminf</code>	<code>\sin</code>	<code>sin</code>
<code>\cos</code>	<code>cos</code>	<code>\exp</code>	<code>exp</code>	<code>\limsup</code>	<code>limsup</code>	<code>\sinh</code>	<code>sinh</code>
<code>\cosh</code>	<code>cosh</code>	<code>\gcd</code>	<code>gcd</code>	<code>\ln</code>	<code>ln</code>	<code>\sup</code>	<code>sup</code>
<code>\cot</code>	<code>cot</code>	<code>\hom</code>	<code>hom</code>	<code>\log</code>	<code>log</code>	<code>\tan</code>	<code>tan</code>
<code>\coth</code>	<code>coth</code>	<code>\inf</code>	<code>inf</code>	<code>\max</code>	<code>max</code>	<code>\tanh</code>	<code>tanh</code>

TABLE 5.8 – Noms de fonction

### Accents

Les macros servant à accentuer en mode texte ne sont quelquefois plus valides en mode mathématique.

Ce dernier offre 10 sortes d'accents plus 2 de taille variable. Le tableau 5.9 montre ces macros. Les deux accents de taille variable n'ont en fait que trois tailles possibles (les deux plus grandes tailles

<code>\acute</code>	<code>acute</code>	<code>\check</code>	<code>check</code>	<code>\grave</code>	<code>grave</code>	<code>\vec</code>	<code>vec</code>	<code>\x</code>	<code>x</code>
<code>\bar</code>	<code>bar</code>	<code>\ddot</code>	<code>ddot</code>	<code>\hat</code>	<code>hat</code>	<code>\widehat</code>	<code>widehat</code>	<code>\widehat{xy}</code>	<code>\widehat{xyz}</code>
<code>\breve</code>	<code>breve</code>	<code>\dot</code>	<code>dot</code>	<code>\tilde</code>	<code>tilde</code>	<code>\widetilde</code>	<code>widetilde</code>	<code>\widetilde{xy}</code>	<code>\widetilde{xyz}</code>

TABLE 5.9 – Accents mathématiques

étant montrées dans le tableau).

### Synonymes

Certaines macros sont particulièrement utilisées dans les textes mathématiques et elles possèdent des synonymes. Ces synonymes ont pour but de leur donner un nom plus court ou plus parlant (pour un anglophone). Le tableau 5.10 donne les équivalents possibles.

<code>\not=</code> ( $\neq$ )	ou <code>\ne</code> ou <code>\neq</code>		
<code>\geq</code> ( $\geq$ )	ou <code>\ge</code>	<code>\leq</code> ( $\leq$ )	ou <code>\le</code>
<code>\}</code> ( $\}$ )	ou <code>\rbrace</code>	<code>\{</code> ( $\{$ )	ou <code>\lbrace</code>
<code>\leftarrows</code> ( $\leftrightarrow$ )	ou <code>\gets</code>	<code>\rightarrow</code> ( $\rightarrow$ )	ou <code>\to</code>
<code>\wedge</code> ( $\wedge$ )	ou <code>\land</code> (logical and)	<code>\vee</code> ( $\vee$ )	ou <code>\lor</code> (logical or)
<code>\neg</code> ( $\neg$ )	ou <code>\lnot</code> (logical not)	<code>\ni</code> ( $\ni$ )	ou <code>\owns</code>
<code>\ </code> ( $\parallel$ )	ou <code>\Vert</code>	<code> </code> ( $ $ )	ou <code>\vert</code>

TABLE 5.10 – Synonymes de macros mathématiques

# Chapitre 6

## Commandes de Base : structurer un document

### Sommaire

---

<b>6.1 Quelques présentations plus évoluées . . . . .</b>	<b>62</b>
6.1.1 Structuration des documents . . . . .	62
6.1.2 Table des matières, index . . . . .	63
6.1.3 En-têtes, pieds de page, notes . . . . .	64

---

## 6.1 Quelques présentations plus évoluées

### 6.1.1 Structuration des documents

#### 6.1.1.1 Commandes pour le plan

##### Présentations évoluées : structuration et plan de document

L<sup>A</sup>T<sub>E</sub>X connaît 7 niveaux de plan. Dans l'ordre d'importance on trouve :

- `\part`,
- `\chapter`,
- `\section`,
- `\subsection`,
- `\subsubsection`,
- `\paragraph`
- et `\subparagraph`.

La syntaxe est on ne peut plus simple : il suffit d'indiquer la macro suivie par un groupe qui donnera le titre. Ainsi les trois titres précédents ont été obtenus avec les lignes :

```
\section{Quelques présentations plus évoluées}
\subsection{Structuration des documents}
\subsubsection{Commandes pour le plan}
```

##### Structure selon la classe

Dans la première édition <sup>1</sup> les titres de chapitre de ce manuel étaient composés avec la macro `\section` au lieu de la macro `\chapter`. C'était que pour limiter la colère à la photocopie, le style de document était `article` et non pas `report` ou `book`. Dans la classe `article` la macro `\chapter` n'existe pas. D'autre part, la mise en page n'est pas la même en fonction de la classe de document. Essentiellement, les espaces verticaux seront plus ou moins importants.

Par exemple, un `\chapter` prendra une page complète et sera composé sur une page impaire dans le style `book` alors qu'avec la classe `report` ce ne sera pas le cas.

Il serait trop long de décrire exactement ce que font ces 7 macros selon les 3 classes possibles de document. Il suffit d'être logique et tout se passera bien.

##### Annexes

Les annexes sont introduites par la macro `\appendix`. À la suite de cette macro, les chapitres n'auront plus les mêmes numérotations (emploi de lettres romaines majuscules, A, B, ..., à la place des chiffres arabes) et le mot " Chapitre " sera remplacé par le mot " Annexe ". Attention, ce dernier point ne sera vrai que si le document est francisé avec, par exemple, l'appel de `\usepackage [frenchb] {babel}` au niveau du préambule.

##### Table des matières

Si le document a été correctement structuré avec ces 7 macros, il devient simplissime de construire la table des matières.

En effet, un appel à la macro `\tableofcontents` suffit à faire tout le travail. Certaines présentations nécessitent deux compilations successives pour être correctes. La construction d'une table des

---

1. Celle de Jean-Côme Charpentier

matières fait partie de celles-ci. La première compilation construit un fichier auxiliaire dans lequel sont rangés les renseignements concernant les titres (énoncés et numéros de page) et la seconde compilation lit ce fichier pour construire effectivement la table des matières.

### 6.1.1.2 Références

#### Label et références

Faire référence à un emplacement d'un document peut être une tâche ingrate si elle n'était pas automatisée. En effet, il faudrait être sûr de la page qu'on veut référencer ainsi que du numéro de titre, de tableau, de figure.

Pour cela, il suffit de placer la macro `\label` suivi d'un groupe donnant le mot-clé permettant de retrouver cette référence. Ensuite, la macro `\ref` suivi d'un groupe ayant le même mot-clé donnera le numéro de la structure référencée (titre, tableau, figure, équation) et la macro `\pageref` donnera son numéro de page. Pour arriver à ceci,  $\TeX$  écrit ces informations dans un fichier auxiliaire lors de la compilation et il est nécessaire de compiler le source une seconde fois pour que ces informations puissent être lues.

Par exemple, les trois titres de ce début de chapitre n'étaient pas tout à fait ceux que j'ai montrés ci-dessus. Sans mensonge cette fois, les titres ont été tapés comme suit :

```
\section{Quelques présentations plus évoluées}\label{presentation}
\subsection{Structuration des documents}\label{structure}
\subsubsection{Commandes pour le plan}\label{commandeplan}
```

Ainsi, la phrase :

```
Les commandes relatives au plan peuvent être vues à la
section~\ref{commandeplan} page~\pageref{commandeplan}.
```

sera composée comme suit :

Les commandes relatives au plan peuvent être vues à la section 6.1.1.1 page 62. (Vous pouvez vérifier, je suis certain du résultat puisque ce n'est pas moi qui ai écrit les numéros.)

## 6.1.2 Table des matières, index

### Table des matières... rien de plus simple

Comme on vient de le voir, la table des matières est on ne peut plus simple à produire : il suffit de taper la macro `\tableofcontents` au niveau où on veut que celle-ci soit écrite et d'avoir pris soin de construire le plan avec les macros vues à la section 6.1.1.1.

Notons que dans la typographie française, la table des matières doit se trouver à la fin de l'ouvrage. Un sommaire avec les titres des chapitres se trouve lui au début de l'œuvre. Nous pouvons obtenir ceci avec l'extension `\usepackage{shorttoc}`.

De même, les sommaires en début de chapitre ont été obtenus avec l'extension `\usepackage{minitoc}`.

Nous détaillerons la syntaxe exacte dans le chapitre " extensions "9.

### Index

La construction d'un index est un peu plus délicate et je ne décrirai pas toutes les possibilités. En premier lieu, il faut dire que le document comportera un index en appelant l'extension `makeidx` avec la syntaxe :

```
\usepackage{makeidx}
```

puis spécifier, également dans le préambule la macro `\makeindex`. L'index proprement dit est construit grâce à la macro `\printindex` placée là où on veut que l'index soit écrit.

Pour mettre un mot dans un index, il suffit d'employer la macro `\index` suivie par un groupe donnant le mot (ou la suite de mots) à indexer. Cette macro présente des syntaxes différentes permettant des comportements variés. Pour ne pas alourdir l'exposé plus que de raison, je n'indiquerai que la plus utile. Avant de la voir, il faut comprendre le mécanisme avec lequel est construit l'index.

Lors de la compilation, à chaque fois que  $\text{\LaTeX}$  voit la macro `\index`, il écrit le mot accompagné de son numéro de page dans un fichier auxiliaire. Ce fichier ne peut pas encore servir à fabriquer directement l'index car les mots sont rangés dans l'ordre de leur apparition dans le texte alors qu'un index donne un classement alphabétique. Ce tri est assuré par un programme externe (*makeindex*) qu'il faut donc appeler en donnant le nom du document en paramètre. Ce programme externe construit un autre fichier où les entrées ont été classées par ordre alphabétique et où les numéros de page ont été regroupés pour chaque entrée. Une seconde compilation produira alors un index correct.

Maintenant, supposons que l'on veuille faire apparaître le symbole  $\zeta$  au niveau de l'index tout en voulant le classer comme s'il s'agissait du mot " zeta " (entre zester et zétète en quelque sorte). Écrire `\index{\(\zeta\)}` ne produira pas ce résultat car  $\zeta$  sera classé au niveau des mots commençant par le caractère contre-oblique (très au début de l'index, avant toutes les lettres romaines). Pour pouvoir " tromper " le programme chargé du tri, il suffit de taper `\index{zeta@(\zeta)}` où le caractère @ sert à séparer la partie gauche qui servira de référence pour l'ordre alphabétique de la partie droite qui sera affichée réellement au niveau de l'index. Cette façon de faire est très utile lorsqu'on met des mots accentués dans un index car le programme de tri ne range pas les lettres accentuées au même niveau que les autres lettres.

### 6.1.3 En-têtes, pieds de page, notes

#### Faire des pieds de... page

Nous allons voir tous les éléments d'une page qui ne font pas partie du corps de texte, c'est-à-dire tout ce qui se trouve dans les marges. L'en-tête se situe au niveau de la marge supérieure, le pied de page et les notes de bas de page au niveau de la marge inférieure et les notes marginales au niveau des marges gauche et droite (intérieure et extérieure dans le cas d'un document composé en recto-verso).

#### Types d'en-tête et de pied de page

Par défaut,  $\text{\LaTeX}$  propose quatre types d'en-tête et pied de page :

**empty** l'en-tête et le pied de page sont vides ;

**plain** l'en-tête est vide, le numéro de page est centré au niveau du pied de page ;

**headings** l'en-tête donne des renseignements sur le titre de chapitre (et/ou de section) courant ainsi que le numéro de page (c'est le style qui a été choisi pour ce manuel) ;

**myheadings** l'en-tête est personnalisée grâce à des macros spéciales (nous ne parlerons pas de ce type dans ce manuel).

#### types d'en-tête et pied de page selon la classe

En réalité, les pages de début de chapitre et/ou de section modifient l'en-tête et le pied de page automatiquement, les règles étant différentes en fonction de la classe choisie pour le document (*book*, *report* ou *article*). Là aussi, nous ne lancerons pas dans l'étude détaillée de tous les cas possibles ; il suffit de laisser faire  $\text{\LaTeX}$  pour être sûr d'avoir une présentation acceptable !



La macro permettant d'indiquer la forme voulue est `\pagestyle` suivi d'un groupe indiquant le type choisi. Normalement, cette indication est donnée au niveau du préambule puisqu'elle doit porter sur l'ensemble du document. Ainsi, la commande :

```
\pagestyle{headings}
```

a été écrite dans le préambule de ce manuel.

### **thispagestyle**

On peut vouloir modifier localement cette présentation. Pour cela, il suffit d'appeler `\thispagestyle` suivie, là aussi, d'un groupe indiquant le type choisi pour la page courante. Seule la page où se situe cette macro prendra le style précisé, les pages suivantes reprenant le style indiqué avec la macro générale `\pagestyle`.

### **exception**

Il existe une petite exception (importante en pratique) pour la première page du document. Si on veut obtenir un style particulier pour cette première page, il faut taper la macro `\thispagestyle` au niveau du préambule. Par exemple, l'auteur de ce manuel a tapé la commande :

```
\thispagestyle{empty}
```

juste avant le `\begin{document}` afin d'obtenir une page de titre sans en-tête ni pied de page.

### **numérotation des pages**

Après la page de couverture, ce manuel présente une table des matières dont les pages sont numérotées en minuscules romaines puis le texte principal commence en passant à une numérotation arabe et en revenant à la page 1. Tout ceci est facile à mettre en œuvre grâce à la macro `\pagenumbering` suivi du type de numérotation désirée :  $\TeX$  définit 5 types de numérotation :

**arabic** donne les numéros 1, 2, 3, 4, ... ;

**roman** donne les numéros i, ii, iii, iv, ... ;

**Roman** donne les numéros I, II, III, IV, ... ;

**alph** donne les numéros a, b, c, d, ... ;

**Alph** donne les numéros A, B, C, D, ... ;

De plus, un appel à la macro `\pagenumbering` remet le compteur de page à 1.

Ainsi, dans la première version de ce manuel, une fois que la page de titre a été composée, les trois lignes suivantes étaient :

```
\newpage\pagenumbering{roman}
\tableofcontents
\newpage\pagenumbering{arabic}
```

Traduisons ces trois lignes en français :

- commencer une nouvelle page (`\newpage`) ;
- passer alors en numérotation romaine en commençant à " i " (`\pagenumbering{roman}`) ;
- construire la table des matières (`\tableofcontents`) ;
- commencer une nouvelle page (`\newpage`) ;
- passer alors en numérotation arabe en commençant à " 1 ".

Dans la version " UREM-U.L.B. ", cependant, cette manière de faire étant trop "  $\TeX$  " et comme nous avons préféré produire un livre, nous donc avons utilisé les commandes suivantes :

- `\frontmatter` pour la préface et la table des matières,

- `\mainmatter` pour le corps de l'ouvrage,
  - `\appendix` pour les listes de tableaux et d'illustration et
  - `\backmatter` pour la bibliographie.
- Ceci est plus dans la logique "  $\text{\LaTeX}$  " et aboutit quasiment au même effet.

### notes de bas de page

Il nous reste à voir les notes de bas de page<sup>2</sup>

Comme et les notes marginales.

celle-  
ci.

Les notes de bas de page sont appelées grâce à la macro `\footnote` suivie d'un groupe donnant le contenu de la note. En particulier le numéro de l'appel est géré de façon automatique (par défaut, il est remis à 1 au début de chaque section avec la classe `article` ou au début de chaque chapitre avec les classes `book` et `report`. Quant aux notes marginales, elles sont appelées avec la macro `\marginpar` et sont suivies, là aussi, d'un groupe indiquant leur contenu.

Les notes de bas de page, comme leur nom l'indique, sont placées en bas de page, séparées du corps de la page par un filet (tout ceci étant bien sûr paramétrable). Les notes marginales sont écrites dans la marge en face du paragraphe dans lequel la macro a été appelée. Ainsi, le début du paragraphe précédent a été tapé de cette façon :

Il nous reste à voir les notes de bas de page `\footnote{Comme celle-là.}` et les notes marginales `\marginpar{Comme celle-ci.}`. Les notes de bas de page sont...

---

2. Comme celle-là.

# Chapitre 7

## Commandes de Base : Fontes, espaces

### Sommaire

---

<b>7.1 Mises en pages via les fontes, les espaces, ...</b> . . . . .	<b>68</b>
7.1.1 Commandes de fonte . . . . .	68
7.1.2 Commandes de paragraphe . . . . .	70
7.1.3 Espacements . . . . .	74

---

## 7.1 Mises en pages via les fontes, les espaces, ...

### 7.1.1 Commandes de fonte

Une fonte est l'ensemble des caractères présentant une forme commune (même taille, même type de dessin) ; le mot de " police " est un néologisme informatique qu'il vaut mieux éviter. On peut classer les opérations sur les fontes en deux grandes catégories : les macros qui vont agir sur la taille du texte et les macros qui vont agir sur le type des caractères (italique, romain, sans empattement, etc.).

#### 7.1.1.1 Taille des caractères

Si on ne dit rien du tout, les caractères sont pris dans la fonte Computer Modern (inventée par Knuth et ressemblant un peu à la fonte Times) en taille 10. C'est ce que vous avez sous les yeux en ce moment.

Pour changer de taille de caractère,  $\LaTeX$  propose 10 macros présentées dans le tableau 7.1 : Attention à leur utilisation. Si on met une de ces macros dans le source, tout le texte qui va suivre sera

Macro	Nom français	Résultat
<code>\tiny</code>	minuscule	Voilà
<code>\scriptsize</code>	taille scripte	Voilà
<code>\footnotesize</code>	taille note de bas de page	Voilà
<code>\small</code>	petit	Voilà
<code>\normalsize</code>	taille normale	Voilà
<code>\large</code>	grand	Voilà
<code>\Large</code>	Grand	Voilà
<code>\LARGE</code>	GRAND	Voilà
<code>\huge</code>	énorme	Voilà
<code>\Huge</code>	Énorme	Voilà

TABLE 7.1 – Macros de taille de caractère

de la taille spécifiée. Une façon naturelle de revenir à la taille précédente serait d'écrire `\normalsize` mais ce n'est pas une si bonne idée car la taille précédente n'était pas forcément la taille normale et, en plus, c'est long à écrire ! Une façon beaucoup plus propre et rapide est d'utiliser un groupe. Petit exemple pour bien comprendre :

**Jamais !** Vous entendez ? **Jamais** je ne le ferai.  
D'accord. Ne vous fâchez pas.

```

1  {\Large Jamais !} Vous entendez ?
2  {\LARGE Jamais} je ne le ferai.
3
4  {\tiny D'accord.} Ne vous f^achez pas

```

#### 7.1.1.2 Types de fonte

Il existe 9 paires de macros permettant de spécifier le type de caractère voulu. Ces macros se répartissent en trois groupes : les macros indiquant la forme (droit (up), italique (italic), penché

(slanted) et petite capitale (small cap)), les macros indiquant la graisse (moyen (medium) ou gras (boldface)) et les macros indiquant la famille (romain (roman), sans empattement (sans serif) et non proportionnel dite type machine à écrire (typewriter)). Le tableau 7.2 présente ces 18 macros.

Macro I	Macro II	Résultat
Macros pour la forme		
<code>\textup</code>	<code>\upshape</code>	Un exemple pour voir
<code>\textit</code>	<code>\itshape</code>	<i>Un exemple pour voir</i>
<code>\textsl</code>	<code>\slshape</code>	<i>Un exemple pour voir</i>
<code>\textsc</code>	<code>\scshape</code>	UN EXEMPLE POUR VOIR
Macros pour la graisse		
<code>\textmd</code>	<code>\mdseries</code>	Un exemple pour voir
<code>\textbf</code>	<code>\bfseries</code>	<b>Un exemple pour voir</b>
Macros pour la famille		
<code>\textrm</code>	<code>\rmfamily</code>	Un exemple pour voir
<code>\textsf</code>	<code>\sffamily</code>	Un exemple pour voir
<code>\texttt</code>	<code>\ttfamily</code>	Un exemple pour voir

TABLE 7.2 – Macros de type de caractère

Les premières lignes de chaque série de macros (`\textup`, `\textmd`, `\textrm` et les macros de type II correspondantes) sont les types par défaut des caractères.

La différence entre les macros de type I et celles de type II porte sur la façon d'indiquer ce qui va être affecté par le changement. Les macros de type II se comportent comme les macros de changement de taille vues précédemment. Les macros de type I n'affectent que ce qui suit immédiatement. Voici un exemple pour bien comprendre :

Qui, sur la terre, voudrait utiliser du **gras** sans empattement ?  
I want **that** !

```

1  {\sffamily {\bfseries Qui}, sur la
   terre,
2  voudrait utiliser du {\bfseries gras
   }
3  sans empattement ?}
4
5  \textbf I want \textbf that !

```

On notera la possibilité de mélanger les styles et l'utilisation de groupes à l'intérieur de groupes pour limiter certaines actions comme il a été fait au premier paragraphe.

Les macros de type I semblent assez peu utiles. Dans le second paragraphe, seul le " t " du mot " that " a été mis en gras, et écrire :

```
\textbf t\textbf h\textbf a\textbf t
```

pour obtenir le résultat voulu (" that " entièrement en gras) est visiblement idiot. En fait, là aussi, il faut penser au groupe.

I want **that**

```
1  \textbf I want \textbf{that}
```

C'est le groupe entier qui a été maintenant mis en gras. Du coup, l'emploi des macros de type I ou de type II est plus ou moins une affaire de goût puisque la seule différence va être de placer la macro soit juste avant le groupe (type I), soit à l'intérieur du groupe (type II). La pratique montre que les utilisateurs choisissent souvent le type I pour de petits textes (quelques mots) et le type II pour des passages plus importants. Le stage de perfectionnement en indiquera la raison.

Notez que la déclaration `\textbfthat` aurait conduit à une erreur de compilation puisque l'ordinateur aurait cherché à savoir ce que devait faire la macro dont le nom est `textbfthat` et que cette prétendue macro n'existe pas !

Il existe une macro particulière qui se charge d'écrire un texte en italique à condition que les caractères en cours ne soit justement pas en italique. Si le texte en cours est composé en italique, cette macro remet le texte droit. Il s'agit de la macro `\emph` (de *emphasis* en anglais) et son comportement est exactement celui qu'il faut avoir lorsqu'on désire mettre un mot en évidence dans un passage.

<p>Dans le livre <i>TEX</i> est meilleur <i>que les autres</i>.          Nous verrons que <i>TEX</i> est <i>meilleur</i> que les autres.</p>
--

```

1 Dans le livre \textit{\LaTeX{}} est
2   \emph{meilleur} que les autres}.
3
4 Nous verrons que \LaTeX{} est
5   \emph{meilleur} que les autres.
```

## 7.1.2 Commandes de paragraphe

Un paragraphe est, par défaut, indenté et justifié (les bords gauche et droit étant alignés avec les marges gauche et droite). Il est quelquefois souhaitable de ne pas obtenir cette présentation et *TEX* offre un choix plantureux de commandes permettant d'obtenir des effets très spéciaux. Il est hors de question de faire le tour de la question et nous nous contenterons de citer les macros les plus utiles.

Avant de voir ces macros, il est bon de savoir qu'un paragraphe, pour *TEX*, est un texte qui se termine soit par une ligne vide (comme on a déjà vu), soit par la macro `\par`<sup>1</sup>.

Il existe quatre présentations classiques de paragraphe : justifié (ce qui est la présentation par défaut), au fer à gauche<sup>2</sup> (seul le bord gauche du paragraphe est aligné avec la marge), au fer à droite (le bord droit du paragraphe aligné avec la marge de droite) et centré. Pour obtenir les trois dernières présentations, il existe à chaque fois une macro et un environnement. Voyons les environnements en premier :

---

1. En réalité, il y a d'autres possibilités que nous ne verrons pas dans ce manuel.

2. La dénomination exacte serait plutôt " en drapeau au fer à gauche " ce qui est plus explicite.

Voici un paragraphe normal (avec une indentation de première ligne nulle pour gagner de la place). Cet exemple suffit à voir l'alignement des bords gauche et droit.

Voici un paragraphe au fer à droite où l'alignement ne se fait plus qu'au niveau du bord droit. La syntaxe anglaise peut être traduite par *aligné à droite*.

Un paragraphe au fer à gauche. La syntaxe anglaise est évidemment cohérente en remplaçant droite par gauche.

Paragraphe centré. Seules les lignes relativement courtes ont un intérêt sinon, on ne voit pas très bien le caractère centré de la chose !

```

1  Voici un paragraphe normal (avec une
2  indentation de premi\`ere ligne nulle
   pour
3  gagner de la place). Cet exemple
   suffit
4  \`a voir l'alignement des bords gauche
   et
5  droit.
6
7  \begin{flushright}
8  Voici un paragraphe au fer \`a
   droite o\`u
9  l'alignement ne se fait plus qu'au
10  niveau du bord droit. La syntaxe
11  anglaise peut \^etre traduite par
12  \emph{align\`e\`a~droite}.
13  \end{flushright}\par % pour changer un
   peu
14  \begin{flushleft}
15  Un paragraphe au fer \`a gauche. La
   syntaxe
16  anglaise est \`evidemment coh\`
   erente en
17  rempla\c{c}ant droite~par~gauche.
18  \end{flushleft}
19
20  \begin{center}
21  Paragraphe centr\`e. Seules les
   lignes
22  relativement courtes ont un int\`er
   \^et sinon,
23  on ne voit pas tr\`es bien le caract
   \`ere
24  centr\`e de la chose !
25  \end{center}

```

Les macros permettant d'obtenir ces présentations sont respectivement `\raggedleft`, `\raggedright` et `\centering`. On notera l'inversion de *left* et *right*. Cela s'explique par la traduction de *flush* qui veut dire ici *aligné* et de *ragged* qui signifie *déchiqueté* ; si un paragraphe est aligné d'un côté, il est déchiqueté de l'autre !

Attention, ces macros portent sur tout ce qui suit, il est donc nécessaire de les inclure dans des groupes si on ne veut pas que leurs actions se poursuivent jusqu'à la fin du document.

