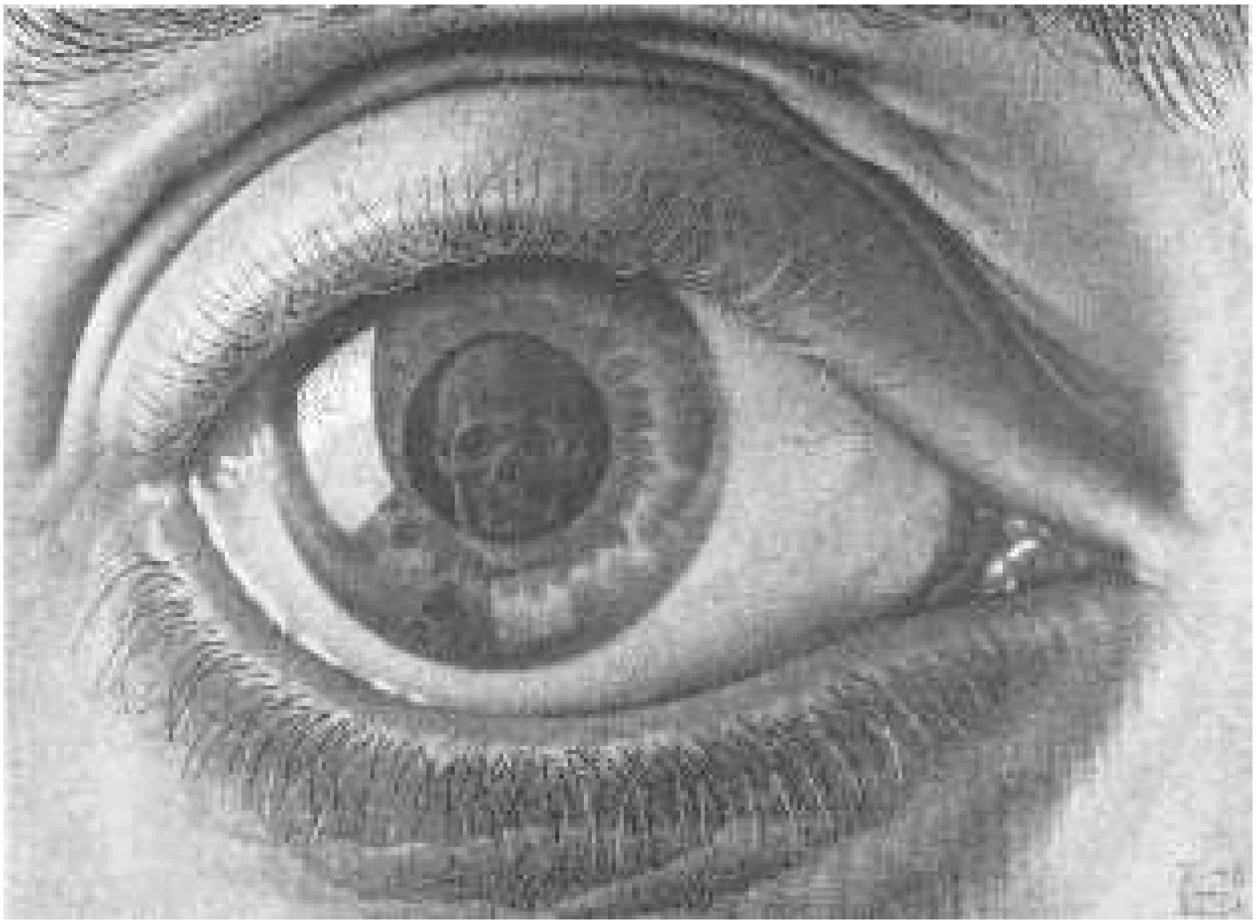
Joli manuel pour $\LaTeX\,2_\varepsilon$

Guide local de l'ESIEE*

Benjamin BAYART (Geut)

Compilé le 18 décembre 1995



*École Supérieure d'Ingénieurs en Électrotechnique et Électronique

Résumé

Durant l'été 1994, une nouvelle version de LaTeX est apparue, et a été quasi-immédiatement installée sur le réseau de l'école par mes bons soins. Je décidais, à ce moment précis, de rédiger une micro-documentation sur les différences entre cette version et l'ancienne. En mûrissant, cette documentation a grandi. Elle décrit maintenant même les bases de LaTeX 2_ε et non plus seulement ses différences avec LaTeX 2.09. Elle décrit également un grand nombre d'extensions utiles à tout et à tous.