Voici un paragraphe au fer à droite où l'alignement ne se fait plus qu'au niveau du bord droit. La syntaxe anglaise peut être traduite par *déchiqueté* à gauche. Un paragraphe au fer à gauche. La syntaxe anglaise est évidemment cohérente en remplaçant gauche par droite. Paragraphe centré. Seules les lignes relativement courtes ont un intérêt sinon, on ne voit pas très bien le caractère centré de la chose !

```

1  {\raggedleft Voici un paragraphe au
   fer \'a
2  droite o\'u l'alignement ne se fait
   plus
3  qu'au niveau du bord droit. La
   syntaxe
4  anglaise peut \^etre traduite par
5  \emph{d\'echiquet\'e\'~\'a~gauche}.\
   par}
6  {\raggedright Un paragraphe au fer \'a
   gauche.
7  La syntaxe anglaise est \'evidemment
8  coh\'erente en rempla\c{c}ant gauche
   ~par~droite.
9  \par}
10 {\centering Paragraphe centr\'e'.
    Seules les
11  lignes relativement courtes ont un
    int\'er\^et
12  sinon, on ne voit pas tr\'es bien le
13  caract\'ere centr\'e de la chose !\
    par}

```

On notera que le paragraphe a été déclaré à l'intérieur des accolades externes. En effet, ces macros ne portent que sur des paragraphes (c'est logique) donc n'auront aucune action sur une portion de texte qui n'est pas un paragraphe.

Exemple qui tombe à l'eau.  
C'est mieux ainsi !

```

1  {\centering Exemple qui tombe \'a l'
   eau.}\par
2  {\centering C'est mieux ainsi !}\par}

```

La dernière chose que nous allons voir ici à propos de la forme des paragraphes est l'indentation et le saut vertical entre deux paragraphes.

Les paragraphes commencent par défaut avec un retrait d'alinéa au niveau de leur première ligne. On peut inhiber cette fonction en faisant précéder le paragraphe de la macro `\noindent` comme dans le paragraphe qui suit immédiatement. On remarquera aussi qu'à partir de ce paragraphe, l'indentation n'est plus la même que d'habitude ainsi que le saut vertical entre les paragraphes. Le comportement normal sera retrouvé à la prochaine section.

Ce retrait peut être défini par l'utilisateur. Normalement, ce type de déclaration se fait au niveau du préambule puisqu'il est rare de changer d'indentation au milieu d'un document. Ainsi, ce manuel a défini une indentation de 8 mm pour l'ensemble du texte mais depuis le paragraphe précédent et jusqu'à la fin de cette section, l'indentation (horrible) a été portée à 3 cm. Ces définitions (ou redéfinitions) ne se font pas à partir d'une macro mais avec ce que  $\text{\LaTeX}$  appelle des *dimen* (dimensions). Pour modifier une dimension, il suffit d'employer la syntaxe :

```
\setlength{dimen}{longueur}
```

$\text{\TeX}$  étant très scrupuleux, il permet d'utiliser toutes les unités typographiques usuelles anglo-saxonnes et françaises ainsi que les unités courantes du système internationale plus une unité à lui qui est le point d'échelle et trois unités qui dépendent de la fonte en cours. Le tableau 7.3 montre toutes les unités à disposition.



unité	nom anglais	nom français	correspondances	
<b>1 bp</b>	big point	gros point	= 1,003 74 pt = 0,352 68 mm = 0,938 06 dd	= 65 781 sp = 1,388 55 × 10 <sup>-2</sup> in
<b>1 cc</b>	cicero	cicéro	= 12,840 1 pt = 4,512 51 mm = 12 dd	= 841 789 sp = 0,177 658 in
<b>1 cm</b>	centimeter	centimètre	= 28,452 74 pt = 10 mm = 26,591 dd	= 1 864 679 sp = 0,393 692 in
<b>1 dd</b>	didot point	point didot	= 1,07 pt = 0,376 066 mm	= 70 124 sp = 1,480 865 × 10 <sup>-2</sup> in
<b>1 in</b>	inch	pouce	= 72,27 pt = 25,4 mm = 67,541 5 dd	= 4 736 286 sp
<b>1 pc</b>	pica	pica	= 12 pt = 4,217 18 mm = 11,214 86 dd	= 786 432 sp = 0,166 031 in
<b>1 pt</b>	point	point		= 65 536 sp = 1,382 446 × 10 <sup>-2</sup> in
<b>1 mm</b>	millimeter	millimètre	= 2,845 26 pt = 2,659 09 dd	= 186 467 sp = 3,936 768 × 10 <sup>-2</sup> in
<b>1 sp</b>	scale point	point d'échelle	= 1,528 78 × 10 <sup>-5</sup> pt = 5,362 851 × 10 <sup>-6</sup> mm = 1,426 × 10 <sup>-5</sup> dd	= 2,109 × 10 <sup>-7</sup> in
<b>1 em</b>	em-space	cadratin	Il s'agit normalement de la largeur de la lettre " M " de la fonte courante mais il s'agit en fait d'une unité déclarée par le concepteur de la fonte qui a le droit de faire ce qu'il veut.	
<b>1 ex</b>	ex-space	hauteur de 'x'	Il s'agit normalement de la hauteur de la lettre " x " de la fonte courante. La remarque a propos de l'unité <b>em</b> s'applique.	
<b>1 mu</b>	math unit	unité mathématique	Correspond à 1/18 <sup>e</sup> de cadratin dans la fonte mathématique courante. Cette unité n'a de sens qu'en mode mathématique.	

TABLE 7.3 – Unités de mesures

La dimension indiquant l'importance de l'indentation est `\parindent` et la syntaxe `\TeX` qui a permis de la spécifier à 3 cm est :

```
\setlength{\parindent}{3cm}
```

En plus de l'indentation, on peut également régler les sauts verticaux entre paragraphes. Là aussi, il existe une dimension qui indique cette distance, en l'occurrence `\parskip`. On

peut évidemment procéder au même type de réglage que pour l'indentation mais, pour cette dimension, on a tout intérêt à utiliser une caractéristique fondamentale des dimensions de  $\text{\TeX}$  : il s'agit en fait de ressorts comportant un certain étirement et compression plutôt que de longueurs fixes. Le mot-clé `plus` introduit l'étirement et le mot-clé `minus` introduit la compression.

Par exemple, dans cette fin de section, la dimension `\parskip` a été fixée à :

```
\setlength{\parskip}{18pt plus4pt minus2pt}
```

c'est-à-dire que l'espace entre deux paragraphes est normalement de 18 points mais qu'il peut en fait varier entre 22 points (18 + 4) et 16 points (18 – 2). En dehors de cette fin de section, dans le reste du manuel, cette dimension a été fixée au niveau du préambule par la syntaxe :

```
\setlength{\parskip}{4pt plus2pt minus1pt}
```

ce qui fait que les paragraphes sont beaucoup moins espacés qu'en ce moment.

Le fait de n'avoir spécifié aucun étirement et compression pour la dimension `\parindent` fait que ces deux composantes sont nulles. Ainsi, l'indentation de paragraphe sera toujours exactement de 3 cm (8 mm dans le reste du manuel). C'est un comportement souhaitable pour l'indentation mais malheureux pour l'espace entre les paragraphes si on souhaite aligner les bas de pages les uns avec les autres.

À partir de maintenant, les paragraphes retrouvent leur indentation et leur saut de paragraphe habituels.

### 7.1.3 Espacements

Comment indiquer à  $\text{\TeX}$  de mettre un certain espace vertical ou horizontal à un emplacement donné ? La technique horrible et très « wordienne » consistant à taper une ribambelle d'espaces pour un grand espace horizontal ou de sauts de paragraphe pour un grand espace vertical et ici totalement inopérante et c'est tant mieux : rappelons que plusieurs espaces au niveau du source seront traduits comme étant un seul espace et que plusieurs lignes vides et/ou commandes `\par` consécutives seront traduites comme étant une seule commande `\par`.

Les deux commandes de base permettant d'obtenir de telles espacements sont `\hspace` et `\vspace` ; la première pour les espacements horizontaux et la seconde pour les espacements verticaux. On fera suivre ces commandes d'un groupe indiquant la longueur voulue. Il devrait être clair que les commandes `\hspace` et `\vspace` n'auront d'action respectivement qu'à l'intérieur d'un paragraphe et qu'à l'extérieur d'un paragraphe (réfléchissez).

<p>Un            espace important</p>   <p>Un saut vertical important.</p>	<pre>1  Un\hspace{1.2cm}espace important 2 3  \vspace{1.2cm} 4  Un saut vertical important.</pre>
---	---

En réalité, les dimensions indiquées comme paramètre de ces deux commandes peuvent comporter une composante d'étirement introduite par le mot `plus` ainsi qu'une composante de compression introduite par le mot `minus`. On parle alors de « ressort » plutôt que de « longueur ». Voici un petit exemple pour bien comprendre.

Un	texte	difficile	à	compo-
ser à cause des contraintes d'espacement.				
Un	texte	difficile	à	composer
à cause des contraintes d'espacement.				

```

1 Un\hspace{1cm}texte\hspace{1cm}
   difficile%
2 \hspace{1cm}\`a\hspace{1cm}composer \`
   a cause
3 des contraintes d'espacement.
4
5 Un\hspace{1cm minus4mm}texte%
6 \hspace{1cm minus4mm}difficile%
7 \hspace{1cm minus4mm}\`a%
8 \hspace{1cm minus4mm}composer \`a
9 cause des contraintes d'espacement.

```

Dans les deux cas, le résultat est évidemment assez laid mais c'est ce qu'on a demandé. Dans le second cas, les espaces indiqués doivent être normalement de 1 cm mais peuvent se comprimer jusqu'à 6 mm. Pourquoi, dans le second cas, les espaces n'ont pas été resserrés au maximum ? Il faut bien comprendre que la spécification « 1cm minus4mm » indique que la largeur doit *normalement* être de 1 cm ; ici, c'est à cause de la césure que  $\text{\LaTeX}$  a préféré réduire un peu cette largeur : il considère qu'une césure n'est pas une bonne chose (en l'occurrence, une césure est moins bien qu'une compression de ressort). S'il n'y avait pas eu de césure, c'est la largeur naturelle (1 cm) qui aurait été gardée.

Tout ce qui vient d'être dit pour le cas horizontal fonctionne de la même façon pour le cas vertical. Le pendant de la césure, dans le cas vertical, est la coupure d'un paragraphe en fin de page.

Les commandes `\hspace` et `\vspace` sont encore plus puissantes que ce qui vient d'être dit : lorsqu'une commande `\hspace` survient en début de ligne ou en fin de ligne, son rôle est inhibé. De même, une commande `\vspace` en début ou en fin de page n'a aucune action. C'est une bonne chose, par exemple, c'est ce qui permet de donner une présentation correcte aux en-têtes de section. De façon terriblement schématique, la commande `\section` dans ce document commence par les commandes suivantes :

```

\par
\vspace{3.5ex plus1ex minus.2ex}

```

Cela signifie que l'espacement vertical précédant une en-tête de section sera normalement de 3,5 ex mais peut en fait varier entre 3,3 ex et 4,5 ex. On trouve une définition similaire pour l'espace entre deux paragraphes (ressort `\parskip`) qui est défini de la façon suivante (définition due à l'auteur du manuel, il ne s'agit pas de la valeur par défaut) :

```

\setlength{\parskip}{4pt plus2pt minus1pt}

```

qui indique que l'espacement entre deux paragraphes sera normalement de 4 points mais pourra en fait varier entre 3 points et 6 points. (Voir section page pour la macro `\setlength`.)

À quoi tout cela peut bien servir ? Les variations sont finalement assez faible (1 point vaut environ un tiers de millimètre) et n'auront pas beaucoup d'influence pour éviter les coupures au milieu d'un paragraphe en fin de page. La raison de cette laxité dans les espacements verticaux est de permettre à  $\text{\LaTeX}$  de remplir entièrement la page, c'est-à-dire de commencer et terminer toutes les pages au même niveau. En pratique, certaines pages étaient trop difficiles à composer en raison de la présence de blocs verticaux importants (tableaux par exemple) mais dans l'ensemble,  $\text{\LaTeX}$  ne s'en est pas trop mal sorti !

Que faire si on veut absolument garder l'espacement, qu'on soit en début ou en fin de ligne dans le cas horizontal ou bien en début ou en fin de page dans le cas vertical ?  $\text{\LaTeX}$  met à disposition les deux commandes `\hspace*` et `\vspace*` qui se comportent strictement comme les commandes non étoilées correspondantes tout en gardant l'espacement quelque soit la configuration du texte. Ces commandes sont à employer en sachant bien ce qu'on veut !

$\LaTeX$  permet d'utiliser des ressorts tout à fait spéciaux qui présentent des composantes infinies ! Comment comprendre ce que  $\LaTeX$  appelle des infinis ? Le mieux est de se dire qu'un ressort infini rend caduque tout ressort fini. Les deux commandes que nous aborderons dans ce manuel sont `\hfill` et `\vfill`. Commençons par un exemple illustrant le comportement de la commande `\hfill` :

Un		texte.
Un	autre	texte.
Un	autre	texte.

```

1 Un\hfill texte.
2
3 Un\hfill autre\hfill texte.
4
5 Un\hfill autre\hfill\hfill texte.%

```

Sur ces, on peut voir que le texte s'est étiré sur toute la largeur de la ligne. En prenant une règle, on pourra vérifier que sur la deuxième ligne, les mots « Un », « autre » et « texte » sont séparés par exactement le même espacement. Plus fort, sur la troisième ligne, il y a deux fois plus d'espace entre « autre » et « texte » qu'entre « Un » et « autre ».

Bien sûr, cette façon de répartir régulièrement du texte au niveau d'une ligne a son pendant vertical avec l'emploi de `\vfill` qui permet de répartir régulièrement des paragraphes dans une page.

Voici un exemple servant à placer un mot sur le premier tiers d'une ligne, la seconde ligne est là pour faire voir que tout fonctionne correctement.

	Super			
Super	Super	Super	Super	

```

1 {}\hfill Super\hfill\hfill{}
2
3 Super\hfill Super\hfill Super\hfill
   Super%

```

La présence des groupes vides est nécessaire car ces commandes en début de ligne ou en fin de ligne n'ont pas d'action (comme `\hspace` et `\vspace`). Il faut donc placer quelque chose qui n'ai aucune répercussion au niveau du document final : le groupe vide est un candidat idéal !

## **Deuxième partie**

### **Deuxième atelier : tableaux et mises en page avancés**



# Chapitre 8

## Tableaux

### Sommaire

---

<b>8.1 Tableaux et mise en page mathématique . . . . .</b>	<b>80</b>
8.1.1 Tableaux . . . . .	80

---

## 8.1 Tableaux et mise en page mathématique

### 8.1.1 Tableaux

#### Les tableaux

Ah ! Les tableaux ! C'est en partie à l'aisance avec laquelle un utilisateur construit un tableau compliqué qu'on reconnaît un expert de  $\text{\LaTeX}$ .

Autant dire qu'il s'agit là d'une notion un peu délicate à manipuler. C'est logique : construire un tableau est quelque chose de très subtil qui nécessite souvent une bonne dose de réflexion. Dans cette section, nous ne verrons que les macros définies directement par  $\text{\LaTeX}$  ; quelques présentations un peu plus compliquées seront présentées lors de l'étude des extensions (Cf. section 10.2.1).

Un tableau est défini grâce à l'environnement `tabular`

Le groupe qui suit le `\begin{tabular}` indique le motif du tableau (nombre et types des colonnes).

Il y a quatre types de base pour les colonnes :

**l** indique une colonne calée à gauche (**left**) ;

**r** indique une colonne calée à droite (**right**) ;

**c** indique une colonne centrée (**center**) ;

**p** $\{d\}$  indique une colonne qui peut accueillir des paragraphes dont la largeur est donnée par la dimension  $d$  (Cf. tableau 7.3 page 73 pour les unités acceptées).

#### Premier exemple de tableau

Nous allons commencer par un tableau très basique : 4 colonnes, une pour chaque type, sans réglures (les lignes horizontales et verticales). Le motif sera alors `lcrp{2cm}` et le tableau complet sera :

```
\begin{tabular}{lcrp{2cm}}
  contenu du tableau
\end{tabular}
```

Le *contenu du tableau* va indiquer le texte qui va apparaître dans chaque cellule du tableau.

Pour passer d'une cellule à une autre, on emploie le caractère spécial `&` et pour passer à la ligne suivante la séquence de caractères `\\`.

Voici deux exemples :

g	c	d	paragraphe de deux centi- mètres
xx	yyy	zz	ppp

```
1 \begin{tabular}{lcrp{2cm}}
2   g & c & d & 
3   paragraphe de deux centim{\e}tres
4   \\
5   xx & yyy & zz & ppp
\end{tabular}
```



g	c	d	paragraphe de deux centi- mètres
xxx	yyy	zzz	pppppp

```

1 \begin{tabular}{lcrp{2cm}}
2 g & c & d &
3 {\raggedright paragraphe de
4 deux centim{\‘e}tres\par} \\
5 xxx &
6 yyy &
7 zzz &
8 pppppp
9 \end{tabular}

```

Remarquons que les deux tableaux ont exactement le même motif (`lcrp{2cm}`). Nous pouvons voir que ce motif ne permet pas d’avoir des tableaux de largeur fixe : le second tableau est plus large que le premier. Pour les colonnes de type l, c et r la largeur sera la largeur de la cellule la plus importante (plus les espaces intercolonnes). En revanche, le rôle même du type p est d’avoir une taille fixe.

### Problème ...

Sans précaution particulière, le motif p produit des résultats peu esthétiques car sa largeur est généralement très faible et le paragraphe est construit par défaut pour être justifié ce qui est incompatible (voyez le piètre résultat du premier tableau).

g	c	d	paragraphe de deux centi- mètres
xx	yyy	zz	ppp

### ... et solution

La solution est, souvent, de forcer la composition au fer à gauche grâce à la macro `\raggedright`. N’oublions pas alors de faire agir cette macro sur un vrai paragraphe (obtenu ici grâce à la macro `\par`).

g	c	d	paragraphe de deux centi- mètres
xxx	yyy	zzz	pppppp

La présence des accolades peut sembler mystérieuse ; sans celles-là, il y aurait eu une erreur à la compilation car la macro `\par` indique de passer à la ligne suivante mais il s’agit d’une commande interdite lorsqu’on se trouve entre deux paragraphes. Les accolades permettent de créer un paragraphe à l’intérieur sans qu’il soit visible à l’extérieur : tordu n’est-ce pas ? C’est le genre de petite feinte qu’il vaut mieux connaître pour ne pas perdre trop de temps lors des premiers pas sous  $\LaTeX$ .

Petit détail sans trop d’importance : la dernière ligne d’un tableau n’a pas besoin de la séquence `\par`, l’environnement en ajoute une de façon automatique si l’utilisateur ne l’a pas spécifié. Plus important : on peut terminer une ligne avant que toutes les colonnes aient été remplies, les cellules correspondantes seront vides. Pour un tableau sans réglures, cela n’a aucune importance, mais avec des réglures, il faut faire attention à ce que l’on fait car ces dernières disparaîtront également (cela peut très bien être le résultat voulu).

### 8.1.1.1 Les réglures

#### Les réglures

Pour obtenir des tableaux qui en soient vraiment, il reste à gérer les réglures.

Les réglures *verticales*, celles séparant les colonnes, sont déclarées au niveau du motif avec le caractère |.

Les réglures *horizontales* sont déclarées dans le contenu du tableau avec la macro \hline qui fera suite à la fin de ligne.

Reprenons l'exemple précédent en ajoutant des réglures :

gauche	centré	droite	paragraphe de deux centi- mètres
xxx	yyy	zzz	ppp
xxx xxx	yyy yyy	zzz zzz	ppp ppp

```

1 \begin{tabular}{|l|c|r||p{2cm}|}
2 \hline
3 gauche & centr\`e & droite &
4 {\raggedright paragraphe de deux
5 centim\`etres\par}\\
6 \hline
7 xxx & yyy & zzz & ppp\\
8 \hline\hline
9 xxx xxx & yyy yyy & zzz zzz & ppp ppp\\
10 \hline
11 \end{tabular}

```

L'inclusion d'une triple ligne verticale n'est sans doute pas très heureuse : elle a été faite pour montrer que c'était possible ! De même, on peut réaliser une double réglure horizontale (ou triple, quadruple, ...) en répétant la macro \hline. Toujours dans le registre des points inesthétiques, on voit que le \par a introduit un espacement vertical parasite ; nous verrons lors du stage de perfectionnement comment y remédier.

### 8.1.1.2 multicolumn

#### multicolumn

On peut vouloir ne pas tout le temps suivre le motif par défaut du tableau : c'est extrêmement fréquent pour les lignes de titres qui peuvent être centrées alors que le reste de la colonne sera calé à gauche par exemple. De plus, il peut arriver qu'on veuille fusionner deux (ou plus) cellules adjacentes. Pour toutes ces manœuvres, la macro \multicolumn apporte la solution. Sa syntaxe est un peu lourde :

```
\multicolumn{nb}{motif}{texte}
```

où *nb* indique le nombre de colonnes devant être fusionnées (pour un changement de motif sans fusion, on prendra  $nb = 1$ ), *motif* est le motif de remplacement pour cette cellule et *texte* est le texte qui sera placé dans cette cellule.

Voici un premier exemple moins artificiel que les précédents où on ne fait que modifier le motif pour obtenir une ligne de titre.

On remarquera le petit jonglage au niveau des formats des macros `\multicolumn` pour garder les mêmes réglures que le tableau.

Corps	Ø (km)	densité
Soleil	1 392 000	1,409
Mercure	4 840	5,50
Vénus	12 390	5,25
Terre	12 760	5,517
Mars	6 800	3,94

```

1 \begin{tabular}{|l|r|l|}
2 \hline
3 \multicolumn{1}{|c|}{Corps} & & \\
4 \multicolumn{1}{c|}{\0\ (km)} & & \\
5 \multicolumn{1}{c|}{densit\'e} \\ \hline
6 Soleil & 1\,392\,000 & 1,409 \\
7 Mercure & 4\,840 & 5,50 \\
8 V\'enus & 12\,390 & 5,25 \\
9 Terre & 12\,760 & 5,517 \\
10 Mars & 6\,800 & 3,94 \\
11 \end{tabular}

```

La macro `\`, n'a rien à voir avec les tableaux : il s'agit d'une espace fine (on verra les différents types d'espaces à la section 10.1).

L'exemple suivant illustre l'utilisation de la macro `\multicolumn` pour fusionner plusieurs cellules. En raison de la taille nécessaire, exceptionnellement, le résultat sera présenté sous le source :

*Exemple 1.*

système RVB						
	couleur primaire			couleur secondaire		
nom	rouge	vert	bleu	jaune	magenta	cyan
composition	R	V	B	RV	RB	VB

### Répétition de motif

En premier lieu, on peut remarquer que le motif est un peu répétitif et on conçoit que pour un motif devant désigner 10 cellules centrées avec une ligne verticale entre chaque cellule, l'écriture :

```
|c|c|c|c|c|c|c|c|c|c|
```

sera un peu embêtante à taper, sans compter les risques d'erreur (taper 9 ou 11 cellules au lieu de 10).

### Raccourci pour la répétition

LaTeX permet un raccourci d'écriture sous la forme :

```
*{n}{motif à répéter}
```

où  $n$  est le nombre de répétitions désirées.

Par exemple :

```
|*{10}{c|}
```

qui indique que le motif sera constitué d'un " | ", puis d'une répétition de 10 fois le texte " c | " ce qui donnera bien le résultat souhaité. L'astérisque est suivi de deux groupes : le premier indique le nombre de répétitions et le second groupe le sous-motif qui devra être répété. Dans notre exemple, cela vaut à peine le coup puisque il n'y a que deux répétitions. Enfin ! Pour des raisons pédagogiques, voici la traduction du motif précédent :

```
|1|*{2}{|c|c|c|}
```

### Pour les esprits torturés

Un esprit torturé pourra remarquer que le motif répété est lui-même constitué d'une répétition de sous-motifs et pourra penser à écrire un motif du type :

```
|1|*{2}{|*{3}{c|}}
```

Cela marchera parfaitement mais on ne peut raisonnablement pas recommander ce genre d'acrobaties en raison de la lisibilité plus que douteuse du source pour un humain normalement constitué !

Plus important en pratique, le tableau précédent aurait pu être plus logiquement présenté sous la forme :

	système RVB					
	couleur primaire			couleur secondaire		
nom	rouge	vert	bleu	jaune	magenta	cyan
composition	R	V	B	RV	RB	VB

Pour cela, il va falloir supprimer certaines parties de quelques réglures. Pour les réglures verticales, il n'y a pas de problème particulier : il suffit d'employer `\multicolumn` en redéfinissant le motif de la cellule pour faire disparaître la réglure. Ainsi, dans l'exemple précédent, les deux cellules vides étaient définies de la façon suivante :

```
\multicolumn{1}{c|}{}&
```

Donc sans réglure verticale à gauche mais quand même la réglure verticale à droite et un texte vide (on notera que le choix de `c` au lieu de `l` ou `r` n'a strictement aucune importance).

Pour les réglures horizontales, il va falloir disposer d'une autre macro que `\hline` puisque celle-ci trace une réglure sur toute la largeur du tableau. La macro magique qui permet de réaliser des réglures horizontales partielles est `\cline` suivi par un groupe indiquant sur quelles cellules adjacentes doit s'étendre la réglure ; ceci est précisé avec la syntaxe `{colonne_début-colonne_fin}` où `colonne_début` et `colonne_fin` sont les numéros de colonne dans le tableau. Dans notre exemple, on veut avoir des réglures horizontales s'étendant de la colonne 2 jusqu'à la colonne 7.

Le source complet est :

```
\begin{tabular}{|l|c|c|c||c|c|c|}
\cline{2-7}
\multicolumn{1}{c|}{} &
\multicolumn{6}{c|}{système RVB} \\ \cline{2-7}
\multicolumn{1}{c|}{} &
\multicolumn{3}{c||}{couleur primaire} &
\multicolumn{3}{c|}{couleur secondaire} \\ \hline
nom & rouge & vert & bleu & jaune & magenta & cyan \\ \hline
composition & R & V & B & RV & RB & VB \\ \hline
\end{tabular}
```

### Un peu de magie

Cette section a été un peu longue mais patience, il ne reste plus qu'un point à découvrir sur la gestion de base des tableaux. Considérons le tableau suivant :

Le nombre 2 est	premier	Le nombre 3 est	premier
Le nombre 4 est non	premier	Le nombre 5 est	premier
Le nombre 6 est non	premier	Le nombre 7 est	premier
Le nombre 8 est non	premier	Le nombre 9 est non	premier
Le nombre 10 est non	premier	Le nombre 11 est	premier

Cela n'a rien de particulièrement excitant ! De plus, avec ce qui a été présenté jusqu'à maintenant, il est tout à fait possible de le composer même si cela va entraîner un côté un peu pénible.

En fait, le côté un peu magique de l'affaire et qui ne transparait pas au niveau de la sortie est que le corps de ce tableau a été tapé sous la forme :

```
2 &      & 3 &      \\
4 & non & 5 &      \\
6 & non & 7 &      \\
8 & non & 9 & non \\
10 & non & 11 &      \\
```

On voit que seules les éléments variables ont été spécifiés. Les " colonnes " présentant toujours le même matériel (texte ou espacement) ont été spécifiées au niveau du motif. La méthode consiste à remplacer l'espace intercolonne par ce qu'on veut grâce à la syntaxe `@{matériel}`. Cela ne pose pas trop de problème, la seule chose à bien se souvenir est qu'il s'agit d'un *remplacement* de l'espace intercolonne ce qui fait que celui n'existe plus et qu'il faudra prévoir les espaces nécessaires. Voyez, par exemple, le " est " entouré de deux espaces ; les espaces suivant ou précédant le caractère & disparaissent au niveau de la sortie ce qui fait qu'il est préférable de les spécifier au niveau du motif si on ne veut pas à avoir à taper une ribambelle de `\_` (macro espace).

Voici le motif qui a permis la construction du tableau précédent :

```
*{2}{@{Le nombre }l@{ est }r@{ premier\hspace{1cm}}}
```

Ceci peut être très utile pour des systèmes d'équations :

$2x + 0y + 3z = 5$
$4x + y + 5z = 0$
$6x + 3y + 7z = -2$
$8x + 7y + 9z = 4$
$10x + 3y + 11z = -5$

```
1 \begin{tabular}{r@{x + }r@{y + }r@{z = }
  r}
2 2 & 0 & 3 & 5 \\
3 4 & & 5 & 0 \\
4 6 & 3 & 7 & - 2 \\
5 8 & 7 & 9 & 4 \\
6 10 & 3 & 11 & - 5 \\
7 \end{tabular}
```



# Chapitre 9

## Extensions

### Sommaire

---

<b>9.1 Extensions utiles</b> . . . . .	<b>88</b>
9.1.1 geometry . . . . .	88
9.1.2 Extension pour les polices de caractères . . . . .	88
<b>9.2 Extensions spécialisées</b> . . . . .	<b>92</b>
9.2.1 gnuplottex . . . . .	92
9.2.2 answers . . . . .	92
9.2.3 pst-pdf . . . . .	92
9.2.4 showexpl . . . . .	92
9.2.5 wallpaper . . . . .	93
9.2.6 hyperref . . . . .	93
9.2.7 ifthen . . . . .	93

---

## 9.1 Extensions utiles

### Présentation des extensions

Ce chapitre présentera, de façon relativement succincte parfois, des extensions particulièrement utiles et assez souvent employées. Ces extensions (ou packages) font obligatoirement partie de toutes les distributions ; vous êtes donc assuré de les avoir si vous installez un système  $\text{\TeX}$  sur votre ordinateur.

Les extensions présentées sont celles que l’auteur utilise. Il existe assez souvent d’autres extensions permettant d’obtenir des résultats équivalents. Choisir telle ou telle extension est affaire d’habitude, de goût, de hasard, ...

#### 9.1.1 geometry

##### Présentation de geometry

Comme nous l’avons dit en début de manuel,  $\text{\TeX}$  a été construit par un américain pour composer des textes sur des feuilles américaines ce qui ne correspond pas à notre bon vieux format A4. Le package *geometry* permet de ne pas se perdre dans la jungle des registres de dimension permettant de régler les différents paramètres de la page. Son utilisation est très simple et les paramètres les plus importants sont très compréhensibles.

##### Syntaxe de geometry

Bien évidemment, il faudra charger ce package avec la commande :

```
\usepackage{geometry}
```

au niveau du préambule. Ce package fournit l’unique macro `\geometry` qui s’utilise en indiquant ce qu’on désire dans le groupe qui la suit.

Voici la liste des possibilités les plus utiles. Chaque demande devra être séparée des autres par une virgule :

- `a4paper` indique que la page physique fait  $21 \times 29,7$  ; il existe également tout un tas d’autres format possible (`a0paper`, ..., `a5paper`, `b0paper`, ..., `b5paper`, `letterpaper`, `legalpaper` et `executivepaper`) ;
- `twoside` indique que le document sera composé en recto-verso alors que `oneside` indique que le document sera composé en recto simple ;
- `nohead` supprime l’espace réservé à l’en-tête ; `nofoot` supprime l’espace réservé au pied de page et `noheadfoot` supprime ces deux espaces ;
- `top=val`, `bottom=val`, `right=val` et `left=val` fixent l’importance des marges (respectivement supérieure, inférieure, droite et gauche) ;
- `marginparwidth=val` indique la largeur des notes marginales.

Il est possible d’en faire encore plus : ces options sont les plus importantes. À titre d’exemple, voici l’appel à cette macro qui a été utilisé pour produire ce manuel :

```
\geometry{a4paper,twoside,left=1.5cm,right=1.5cm,marginparwidth=1.2cm,%
marginparsep=3mm,top=2cm,bottom=2cm}
```

#### 9.1.2 Extension pour les polices de caractères

**times et compagnie**



Si vous en avez assez des sempiternelles fontes Computer Modern, il est possible de composer des documents avec d'autres familles de fontes. Le problème est que dans le monde T<sub>E</sub>X on n'aime pas l'à peu près donc les fontes disponibles sont d'excellentes qualités et par conséquent soit rares, soit chères. Oubliez les fontes True Type qui ne correspondent de toute façon pas aux critères de qualité requis bien qu'il soit possible de transformer ce genre de fontes sous une forme utilisable par T<sub>E</sub>X.

### **METAFONT et les fontes PostScript**

En résumé et en très gros, deux grands types de fontes sont utilisées dans le monde T<sub>E</sub>X : les fontes, telles que les Computer Modern, fabriquées à partir du programme METAFONT et les fontes PostScript. Ces fontes sont radicalement différentes. Les fontes produites à partir de METAFONT sont des dessins bitmaps (chaque caractère est décrit sous la forme d'une grille de points) alors que les fontes PostScript sont des fontes vectorielles. La plupart du temps, les fontes METAFONT appartiennent au domaine public (elles sont donc accessibles gratuitement, librement distribuables et/ou modifiables) alors que la plupart des fontes PostScript sont soumises à un copyright classique et sont payantes.

### **Les fontes PostScript**

Les fontes PostScript offrent un vaste choix de possibilités car elles sont produites par des structures commerciales dont c'est, au moins en partie, le rôle et qui disposent d'un personnel hautement qualifié alors que les fontes METAFONT sont fabriquées par des passionnés de façon autonome sur leur temps libre. Or, produire une fonte de qualité requiert une quantité de travail énorme (j'ai essayé) et beaucoup de connaissances ce qui explique la relative rareté des fontes METAFONT, du moins celles de qualité.

### **La fonte Times**

La gestion des fontes est une tâche assez obscure et nous n'allons certainement pas entrer dans les détails de ce qui se passe en coulisse. Des extensions permettent un minimum de travail. Par exemple, pour composer l'ensemble du document en fonte Times, il suffit de spécifier :

```
\usepackage{times}
```

au niveau du préambule et c'est tout.

### **Autres fontes**

Les distributions proposent une vingtaine de package de ce type mais je conseille l'essai des extensions suivantes qui sont les plus complètes :

- bookman ;
- newcent ;
- palatino ;
- times.

### **Les fontes Dingbat**

Le package pifont est un peu spécial car la fonte appelée renferme des symboles et parce que, contrairement aux autres extensions, il définit quelques macros permettant quelques présentations particulières. Le tableau 9.1 indique tous les symboles existant dans la fonte Dingbat accessibles grâce à la syntaxe :

```
\ding{numéro}
```

### Les fontes de symboles

De même, on a accès facilement à une fonte dite " de symboles " avec la syntaxe :

`{\Pifont{psy} texte }`

ou

`\Pisymbol{psy}{numéro}`

Le tableau 9.2 indique l'ensemble de ces symboles

40 ✈	41 ✉	32 ☞	33 ✂	34 ✂	35 ✂	36 ✂	37 ☞	38 ☞	39 ☞
50 ✂	51 ✓	52 ✓	53 ✕	54 ✕	55 ✕	56 ✕	57 ✕	58 ✕	59 ✕
60 ✕	61 †	62 †	63 †	64 ✕	65 ✕	66 ✕	67 ✕	68 ✕	69 ✕
70 ✕	71 ✕	72 ★	73 ☆	74 ☆	75 ☆	76 ☆	77 ☆	78 ☆	79 ☆
80 ☆	81 ✕	82 ✕	83 ✕	84 ✕	85 ✕	86 ✕	87 ✕	88 ✕	89 ✕
90 ✕	91 ✕	92 ✕	93 ✕	94 ✕	95 ✕	96 ✕	97 ✕	98 ✕	99 ✕
100 ✕	101 ✕	102 ✕	103 ✕	104 ✕	105 ✕	106 ✕	107 ✕	108 ●	109 ○
110 ■	111 □	112 □	113 □	114 □	115 ▲	116 ▼	117 ◆	118 ◆	119 ●
120	121	122	123 ‘	124 ’	125 “				
	161 ♪	162 ♪	163 ♪	164 ♥	165 ♡	166 ♡	167 ♡	168 ♣	169 ♦
170 ♥	171 ♠	172 ①	173 ②	174 ③	175 ④	176 ⑤	177 ⑥	178 ⑦	179 ⑧
180 ⑨	181 ⑩	182 ①	183 ②	184 ③	185 ④	186 ⑤	187 ⑥	188 ⑦	189 ⑧
190 ⑨	191 ⑩	192 ①	193 ②	194 ③	195 ④	196 ⑤	197 ⑥	198 ⑦	199 ⑧
200 ⑨	201 ⑩	202 ①	203 ②	204 ③	205 ④	206 ⑤	207 ⑥	208 ⑦	209 ⑧
210 ⑨	211 ⑩	212 →	213 →	214 ↔	215 ↕	216 ↘	217 →	218 ↗	219 →
220 →	221 →	222 →	223 →	224 →	225 →	226 →	227 →	228 →	229 →
230 →	231 →	232 →	233 →	234 →	235 →	236 →	237 →	238 →	239 →
	241 →	242 →	243 →	244 →	245 →	246 →	247 →	248 →	249 →
250 →	251 →	252 →	253 →	254 →					

TABLE 9.1 – Caractères de la fonte ZapfDingbat

#### 9.1.2.1 Talon Réponse

On peut également construire des listes et des suites de symboles occupant une partie d'une ligne :

- \* petite liste charmante ;
- \* peut-être un peu niaise.

COUPON À RENVOYER

☞ ☞ ☞ ☞ ☞ ☞ ☞ ☞ ☞ ☞ ☞

*La section du Canon (κατατομη κανόνος)*

```

1 \parskip=4pt plus2pt minus1pt \
   parindent=8mm
2 \begin{dinglist}{93}
3   \item petite liste charmante ;
4   \item peut-être un peu niaise.
5 \end{dinglist}
6
7 {\centering COUPON \‘A RENVOYER\par}
8 \dingline{34}
9
10 \emph{La section du Canon}
11 ({\Pifont{psy}katatomh kan\’onoV})
    
```

40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	;
60	<	61	=	62	>	63	?	64	≅	65	A	66	B	67	X	68	Δ	69	E
70	Φ	71	Γ	72	H	73	I	74	∅	75	K	76	Λ	77	M	78	N	79	O
80	Π	81	Θ	82	P	83	Σ	84	T	85	Υ	86	ζ	87	Ω	88	Ξ	89	Ψ
90	Z	91	[	92	∴	93	]	94	⊥	95	-	96		97	α	98	β	99	χ
100	δ	101	ε	102	φ	103	γ	104	η	105	ι	106	φ	107	κ	108	λ	109	μ
110	ν	111	ο	112	π	113	θ	114	ρ	115	σ	116	τ	117	υ	118	ϖ	119	ω
120	ξ	121	ψ	122	ζ	123	{	124		125	}								
		161	Υ	162	'	163	≤	164	/	165	∞	166	f	167	♣	168	♦	169	♥
170	♠	171	↔	172	←	173	↑	174	→	175	↓	176	°	177	±	178	"	179	≥
180	×	181	∞	182	∂	183	•	184	÷	185	≠	186	≡	187	≈	188	...	189	
190	—	191	↵	192	⌘	193	⌚	194	⌘	195	∅	196	⊗	197	⊕	198	∅	199	∩
200	∪	201	⊃	202	⊇	203	⊄	204	⊂	205	⊆	206	∈	207	∉	208	∠	209	∇
210	®	211	©	212	™	213	∏	214	√	215	·	216	¬	217	∧	218	∨	219	↔
220	⇐	221	↑	222	⇒	223	↓	224	◇	225	⟨	226	®	227	©	228	™	229	Σ
230	∫	231		232	∫	233	∫	234		235	∫	236	∫	237	{	238	∫	239	
		241	⟩	242	∫	243	∫	244		245	J	246	∫	247		248	J	249	∫
250		251	∫	252	∫	253	∫	254	J										

TABLE 9.2 – Caractères de la fonte Symbol

## 9.2 Extensions spécialisées

### La magie $\LaTeX$ commence

Les extensions présentées ici sont souvent plus délicates à utiliser mais permettent de réaliser de petits miracles : génération automatisée de graphiques, solutions d'exercices venant se placer automatiquement au bon endroit soit en dessous de la question, soit sur une feuille part, génération de QCM, notes lacunaires ...

### 9.2.1 gnuplottex

#### Présentation de gnuplottex

L'extension gnuplottex permet de faire appel à gnuplot, un logiciel standard de graphisme scientifique, pour tracer un graphique.

#### Syntaxe de gnuplottex

Bien évidemment, il faudra charger ce package avec la commande :

```
\usepackage{gnuplottex}
```

au niveau du préambule. Ce package fournit l'environnement `\begin{gnuplot} ... \end{gnuplot}` qui accepte des commandes gnuplot. Cette démarche est très gourmande en mémoire et il est préférable d'utiliser une syntaxe plus subtile que nous verrons quand nous en serons à la partie du cours consacrée au graphisme.

$\LaTeX$  doit être capable de traiter une commande externe.

### 9.2.2 answers

#### Présentation de answers

L'extension answers permet de gérer des corrections d'exercices ou de contrôles.

### 9.2.3 pst-pdf

#### Présentation de pst-pdf

L'extension pst-pdf permet d'inclure des graphiques pstricks dans un document traité par pdfLatex.

D'abord lancer latex : `latex document.tex dvips -o document-pics.ps document.dvi ps2pdf document-pics.ps bibtex document.aux pdflatex document.tex`

### 9.2.4 showexpl

#### Présentation de showexpl

L'extension showexpl permet de visualiser le code  $\LaTeX$  juxtaposé avec son résultat. C'est l'extension utilisée dans ce cours.

### **9.2.5 wallpaper**

#### **Présentation de wallpaper**

L'extension wallpaper permet d'avoir un " papier peint ". C'est à dire un motif se répétant comme image de fond.

### **9.2.6 hyperref**

#### **Présentation de hyperref**

L'extension hyperref permet d'avoir une table des matières " cliquable ".

### **9.2.7 ifthen**

#### **Présentation de ifthen**

L'extension ifthen permet de traiter des conditions.

Par exemple, si c'est la version du professeur j'imprime un mot. Si c'est la version de l'élève, j'imprime des petits points pour un texte lacunaire.



# Chapitre 10

## Mathématiques avancées

### Sommaire

---

<b>10.1 Mathématiques complexes</b> . . . . .	<b>96</b>
10.1.1 Fontes mathématiques . . . . .	96
10.1.2 Grands opérateurs . . . . .	97
10.1.3 Délimiteurs . . . . .	99
10.1.4 Espacements . . . . .	99
10.1.5 Empiler verticalement . . . . .	100
10.1.6 Matrices et environnement array . . . . .	102
<b>10.2 amsmath</b> . . . . .	<b>104</b>
10.2.1 Tableaux évolués . . . . .	105
10.2.2 multicol . . . . .	109

---

## 10.1 Mathématiques complexes

### Mathématiques complexes

Voici le deuxième chapitre consacré aux mathématiques. Nous verrons encore quelques points supplémentaires lors de la présentation de l'extension *amsmath* à la section 10.2. Pour l'instant, nous en resterons à ce que  $\text{\TeX}$  offre de manière standard.

#### 10.1.1 Fontes mathématiques

##### Fontes mathématiques

Les textes mathématiques ne sont pas composés de la même façon que les textes courant, les macros gérant ces deux mondes sont différentes et même la façon d'agencer les lettres les unes à côté des autres n'est pas la même. Voici le mot difficile écrit en mode mathématique et en italique, la différence saute aux yeux :

mode mathématique : `\(difficile\)`    *difficile*  
 mode texte : `\textit{difficile}`    *difficile*

Les caractères sont les mêmes mais les espacements ne sont pas du tout les mêmes et les ligatures ne se font plus dans les modes mathématiques puisque pour  $\text{\TeX}$ , la formule `\(difficile\)` est comprise comme étant en fait `\(di^3f^2cle\)` ( $di^3f^2cle$ ), c'est-à-dire en tant que produit de plusieurs variables !

Les fontes mathématiques présentent les mêmes possibilités de modification que celles du texte courant mais en modifiant le nom des macros. Ainsi, certaines macros `\text...` vues à la section 7.1.1.2 ont un équivalent dans les modes mathématiques en changeant le "text" en "math".

Le tableau 10.1 montre l'effet de toutes ces macros existant en mode mathématique : on voit

Source	Résultat
<code>\(f\in ab\log[\alpha](\Phi_1\cup F)\)</code>	$f \in ab\log[\alpha](\Phi_1 \cup F)$
<code>\(f\in \mathbf{ab}\log[\alpha](\Phi_1\cup F)\)</code>	$f \in \mathbf{ab}\log[\alpha](\Phi_1 \cup F)$
<code>\(f\in \mathit{ab}\log[\alpha](\Phi_1\cup F)\)</code>	$f \in ab\log[\alpha](\Phi_1 \cup F)$
<code>\(f\in \mathrm{ab}\log[\alpha](\Phi_1\cup F)\)</code>	$f \in ab\log[\alpha](\Phi_1 \cup F)$
<code>\(f\in \mathsf{ab}\log[\alpha](\Phi_1\cup F)\)</code>	$f \in ab\log[\alpha](\Phi_1 \cup F)$
<code>\(f\in \mathtt{ab}\log[\alpha](\Phi_1\cup F)\)</code>	$f \in ab\log[\alpha](\Phi_1 \cup F)$

TABLE 10.1 – Fontes mathématiques

que seules les lettres romaines, les chiffres et les majuscules grecques sont affectés par ces macros. Les exemples montrent également la différence entre rien du tout et `\mathit{}` car, par défaut, les chiffres et les majuscules grecques ne sont pas en italique.

##### Problème d'exposants

$\text{\TeX}$  gère automatiquement la taille des caractères dans les formules en fonction de leurs emplacements (exposant, exposant d'exposant, composantes de fraction, etc.) et du mode hors texte ou en texte. En tout, il y a quatre tailles prédéfinies : la taille hors texte (`display`), texte (`text`), scripte (`script`) et sous-scripte (`scriptscript`). La taille peut alors être forcée grâce aux macros `\displaystyle`, `\textstyle`, `\scriptstyle` et `\scriptscriptstyle` (la macro `\displaystyle` avait été présentée



brèvement à la section 5.1.3). Tout ce qui se trouve après ces macros va prendre la taille spécifiée, à moins que des automatismes de  $\text{\LaTeX}$  entrent en jeu pour réduire cette taille (fraction, exposant, ...).

Voici un exemple de leur utilisation :

<p>Un nombre énorme : <math>2^{2^{2^{2^{2^{2^2}}}}}} \approx 3,4 \times 10^{38}</math></p> <p>Un nombre énorme : <math>2^{2^{2^{2^{2^{2^2}}}}} \approx 3,4 \times 10^{38}</math></p>
--

```

1 Un nombre \'enorme :
2 \((2^{2^{2^{2^{2^{2^2}}}}})\approx
3 3{,}4\times10^{38}\)
4
5 Un nombre \'enorme :
6 \((2^{2^
7   {\scriptstyle2^
8     {\scriptstyle2^
9       {\scriptstyle2^
10        {\scriptstyle2^
11         {\scriptstyle2^
12          {\scriptstyle2}}}}}})\
13   approx
14   3{,}4\times10^{38}\)

```

ou encore, plus parlant :

<p>Le début du développement du nombre d'or en fraction continue donne :</p> $1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}}} = 1,6$ <p>Le début du développement du nombre d'or en fraction continue donne :</p> $1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}}}} = 1,6$
---

```

1 Le d\'ebut du d\'eveloppement du
2   nombre d'or en
3   fraction continue donne :
4   \[1+\frac{1}{1+
5     \frac{1}{1+
6       \frac{1}{1}}}}=1{,}6\]
7
8 Le d\'ebut du d\'eveloppement du
9   nombre d'or en
10  fraction continue donne :
11  \[1+\displaystyle\frac{1}{1+
12    \displaystyle\frac{1}{1+
13      \displaystyle\frac
14        {1}{1}}}}=1{,}6\]

```

## 10.1.2 Grands opérateurs

### Grands opérateurs

Les grands opérateurs sont des opérateurs pouvant changer de taille en fonction du mode mathématique et s'écrivant souvent avec des indices et/ou des exposants. Le tableau 5.4 page 58 en dresse la liste complète ; nous allons maintenant voir comment les utiliser.

Regardons tout ceci avec l'opérateur  $\sum$ , les autres opérateurs se comportent exactement de la même façon mis à part  $\int$  et  $\oint$  sur lesquels il faudra un petit complément.

### Indice et exposant d'opérateurs

Ces opérateurs peuvent avoir un indice et/ou un exposant qui seront placés de façon traditionnelle dans une formule en texte et au-dessus et au-dessous du symbole dans une formule hors texte :

On sait que  $\sum_{n=0}^{\infty} \frac{1}{n!}$  peut être la définition de  $e$ .  
On sait que

$$\sum_{n=0}^{\infty} \frac{1}{n!}$$

peut être la définition de  $e$ .

```
1 On sait que
2 \(\sum_{n=0}^{\infty} \frac{1}{n!}\)
3 peut \^etre la d\efinition de \e\).
4
5 On sait que
6 \[\sum_{n=0}^{\infty} \frac{1}{n!}\]
7 peut \^etre la d\efinition de \e\).
```

Comme il a été précisé à la section 5.1.1, on peut être tenté d'écrire quelque chose du type : " On sait que  $\sum_{n=0}^{\infty} \frac{1}{n!}$  peut être la définition de  $e$ ". Toujours pour ne pas massacrer l'interligne, cette façon de faire est fortement déconseillée ! Incidemment, le code pour la formule qui a permis cette horreur a été :

```
\(\displaystyle\sum_{n=0}^{\infty} \frac{1}{n!}\)
```

### Indice et exposant d'intégrales

Les deux opérateurs `\int` et `\oint` ne se comportent pas comme les autres au niveau du placement des exposants et indices. En effet, il est assez peu courant de voir quelque chose du style :

$$\int_0^2 t^2 dt$$

et même en mode hors texte, l'exposant et l'indice occupent leur position habituelle, contrairement aux autres grands opérateurs.

Par exemple :

Comme  $\int_0^2 t dt = 2$ , un triangle ...

Comme

$$\int_0^2 t dt = 2$$

un triangle ...

```
1 Comme \(\int_0^2 t,dt=2\),
2 un triangle \ldots
3
4 Comme
5 \[\int_0^2 t,dt=2\]
6 un triangle \ldots
```

### limits ...

Mais alors, comment  $\int_0^2 t^2 dt$  a-t-il été tapé ? En fait, on peut forcer la position des exposants et indices au-dessus et au-dessous d'un opérateur quelconque quelle que soit sa taille mathématique en le faisant suivre de la macro `\limits`. L'exemple donné ci-dessus a été composé de cette façon :

```
\(\int\limits_0^2 t^2,dt\)
```

### ... et nolimits

De même, on peut forcer la position des exposants et indices à leurs positions habituelles en faisant suivre l'opérateur de la macro `\nolimits`. Par exemple :

$$\sum_{i=1}^n a_i = \frac{\sum_{i=1}^n i}{\sum_{i=1}^n i^2}$$

```
1 \[\sum\nolimits_{i=1}^n a_i
2 =\frac{\sum_{i=1}^n i}{\sum_{i=1}^n
i^2}\]
```

### 10.1.3 Délimiteurs

#### Délimiteurs

Les délimiteurs sont des symboles destinés à encadrer des sous-formules. La liste complète des délimiteurs a été présentée au tableau 5.5 page 59. Si ces symboles n'étaient que cela, ils ne seraient guère utiles et n'auraient de sens que pour l'humain. En réalité, les délimiteurs peuvent voir leur taille modifiée de façon fine, soit manuellement, soit automatiquement. Nous ne verrons ici que la façon automatique (de toute façon plus utile) et nous renverrons le lecteur avide au stage de perfectionnement ou à des ouvrages spécialisés !

Dans la formule  $f(x)$ , les deux parenthèses sont des délimiteurs. Soyons logique : la parenthèse gauche est appelée " délimiteur gauche " et je vous laisse deviner comment on appelle la parenthèse droite. La sous-formule, dans le cas présent est composée de la seule lettre  $x$ . Construisons maintenant une formule avec des délimiteurs encadrant une sous-formule un peu plus complexe, par exemple :

$f\left(\frac{1+x}{x}\right) = x^2 + \left(\frac{1}{x}\right)^2$	<code>1 \[f(\frac{1+x}{x})=x^2+(\frac{1}{x})^2\]</code>
--	---

#### left ... right

Le résultat est tout à fait ridicule. Les deux macros qui permet de régler automatiquement la taille des délimiteurs à la hauteur de la sous-formule sont `\left` à mettre immédiatement avant le délimiteur gauche et `\right` à mettre immédiatement avant le délimiteur droit. Le même exemple donnera alors :

$f\left(\frac{1+x}{x}\right) = x^2 + \left(\frac{1}{x}\right)^2$	<code>1 \[f\left( \frac{1+x}{x} \right)</code> <code>2 =x^2+\left( \frac{1}{x} \right)^2\]</code>
--	--

On peut imbriquer des couples `\left... \right` mais il faut toujours qu'un `\left` s'équilibre avec un `\right` sous peine de voir  $\text{\LaTeX}$  protester avec véhémence.