Renvoi

Pour tout ce qui touche de près ou de loin à LaTeX 2_ε, je ne puis que renvoyer le lecteur passionné aux excellents ouvrages de Leslie Lamport *LaTeX: A document preparation system* [61] (deuxième édition, revue et adaptée à LaTeX 2_ε) et de Frank Mittelbach, Michel Goossens, et Alexander Samarin *The LaTeX Companion* [81] qui contient une description complète de nombreux packages existants ou en préparation pour le projet LaTeX3.

Ces deux ouvrages sont disponibles en Doc-Tek, et, si un jour GEUT à des finances, alors ils seront empruntables aussi auprès du responsable (moi, à l'heure actuelle, mais pas éternellement).

Pour plus de détails sur chacun des packages que je décris dans le présent document, se référer à l'ouvrage associé que je cite en bibliographie.

Table des matières

Liste des figures

Liste des tableaux

Introduction

1 Avertissement

Ce document que tu as entre les mains n'est pas un document de référence. Il n'engage que son lecteur (toi) et donc pas son auteur (moi).

Cependant, je pense qu'il ne comporte plus trop d'erreurs. Il se veut conforme aux documentations citées en bibliographie. Un moyen simple de savoir s'il est obsolète est de savoir si l'un de ces documents l'est.

Autre chose enfin, ce document est le «Local guide» de l'ESIEE, c'est-à-dire que lorsque j'ai besoin de faire référence à une installation type, c'est à celle de l'ESIEE que je me réfère. Je ne suis pas omniscient, donc je ne peut pas en décrire d'autres. Toutes mes excuses pour cette gêne aux gens extèrieurs à l'ESIEE qui me lisent.

2 Introduction

Cette remise à jour du «Joli Manuel pour LaTeX 2_ε» n'a que très peu de choses de plus que l'édition précédente. On retiendra tout de même la partie sur la réalisation d'un index, celle sur les incompatibilités entre packages, l'ajout d'un CV dans les documents types, une partie sur l'utilisation du landscape, quelques remarques sur `babel`, quelques packages nouvellement décris (`endnotes`, `fn2end`, `oldstyle`, `lastpage` et d'autres), une diffusion à l'extèrieur de l'ESIEE et une partie sur le lien entre Mathematica et LaTeX.

3 Introduction à la seconde édition

Ce fascicule que tu tiens dans tes petites mains potelées est la seconde édition du «Joli Manuel pour LaTeX 2_ε». Elle est une remise à jour majeure de la précédente (la première édition), et contient les informations que contenait son aîné («Joli Manuel pour LaTeX») qui était consacré à LaTeX 2.09. C'est donc le troisième document que je diffuse à l'ESIEE sur LaTeX.

Il a une longue histoire pleine de rebondissements. Son origine est le fait qu'un ami m'avais demandé un fichier d'exemple contenant un peu de tout. Ensuite, on me fît remarquer qu'un tel document pouvait être d'une grande utilité, ce qui le fit se répandre sur le réseau de l'école.

Vint ensuite l'épisode GEUT, ce groupe que j'avais à cœur de créer, pour essayer d'insufler une certaine solidarité aux utilisateurs de LaTeX. Lors de l'apparition de LaTeX 2_ε à l'ESIEE, c'est à dire dés sa sortie comme version officielle de LaTeX en juin 1994, j'entrepris la rédaction d'une documentation spécifique. Elle a évolué jusqu'à atteindre le stade où tu la trouves.

Cette histoire se termine, pour l'heure, sur un constat d'échec. L'administration de l'école m'a en effet demandé de ne pas poursuivre mes travaux. Toutefois, une solution semble se profiler à l'horizon. Il reste donc un très mince espoir.

4 Introduction à la première édition

Ce manuel n'est pas un manuel pour débutant sous LaTeX, il suppose que tu connais déjà au moins les rudiments de LaTeX2.09. Si tel n'était pas le cas, alors reporte toi à son grand frère comme c'est dit dans l'avertissement que si tu l'as pas lu, il faudra bien le lire. LaTeX évolue, alors la doc aussi. Il y a plein de nouveautés, de changements, de coups de génie... Il était nécessaire de régler ce problème rapidement en indiquant dans un petit fascicule ce qui fonctionne à l'ESIEE et ce qui n'y fonctionne pas.