### 10.1.4 Espacements

#### Espaces

Comme nous l'avons déjà dit, un espace tapé au niveau d'une formule ne sert à rien sauf à stopper un nom de macro ou bien à présenter les choses de façon lisible pour un humain. Or, il peut arriver que l'on veuille insérer des espaces au niveau d'une formule. Les macros d'espacement horizontal vues pour le texte — `\_` (macro espace), `\hspace` — ainsi que l'espace insécable `\sim` fonctionnent également en mode mathématique.

#### Sept types d'espaces

Le tableau 10.2 montre sept types d'espaces que  $\text{\LaTeX}$  permet en mode mathématique (dont quatre disponibles en mode texte) :

Macro	Signification	Exemple	Mode texte
<code>\!</code>	espace négative		non
<code>\,</code>	espace fine		oui
<code>\:</code>	espace moyenne		non
<code>\;</code>	grande espace		non
<code>\_</code>	espace intermot		oui
<code>\quad</code>	espace cadratin		oui
<code>\qquad</code>	double espace cadratin		oui

TABLE 10.2 – Macros d’espacements

### Méfiez-vous de la racine carrée!

Le tableau 10.3 montre quelques utilisations classiques de ces différentes espaces qu’il est bon de connaître pour obtenir une lecture facile d’un texte. On arrive vraiment à de la typographie très fine et il n’est pas obligatoire de se souvenir de toutes ces présentations. D’autre part, certaines constructions sont sujettes à discussions (parfois enflammées) donc vous prenez et vous en faites ce que vous voulez ! Tous ces exemples ont été repris tels quels dans le  $\TeX$ book. Pensez-y surtout pour les intégrales multiples et méfiez-vous de la racine carrée : avec ces deux conseils, vous pourrez éviter les présentations malheureuses.

Formule	Résultat	sans espace
<code>\((2n)!/(n!, (n+1)!)\)</code>	$(2n)!/(n!(n+1)!)$	$(2n)!/(n!(n+1)!)$
<code>\[\frac{52!}{13!13!26!}\]</code>	$\frac{52!}{13!13!26!}$	$\frac{52!}{13!13!26!}$
<code>\(\sqrt{2}\,x\)</code>	$\sqrt{2}x$	$\sqrt{2}x$
<code>\(\sqrt{\log x}\)</code>	$\sqrt{\log x}$	$\sqrt{\log x}$
<code>\(O(1/\sqrt{n})\)</code>	$O(1/\sqrt{n})$	$O(1/\sqrt{n})$
<code>\([0,1)\)</code>	$[0,1)$	$[0,1)$
<code>\(\log n, (\log \log n)^2\)</code>	$\log n (\log \log n)^2$	$\log n (\log \log n)^2$
<code>\(x^2/2\)</code>	$x^2/2$	$x^2/2$
<code>\(n/\log n\)</code>	$n/\log n$	$n/\log n$
<code>\(\Gamma_2 + \Delta^2\)</code>	$\Gamma_2 + \Delta^2$	$\Gamma_2 + \Delta^2$
<code>\(R_i^j{}_{kl}\)</code>	$R_i^j{}_{kl}$	$R_i^j{}_{kl}$
<code>\(\int_0^x \int_0^y dF(u,v)\)</code>	$\int_0^x \int_0^y dF(u,v)$	$\int_0^x \int_0^y dF(u,v)$
<code>\(\int \int_D dx, dy\)</code>	$\int \int_D dx dy$	$\int \int_D dx dy$

TABLE 10.3 – Exemples classiques de correction d’espacement

## 10.1.5 Empiler verticalement

### Deux dimensions

Jusqu'ici, mises à part les fractions, les éléments d'une formule mathématique étaient composés de gauche à droite mais les mathématiciens aiment beaucoup travailler en deux dimensions et la fin de ce chapitre montre comment procéder verticalement dans une formule de math.

### Barre au-dessus d'une formule

Nous avons déjà vu les accents mathématique et le tableau 5.9 page 60 en dressait la liste complète. Il existait un accent permettant de surligner une lettre (`\bar`) mais son emploi pour placer une barre au-dessus d'une formule plus longue n'est pas correct :

$\bar{x} + \bar{y}$	1 <code>\(\bar{x} \bar{y} + \bar{x} \bar{y}\)</code>
---------------------	--

Pour réaliser cela, il faut employer la macro `\overline` suivie du groupe sur lequel on veut placer la barre :

$\overline{x + y}$	1 <code>\(\overline{x+y}\)</code>
--------------------	-----------------------------------

### Barre sous la formule

De même, il existe la macro `\underline` qui permet de placer une barre sous la formule :

$\underline{x + y}$	1 <code>\(\underline{x+y}\)</code>
---------------------	------------------------------------

Cette dernière macro est également disponible en mode texte mais le soulignement d'un texte est généralement une mauvaise idée : en fait une mauvaise habitude qui provient du temps des machines à écrire. Mettre un texte en évidence se fait **toujours** en le composant dans une autre fonte que le texte alentour (soit en gras, soit en italique par exemple).

### Accolade horizontale

Les deux macros `\overbrace` et `\underbrace` permettent respectivement de surligner ou de souligner avec une accolade horizontale à la place de la barre. Le groupe qui suit la macro indique ce qui doit être inclus au niveau de l'accolade. On peut écrire quelque chose au-dessus de l'accolade supérieure ou au-dessous de l'accolade inférieure en utilisant respectivement l'exposant ou l'indice. L'exemple suivant montre tout cela :

Pour $n$ pair, $2^n = \overbrace{2 \times \cdots \times 2}^{\frac{n}{2}} \times \underbrace{2 \times \cdots \times 2}_{\frac{n}{2}}$	1 Pour $(n)$ pair, $(2^n =$ 2 $\overbrace{2 \times \cdots \times 2}$ 3 $\textstyle\frac{n}{2}$ 4 $\times$ 5 $\underbrace{2 \times \cdots \times 2}$ 6 $\textstyle\frac{n}{2})$
--	---

### Mélanger accolades supérieures et inférieures

Encore plus fort, on peut mélanger des accolades supérieures et inférieures. Il n'y a que la sagesse de l'auteur qui limite les possibilités car la lisibilité devient franchement mauvaise et la frappe un peu délicate à mener sans se tromper ! Un premier exemple sage :

$2^n = \overbrace{2 \times 2 \times \cdots \times 2}^n \times \underbrace{2 \times \cdots \times 2}_{n-2}$	1 $(2^n = \overbrace{2 \times \underbrace{2 \times \cdots \times 2}_{n-2}}$ 2 $\times 2 \times \cdots \times 2)$ 3 $\times 2^n$
--	---

**Tout est permis!**

Et un exemple délirant pour voir que tout est permis !

$$\overbrace{\overbrace{\overbrace{2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2}^2}^4}^8$$

```

1  \(\overbrace{\overbrace{\overbrace
2  {2\times2}^2
3  \times\overbrace
4  {2\times2}^2
5  }^{\scriptstyle 4}
6  \times\overbrace{\overbrace
7  {2\times2}^2
8  \times\overbrace
9  {2\times2}^2
10 }^{\scriptstyle 4}
11 }^{\scriptstyle 8}\)

```

**stackrel**

On peut placer n'importe quoi au-dessus de n'importe quoi pour produire une relation (donc avec des espacements autour comme pour " = "). Pour cela,  $\text{\LaTeX}$  propose la macro `\stackrel` (stack signifie pile et rel et là pour rappeler qu'on construit une relation). Cette macro place l'élément qui la suit immédiatement au-dessus de l'élément qui est spécifié ensuite. Il est plus sage de ne pas trop réfléchir et de placer deux groupes à la suite de cette macro.

Voici deux exemples utiles :

$$\vec{x} \stackrel{\text{d\'ef}}{=} (x_1, x_2, \dots, x_n)$$

$$x \stackrel{f}{\mapsto} f(x)$$

```

1  \(\vec{x}\stackrel{\mathrm{d\'ef}}{=}
2  (x_1,x_2,\ldots,x_n)\)
3
4  \(\x\stackrel{f}{\longmapsto} f(x)\)

```

Le premier exemple montre qu'on peut être conduit assez souvent à mettre du texte véritable à l'intérieur d'une formule de math. Pour cela, il existe la macro `\mbox` qui construit le contenu de ce qui suit comme s'il s'agissait de texte ordinaire sauf qu'aucune coupure ne pourra être effectuée à ce niveau.

Ainsi, le premier exemple aurait pu être tapé sous la forme :

```
\(\vec{x}\stackrel{\mbox{d\'ef}}{=} (x_1,x_2,\ldots,x_n)\)
```

**10.1.6 Matrices et environnement array****environnement array**

De façon générale, les matrices sont considérées comme des tableaux, la seule différence étant que les tableaux étaient construits en utilisant l'environnement `tabular` alors que les matrices seront construites avec l'environnement `array`. L'énorme avantage est d'être en mode mathématique.

**Matrices**

Un premier exemple pour voir que ce n'est vraiment pas différent :

$$\begin{array}{cccc} 1-\lambda & 2 & 3 & 4 \\ 2 & 3-\lambda & 4 & 1 \\ 3 & 4 & 1-\lambda & 2 \\ 4 & 1 & 2 & 3-\lambda \end{array}$$

```
1 \[ \begin{array}{*{4}{c}}
2 1-\lambda & 2 & 3 & 4 \\
3 2 & 3-\lambda & 4 & 1 \\
4 3 & 4 & 1-\lambda & 2 \\
5 4 & 1 & 2 & 3-\lambda \\
6 \end{array} \]
```

Pour les " vraies " matrices, il faudra ajouter des parenthèses sous la forme de délimiteurs extensibles (donc avec les préfixes `\left` et `\right`).

### Déterminants

La forme pour les déterminants suivra le même principe en prenant des barres verticales comme délimiteurs. L'exemple précédent devient alors :

$$\left| \begin{array}{cccc} 1-\lambda & 2 & 3 & 4 \\ 2 & 3-\lambda & 4 & 1 \\ 3 & 4 & 1-\lambda & 2 \\ 4 & 1 & 2 & 3-\lambda \end{array} \right|$$

```
1 \[
2 % \varphi(\lambda) =
3 \left|
4 \begin{array}{*{4}{c}}
5 1-\lambda & 2 & 3 & 4 \\
6 2 & 3-\lambda & 4 & 1 \\
7 3 & 4 & 1-\lambda & 2 \\
8 4 & 1 & 2 & 3-\lambda \\
9 \end{array} \right| \]
```

### délimiteur vide

$\TeX$  définit le délimiteur " vide " qui ne produit aucun symbole. Il est souvent utile dans certains cas faisant appel aux tableaux. Par exemple :

$$\left. \begin{array}{l} x \in E \\ \text{ou} \\ x \in F \end{array} \right\} \iff x + x^2 = 0$$

```
1 \[ \left. \begin{array}{l}
2 x \in E \\
3 \quad \quad \quad \\
4 x \in F \\
5 \end{array} \right\} \iff x+x^2=0
6 \]
```

Un autre exemple fréquent est donné par la construction :

$$P_{i,j} = \begin{cases} 0 & \text{si } i-j \text{ est impair,} \\ i!(-1)^{(i-j)/2} & \text{si } i-j \text{ est pair.} \end{cases}$$

```
1 \[ P_{i,j} = \left\{ \begin{array}{ll}
2 \quad 0 & \text{\mbox{si } \((i-j\) est impair} \\
3 \quad , \\
4 \quad i!(-1)^{\{(i-j)/2\}} & \text{\mbox{si } \((i-j\) est pair}.} \\
5 \quad \end{array} \right. \\
6 \]
```

Cet exemple sera repris lors de l'étude de l'extension *amsmath* à la section 10.2 page 104. Notons ici l'utilisation d'un mode mathématique en texte (le `\(i-j\)`) à l'intérieur d'un mode texte (les boîtes `\mbox`) à l'intérieur d'un mode mathématique hors texte (la formule complète) !

## 10.2 amsmath

### L'American Mathematical Society

L'American Mathematical Society est la dépositaire du logiciel T<sub>E</sub>X. Il s'agit d'une importante association américaine de mathématiciens. Lorsque ceux-ci ont eu le programme de Knuth en leur possession, ils se sont empressé de construire toute une série de macros permettant des constructions mathématiques complexes dont ils avaient besoin. Toutes ces macros ont été réunies dans plusieurs extensions dont la plus importante est *amsmath*. Il est hors de question de présenter toutes les possibilités de ce package en raison de la profusion de possibilités mais nous verrons les possibilités offertes les plus utiles.

### Listes des extensions AMS

Les extensions disponibles sont *amsmath*, *amstext*, *amsfonts*, *amssymb* et *amscd*. L'extension *amscd* permet la construction de diagrammes commutatifs, *amssymb* et *amsfonts* permettent d'avoir un accès à quelques lettres hébraïques, à quelques symboles très spécialisés, à une fonte gothique, une fonte pour l'écriture des ensembles " comme à la main ", des fontes grasses, *amstext* permet de placer du texte normal dans des formules mathématiques de façon intelligente. Tous les exemples qui suivent n'ont eu besoin que des packages *amsmath* et *amsfonts*.

### Exemples

Voici d'abord quelques exemples d'utilisation de macros particulières :

$$\int_C \vec{V} \cdot d\vec{M} = \iint_D \left( \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy$$

$$\overrightarrow{A(x)B(x)} = \vec{g}_x$$

$$\varliminf_{n \rightarrow \infty} u_n = \overline{\lim}_{n \rightarrow \infty} u_n = \mathbf{a}^\lambda \iff \lim_{n \rightarrow \infty} u_n = \mathbf{a}^\lambda$$

```

1 \[\int_C \vec V \cdot d\vec M
2   = \iint_D \left( \frac{\partial Q}{\partial x}
3     - \frac{\partial P}{\partial y} \right) dx
4     \, dy
5 \] % int\ 'egrales multiples
6     automatiques
7
8 \[\overrightarrow{A(x)B(x)}
9   = \overrightarrow{g_x} \] % grandes
10    fl\ 'eches
11
12 \[\varliminf_{n\to\infty} u_n
13   = \varlimsup_{n\to\infty} u_n
14   = \boldsymbol{a^\lambda}
15   \iff \lim_{n\to\infty} u_n
16   = \boldsymbol{a^\lambda} \]
```

### array

Une série de macros permettent de construire des structures classiques faisant normalement appel à l'environnement `array`. Voici un exemple permettant les structures de choix :

$$|x| = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x \leq 0 \end{cases}$$

```

1 \[|x| =
2   \begin{cases}
3     x & \text{si } (x \geq 0) \\
4     -x & \text{si } (x \leq 0) \\
5   \end{cases} \]
```



## Matrices

Le package *amsmath* définit cinq environnements permettant de construire des matrices (*matrix* qui ne met aucun délimiteur, *pmatrix* qui entoure la matrice avec des parenthèses, *bmatrix* avec des crochets, *vmatrix* avec des traits verticaux et *Vmatrix* avec des double traits verticaux) :

$$\begin{pmatrix} 1 & \cos x \\ -\cos x & 1 \end{pmatrix}$$

$$\begin{vmatrix} \cos(x) - X & -\sin x \\ \sin x & \cos(x) - X \end{vmatrix}$$

```

1  \[\begin{pmatrix}
2    1 & \cos x \\
3    -\cos x & 1 \\
4  \end{pmatrix}\]
5
6  \[\begin{vmatrix}
7    \cos(x)-X & -\sin x \\
8    \sin x & \cos(x)-X \\
9  \end{vmatrix}\]
```

### 10.2.1 Tableaux évolués

#### Tableaux évolués

Vous vous êtes peut-être aperçu que la construction de tableaux est quelquefois un peu pénible et des problèmes surgiront presque obligatoirement si vous vous mettez à composer des textes de façon soutenue avec  $\text{\LaTeX}$ . Ces problèmes ont été évoqués sur des forums de discussion et certaines personnes, très à l'aise dans la programmation de  $\text{\LaTeX}$  ont créé des packages résolvant automatiquement les problèmes rencontrés. Là aussi, il existe beaucoup d'extensions dont le but est de résoudre certains problèmes liés aux tableaux ; nous ne ferons pas vœux d'exhaustivité et nous ne présenterons que les plus classiques. Seul le package *array* sera présenté avec quelques détails, les autres packages ne seront qu'évoqués, éventuellement avec un exemple représentatif.

#### Commandes de motif

Le package *array* offre la possibilité de dire plus de choses au niveau du motif d'un tableau. Le tableau 10.4 donne le récapitulatif des commandes qu'on peut indiquer au niveau du motif d'un tableau.

Un petit exemple va permettre de voir ceci en pratique. Il est bien entendu hors de question de voir toutes les subtilités du package *array*. [COM] consacre une quarantaine de pages aux extensions dédiées aux tableaux (en fait, un chapitre entier) et la description du package *array* en prend une quinzaine.

Anciennes commandes	
l	Colonne alignée à gauche
c	Colonne centrée
r	Colonne alignée à droite
p{ <i>larg.</i> }	Paragraphe de largeur <i>larg.</i> dont le haut sera aligné avec le haut de la ligne du tableau.
{ <i>instr.</i> }	Remplace l'espace intercolonne par <i>instr.</i>
Ancienne commande modifiée	
	Insère une réglure verticale. Contrairement à ce qui se passait avec la commande $\text{\TeX}$ , la largeur de la colonne sera agrandie de la largeur de cette réglure.
Nouvelles commandes	
m{ <i>larg.</i> }	Paragraphe de largeur <i>larg.</i> centrée verticalement dans la ligne du tableau.
b{ <i>larg.</i> }	Paragraphe de largeur <i>larg.</i> dont le bas sera aligné avec le bas de la ligne du tableau.
>{ <i>instr.</i> }	Utilisée avant les commandes l, r, c, p, m ou b, elle insère <i>instr.</i> au début de la colonne.
<{ <i>instr.</i> }	Utilisée après les commandes l, r, c, p, m ou b, elle insère <i>instr.</i> à la fin de la colonne.
!{ <i>instr.</i> }	Peut être utilisée n'importe où comme la commande   sauf qu'au lieu d'obtenir une réglure verticale, c'est <i>instr.</i> qui sera inséré. En particulier, cette commande ne supprime pas l'espace intercolonne.

TABLE 10.4 – Commandes des motifs de tableaux

ABEL	Niels Henrick	Il montre l'impossibilité de résoudre les équations de degré 5 par radicaux.	1802 ap. J.-C. – 1829 ap. J.-C.
ALEMBERT	Jean LE ROND D'	Collaborateur à l'encyclopédie de Diderot. Il a effectué d'importants travaux sur les dérivées partielles, les nombres complexes et la mécanique	1717 ap. J.-C. – 1783 ap. J.-C.
APOLLONIUS	de Perge	Il a étudié les sections planes du cône (les coniques). En fait, on sait très peu de choses sur ce mathématicien.	262 av. J.-C. – ~ 180 av. J.-C.

```

1 \begin{tabular}{|>{\scshape}llm{4.25cm}%
2   r<{\scshape j-c.}@{ -- }l<{\scshape j-c.}|}
3 \hline
4 Abel & Niels Henrick &
5 Il montre l'impossibilit\`e de r\`esoudre les \`equations de degr\`e 5 par
6   radicaux. &
7 1802 ap. & 1829 ap. \\ \hline
8 Alembert & Jean \textsc{le Rond d'} &
9 Collaborateur \`a l'encyclo\`edie de Diderot. Il a effectu\`e d'importants
10   travaux sur les
11   d\`eriv\`ees partielles, les nombres complexes et la m\`ecanique &
12 1717 ap. & 1783 ap. \\ \hline
13 Apollonius & de Perge &
14 Il a \`etudi\`e les sections planes du c\`one (les coniques). En fait, on sait tr
15   \`es peu de
16   choses sur ce math\`ematicien. &
17 262 av. & \(\sim\) 180 av. \\ \hline
18 \end{tabular}

```

## tabularx

L'extension *tabularx* permet de composer des tableaux ayant une largeur totale précise. Pour cela, ce package fournit l'environnement `tabularx` qui fonctionne strictement de la même façon que le `tabular` classique de  $\text{\TeX}$  sauf que l'appel comporte un argument supplémentaire donnant la largeur totale du tableau et qu'il existe la commande `X` qu'on peut mettre dans le motif du tableau, cette commande indiquant la ou les colonnes qui seront extensibles afin d'amener le tableau à la taille voulue. Si plusieurs colonnes sont spécifiées " `X` ", elles se partageront la place restant et seront donc de même largeur. Il existe une astuce permettant de surmonter cette limitation mais sa présentation sortirait du cadre de ce stage.

Ainsi, le tableau 10.4 commençait de cette façon :

```
\noindent
\begin{tabularx}{\linewidth}{|c|X|}
\hline
\multicolumn{2}{|c|}{\textbf{Anciennes commandes}} \\\hline
\texttt{l} & Colonne align\`ee \`a gauche \\\
\texttt{c} & Colonne centr\`ee \\\
\texttt{r} & Colonne align\`ee \`a droite \\\
...
```

Ce qui a permis d'avoir un tableau ayant exactement la largeur du corps de la page (`\linewidth`).

## supertabular

Le package *supertabular* permet de réaliser des tableaux s'étendant sur plusieurs pages. En effet, un tableau classique ne peut absolument pas franchir la frontière d'une page ce qui peut provoquer soit un grand espace vide soit un débordement dans la marge lorsque des tableaux deviennent volumineux.

## dcolumn

Le package *dcolumn* permet des alignements sur le point (ou la virgule) décimal.

## hhline

Le package *hhline* propose un grand choix pour la construction de réglures doubles dans les tableaux. Le principe est de remplacer la macro classique `\hline` par la macro `\hhline` qui offre beaucoup plus de possibilité. Sa syntaxe générale est :

```
\hhline{motif}
```

On commencera avec un petit exemple de ses possibilités et le tableau 10.5 récapitulera les commandes de motif de cette macro.

a	b	A	+
1	2	$\alpha$	-
3	4	$\beta$	$\times$
5	6	$\Gamma$	$\div$

```
1 \setlength{\arrayrulewidth}{1pt}
2 \begin{tabular}{|cc|c|c|}
3   \hhline{|t :== :t :== :t|}
4   a & b & A & \(\+\) \\\
5   \hhline{| :== :|-|~|}
6   1 & 2 & \(\alpha\) & \(-\) \\\
7   \hhline{||--||~ :||}
8   3 & 4 & \(\beta\) & \(\times\) \\\
9   \hhline{||--||-|~|}
10  5 & 6 & \(\Gamma\) & \(\div\) \\\
11  \hhline{|b :==#== :b|}
12 \end{tabular}
```

Éléments faisant la largeur d'une cellule.	
=	Une double ligne horizontale.
-	Une ligne horizontale.
~	Un vide.
Éléments occupant un coin de cellule.	
	Une (double) ligne horizontale coupée par une ligne verticale.
:	Une double ligne horizontale non coupée par une ligne verticale.
#	Une double ligne horizontale coupée par une double ligne verticale.
t	La ligne supérieure d'une double ligne horizontale.
b	La ligne inférieure d'une double ligne horizontale.

TABLE 10.5 – Syntaxe de la macro `\hline`

L'exemple montre également la façon d'obtenir des réglures n'ayant pas l'épaisseur standard (0,4 point). La dimension `\arrayrulewidth` est déclarée par  $\LaTeX$ , il ne s'agit pas d'un ajout de ce package.

### **multirow**

Le dernier package que nous évoquerons ici est *multirow* qui permet de fusionner des cellules verticalement (la macro `\multicolumn` de  $\LaTeX$  permettait une fusion horizontale). Ce package fournit la macro `\multirow` (original n'est-ce pas ?) qui a la syntaxe de base suivante :

```
\multirow{nbligne}{largeur}{contenu}
```

où *nbligne* désigne le nombre de lignes qui devront être fusionnées, *largeur* la largeur de la grande cellule créée et *contenu* le contenu de la cellule qui sera considérée comme un paragraphe (donc pourra être composée sur plusieurs ligne de texte).

Voici un exemple de tableau composé grâce à cette macro :

Planète	Satellite(s)
Terre	Lune
Mars	Phobos Deimos
Jupiter	Io Europe Ganymède Callisto 8 petits

```

1 \newlength{\Lmax}
2 \settowidth{\Lmax}{Jupiter}
3 \begin{tabular}{|l|l|}
4 \hline
5 Plan\`ete & Satellite(s) \\ \hline
6 Terre & Lune \\ \hline
7 \multirow{2}{\Lmax}{Mars} & Phobos
8 \quad \quad \quad & Deimos \\ \hline
9 \multirow{5}{\Lmax}{Jupiter} & Io \\
10 \quad \quad \quad & Europe \\
11 \quad \quad \quad & Ganym\`ede \\
12 \quad \quad \quad & Callisto \\
13 \quad \quad \quad & 8 petits \\
14 \hline
15 \end{tabular}

```

## 10.2.2 multicol

### multicol

Le package *multicol* permet la composition de parties de document sur plusieurs colonnes. En fait,  $\LaTeX$  offre un embryon de possibilité mais les restrictions sont trop fortes (à mon goût du moins) puisque le changement de nombre de colonnes provoque un saut de page, que le nombre de colonnes est limité à 2 et que la fin d'un passage en deux colonnes n'est pas équilibrée (la colonne de droite n'est pas remplie comme la colonne de gauche). L'extension *multicol* surmonte ces trois limitations puisque le nombre de colonnes maximum autorisé est de 10 (je vois mal comment dépasser ce nombre sans obtenir un résultat illisible), qu'il est maintenant possible de mélanger des nombres de colonnes différents sur une même page et qu'on peut gérer les problèmes d'équilibre de colonnes (le défaut étant que les colonnes sont parfaitement équilibrées, y compris celles de la dernière page).

### syntaxe de multicol

Une fois cette extension chargée, on change le nombre de colonnes grâce à la syntaxe :

```
\begin{multicols}{n}  
  texte ...  
\end{multicols}
```

où  $n$  représente le nombre de colonnes voulues. Le seul autre point important concernant ce package est la possibilité de modifier la largeur du filet séparateur (une largeur nulle permettant une absence de filet). La dimension à modifier est `\columnseprule`. Pour mémoire, la largeur par défaut des réglures dans  $\LaTeX$  (par exemple ceux des tableaux) est de 0,4 pt.



## **Troisième partie**

**Troisième atelier : programmation, macros, compteurs, graphisme, programmes externes**





# Chapitre 11

## Graphisme

### Sommaire

---

<b>11.1 Petit aperçu des possibilités</b> . . . . .	<b>114</b>
11.1.1 Quels graphiques? . . . . .	114
<b>11.2 Deux choix possibles</b> . . . . .	<b>114</b>
11.2.1 Inclusion d'images externes . . . . .	114
11.2.2 L'environnement "figure" . . . . .	120
11.2.3 Syntaxe de "includegraphics" . . . . .	120
11.2.4 Des commandes dans le code source . . . . .	123
11.2.5 Graphismes avec appel à des programmes extérieurs . . . . .	127

---

## 11.1 Petit aperçu des possibilités

Utiliser  $\LaTeX$  sans étudier ses possibilités graphiques nous a semblé dangereusement limitatif. En effet, certains objets de  $\LaTeX$  sont des graphiques même si cela n’apparaît pas immédiatement.

Nous allons donner une vue générale des différentes possibilités ceci afin que les choix à faire en terme de graphisme soient plus clairs :

- Quand faut-il inclure des images et quels formats d’images choisir ?
- Quand et comment générer des images via des extensions spécifiques ?
- Comment lier les objets graphiques à la structure du document (numéros et tables des figures) et ajouter des légendes à ces objets.

Nous ferons alors une exploration rapide d’extensions permettant de générer des graphiques sous  $\LaTeX$ .

Plus tard, nous aborderons les bases de notre extension préférée : Tikz.

Nous étudierons certains outils externes qui peuvent être appelés par  $\LaTeX$  et discuterons de certains programmes qui se “marient” bien avec  $\LaTeX$ .

### 11.1.1 Quels graphiques ?

Initialement,  $\TeX$  avait été pensé pour produire des ouvrages mathématiques et informatiques. Par conséquent les possibilités graphiques natives sont extrêmement limitées. D’autre part, à l’heure actuelle, les graphiques évoluées font la part belle au format PostScript et, depuis moins longtemps au format PDF, formats qui n’existaient pas lorsque Knuth a programmé le logiciel  $\TeX$ . En fait, cela montre les extraordinaires possibilités de ce logiciel puisqu’il a été possible de construire des extensions permettant d’utiliser ces formats.

## 11.2 Deux choix possibles

### J’ai trouvé la voie !

Il y a deux voies possibles :

- construire une image grâce à un programme externe et l’inclure sous forme d’un fichier (PostScript, PDF, JPEG, PNG, TIFF, ...) ;
- utiliser des extensions permettant de mettre des commandes graphiques dans le source du document (Pstricks, Pgf/Tikz, Metapost, Asymptote, ...).

Ces deux méthodes présentent chacune des avantages et des inconvénients.

### 11.2.1 Inclusion d’images externes

#### 11.2.1.1 Nature de l’affichage

#### Qu’est ce qu’une image ?

Pour faire un choix rationnel du type de fichier à choisir, il faut comprendre le processus qui aboutit à l’affichage final.

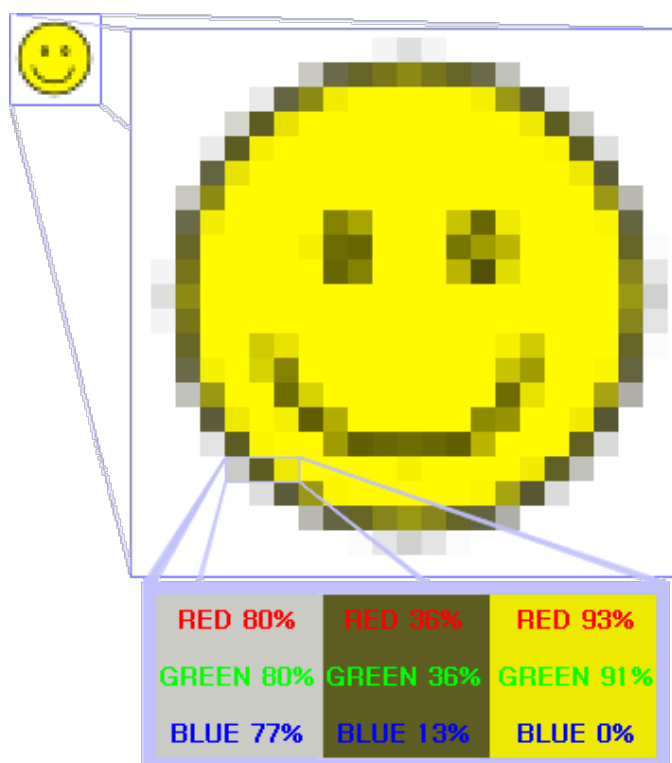
Quelque soit le type de fichier choisit, “in fine” il faut l’imprimer ou l’afficher.

Les “pixels” de l’écran ou les “gouttes d’encre” de l’imprimante imposent un découpage de l’image et la réalisation d’une approximation de l’image de départ <sup>1</sup>.

---

1. Pour ceux d’entre vous qui ont fait de la photographie “argentique”, ceci correspond au grain du papier photo.

Ce processus est souvent appelé “pixelisation”. Le vocabulaire correct que nous utiliserons est “rastérisation”.



### 11.2.1.2 Packages et types d’images

#### Les packages et commandes

Il faut charger l’extension `\usepackage{graphicx}`<sup>2</sup> ou l’extension `\usepackage{graphics}`<sup>3</sup>.

Les deux extensions accompagnent toutes distribution  $\text{\LaTeX}$  de base, et en général, il n’y a pas de raison de ne pas toujours utiliser *graphicx*.

En effet, si on utilise `\usepackage{graphics}`, le fichier image doit être au format “ postscript encapsulé ” et l’extension du fichier doit être “ .eps ”.

Alors que `\usepackage{graphicx}` permettra d’inclure d’autres formats d’images (pdf, jpg, png, ...).

La commande à utiliser est : `\includegraphics[options]{Chemin_accès_du_fichier\Nom_Fichier}`.

#### Inclusion d’images postscripts

Le format le plus “ natif ” à utiliser sous  $\text{\LaTeX}$  est le postscript. Mais ce format d’image est de moins en moins utilisé.

#### Inclusion d’images PDF, PNG, JPEG

Avec “ pdflatex ”, d’autres formats d’images peuvent être utilisés.

Ceci est cependant générateur d’autres problèmes comme par exemple, la gestion de la “ bounding box ”.

2. Qui est plus moderne et fonctionne aussi avec du PDF.

3. Qui est “ orientée ” postscript.

## Création d'images externes : problèmes de cadres

Le postscript encapsulé présente à  $\text{\LaTeX}$  des informations sur la taille de l'image.  $\text{\LaTeX}$  en général attend cette information puisque l'image étant un " flottant ",  $\text{\LaTeX}$  doit savoir comment la placer.

Si l'image n'est pas du postscript encapsulé,  $\text{\LaTeX}$  va souvent se plaindre de la " bounding box " <sup>4</sup> manquante.

Une solution est le programme " ebb " qui va créer un fichier de " bounding box ". Il faut alors transmettre ce fichier à  $\text{\LaTeX}$ .

### 11.2.1.3 Types de fichiers images

#### Rendre à César... pardon à René!

L'information qui suit est largement inspirée de l'article de Klaus Höppner traduit par René Fritz <sup>5</sup> sous le titre : *Stratégies pour inclure des graphiques dans des documents en  $\text{\LaTeX}$* . Nous tenons à remercier René Fritz de nous avoir autorisé à reproduire son travail en le placant sous une licence libre. Sa seule demande a été que l'origine de l'information soit mentionnée, ce que nous faisons ici.

L'article original peut être trouvé ici : <http://www.gutenberg.eu.org/IMG/pdf/Graphiques.pdf>

#### Présentation des formats

En général, les graphiques existent sous deux types de formats :

- les vectoriels
- et les matriciels.

#### Images Vectorielles

Les images vectorielles sont mises en place en dessinant ou en remplissant des objets *géométriques* tels que des lignes, des courbes de Bézier, des polygones, des cercles, etc. Les propriétés de ces objets sont stockées *mathématiquement*.

Les graphismes vectoriels sont en général des dispositifs indépendants.

Il est facile de les mettre à l'échelle ou de les tourner sans perte de qualité, car leur rasterisation <sup>6</sup> est effectuée par l'imprimante ou son pilote.

#### Images matricielles

Ce sont des images dont l'information est décrite sous la forme d'une matrice de points : bitmap en anglais. Le terme bitmap est un anglicisme à éviter.

La mise à l'échelle d'images matricielles ainsi que leur rotation produira une perte de qualité.

L'impression ou l'affichage avec un dispositif ayant une autre résolution <sup>7</sup> provoquera aussi de mauvais résultats.

Une image vectorielle peut contenir une image matricielle. La réciproque n'est pas vraie.

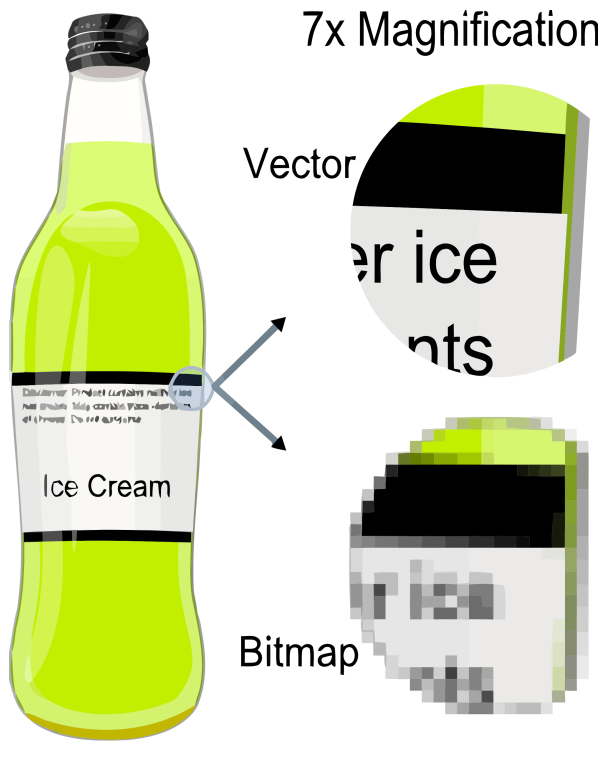
---

4. C'est à dire du cadre de l'image.

5. René est membre de GUTenberg : Le Groupe francophone des utilisateurs de TeX

6. Voici l'occasion de définir ce terme : Procédé qui consiste à convertir une image vectorielle en une image matricielle destinée à être affichée sur un écran ou imprimée par un matériel d'impression.

7. même une résolution plus basse!



### Classes d'images matricielles

Il y a différentes sortes d'images matricielles :

- sans compression (qui peut rendre vos fichiers vraiment énormes, tributaires de la résolution et de la profondeur de couleur, et je ne les traiterai pas ici),
- avec des méthodes de compression qui préservent complètement la qualité de l'image tout en réduisant la taille des données<sup>8</sup>,
- et des méthodes de compression avec pertes (lossy compression)<sup>9</sup> qui provoquent une réduction conséquente de la qualité de l'image mais aussi de la taille du fichier.

### Images matricielles avec compression conservative

Les bitmaps sans pertes de compression mémorisent les informations de l'image sous forme de pixels, chaque pixel étant d'une couleur donnée.

En principe, la qualité d'une image est meilleure avec une résolution et une profondeur d'échantillonnage accrues.

Ainsi, les fichiers GIF utilisent une profondeur d'échantillonnage de *8 bits* par pixel, ceci permet d'indexer *256 couleurs*.

Une image matricielle avec une profondeur d'échantillonnage de *24 bits* indexera approximativement 16 millions de couleurs.

### Bitmaps avec pertes de compression

Les bitmaps avec pertes de compression utilisent le fait que l'œil humain distingue assez bien de petites différences de luminosité sur une zone relativement vaste, mais ne perçoit pas exactement la force d'une variation de luminosité à fréquence élevée.

8. On parle alors de compression conservative

9. ou compression non conservative : Type de compression qui élimine les informations qui ne sont pas absolument nécessaires pour l'appréciation visuelle d'images numérisés afin d'obtenir le meilleur taux de compression possible.

Pour faire une analogie acoustique<sup>10</sup>, après le bruit du tonnerre, nous n'entendons pas le bruit du moustique. Après l'éblouissement dû à un flash, nous ne voyons pas l'allumette.

Pour cette raison, les composants d'une région de haute fréquence peuvent être réduits, conduisant à des fichiers de plus petites tailles.

Ceci fonctionne bien pour des photos qui contiennent généralement des transitions en douceur des couleurs, mais pour les graphiques avec une frontière, des artefacts peuvent se produire.

### Exemple



#### 11.2.1.4 Exemples de formats les plus utilisés

##### Formats courants

Il existe très nombreux formats de graphiques, et je ne m'intéresserai qu'à ceux qui sont plus souvent utilisés.

##### EPS

Le postscript encapsulé (ou *Encapsulated PostScript*) est surtout utilisé pour les images vectorielles mais peut contenir des images matricielles.

##### PDF

Comme le postscript encapsulé, du vectoriel qui peut contenir du matriciel.

---

10. Les formats jpeg, mpeg et mp3 obéissent non seulement au même principe mais au même algorithme!

**PNG**

Le *Portable Network Graphics* a été introduit suite à un problème de licence du format *GIF*.

Il est fort utilisé pour les images des pages web.

Ce format permet aussi bien la compression conservative que non conservative.

Il faut faire attention aux options lors de l'enregistrement d'une image donc.

**JPEG**

JPEG sigle de *Joint Photographic Experts Group* est un format matriciel avec pertes.

Il est souvent utilisé pour les photographies (La plupart des appareils photo numériques produisent des fichiers JPEG).

**TIFF**

TIFF ou *Tag(ged) Image File Format* est un format matriciel.

Il est parfois utilisé pour des images de haute qualité, en partie parce qu'il supporte l'espace colorimétrique CMYK<sup>11</sup>, important notamment pour l'impression commerciale.

Ce format permet aussi bien la compression conservative que non conservative.

**11.2.1.5 Quel format choisir?****Formats dans la pratique**

Maintenant, la question est : Quel format doit-je utiliser pour quel but ?

Bien qu'il n'y ait pas une véritable réponse à cette question, voici mes conseils.

**Vive la liberté**

Pour les dessins (par exemple, des dessins techniques ou distribution de points ) utilisez des graphiques vectoriels.

Ils vous donnent le maximum de liberté pour manipuler une image que, lors de son inclusion dans un document, vous avez souvent besoin de redimensionner pour l'adapter à votre présentation.

En outre, ils sont indépendants du périphérique de sortie, et donc vous pouvez zoomer sur l'image dans votre visualiseur de document sans voir chacun des pixels.

Les outils de dessin offerts par les distributions TEX – notamment TIKZ, PSTricks et MetaPost, peuvent habituellement produire de façon native des sorties en EPS ou PDF.

La plupart des programmes de dessin vectoriels comme xfig et Corel Draw offrent également une fonctionnalité d'exportation pour produire des sorties en EPS ou PDF (même si parfois bogués).

C'est aussi le cas des programmes de géométrie dynamique ou d'algèbre.

**Coincé avec du matriciel**

Si vous êtes coincé avec des images matricielles, utilisez le format PNG pour les images avec de fortes transitions de couleur, comme le passage du noir au blanc.

---

11. Palette de couleurs CMJN : cyan, magenta, jaune, noir (CMYK en anglais : cyan, magenta, yellow, black).

## Y'a pas photo

Pour les photographies, vous pouvez utiliser le format JPEG dans la plupart des cas.

En effet, la perte de qualité par compression est normalement imperceptible à l'impression.

Sur la plupart des dispositifs, une résolution de 100 à 200 dpi sera suffisante.

Rappelez-vous que la résolution de l'écran est normalement d'environ 75 à 100 dpi, et que les imprimantes couleur prétendent avoir à haute résolution, mais ont des tirages couleur lissés, de sorte que vous aurez du mal à remarquer la différence par rapport au format JPEG avec une meilleure résolution.

### 11.2.2 L'environnement “figure”

#### Faire bonne “figure”

Il y a un environnement important pour gérer les images dans un texte, c'est l'environnement “figure”.

#### Intêret

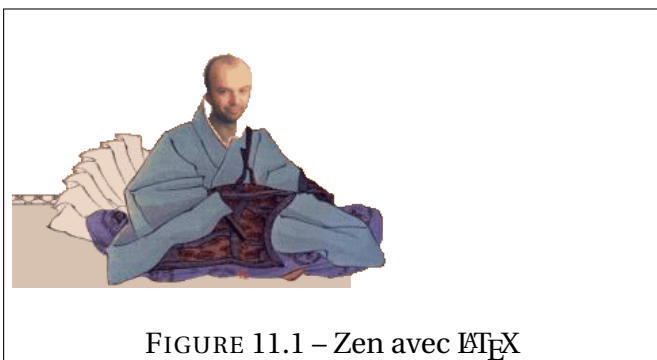
- Cet environnememnt va donner automatiquement un numéro à la figure.
- La commande `\listoffigures` à la fin du document va créer automatiquement un liste des figures.
- La commande (optionnelle) `\caption{texte sous la figure}` va, elle, ajouter un texte.
- Et il est également possible d'ajouter un “label” pour faire automatiquement référence à la figure et à son numéro de page ailleurs dans le texte.

Voici un exemple :

```
\begin{figure}
  \includegraphics{Graphisme/images/computer-science-mindmap.pdf}
  \caption{Carte heuristique avec Pgf/Tikz}
  \label{tikzmindmap}
\end{figure}
```

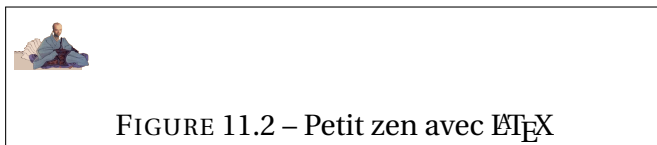
### 11.2.3 Syntaxe de “includegraphics”

#### Différentes options du includegraphics



```
1 \begin{figure}
2   \includegraphics[width=5cm]{
3     Graphisme/images/yvzen.png}
4   \caption{Zen avec  $\LaTeX$ {}}
5   \label{fig :yvzen1}
6 \end{figure}
```



**width**

```

1 \begin{figure}
2   \includegraphics%
3   [width=1cm]%
4   {Graphisme/images/yvzen.png}
5   \caption{Petit zen avec \LaTeX{}}
6   \label{fig :yvzen2}
7 \end{figure}

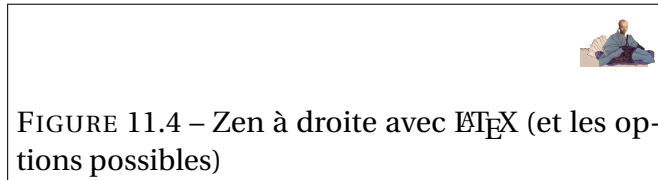
```

**centrage**

```

1 \begin{figure}
2   \begin{center}
3     \includegraphics[width=1cm]{
4       Graphisme/images/yvzen.png}
5     \caption{Petit zen au milieu avec \
6       LaTeX{}}
7     \label{fig :yvzen3}
8   \end{center}
9 \end{figure}

```

**pousser à droite**

```

1 \begin{figure}[htbp] %[htbp]
2   \hfill \includegraphics[width=1cm]{
3     Graphisme/images/yvzen.png}
4   \caption{Zen \‘a droite avec \LaTeX
5     {} (et les options possibles)}
6   \label{fig :yvzen4}
7 \end{figure}

```

**tourner**

```

1 \begin{figure}
2   \begin{center}
3     \includegraphics%
4     [width=3cm,angle=180]%
5     {Graphisme/images/yvzen.png}
6     \caption{Pas si zen avec \LaTeX{}}
7     \label{fig :yvzen3}
8   \end{center}
9 \end{figure}

```

**autres options**

Les options de `\includegraphics` sont : séparées par des virgules. Voici quelques exemples :

- `height` (hauteur, unités : pt, cm, mm, inch, ...)
- `scale` (mise à échelle) `scale=0.3`
- `bb BoundingBox` : `bb=(bblx bbly bbux bbuy)`
- `clip=true, draft=true, command=...`

### Précisions sur les chemins d'accès

Le chemin d'accès au fichier peut être absolu ou relatif et la syntaxe dépend du système opératoire utilisé.

Ce qui signifie (en français de France) que je peux écrire :

- `\includegraphics{intro/stallman.ps}`
- ou `\includegraphics{/home/yves/docs/.../cours/latex/Ateliers_urem/cours/intro/stallman.ps}`

La première méthode est clairement plus facile. Elle permet aussi de déplacer tout un projet (comme ce cours) sans devoir modifier à la main toutes les commandes d'inclusion de fichier.

### Systemes opératoires et chemins d'accès

Le système opératoire (ou “ operating system ”), c’est le programme (ou plutôt les programmes) qui fait fonctionner votre ordinateur et qui permet à tous les autres programmes de fonctionner.

Exemples :

- MacOSX,
- Les divers Windows,
- Dos,
- les Unix's et Linux's,
- et plein d'autres.

Sous Dos et Windows, la différence est l'utilisation de la contre oblique comme séparateur de nom de dossier.

- Linux : `\includegraphics{intro/stallman.ps}`
- Dos : `\includegraphics{c : \ex_LaTeX_1 \stallman.ps}`

#### 11.2.3.1 Ce manuel

##### Les flottants

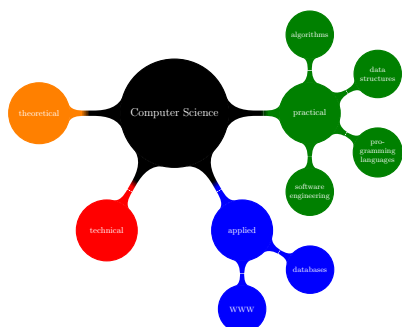
Le manuel lui-même comprend des constructions qui n'ont pas été évoquées jusqu'ici. Par exemple, les tableaux ayant une légende et un numéro sont ce que  $\LaTeX$  appelle des “ flottants ”. Un flottant est une structure qui sera placée dans le document de façon automatique par  $\LaTeX$  en fonction de la place disponible, donc pas nécessairement au niveau où cette structure a été tapée dans le source.  $\LaTeX$  prévoit des flottants pour les tableaux et pour les figures. On notera la présence d'une *Liste des tableaux* à la fin du manuel, liste qui a été créée automatiquement.

##### Inclusion dans un paragraphe

La première version de cet ouvrage présentait des inclusions de dessins dans des paragraphes avec un découpage du texte autour du dessin. Là aussi, tout est fait de façon automatique, le source indique seulement qu'il faut placer un certain dessin et le texte qui l'entourera mais sans se soucier de savoir où se feront les coupures de lignes et quand les lignes auront de nouveau toute la largeur de la page à leur disposition.

##### Juxtaposition du code source et du résultat

Les exemples présentant un source et son résultat (soit côte à côte, soit l'un en-dessous de l'autre) ont été faits de façon automatique grâce à un environnement d'un package particulier. Le texte n'a été tapé qu'une seule fois de telle sorte que le résultat montré est forcément celui obtenu par le source en vis-à-vis.



## Tableaux

Les tableaux 9.1 et 9.2 page 90 auraient été très pénibles à taper si l’auteur n’avait pas créé quelques macros pour se débarrasser d’une bonne partie du travail répétitif. Cela devrait d’ailleurs devenir un réflexe lorsqu’on travaille sous  $\LaTeX$  : tout travail répétitif devrait faire l’objet d’une ou de plusieurs macros (ou environnements).

### 11.2.4 Des commandes dans le code source

#### Retour aux sources

Une série d’extensions existent qui permettent d’inclure du code source “ spécial ” dans le code source  $\LaTeX$ .

#### PsTricks

Le plus riche mais il pose des problèmes avec la production de PDFs sous “ pdf $\LaTeX$  ”

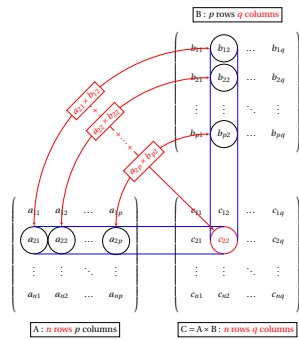
#### Pgf Tikz

Le préféré de l’UREM! Voir : <http://sourceforge.net/projects/pgf/>

Des exemples : <http://www.texample.net/tikz/>

Pgf/Tikz peut faire appel à gnuplot comme nous le verrons plus loin et profiter des capacités de calcul de ce dernier.

#### Pgf Tikz : exemples



1

## Gnuplot et Pgf/Tikz

Pgf/Tikz peut faire appel à gnuplot pour tracer des courbes de fonctions plus jolies.

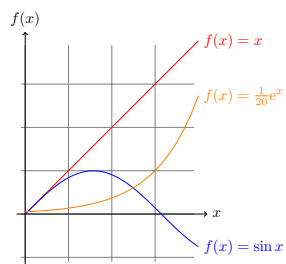
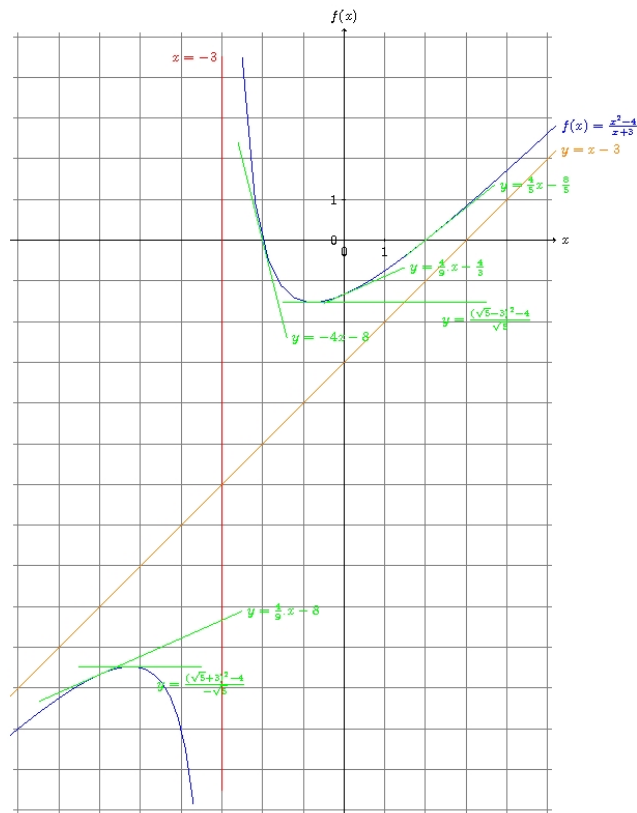


FIGURE 11.6 – Graphes de fonctions grâce à Gnuplot et Tikz

## 13 Le graphique



11

**Metapost**

Dérivé de Metafont et adoré par certains.

Voir : <http://fr.wikipedia.org/wiki/MetaPost>

Metapost

- crée des images en postscript encapsulé, (pas de pdf)
- implique de créer des fichiers externes avec le code metapost,
- ce qui implique de lancer `latex mpost nom_dufichier.mp` et à nouveau `latex`.
- La précision mathématique est celle de *metafont* c'est à dire très limitée.
- N'a plus l'air activement développé.

**Asymptote**

Inspiré de Metapost mais avec un syntaxe à la C++.

Voir : <http://asymptote.sourceforge.net/>

Asymptote

- a des sorties eps, pdf, ...

- permet la 3d,
- les animations...
- le code asymptote peut être inclus dans le document  $\LaTeX$  via `\usepackage{asymptote}` et `\begin{asy} ... \end{asy}`.

C'est parce que Tikz est associé à "beamer" qu'asymptote n'a pas été "retenu" à l'UREM.

#### 11.2.4.1 Dessins scientifiques

Comme  $\LaTeX$  a été très vite employé par l'ensemble des scientifiques, des extensions diverses et variées ont fleuri pour permettre des compositions faciles de schémas dans différentes disciplines.

Il est hors de question de montrer toutes les possibilités.

Des extensions permettent ainsi de construire des graphes, des arbres généalogiques ou syntaxiques ou ..., des nœuds, des molécules (avec toute la panoplie de molécules cycliques, de liaisons diverses, ...), des diagrammes de Feynman, des diagrammes de cycles pour les ordinateurs, des schémas optiques, des schémas électroniques, des circuits analogiques ou numériques.

#### Chimie

L'extension `ppchtex` permet de dessiner des molécules organiques. Il n'est pas trivial à utiliser.

voir : <http://dev.ulb.ac.be/urem/Dessiner-des-molecules-avec-LaTeX>

#### 11.2.4.2 Jeux

Les jeux nécessitent des compositions très spécialisées qui doivent suivre des règles strictes pour être facilement lisibles de la part d'un habitué (présentation d'une partie, diagrammes divers, ...). Des extensions permettent de gérer parfaitement les échecs, les échecs chinois, le go, le backgammon, les jeux de cartes (bridge par exemple), les mots croisés ou fléchés.

### 11.2.5 Graphismes avec appel à des programmes extérieurs

Une autre voie consiste à faire appel à des programmes annexes qui produisent des fichiers pouvant être inclus dans un source  $\LaTeX$ , soit sous forme d'images PostScript (ou d'autres formats avec PDF $\LaTeX$ ), soit directement sous une forme directement compréhensible par  $\LaTeX$ . Là aussi, il n'est pas possible de réaliser une présentation exhaustive, on pourra citer les programmes utilisés au moins une fois par l'auteur : Gnuplot (tracés de courbes), Scilab (mathématique formelle), Xfig (dessin vectoriel), METAPOST (comme METAFONT mais produisant des figures PostScript au lieu des fontes), etc.

Il faut, en général passer des arguments supplémentaire à  $\LaTeX$  comme `-enable-write18` ou `-shell-escape` et compiler deux fois.

#### gnuplot

Gnuplot a servi pour réaliser le graphique de fonction dans l'exemple d'examen de mathématique de l'atelier II.

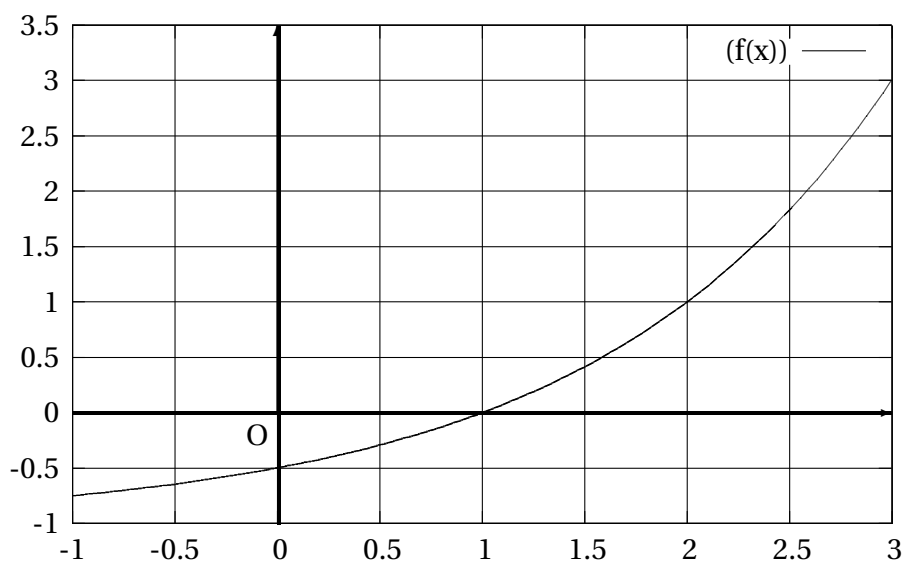
Il faut charger l'extension `\usepackage{gnuplottex}` et enfermer les commandes gnuplot dans un environnement.

Voici le code utilisé :

```

\begin{gnuplot}
f(x)=-1+0.5*2**(x)
set xrange [-1 :3]
set label "0" at first -0.15, first -0.2
set xzeroaxis lt 5
set yzeroaxis lt 5
set arrow from 0,0 to 3,0 ls 4
set arrow from 0,0 to 0,3.5
set grid
plot (f(x))
\end{gnuplot}

```



### Création d'images externes : les bons programmes

En vrac : xfig, gnuplot, scilab, geogebra ...

Ces programmes sont tous capables de générer directement du postscript encapsulé (ou du pdf) utilisable donc directement sous  $\text{\LaTeX}$ .

Ils peuvent aussi parfois être appelé par  $\text{\LaTeX}$  et les commandes spécifiques sont alors incluses dans le code  $\text{\LaTeX}$  moyennant le chargement d'une extension et un environnement spécifique.

### xfig

Xfig n'est pas très " joli " mais il permet de créer des images avec du texte et ce texte sera automatiquement dans la police de caractère de votre document.

### TeXGraph

TeXGraph mérite une mention spéciale. Vous réalisez un dessin et TeXGraph va générer le code correspondant soit en pstricks soit en tikz ou au choix en :

- $\text{\LaTeX}$ ,
- postscript,
- pdf,
- et bien d'autres

Voir : <http://texgraph.tuxfamily.org/>



# Chapitre 12

## Graphisme avec Tikz

### Sommaire

---

<b>12.1 Présentation de Pgf/Tikz</b> . . . . .	<b>130</b>
12.1.1 Créateur et acronymes . . . . .	130
12.1.2 Justification du choix de Tikz . . . . .	130
12.1.3 Motivation des choix pédagogiques . . . . .	130
<b>12.2 Syntaxe de Tikz</b> . . . . .	<b>130</b>
12.2.1 Appeler Tikz . . . . .	130
12.2.2 Les bases de pgf/tikz . . . . .	131
<b>12.3</b> . . . . .	<b>138</b>
12.3.1 Path, coordinate et node . . . . .	138
12.3.2 Opérations répétitives et programmation . . . . .	140
12.3.3 Graphes de fonctions . . . . .	140

---

## 12.1 Présentation de Pgf/Tikz

### 12.1.1 Créateur et acronymes

#### Pgf et Tikz

Pgf/Tikz est un “tandem” créé par Till Tantau.

Pgf est l’acronyme de “Portable Graphics Format”. C’est un programme et un langage de *bas niveau*.

Tikz est un acronyme récursif pour “TikZ ist kein Zeichenprogramm” (Tikz n’est pas un programme graphique!) C’est un programme et un langage de *haut niveau*.

### 12.1.2 Justification du choix de Tikz

#### Pourquoi Tikz?

Nous utilisons Tikz pour toute une série de raisons :

- Il a des sorties eps *et* pdf, mais aussi svg et dvi.
- Il permet les animations,
- la 3d,
- des calculs (opérations élémentaires, intersections, tangentes, ...).

C’est parce que Tikz est associé à “beamer” qu’il a été “retenu” à l’UREM.

Ceci nous permet de faire des présentations, des affiches et d’utiliser le même code pour produire aussi articles et livres via : `\usepackage{beamerarticle}`.

### 12.1.3 Motivation des choix pédagogiques

#### Mais pourquoi ne dit il rien de ... ???

Pgf/Tikz n’est pas un programme “pédagogique”.

Sa syntaxe est relativement simple mais est néanmoins très riche.

Le manuel “officiel” fait plus de 500 pages!

Pour réaliser cette courte introduction, nous avons utilisé le manuel officiel et divers documents.

Nous irons à l’essentiel en ne présentant pas toutes les options et choix possibles!

Pour avancer rapidement, nous “mélangerons” parfois les concepts et ferons beaucoup appel à des exemples.

## 12.2 Syntaxe de Tikz

### 12.2.1 Appeler Tikz

#### Commencer par le commencement.

Pour faire des illustrations avec pgf/tikz, il faut commencer

- par charger le “package” tikz
  - “`\usepackage{tikz}`”
  - et faire éventuellement des appels à des bibliothèques diverses, par ex. : “`\usetikzlibrary{calc,arrows,sh`

Le code d’une illustration tikz peut alors être inclus dans le document  $\LaTeX$  via

- `\begin{tikzpicture} ... \end{tikzpicture}`. (pour des figures complexes avec plusieurs commandes)
- ou `\tikz ...` (pour une seule commande).
- Les commandes se terminent avec un *point virgule* “ ; ”.

## 12.2.2 Les bases de pgf/tikz

### Programme de base pour Tikz

Nous allons explorer ici la syntaxe de base de Tikz :

- les objets graphiques de base :
  - points et systèmes de coordonnées,
  - rectangles, grilles, cercles, ellipses et arcs;
  - lignes brisées et courbes,
- les notions de :
  - commande de dessin et ses options (draw),
  - chemin (path),
  - node.
- Comment faire des graphes de fonctions sur base de tout cela.

### 12.2.2.1 Points et coordonnées

#### Soyons ... un peu carrés

Il y a plusieurs façons de donner la position d'un point avec Tikz.

Limitons nous pour l'instant

- aux coordonnées cartésiennes
- et aux coordonnées polaires.

#### Soyons cartésiens

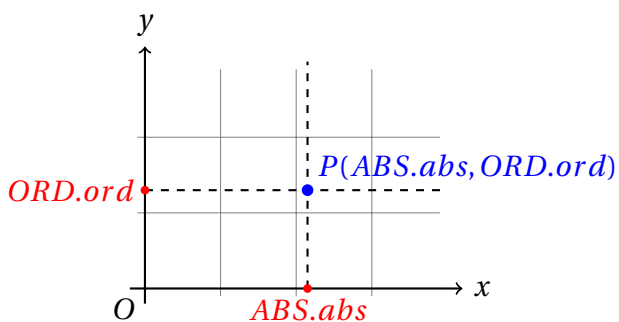


FIGURE 12.1 – Un point avec PgfTikz

La syntaxe est donc “(ABS . abs , ORD . ord)”, où :

- le séparateur entre abscisse “ABS . abs” et ordonnée “ORD . ord” est la *virgule* “ , ”;
- le séparateur entre entiers “ABS ORD” et décimaux “abs ord” est le *point* “ . ”.

#### Les coordonnées polaires

La syntaxe est donc “(Angle . a : Rayon . r)”, où :

- l'*angle* est en *degré*;
- le séparateur entre angle “Angle . a” et rayon “Rayon . r” est le *double point* “ : ”;
- le séparateur entre entiers “Angle Rayon” et décimaux “a r” est le *point* “ . ”.

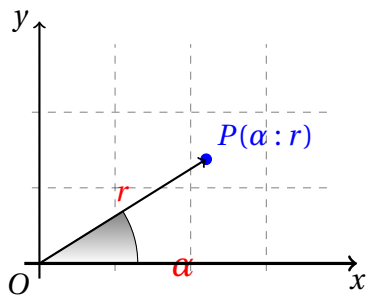
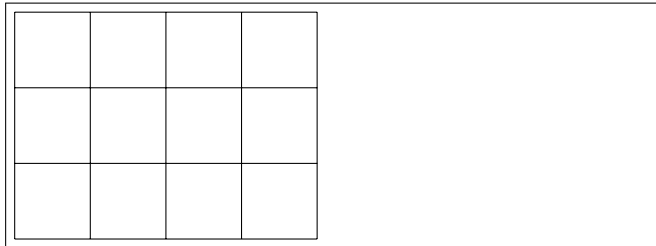


FIGURE 12.2 – Les coordonnées polaires avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw (0,0) grid (4,3) ;
3 \end{tikzpicture}

```

FIGURE 12.3 – Une grille avec PgfTikz

### Remarque sur les unités

Je viens de dire à l'instant qu'il faut donner l'*angle* en *degré*, et n'ai rien dit, en terme d'unité pour le rayon.

Comme je n'ai pas fixé d'unité, tikz choisit par défaut le *cm*. Comme l'image choisie est vectorielle, tout est à l'échelle. Il est possible de choisir différentes unités de longueur : *cm*, *mm*, *inch* ou *pt*. L'essentiel est de ne pas les mélanger.

### 12.2.2.2 Les formes

#### Les formes pas la forme

Commençons avec des formes données en coordonnées cartésiennes.

#### Syntaxe des grilles en coordonnées cartésiennes

Pour ce faire, créons une grille, presque un repère : Ce faisant, j'utilise les coordonnées cartésiennes, puisque le *coin inférieur gauche* et le *coin supérieur droit* de la grille sont ici donnés en coordonnées cartésiennes.

La commande de dessin est `\draw`.

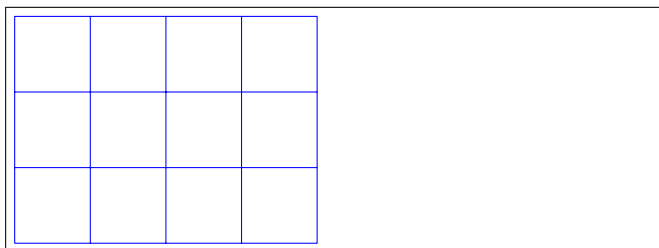
Et mettons un peu de couleur! `[color=blue]` colore en bleu ce qui sera représenté suite à la commande `\draw`.

Nous aurions pu mettre l'option `[blue]` qui est équivalente.

C'est ce que nous ferons par la suite.

#### Syntaxe des cercles en coordonnées cartésiennes

Syntaxe : (position du centre) circle (longueur du rayon)

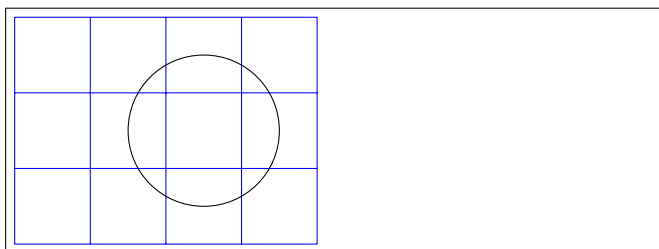


```

1 \begin{tikzpicture}
2   \draw[color=blue] (0,0) grid (4,3) ;
3 \end{tikzpicture}

```

FIGURE 12.4 – Une grille bleue avec PgfTikz



```

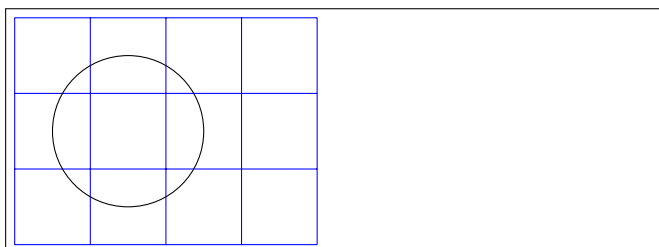
1 \begin{tikzpicture}
2   \draw[blue] (0,0) grid (4,3) ;
3   \draw (2.5,1.5) circle (1) ;
4 \end{tikzpicture}

```

FIGURE 12.5 – Un cercle avec PgfTikz

### Les coordonnées polaires

Le même résultat peut être obtenu via les coordonnées polaires.



```

1 \begin{tikzpicture}
2   \draw[blue] (0,0) grid (4,3) ;
3   \draw (45 :1.5 * 1.4142) circle (1) ;
4 \end{tikzpicture}

```

FIGURE 12.6 – Un cercle en coordonnées polaires avec PgfTikz

Remarquez que j'introduis ici un premier calcul (certes élémentaire), via la longueur  $1.5 * \sqrt{2}$ .

### Les points

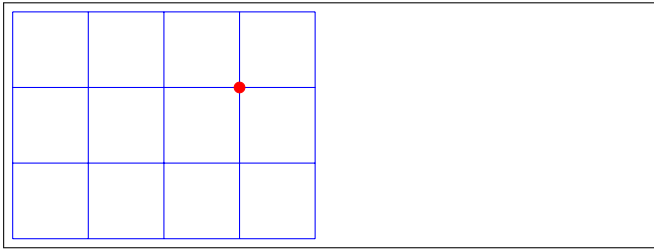
Voici un cercle, la position du point est celle du centre et l'option `fill` et la taille du rayon sont utilisés pour représenter un point.

### Exemples de cercles

Les exemples repris ici sont extraits du magnifique document de Andrew Mertz.

### Les ellipses

Pour dessiner des ellipses, on donne les *centres* et les longueurs des *axes* horizontaux et verticaux.



```

1 \begin{tikzpicture}
2   \draw[blue] (0,0) grid (4,3) ;
3   \draw[fill,red] (3,2) circle (2pt) ;
4 \end{tikzpicture}

```

FIGURE 12.7 – Un cercle avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw (0,0) circle (1cm)
3         circle (0.6cm)
4         circle (0.2cm) ;
5 \end{tikzpicture}

```

FIGURE 12.8 – Cercles concentriques avec PgfTikz

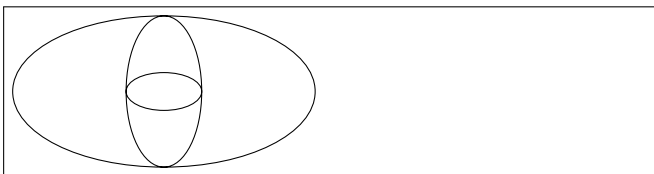


```

1 \begin{tikzpicture}
2   \draw (0,0) circle (1cm) ;
3   \draw (0.5,0) circle (0.5cm) ;
4   \draw (0,0.5) circle (0.5cm) ;
5   \draw (-0.5,0) circle (0.5cm) ;
6   \draw (0,-0.5) circle (0.5cm) ;
7 \end{tikzpicture}

```

FIGURE 12.9 – Cercles excentriques avec PgfTikz

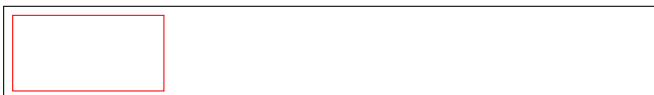


```

1 \begin{tikzpicture}
2   \draw (0,0) ellipse (2cm and 1cm)
3         ellipse (0.5cm and 1 cm)
4         ellipse (0.5cm and 0.25cm)
5         ;
6 \end{tikzpicture}

```

FIGURE 12.10 – Ellipses avec PgfTikz

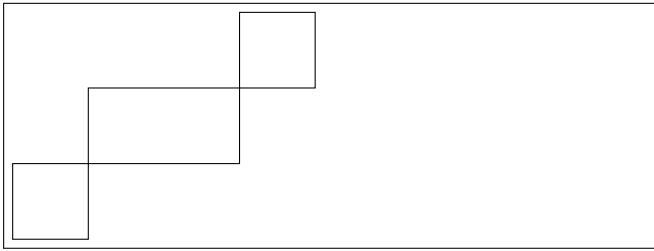


```

1 \begin{tikzpicture}
2   \draw[red] (0,0) rectangle (2,1) ;
3 \end{tikzpicture}

```

FIGURE 12.11 – Rectangle avec PgfTikz

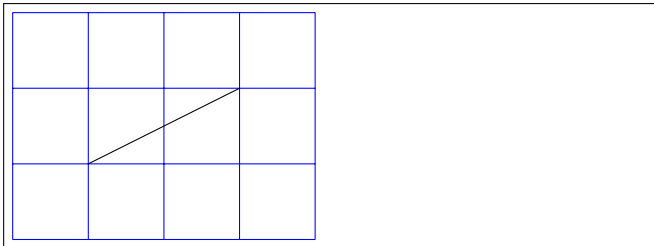


```

1 \begin{tikzpicture}
2   \draw (0,0) rectangle (1,1)
3         rectangle (3,2)
4         rectangle (4,3) ;
5 \end{tikzpicture}

```

FIGURE 12.12 – Rectangles avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw[blue] (0,0) grid (4,3) ;
3   \draw (1,1) -- (3,2) ;
4 \end{tikzpicture}

```

FIGURE 12.13 – Un segment avec PgfTikz

### Les rectangles

Comme pour la grille, il faut renseigner les coordonnées du coin inférieur gauche et du coin supérieur droit du rectangle.

#### 12.2.2.3 Les lignes brisées

##### Les lignes droites

Commençons plus exactement par les segments!

##### Malevitch revisité

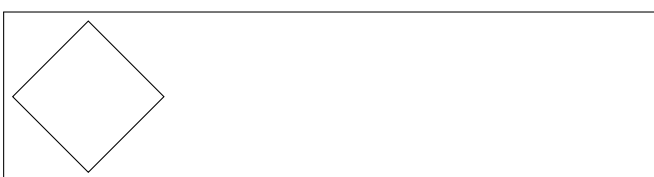
Les exemples repris ici sont extraits du magnifique document de Andrew Mertz.

##### cycle

#### 12.2.2.4 Les lignes courbes

##### Courbes de Bézier

Il faut donner un point de départ, un d'arrivée et deux points appartenant aux tangentes.

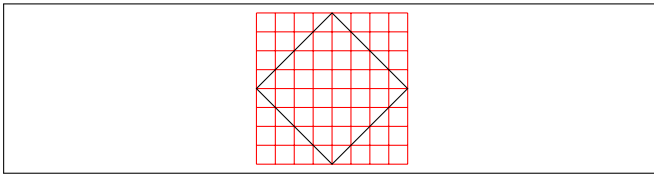


```

1 \begin{tikzpicture}
2   \draw (1,0) -- (0,1) --
3         (-1,0) -- (0,-1) -- cycle ;
4 \end{tikzpicture}

```

FIGURE 12.14 – Carré sur pointe avec PgfTikz



```

1 \begin{center}
2   \begin{tikzpicture}
3     \draw[step=0.25cm,red]
4       (-1,-1) grid (1,1) ;
5     \draw (1,0) -- (0,1) --
6       (-1,0) -- (0,-1) -- cycle ;
7   \end{tikzpicture}
8 \end{center}

```

FIGURE 12.15 – Carré avec grille rouge avec PgfTikz

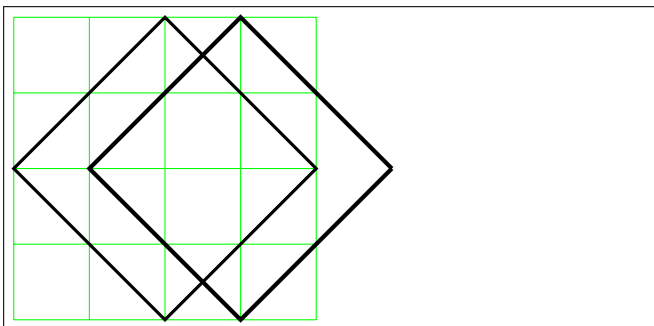


```

1 \begin{tikzpicture}
2   \draw[step=0.5cm,green]
3     (-1,-1) grid (1,1) ;
4   \draw (1,0) -- (0,1) -- (-1,0)
5     -- (0,-1) -- cycle ;
6 \end{tikzpicture}

```

FIGURE 12.16 – Carré avec grille verte avec PgfTikz

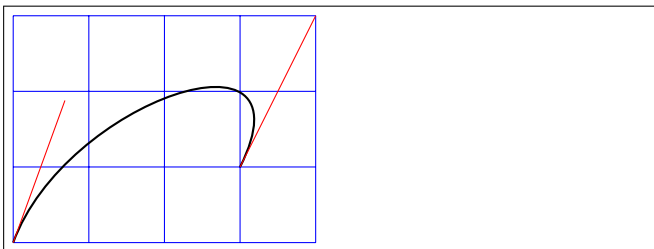


```

1 \begin{tikzpicture}[scale=2]
2   \draw[step=0.5cm,green]
3     (-1,-1) grid (1,1) ;
4   \draw[very thick] (1,0) -- (0,1) --
5     (-1,0)
6     -- (0,-1) -- cycle ;
7   \draw[xshift=0.5cm,ultra thick] (1,0)
8     -- (0,1) -- (-1,0)
9     -- (0,-1) -- (1,0) ;

```

FIGURE 12.17 – Cycle avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw[blue] (0,0) grid (4,3) ;
3   \draw[thick] (0,0)..controls(70 :2)
4     and(4,3)..(3,1) ;
5   \draw[red] (0,0)--(70 :2)
6     (4,3)--(3,1) ;
7 \end{tikzpicture}

```

FIGURE 12.18 – Une courbe de Bézier avec PgfTikz





```

1 \begin{tikzpicture}
2   \draw (0 :1cm) -- (0 :2cm)
3         arc (0 :60 :2cm) -- (60 :1cm)
4         arc (60 :0 :1cm) -- cycle ;
5 \end{tikzpicture}

```

FIGURE 12.19 – Arcs avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw (0,0) -- ++(1,0) --
3         ++(1,1) -- ++(1,-1) ;
4 \end{tikzpicture}

```

FIGURE 12.20 – Points avec déplacements relatifs avec PgfTikz

### Les arcs

Syntaxe : (Angle de départ, angle d'arrivée, rayon) Attention! Un arc démarre depuis le dernier point donné.

#### 12.2.2.5 Les déplacements

##### Déplacements relatifs

Le déplacement est donné par un vecteur partant du point précédent.

##### Déplacements absolus

Le déplacement est donné par un vecteur partant du premier point.

#### 12.2.2.6 Flèches

##### de tout bois

Ne serait ce que pour dessiner un référentiel cartésien, les flèches sont nécessaires.

Tikz en possède une panoplie

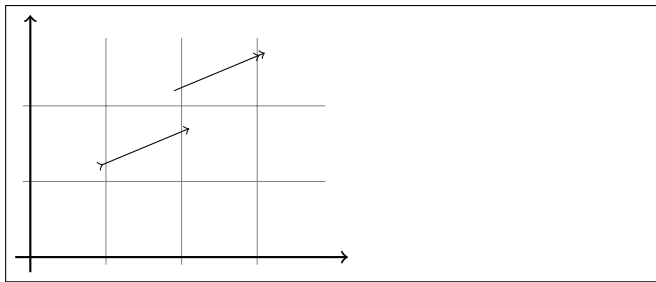


```

1 \begin{tikzpicture}
2   \draw (0,0) -- +(1,0) --
3         +(0,-1) -- +(-1,0) --
4         +(0,1) ;
5 \end{tikzpicture}

```

FIGURE 12.21 – Points avec déplacements absolus avec PgfTikz

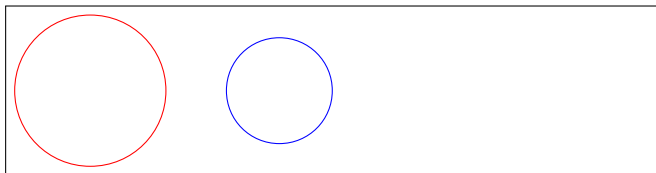


```

1 \begin{tikzpicture}
2   \draw[very thin,color=gray]
3     (-0.1,-0.1) grid (3.9,2.9) ;
4   \draw[thick,->]
5     (-0.2,0) -- (4.2,0) ;
6   \draw[thick,->]
7     (0,-0.2) -- (0,3.2) ;
8   \draw[>->]
9     (0.9,1.2) -- (2.1,1.7) ;
10  \draw[->>]
11    (1.9,2.2) -- (3.1,2.7) ;
12 \end{tikzpicture}

```

FIGURE 12.22 – Flèches avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw[red] (0,0) circle (1cm) ;
3   \path[draw,blue] (2.5,0) circle (0.7cm)
4     ;
5 \end{tikzpicture}

```

FIGURE 12.23 – Path avec PgfTikz

## 12.3

### 12.3.1 Path, coordinate et node

#### 12.3.1.1 Path

##### path

Donner un "path", c'est "expliquer son chemin à Tikz"! Les commandes peuvent presque toujours s'écrire selon un `\path`. Ceci peut s'avérer pratique et simplifier l'écriture mais c'est souvent déroutant quand on lit du code.

#### 12.3.1.2 Coordinate

##### Coordonnées sans dessiner

Il est souvent pratique de pouvoir référencer un point sans pour autant le dessiner de suite.

Ceci afin de référencer un point plusieurs fois mais en n'ayant le modifier qu'à un seul endroit.

##### path et coordinate

Ici aussi l'utilisation de `\path` est courante Attention que la syntaxe des commandes est différente.

##### Pentagone et autres

Plusieurs choses sont à remarquer dans l'exemple du pentagone :



```

1 \begin{tikzpicture}
2   \coordinate (Origine) at (0,0) ;
3   \coordinate (I) at (1,0) ;
4   \coordinate (J) at (0,1) ;
5   \draw[->] (Origine) -- (I) ;
6   \draw[->] (Origine) -- (J) ;
7 \end{tikzpicture}

```

FIGURE 12.24 – coordinate avec PgfTikz



```

1 \begin{tikzpicture}
2   \path (Origine) coordinate (0,0) ;
3   \coordinate (I) at (1,0) ;
4   \coordinate (J) at (0,1) ;
5   \draw[->] (Origine) -- (I) ;
6   \draw[->] (Origine) -- (J) ;
7 \end{tikzpicture}

```

FIGURE 12.25 – Path et coordinate avec PgfTikz

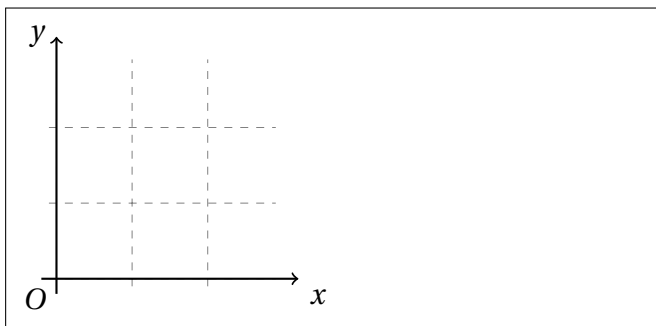


```

1 \begin{tikzpicture}
2   % Define the points of a regular
3   % pentagon
4   \path (0,0) coordinate (origin) ;
5   \path (0 :1cm) coordinate (P0) ;
6   \path (1*72 :1cm) coordinate (P1) ;
7   \path (2*72 :1cm) coordinate (P2) ;
8   \path (3*72 :1cm) coordinate (P3) ;
9   \path (4*72 :1cm) coordinate (P4) ;
10  % Draw the edges of the pentagon
11  \draw (P0) -- (P1) --
12        (P2) -- (P3) --
13        (P4) -- cycle ;
14  % Add "spokes"
15  \draw (origin) -- (P0) (origin) -- (P
16        1)
17        (origin) -- (P2)
18        (origin) -- (P3) (origin) -- (P
19        4) ;
20 \end{tikzpicture}

```

FIGURE 12.26 – Pentagone avec PgfTikz



```

1 \begin{tikzpicture}
2   \draw[very thin,gray,dashed]
3     (-0.1,-0.1) grid (2.9,2.9) ;
4   \draw[thick,->]
5     (-0.2,0) -- (3.2,0)
6     node[below right] {$x$} ;
7   \draw[thick,->]
8     (0,-0.2) -- (0,3.2)
9     node[left] {$y$} ;
10  \path (0,0) coordinate (Origine) ;
11  \draw (Origine)
12    node[below left] {$0$} ;
\end{tikzpicture}

```

FIGURE 12.27 – node avec PgfTikz

- `\path (0,0) coordinate (origin)` ; est équivalent à `\path (Origine) coordinate (0,0)` ;
- `(1*72 :1cm)` , `(2*72 :1cm)` , ... La rotation se fait via un calcul répétitif sur les angles.

### 12.3.1.3 Nodes

#### node = disposition de matériel

Et si nous voulons nommer les axes de notre référentiel cartésien ?

La structure à utiliser est un "node".

#### Syntaxe de node

Un node sert à placer du "matériel" au sens de  $\text{\TeX}$ , càd. du texte, une formule, ...

Les options se placent comme pour les autres commandes entre crochets : "[ ]".

Les options sont séparées par des virgules.

Les options de placement sont particulièrement importantes : `above` `below` `right` `left` OU `north` `south` `east` `west`.

Le matériel se place entre accolades : "{ }". Ex. :  $\{x\}$

## 12.3.2 Opérations répétitives et programmation

### foreach, mathparse et les polygones

#### A little bit of foreach

## 12.3.3 Graphes de fonctions

### Graphe = plot

Trois manières de faire!

- Avec *Tikz* qui délègue les calculs à  $\text{\TeX}$ ;
- avec *Tikz* qui délègue les calculs à *gnuplot*;
- avec *pgfplots*.

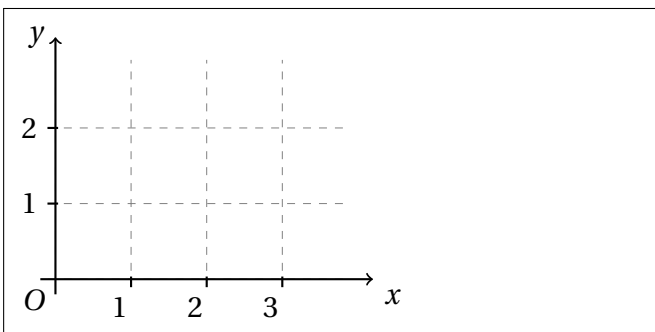


```

1 \begin{tikzpicture}
2   \pgfmathparse{15.0}\let\k\pgfmathresult
3   ;
4   \path (0,0) coordinate (origin) ;
5   \foreach \x in {1,2,...,\k}
6     \draw (origin)-- (\x *360.0/\k
7       :1)
8     -- ({(\x +1)}*360/\k :1cm) ;
9 \end{tikzpicture}

```

FIGURE 12.28 – Boucle avec PgfTikz

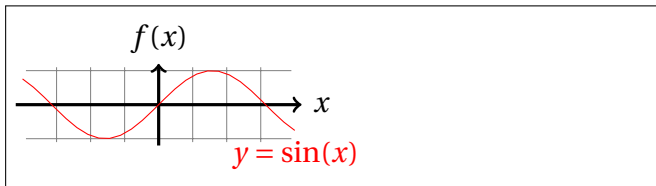


```

1 \begin{tikzpicture}
2   \draw[very thin,gray,dashed]
3     (-0.1,-0.1) grid (3.9,2.9) ;
4   \draw[thick,->]
5     (-0.2,0) -- (4.2,0)
6     node[below right] {$x$} ;
7   \draw[thick,->]
8     (0,-0.2) -- (0,3.2)
9     node[left] {$y$} ;
10  \path (0,0) coordinate (Origine) ;
11  \draw (Origine)
12  node[below left] {$O$} ;
13  \foreach \x in {1,2,3}
14    \draw[thick] (\x cm,1pt) -- (\x
15      cm,-3pt)
16    node[anchor=north,xshift
17      =-0.15cm] {$\x$} ;
18  \foreach \y/\ytext in {1,2}
19    \draw[thick] (1pt,\y cm) -- (-3
20      pt,\y cm) node[anchor=east]
21    {$\ytext$} ;
22 \end{tikzpicture}

```

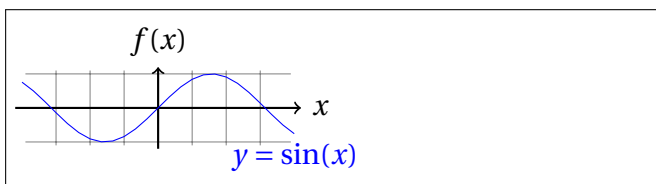
FIGURE 12.29 – Boucles avec PgfTikz



```

1 \begin{tikzpicture}
2   [scale=0.45,domain=-4 :4]
3   \draw[very thin,gray] (-3.9,-1.1) grid
4     (3.9,1.1) ;
5   \draw[very thick,->] (-4.2,0) --
6     (4.2,0) node[right] {$x$} ;
7   \draw[very thick,->] (0,-1.2) --
8     (0,1.2) node[above] {$f(x)$} ;
9   \draw[red] plot
10    (\x,{sin(deg(\x))}) node[below=0.5] {$y$
11      = \sin(x)} ;
12 \end{tikzpicture}

```

FIGURE 12.30 – Sin(x) avec Tikz et le moteur de calcul de  $\text{\LaTeX}$ 

```

1 \begin{tikzpicture}
2   [scale=0.45,domain=-4 :4]
3   \draw[very thin,gray] (-3.9,-1.1)
4     grid (3.9,1.1) ;
5   \draw[thick,->] (-4.2,0) -- (4.2,0)
6     node[right] {$x$} ;
7   \draw[thick,->] (0,-1.2) -- (0,1.2)
8     node[above] {$f(x)$} ;
9   \draw[blue] plot[id=sin]
10    function{sin(x)}
11    node[below] {$y = \sin(x)$} ;
12 \end{tikzpicture}

```

FIGURE 12.31 – Sin(x) avec Tikz

+ et -

- *Tikz* +  $\text{\LaTeX}$  : Toutes plateformes, lent, peu précis.
- *Tikz* + gnuplot ; facile sous linux, plus difficile ailleurs, précis.
- *pgfplots*. autre syntaxe.

Quelques difficultés ici :

- La syntaxe :  $(\x,{\sin(\text{deg}(\x))})$  avec  $\x$
- Les calculs sont fait par  $\text{\LaTeX}$ . C'est très gourmand en ressources systèmes, lent et imprécis!

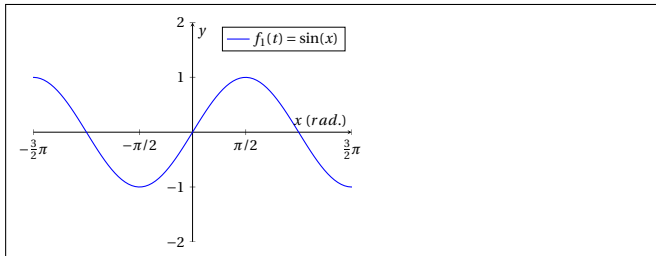
### Gnuplot = superplot

Sous linux, le programme "gnuplot" est standard et permet des *superplots*<sup>1</sup> en combinaison avec tikz!

- Les calculs sont fait par gnuplot.
- Il faut compiler deux fois. La première compilation se fait ainsi : `pdflatex --shell-escape FICHIER.tex`

### Linux : il y a moins bien mais c'est plus cher!

1. Ça n'a rien d'officiel. Mais j'adore!



```

1      \def\monpi{3.1415926535}
2 \begin{tikzpicture}[scale=0.5]
3 \begin{axis}[
4   xlabel=$x\ (rad.)$,
5   ylabel={$y$},
6   width=10cm, height=210pt,
7   xmin=-4.7124, xmax=4.7124,
8   ymin=-2, ymax=2,
9   xtick={-4.7124,-1.5708,...,10},
10  xticklabels={-$\frac{3}{2}\pi$, $-\pi$,
11             $\pi/2$, $\frac{3}{2}\pi$},
12  axis x line=center,axis y line=center,
13  no markers,
14  samples=100]
15  \addplot[mark="",color=blue,thick]
16     {\sin(x*180/\monpi)};
17  \legend{$f_1(t)=\sin(x)$};
18 \end{axis}
19 \end{tikzpicture}

```

FIGURE 12.32 – Sin(x) avec Pgfplots

Les avantages de l'utilisation de gnuplot tiennent, entre autres, à ses capacités de calcul qui sont supérieures à celle de  $\text{\LaTeX}$ .

$\text{\LaTeX}$  va utiliser beaucoup de mémoire pour faire des calculs peu précis. Quelques graphiques de ce type dans un document et il n'est plus possible de compiler!

Sous windows, il faut renommer `wgnuplot.exe` en `gnuplot.exe`.

Ensuite il faut faire en sorte que ce `gnuplot.exe` soit dans le PATH.

Tout ceci n'est pas très difficile mais celà en décourage plus d'un!

### Graphe de sin(x) avec pgfplots

Cette solution est à éviter parce qu'elle implique une syntaxe différente, celle de "pgf"!

De plus, il faut charger un package supplémentaire : `\usepackage{pgfplots}`

Mais, ça marche!





# Chapitre 13

## Macros et compteurs

### Sommaire

---

<b>13.1 Macros</b> . . . . .	<b>146</b>
13.1.1 Principe des macros . . . . .	146
13.1.2 Macros à paramètres . . . . .	147
13.1.3 Compteurs et dimensions . . . . .	149
<b>Liste des tableaux</b> . . . . .	<b>155</b>
<b>Liste des illustrations</b> . . . . .	<b>155</b>

---

## 13.1 Macros

### 13.1.1 Principe des macros

#### Macro = macro-commande

Nous avons étudié un certain nombre de macros jusqu'à maintenant et nous pouvons nous faire une idée générale de ce qu'est une macro : une commande permettant des actions complexes. En réalité, la signification du terme "macro" est un peu plus précise : une macro est un mini-programme construit à partir de commandes directement compréhensibles par le compilateur (en l'occurrence  $\text{\TeX}$ ) et/ou d'autres macros définies préalablement. Ainsi, un utilisateur pourra vouloir se construire ses propres macros pour automatiser certaines tâches répétitives.

#### Un exemple pour comprendre

Reprenons un des exemples précédents où nous avons écrit la formule :

$$\vec{x} \stackrel{\text{déf}}{=} (x_1, x_2, \dots, x_n)$$

et supposons que le symbole " $\stackrel{\text{déf}}{=}$ " soit souvent utilisé dans le document. Il y aura tout intérêt à définir une macro qui se charge de recopier la définition de ce symbole au lieu de retaper à chaque fois tous les ordres nécessaires à son obtention.

#### Nommer la macro

La première étape va être de choisir un nom (les  $\backslash\text{tata}$ ,  $\backslash\text{titi}$  et autre  $\backslash\text{toto}$  sont à proscrire sans hésitation).

Appelons notre macro  $\backslash\text{defeq}$  pour des raisons de cohérence avec les autres macros de  $\text{\TeX}$ .

#### Définir la macro

Ensuite le texte que la macro doit remplacer doit être bien compris et on doit être sûr de son résultat.

Ici, pas de problème, la macro doit remplacer la suite :

```
\stackrel{\mathrm{d\acute{e}f}}{=}
```

Il ne reste alors plus qu'à appeler la macro permettant de construire des macros, en l'occurrence  $\backslash\text{newcommand}$ , suivie du nom de la macro que l'on veut définir puis d'un groupe donnant son contenu.

Avec notre exemple, cela va donner :

```
\newcommand{\defeq}{\stackrel{\mathrm{d\acute{e}f}}{=}}
```

Et voilà, c'est tout !

Il ne reste plus qu'à l'utiliser :

$$\vec{x} \stackrel{\text{déf}}{=} (x_1, x_2, \dots, x_n)$$

```
1 \[ \vec{x} \defeq (x_1, x_2, \ldots, x_n) \]
```

### Attention aux noms de macro

Il y a d'autres façons de définir des macros mais celle exposée ci-dessus est fortement conseillée par les défenseurs de  $\LaTeX$ . Elle offre, entre autres avantages, la sécurité de ne pas redéfinir des macros déjà existantes.

En reprenant l'exemple précédent, on pourrait légitimement se dire que plus un nom de macro est court, plus rapide sera sa frappe et décider d'appeler `\def` notre jolie macro. Si cela avait été possible, il y aurait eu une série de catastrophes et plus rien n'aurait fonctionné correctement car `\def` est une primitive du langage  $\TeX$  abondamment utilisée (par exemple, la macro `\newcommand` utilise cette primitive) et tout ce qui ferait appel directement ou indirectement à `\def` serait corrompu.

D'autre part, les messages d'erreurs affichés par le terminal auraient été réellement abscons et sans rapport évident avec la redéfinition de la primitive donc une erreur très délicate à comprendre.

### 13.1.2 Macros à paramètres

#### Paramètres

Les macros comme celle définie à la section précédente seraient assez utiles mais la puissance de ce concept vient surtout de la possibilité de mettre des paramètres (de 1 à 9).

Illustrons cela par un exemple un peu plus évolué. Supposons qu'à plusieurs endroits d'un ouvrage on veuille écrire des citations avec le texte composé au fer à droite et en italique suivi, à la ligne, du nom de l'auteur composé en petite capitale et enfin du titre de l'ouvrage séparé du nom de l'auteur par une virgule, le tout placé dans un bloc de 6 cm de large.

#### Exemple de macro à paramètres

Voici un petit exemple pour voir sa réalisation sans définir de nouvelle macro :

*C'est parce que quelque chose des  
objets extérieurs pénètre en nous  
que nous voyons les formes et que  
nous pensons.*

Example 2.

ÉPICURE, lettre à Hérodote.

Petite cerise sur le gâteau : la largeur du texte écrit est exactement de 6 cm car on a pris la peine d'inhiber les espaces intercolonnes en mettant les deux `@{}` au début et à la fin du motif du tableau.

Le tout est un peu embêtant à taper et nous voudrions avoir une macro qui se charge du travail, étant entendu que le texte de la citation, le nom de l'auteur et le titre de l'ouvrage sont variables. Pour cela, il va falloir définir une macro à trois paramètres.

#### Déclaration de la macro à paramètres

Appelons-la `\epigraphe`, le début de sa déclaration se fait sous la forme :

```
\newcommand{\epigraphe}[3]
```

c'est-à-dire comme précédemment mais en rajoutant le nombre de paramètres entre crochets.

### Paramètre = dièse

Ensuite, on tape le texte qui remplacera la macro en indiquant la présence du premier paramètre par la séquence #1, celle du deuxième paramètre par #2 et celle du troisième paramètre par #3.

Comme nous avons déjà le modèle, cela ne va pas poser de problème insurmontable, il suffit de remplacer les passages correspondants par les séquences #*n* adéquates :

```
\newcommand{\epigraphe}[3]{
  \begin{flushright}
    \begin{tabular}{@{}p{6cm}@{}}
      {\raggedleft \itshape #1\par}\\
      {\raggedleft \textsc{#2}, #3\par}
    \end{tabular}
  \end{flushright}}
```

L'emploi est on ne peut plus simple, il suffit d'écrire la macro puis les trois éléments (dans l'ordre) et le tour est joué. Par exemple :

*Voici une macro qui parle  
d'elle-même (en logique, on appelle  
cela un quine). Une macro  
compliquée qui fonctionne du  
premier coup est un bonheur rare.*

*Example 3.*

CHARPENTIER, Stage  $\LaTeX$

### Cuisine interne

L'emploi de macros personnelles ne sert pas qu'à remplacer des séquences longues et répétitives de commandes, il permet également de réaliser des documents faciles à maintenir car mieux structurés. Supposons qu'on réalise un document produisant des recettes de cuisines. Chaque recette va commencer avec le nom du plat qu'on écrira dans un corps plus grand que le reste de la fiche.

Il n'y a pas de problème particulier, un source tel que :

```
{\Large Le canard laqu\'e}
```

conviendra parfaitement.

Après des semaines de travail, l'ouvrage comporte maintenant plus de 400 fiches lorsque, tout compte fait, les noms des recettes seraient plus jolis s'ils étaient aussi en gras et centrés en haut de la fiche, c'est-à-dire que chaque fiche devrait commencer sur une nouvelle page. Bien évidemment, les éditeurs de texte sont capables de faire des remplacements automatiques mais il s'agit presque toujours d'opérations dangereuses et qui doivent être effectuée sous contrôle humain ; après tout, d'autres textes ont pu être composés dans ce même corps sans être le nom de la recette.

Une solution beaucoup plus propre et compréhensible serait de définir dès le début du travail une macro `\titrerecette` dont la déclaration serait :

```
\newcommand{\titrerecette}[1]{\Large #1}}
```

La présence des doubles accolades est nécessaire car sinon, l'appel `\titrerecette{Le canard laqu\'e}` est remplacé par le texte :

```
\Large Le canard laqu\'e
```

et l'action de la macro `\Large` se poursuivra au-delà de la zone souhaitée alors qu'avec les doubles accolades, la macro est remplacée par le texte :

```
{\Large Le canard laqu\'e}
```

et tout va bien.

Au moment où on veut modifier la présentation des titres des recettes, il suffira de modifier la définition de la macro au niveau du préambule et tout sera recomposé selon les nouvelles exigences. Ici, la macro deviendrait :

```
\newcommand{\titrerecette}[1]{%
  \newpage
  \begin{center}
    \Large #1
  \end{center}
}
```

Deux petits mots sur cette macro avant de clore le sujet. Nous avons vu précédemment qu'il fallait mettre la macro `\Large` entre accolades pour que son action reste confinée au paramètre de la macro. Ici, ce n'est pas nécessaire car tout environnement se comporte comme des accolades sur ce point. En l'occurrence, l'environnement `center` va limiter la portée de la macro `\Large`. D'autre part, il y a un caractère `%` à la fin de la première ligne. Ce commentaire (vide) sert à ce que `TEX` ne voit pas la fin de la ligne et ne produisent donc pas d'espace parasite. En fait, cette précaution est ici superflue car une espace précédant un saut de page ne fait de mal à personne mais il existe beaucoup de situations où il faut penser à ce " truc ".

### 13.1.3 Compteurs et dimensions

#### Registres

Dans ce manuel, nous avons déjà rencontré les deux dimensions `\parindent` et `\parskip` qui permettaient respectivement d'indiquer l'importance de l'indentation de première ligne et la distance entre deux paragraphes. Nous avons également utilisé, sans le nommer, le registre qui garde le numéro de page en cours. En fait, `LATEX` gère une multitude de choses en se servant de registres de dimension et de compteurs ; de plus, l'utilisateur peut se créer ses propres registres pour des besoins particuliers.

#### Compteurs

Étudions d'abord les compteurs, la syntaxe pour les dimensions ne sera pas trop différente. Si on désire utiliser un compteur qui n'est pas défini par `LATEX`, il va falloir le déclarer. Pour cela, on dispose de la macro `\newcounter` qui sera suivie du nom du compteur (compteur qui, sous `LATEX`, ne s'écrit pas avec la contre-oblique). Nous allons déclarer un compteur qui servira à une macro destinée à numéroter automatiquement les exercices d'un devoir ; appelons-le `numexo`. Sa déclaration sera donc :

```
\newcounter{numexo}
```

Cette déclaration faite, le compteur `numexo` existe et est initialisé à zéro. Il reste à pouvoir modifier sa valeur, à le transmettre à des macros et à placer sa valeur dans un texte. Voyons ces différents points un à un. Avec `LATEX`, il existe deux façons de changer le contenu d'un compteur. On peut placer une valeur particulière grâce à la syntaxe :

```
\setcounter{cpt}{val}
```

qui permet de mettre la valeur *val* dans le compteur *cpt*.

On peut aussi ajouter une valeur en utilisant la construction :

```
\addtocounter{cpt}{val}
```

qui ajoutera la valeur *val* au contenu actuel du compteur *cpt*. Pour une soustraction, il suffit d'ajouter une valeur négative. Pour les multiplications et divisions, vous êtes cordialement invité au stage de perfectionnement ou à lire des documents de référence !

Si le contenu d'un compteur doit être transmis à une macro en tant que paramètre, on pourra utiliser la construction :

```
\value{cpt}
```

Un point plus utile que le précédent consiste à afficher le contenu d'un compteur au niveau du document produit. C'est ce que nous voulons avec le compteur d'exercice. Pour cela,  $\TeX$  n'offre pas moins de 7 macros.

Nous ne verrons ici que les cinq les plus fréquemment employées :

Commande	Type	Exemple
<code>\arabic{cpt}</code>	nombre arabe	1, 2, 3, ...
<code>\roman{cpt}</code>	nombre romain minuscule	i, ii, iii, ...
<code>\Roman{cpt}</code>	nombre romain majuscule	I, II, III, ...
<code>\alph{cpt}</code>	lettre minuscule	a, b, c, ...
<code>\Alph{cpt}</code>	lettre majuscule	A, B, C, ...

Nous sommes maintenant en possession de tout ce qu'il faut pour construire notre macro gérant automatiquement les numéros d'exercices. Cette macro devra définir un nouveau paragraphe, inhiber l'indentation de première ligne et afficher le texte "**Exercice** *n*." où *n* sera le numéro de l'exercice en cours suivie d'une espace horizontale assez généreuse.

Le code réalisant toutes ces actions peut être défini comme suit :

```
\newcommand{\exo}{\par
  \addtocounter{numexo}{1}%
  \noindent
  \textbf{Exercice \arabic{numexo}}\quad}
```

À la suite de quoi on pourra obtenir quelque chose comme :

*Example 4* (firstnumber=1,firstline=2). **Exercice 1** Repérer toutes les fautes de ce manuel (exercice difficile et long).

**Exercice 2** Assez tôt dans ce manuel, il avait été dit que le nombre de macros utilisées jusqu'à ce point était de 78.

Maintenant que ce manuel arrive plutôt vers sa fin, quel est le nombre de macros qui ont été étudiées ?

**Exercice 3** Très souvent, l'augmentation d'un compteur avec la macro `\addtocounter` est de 1 (comme pour l'exemple en cours des exercices).

Pensez-vous vraiment que  $\TeX$  soit si mal fait qu'il n'ait pas pensé à définir un raccourci pour une incrémentation d'une unité ?

Facile n'est-ce pas ? Et surtout, cela évite les erreurs de numérotations qui arrivent fréquemment surtout si on remanie l'ordre et le nombre d'exercices en étant un peu pressé (expérience personnelle).

À propos de l'exercice 3 : la réponse est évidemment “ non ” ! Il arrive effectivement très fréquemment que l'on veuille ajouter 1 à un compteur et la séquence :

```
\addtocounter{cpt}{1}
```

peut être raccourcie en :

```
\stepcounter{cpt}
```

Compliquons un peu les choses en exigeant une macro qui numérote automatiquement les questions à l'intérieur d'un exercice mais qui ait le bon goût de redémarrer à 1 à chaque nouvel exercice.

Pour que cela soit joli, le numéro de la question sera écrit en gras et suivi d'un triangle, d'un point et d'une espace normale. Nous appellerons `\question` cette macro et `numq` le compteur associé au numéro de question.

La définition de la macro `\question` ne pose pas de problème : elle ressemble beaucoup à celle de la macro `\exo` :

```
\newcommand{\question}{\par
  \stepcounter{numq}%
  \noindent
  \textbf{\arabic{numq}} \(\triangleright\).\ }
```

Avec les deux macros `\exo` et `\question` telles qu'elles ont été définies, les choses seront correctes pour le premier exercice mais deviendront incorrectes dès le deuxième exercice car le compteur `numq` continuera sur sa lancée.

Il existe deux solutions à ce problème : soit on modifie la définition de la macro `\exo` en ajoutant la commande :

```
\setcounter{numq}{0}%
```

au début de la macro, soit on utilise une autre façon de définir le compteur `numq` :

```
\newcounter{numq}[numexo]
```

Cette syntaxe indique que `numq` est un compteur qui sera automatiquement remis à zéro à chaque appel de `\stepcounter{numexo}` (il faudra donc obligatoirement modifier la macro `\exo` pour remplacer la commande `\addtocounter{numexo}{1}` par la commande `\stepcounter{numexo}`).

Certaines macros gérant les dimensions ressemblent à celles dédiées aux compteurs. Ainsi, on trouve :

```
\newlength{\long}
```

qui permet de définir la dimension `\long` (on notera la présence de la contre-oblique pour les dimensions alors qu'il n'y en avait pas pour les compteurs). Lorsqu'une nouvelle dimension est créée, elle est initialisée à 1in (1 pouce).

On a également :

```
\setlength{\long}{val}
```

qui permet de spécifier la valeur `val` à placer dans le registre de dimension `\long`. Une longueur doit obligatoirement comporter une des unités listées au tableau 7.3 page 73 et peut comporter une composante plus et/ou une composante minus permettant de donner respectivement un étirement et une compression à cette longueur (T<sub>E</sub>X parle plutôt de ressort), Cf. section 7.1.3 page 74.

Enfin, la dernière macro ressemblant à celles gérant les compteurs est :

```
\addtolength{\long}{val}
```

qui ajoute la valeur *val* au registre `\long`. Les trois composantes (longueur, extension et compression) sont ajoutées séparément.

Trois autres macros permettent de spécifier une longueur calculée à partir d'un matériel donné :

```
\settowidth{\long}{objet}
```

```
\settoheight{\long}{objet}
```

```
\settodepth{\long}{objet}
```

initialisent le registre *long* respectivement avec la largeur, la hauteur et la profondeur de *objet*.

### Exemple

Un petit exemple pour bien comprendre qui utilise le fait que la fonte `\texttt{}{}` a tous ses caractères de la même largeur :

Voici un mot ~~supprimé~~

```
1 \newlength{\retour}
2 \settowidth{\retour}{\texttt{supprim\
   e}}
3 \texttt{Voici un mot
4   supprim\ 'e\hspace{-\retour}////////}
```



## **Quatrième partie**

### **Annexes**



# Liste des tableaux

4.1	Obtention des caractères réservés . . . . .	47
4.2	Accent en mode texte . . . . .	48
4.3	Caractères européens spéciaux . . . . .	49
4.4	Symboles spéciaux . . . . .	49
5.1	Délimiteurs des modes mathématiques . . . . .	53
5.2	Opérateurs binaires . . . . .	57
5.3	Relations . . . . .	58
5.4	Grands opérateurs . . . . .	58
5.5	Délimiteurs . . . . .	59
5.6	Flèches . . . . .	59
5.7	Symboles mathématiques divers . . . . .	59
5.8	Noms de fonction . . . . .	60
5.9	Accents mathématiques . . . . .	60
5.10	Synonymes de macros mathématiques . . . . .	60
7.1	Macros de taille de caractère . . . . .	68
7.2	Macros de type de caractère . . . . .	69
7.3	Unités de mesures . . . . .	73
9.1	Caractères de la fonte ZapfDingbat . . . . .	90
9.2	Caractères de la fonte Symbol . . . . .	91
10.1	Fontes mathématiques . . . . .	96
10.2	Macros d'espacements . . . . .	100
10.3	Exemples classiques de correction d'espacement . . . . .	100
10.4	Commandes des motifs de tableaux . . . . .	106
10.5	Syntaxe de la macro <code>\hline</code> . . . . .	108



# Table des figures

1.1	Schéma de production de documents en $\LaTeX$ . . . . .	6
1.2	Richard Stallman . . . . .	6
1.3	Don Knuth . . . . .	7
1.4	Leslie Lamport . . . . .	8
11.1	Zen avec $\LaTeX$ . . . . .	120
11.2	Petit zen avec $\LaTeX$ . . . . .	121
11.3	Petit zen au milieu avec $\LaTeX$ . . . . .	121
11.4	Zen à droite avec $\LaTeX$ (et les options possibles) . . . . .	121
11.5	Pas si zen avec $\LaTeX$ . . . . .	121
11.6	Graphes de fonctions grâce à Gnuplot et Tikz . . . . .	125
12.1	Un point avec PgfTikz . . . . .	131
12.2	Les coordonnées polaires avec PgfTikz . . . . .	132
12.3	Une grille avec PgfTikz . . . . .	132
12.4	Une grille bleue avec PgfTikz . . . . .	133
12.5	Un cercle avec PgfTikz . . . . .	133
12.6	Un cercle en coordonnées polaires avec PgfTikz . . . . .	133
12.7	Un cercle avec PgfTikz . . . . .	134
12.8	Cercles concentriques avec PgfTikz . . . . .	134
12.9	Cercles excentriques avec PgfTikz . . . . .	134
12.10	Ellipses avec PgfTikz . . . . .	134
12.11	Rectangle avec PgfTikz . . . . .	134
12.12	Rectangles avec PgfTikz . . . . .	135
12.13	Un segment avec PgfTikz . . . . .	135
12.14	Carré sur pointe avec PgfTikz . . . . .	135
12.15	Carré avec grille rouge avec PgfTikz . . . . .	136
12.16	Carré avec grille verte avec PgfTikz . . . . .	136
12.17	Cycle avec PgfTikz . . . . .	136
12.18	Une courbe de Bézier avec PgfTikz . . . . .	136
12.19	Arcs avec PgfTikz . . . . .	137
12.20	Points avec déplacements relatifs avec PgfTikz . . . . .	137

12.21	Points avec déplacements absolus avec PgfTikz . . . . .	137
12.22	Flèches avec PgfTikz . . . . .	138
12.23	Path avec PgfTikz . . . . .	138
12.24	coordinate avec PgfTikz . . . . .	139
12.25	Path et coordinate avec PgfTikz . . . . .	139
12.26	Pentagone avec PgfTikz . . . . .	139
12.27	node avec PgfTikz . . . . .	140
12.28	Boucle avec PgfTikz . . . . .	141
12.29	Boucles avec PgfTikz . . . . .	141
12.30	Sin(x) avec Tikz et le moteur de calcul de $\text{\LaTeX}$ . . . . .	142
12.31	Sin(x) avec Tikz . . . . .	142
12.32	Sin(x) avec Pgfplots . . . . .	143

# **Bibliographie**

## .1 Bibliographie commentée

La liste des références présentée ici n'a pas pour vocation d'être complète. Il s'agit d'ouvrages que je possède, ou que j'ai eu un certain temps en ma possession, et de sites internet sur lesquels j'ai l'habitude d'aller, ce qui me permet de donner mon avis et non un avis pris dans un livre quelconque. Bien entendu, tous les commentaires sont hautement subjectifs ! Les prix indiqués sont ceux pratiqués lorsque j'ai acheté l'ouvrage et sont donc susceptibles d'une certaine variation.

### .1.1 Ouvrages sur la typographie

[LEX] *Lexique des règles typographiques en usage à l'Imprimerie Nationale*, Imprimerie nationale, 1990. (89 F)

Ouvrage de référence sur les questions de typographie. Il est à noter qu'il existe d'autre corpus de règles que celles-ci mais, dans l'ensemble, ces règles sont plus ou moins admises par tout le monde (en France).

[MTF] *Manuel de typographie française*, Yves PERROUSSEAU, Atelier Perrousseau, 1996. (120 F)  
Cet ouvrage présente les règles de typographie avec des références historiques et indique comment les mettre en œuvre sur un système informatique.

[MPI] *Mise en page & imprimerie*, Yves PERROUSSEAU, Atelier Perrousseau, 1999. (200 F)  
Il s'agit du complément logique de l'ouvrage précédent.

[PLT] *Petites leçons de typographie*, Jacques ANDRÉ, document libre, 1990. (0 F)  
Petit document (25 pages de corps) mais bien présenté et clair.

### .1.2 Ouvrages pour débiter

[LGP] *LaTeX, guide pratique*, Christian ROLLAND, Campus Press, 199?. (???) F  
C'est très souvent le premier ouvrage qu'ont eu les LaTeXiens confirmés. La présentation se veut résolument pratique et la nouvelle édition a été fortement augmentée d'où une certaine exhaustivité au niveau des packages importants.

[JML] *Joli manuel pour LaTeX<sub>2 $\epsilon$</sub>* , Benjamin BAYART, document libre, 1995. (0 F)  
Il s'agit d'un document destiné initialement aux élèves de l'ESIEE (ingénieurs en électronique) mais qui a été mis à disposition sur internet. Dès que les rudiments de LaTeX sont connus, c'est-à-dire, par exemple, après un stage comme celui-ci, ce guide est très utile et très complet. Je le recommande chaudement.

[MPM] *A User Manual for METAPOST*, John D HOBBY, document libre, 199?. (0 F)  
Le manuel de référence de METAPOST écrit par son auteur. METAPOST est un logiciel permettant de construire des figures au format PostScript en utilisant une bonne partie de la syntaxe du langage METAFONT. Ce livre est nettement plus accessible que le METAFONTbook mais moins complet. Il existe une excellente traduction faite par Jean-Côme CHARPENTIER et Pierre FOURNIER (un peu de publicité) disponible sur le site Syracuse (Cf. infra).

[LAG] *LaTeX. Apprentissage, guide et référence*, Bernard DESGRAUPES, Vuibert, 2000. (~ 250 F)  
Comme son nom l'indique, il s'agit vraiment d'un livre permettant d'apprendre, d'être guidé et d'avoir une référence (très complète) sur LaTeX. À mon sens, le meilleur ouvrage car il a réussi le pari d'être aussi utile au débutant qu'au LaTeXien confirmé.



### .1.3 Ouvrages plus difficiles

[TEX] *The T<sub>E</sub>Xbook*, Donald KNUTH, Addison-Wesley, 1996. (41,95 \$ US)

LA référence en ce qui concerne le langage T<sub>E</sub>X. Cet ouvrage se veut lisible par un débutant mais, soyons sérieux, la majeure partie de son contenu est d'une très haute technicité. À posséder absolument si on attrape le virus T<sub>E</sub>Xien.

[MTL] *La maîtrise de T<sub>E</sub>X et L<sup>A</sup>T<sub>E</sub>X*, Thomas LACHAND-ROBERT, Masson, 1995. (380 F)

Contrairement à ce que son titre pourrait laisser croire, il n'est quasiment pas question de L<sup>A</sup>T<sub>E</sub>X. Il s'agit de l'ouvrage français le plus complet sur le langage T<sub>E</sub>X, un peu l'équivalent du T<sub>E</sub>Xbook. Indispensable si on ne lit pas l'anglais et qu'on veuille découvrir les entrailles de T<sub>E</sub>X.

[MFB] *The METAFONTbook*, Donald KNUTH, Addison-Wesley, 19?? (??? \$ US)

LA référence en ce qui concerne le langage METAFONT. Les mêmes remarques que celles faites pour le T<sub>E</sub>Xbook s'appliquent à ce livre. Il semble malheureusement épuisé.

[DPS] *A document Preparation System*, Leslie LAMPORT, Addison-Wesley, 1999. (36,95 \$ US)

Un ouvrage de référence sur L<sup>A</sup>T<sub>E</sub>X écrit par son auteur. L'ouvrage ne traite que de L<sup>A</sup>T<sub>E</sub>X et ne présente pas les extensions. Personnellement, je lui préfère la référence suivante mais mon opinion n'est pas forcément toujours partagée !

[COM] *The L<sup>A</sup>T<sub>E</sub>X Companion*, Michel GOOSSENS, Frank MITTELBACH, Alexander SAMARIN, Addison-Wesley, 1994. (36,95 \$ US)

C'est mon livre de chevet ! Il est complet au-delà de toute espérance : un travail titanesque. Une traduction française existe (publiée chez Campus Press) et qui porte le même nom (249 F). La traduction reprend certains points rendus obsolètes (elle date de 2000) et est moins chère que l'ouvrage originale mais la beauté (physique) du livre est moins bonne : les ouvrages de chez Addison-Wesley sont souvent de qualité.

[LGC] *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion*, Michel GOOSSENS, Sebastien RAHTZ, Frank MITTELBACH, Addison-Wesley, 1997. (36,95 \$ US)

C'est l'équivalent du précédent pour tout ce qui touche au graphisme dans le monde L<sup>A</sup>T<sub>E</sub>X. Il est aussi beau et aussi complet que *The L<sup>A</sup>T<sub>E</sub>X Companion*.

[FAQ] *FAQ L<sup>A</sup>T<sub>E</sub>X*, Marie-Paule KLUTH, ???, ???. (??? F)

Pourquoi tous ces points d'interrogations ? Parce que je n'ai pas cet ouvrage que pourtant tout le monde devrait posséder. Pourquoi je n'ai pas cet ouvrage alors que je n'en pense que du bien ? Réponse à la section suivante. À propos, FAQ signifie " Frequently Asked Questions " très joliment traduit en " Foire Aux Questions ".

### .1.4 Références sur internet

<http://www.ctan.org/ctan>

CTAN est l'acronyme de " Comprehensive TeX Archive Network ". En clair, tous les fichiers informatiques qui existent (officiellement) sur TeX se trouvent sur ce site. En conséquence, lorsqu'on cherche la perle rare, c'est un des premiers endroits où aller. Le site indiqué est le site principal basé au États-Unis, il existe des sites miroirs un peu partout dans le monde : la liste de ces sites miroirs se trouve sur tous les sites CTAN.

<http://tex.loria.fr/index.html>

Il s'agit du Loria : un site français de référence pour tout ce qui concerne la documentation de T<sub>E</sub>X et des programmes satellites. J'ai réellement pillé ce site !

<http://gutenberg.eu.org/pub/GUTenberg>

GUTenberg est le pendant français du TUG (T<sub>E</sub>X User's Group). Il s'agit d'une association loi 1901 qui fédère les passionnés de T<sub>E</sub>X et de L<sup>A</sup>T<sub>E</sub>X. Cette association permet à ses adhérents de recevoir un

bulletin d'information, de s'abonner pour un prix modique aux cahiers de GUTenberg et de participer à des réunions aux thèmes divers et variés touchant au monde T<sub>E</sub>X.

<http://melusine.eu.org/syracuse>

Site Poitevin regroupant les utilisateurs des logiciels libres et des utilisateurs des programmes du monde T<sub>E</sub>X. Site très sympathique géré par des personnes non moins sympathiques quoique peu nombreuses ce qui ajoute à leur mérite. C'est par exemple sur ce site qu'a été placée la traduction du manuel de METAPOST réalisée avec Pierre FOURNIER.

[fr.comp.text.tex](mailto:fr.comp.text.tex)

Attention, il ne s'agit pas d'un site mais d'un forum de discussion (newsgroup comme disent ceux qui ont du mal avec la langue française). On y trouve des personnes de tout niveau, on peut se contenter de poser des questions, de lire les questions et réponses des autres ou bien de répondre soi-même à certaines questions. Il s'agit d'un forum très vivant : soyez un minimum poli, posez votre question clairement et vous obtiendrez presque à coup sûr une réponse. Enfin, comme sur tout forum de discussion ou presque, une FAQ est postée tous les mois. Cette FAQ a été établie à partir de celle de Marie-Paule KLUTH qui s'en occupait il y a quelques années. Elle s'enrichit de mois en mois ce qui explique ma réticence à acheter celle du commerce.

# Table des matières

<b>I</b>	<b>Premier atelier : prise en main</b>	<b>1</b>
<b>1</b>	<b>Présentation de <math>\LaTeX</math></b>	<b>3</b>
1.1	$\TeX$ , $\LaTeX$ et logiciel libre . . . . .	4
1.1.1	$\LaTeX$ pour le néophyte . . . . .	4
1.1.2	Historique . . . . .	6
1.1.3	Que signifie logiciel libre ? . . . . .	8
1.2	Avantages et inconvénients de $\LaTeX$ . . . . .	9
1.2.1	Inconvénients ou dix bonnes raisons de ne pas utiliser $\LaTeX$ . . . . .	9
1.2.2	Avantages ou dix bonnes raisons d'utiliser $\LaTeX$ . . . . .	9
<b>2</b>	<b>Distributions <math>\LaTeX</math> et installation</b>	<b>11</b>
2.1	Qu'est ce qu'une "Distribution"? . . . . .	12
2.1.1	Distribution Mik $\TeX$ . . . . .	12
2.1.2	Distribution $\TeX$ live . . . . .	12
2.2	EDI : Environnement de Développement Intégré . . . . .	13
2.2.1	Kile . . . . .	13
2.2.2	$\TeX$ Maker . . . . .	13
2.2.3	$\TeX$ nicenter . . . . .	13
2.2.4	$\TeX$ Shell . . . . .	13
2.2.5	$\TeX$ Shop . . . . .	13
<b>3</b>	<b>Premiers pas</b>	<b>15</b>
3.1	Premiers documents . . . . .	16
3.1.1	Structure d'un fichier $\LaTeX$ . . . . .	16
3.1.1.1	Entête, préambule ou header . . . . .	18
3.1.1.2	Le document . . . . .	19
3.1.1.3	En résumé . . . . .	21
3.1.2	Etude du résultat . . . . .	22
3.1.2.1	Césures . . . . .	22
3.1.2.2	Paragraphes et indentation . . . . .	22
3.1.2.3	Macro commandes . . . . .	22
3.1.2.4	Listes . . . . .	23

3.1.2.5	Typographie . . . . .	23
3.1.2.6	Marges et format de papier . . . . .	23
3.1.2.7	Ligature . . . . .	24
3.1.2.8	En résumé . . . . .	24
3.1.2.9	Caractères réservés . . . . .	24
3.1.3	Deuxième exemple . . . . .	26
3.1.4	Troisième exemple . . . . .	27
3.2	EDI : prise en main de TexnicCenter et premières compilations . . . . .	28
3.2.1	Les différentes sous fenêtres de T <sub>E</sub> XnicCenter . . . . .	28
3.2.2	Compilation de l'exemple . . . . .	32
3.2.2.1	Choix du format de sortie . . . . .	32
3.2.2.2	Compiler . . . . .	33
3.2.2.3	Voir le résultat . . . . .	34
3.2.3	Autres EDI . . . . .	35
3.3	Erreurs de compilation . . . . .	37
<b>4</b>	<b>Commandes de Base : environnements</b>	<b>39</b>
4.1	Commandes de base . . . . .	40
4.1.1	Listes . . . . .	41
4.2	La gestion des erreurs de compilation . . . . .	43
4.3	Commandes de base : la suite . . . . .	47
4.3.1	Caractères particuliers . . . . .	47
4.3.1.1	Les accents . . . . .	47
<b>5</b>	<b>Modes Mathématiques</b>	<b>51</b>
5.1	Mathématiques élémentaires . . . . .	52
5.1.1	Différents modes mathématiques . . . . .	52
5.1.2	Opérations élémentaires . . . . .	53
5.1.3	Structures indispensables . . . . .	54
5.2	Symboles en vrac . . . . .	56
<b>6</b>	<b>Commandes de Base : structurer un document</b>	<b>61</b>
6.1	Quelques présentations plus évoluées . . . . .	62
6.1.1	Structuration des documents . . . . .	62
6.1.1.1	Commandes pour le plan . . . . .	62
6.1.1.2	Références . . . . .	63
6.1.2	Table des matières, index . . . . .	63
6.1.3	En-têtes, pieds de page, notes . . . . .	64

<b>7</b>	<b>Commandes de Base : Fontes, espaces</b>	<b>67</b>
7.1	Mises en pages via les fontes, les espaces, ...	68
7.1.1	Commandes de fonte	68
7.1.1.1	Taille des caractères	68
7.1.1.2	Types de fonte	68
7.1.2	Commandes de paragraphe	70
7.1.3	Espacements	74
<b>II</b>	<b>Deuxième atelier : tableaux et mises en page avancés</b>	<b>77</b>
<b>8</b>	<b>Tableaux</b>	<b>79</b>
8.1	Tableaux et mise en page mathématique	80
8.1.1	Tableaux	80
8.1.1.1	Les réglures	82
8.1.1.2	multicolumn	82
<b>9</b>	<b>Extensions</b>	<b>87</b>
9.1	Extensions utiles	88
9.1.1	geometry	88
9.1.2	Extension pour les polices de caractères	88
9.1.2.1	Talon Réponse	90
9.2	Extensions spécialisées	92
9.2.1	gnuplottex	92
9.2.2	answers	92
9.2.3	pst-pdf	92
9.2.4	showexpl	92
9.2.5	wallpaper	93
9.2.6	hyperref	93
9.2.7	ifthen	93
<b>10</b>	<b>Mathématiques avancées</b>	<b>95</b>
10.1	Mathématiques complexes	96
10.1.1	Fontes mathématiques	96
10.1.2	Grands opérateurs	97
10.1.3	Délimiteurs	99
10.1.4	Espacements	99
10.1.5	Empiler verticalement	100
10.1.6	Matrices et environnement array	102
10.2	amsmath	104
10.2.1	Tableaux évolués	105
10.2.2	multicol	109

<b>III Troisième atelier : programmation, macros, compteurs, graphisme, programmes externes</b>	<b>111</b>
<b>11 Graphisme</b>	<b>113</b>
11.1 Petit aperçu des possibilités . . . . .	114
11.1.1 Quels graphiques? . . . . .	114
11.2 Deux choix possibles . . . . .	114
11.2.1 Inclusion d'images externes . . . . .	114
11.2.1.1 Nature de l'affichage . . . . .	114
11.2.1.2 Packages et types d'images . . . . .	115
11.2.1.3 Types de fichiers images . . . . .	116
11.2.1.4 Exemples de formats les plus utilisés . . . . .	118
11.2.1.5 Quel format choisir? . . . . .	119
11.2.2 L'environnement "figure" . . . . .	120
11.2.3 Syntaxe de "includegraphics" . . . . .	120
11.2.3.1 Ce manuel . . . . .	122
11.2.4 Des commandes dans le code source . . . . .	123
11.2.4.1 Dessins scientifiques . . . . .	127
11.2.4.2 Jeux . . . . .	127
11.2.5 Graphismes avec appel à des programmes extérieurs . . . . .	127
<b>12 Graphisme avec Tikz</b>	<b>129</b>
12.1 Présentation de Pgf/Tikz . . . . .	130
12.1.1 Créateur et acronymes . . . . .	130
12.1.2 Justification du choix de Tikz . . . . .	130
12.1.3 Motivation des choix pédagogiques . . . . .	130
12.2 Syntaxe de Tikz . . . . .	130
12.2.1 Appeler Tikz . . . . .	130
12.2.2 Les bases de pgf/tikz . . . . .	131
12.2.2.1 Points et coordonnées . . . . .	131
12.2.2.2 Les formes . . . . .	132
12.2.2.3 Les lignes brisées . . . . .	135
12.2.2.4 Les lignes courbes . . . . .	135
12.2.2.5 Les déplacements . . . . .	137
12.2.2.6 Flèches . . . . .	137
12.3 . . . . .	138
12.3.1 Path, coordinate et node . . . . .	138
12.3.1.1 Path . . . . .	138
12.3.1.2 Coordinate . . . . .	138
12.3.1.3 Nodes . . . . .	140
12.3.2 Opérations répétitives et programmation . . . . .	140
12.3.3 Graphes de fonctions . . . . .	140

<i>TABLE DES MATIÈRES</i>	167
<b>13 Macros et compteurs</b>	<b>145</b>
13.1 Macros . . . . .	146
13.1.1 Principe des macros . . . . .	146
13.1.2 Macros à paramètres . . . . .	147
13.1.3 Compteurs et dimensions . . . . .	149
<b>IV Annexes</b>	<b>153</b>
Liste des tableaux . . . . .	155
Liste des illustrations . . . . .	155
<b>Bibliographie</b>	<b>159</b>
.1 Bibliographie commentée . . . . .	160
.1.1 Ouvrages sur la typographie . . . . .	160
.1.2 Ouvrages pour débiter . . . . .	160
.1.3 Ouvrages plus difficiles . . . . .	161
.1.4 Références sur internet . . . . .	161