Toutefois, certains d'entre vous ont des documents qui compilent très bien sous LaTeX2.09 et qui risquent de ne pas passer avec la nouvelle version. Dans le soucis de vous rendre service, je garantis encore le bon fonctionnement de l'ancienne version, c'est à dire qu'elle ne fonctionnera pas moins bien qu'au premier juin 1994. Pas forcément mieux. Je pense qu'aucune correction de bugs n'aura lieu. Ou alors à titre absolument exceptionnel. Il est nécessaire que vous appreniez la nouvelle version de LaTeX pour vos nouveaux documents, tout simplement parce que l'ancienne est obsolète et ne survivra pas éternellement. Les garanties que je donne ne sont valable que tant que je garde une certaine influence sur le devenir de LaTeX sur le réseau de l'école, c'est à dire de manière sûre jusqu'en juin 1995, mais probablement pas après.

Bonne lecture.

5 Groupe des Esieeens Utilisateurs de TeX

5.1 Présentation

Le GEUT n'est pas un club, il ne dépend pas exclusivement du BdE[1]. Il n'est pas une association, n'ayant ni finances[2] ni trésorier. Il n'est pas autonome non plus.

Le GEUT doit au BdE de pouvoir publier des documents comme celui-ci, et seulement ça. Le GEUT doit au SMIG l'accès au matériel pour travailler (locaux, machine ...). Le GEUT doit à l'administration son droit d'exister. Dépendant de toutes les parties de l'école, il se veut ouvert à tous : profs, élèves, et administratifs.

Le GEUT n'est pas une secte, il n'y a pas de rite initiatique, tout le monde y est bienvenu et nous sommes prêts à prendre avec nous n'importe qui. La seule obligation de type mystique est un infini respect des pères fondateurs (D.E. Knuth, L. Lamport, F. Mittelbach...) sans qui TeX et LaTeX ne seraient pas.

Le GEUT n'est qu'un groupe (petit) d'élèves qui connaissent un petit peu TeX ou LaTeX et qui sont prêts à empiéter un peu (beaucoup pour certains) sur leurs loisirs pour aider tous les utilisateurs de l'école à produire des documents de bonne facture avec l'un des meilleurs logiciels qui soit.

Le GEUT se propose de vous offrir des documentations en français sur certains thèmes qui intéresseront suffisamment ses membres pour qu'ils écrivent des docs.

5.2 État des lieux

Un prototype d'installation pour PC (MS-DOS) mimant parfaitement celle des stations existe et est, à ce jour, en démonstration sur un des PCs du département info. Elle sera probablement remise à niveau bientôt.

Une installation pour Linux existe et est disponible sur disquette ou sur le réseau.

Une installation pour amiga répondant aux mêmes exigences existe. Nous la devons à Julien Wilk, Yves Surantin et Patrick Reuter.

Il existe des docs en français sur pas mal de thèmes (Mathematica, LaTeX2.09, construction des tableaux, composition de partitions...).

Il existe aussi des docs sur à peu près tout en anglais mais elles ne sont pas toujours disponibles sur papier. De toutes façons, poser la question ne coûte rien et te laisse une chance d'obtenir une doc.

5.3 Projets

Les projets plus ou moins en cours sont les suivants :

- L'installation sur Mac sous System7.
- La tenue à jour du «Joli manuel pour LaTeX 2_ε».
- La tenue à jour de LaTeX.
- L'écriture et la mise au point d'une documentation en HTML.
- L'aide aux utilisateurs par mail.

[1] Il dépend plus du SMIG que du BdE, pour pouvoir travailler et plus du BdE que du SMIG pour diffuser ses docs

[2] La «vente» de cette présente doc à un prix symbolique permettra cependant de faire face à d'éventuels frais du type achat de livres de référence

Overview

6 Présentation générale

LaTeX est un outil de compilation de documents. Ce n'est pas un traitement de texte au sens classique du terme. Il te permettra, à l'aide de mots clef spécifiques de dire comment tu veux voir apparaître ton document, la logique qui le caractérise, tes conventions de mise en page ... pour qu'il puisse, tout seul, produire un document le plus convenable possible.

LaTeX est particulièrement étudié pour les mathématiques, il permet la mise en forme de rapports, de thèses, de polys ... d'une qualité exceptionnelle. On peut, sans trop se tromper, dire que personne ne vient lui faire d'ombre pour tout ce qui touche à l'édition d'équations, et que ses possibilités sont grandes pour tout le reste.

Cependant, LaTeX n'est pas fait pour produire des affiches et ne peut par conséquent pas réaliser simplement des travaux d'édition de texte tordus. Il y a pour cela d'autres outils puissants et gratuits, dont Xfig

Toutefois avant de renoncer à utiliser LaTeX, demande-moi, j'ai quelques jolies choses dans ma besace.

LaTeX compile des fichiers ascii standards, avec l'extension `.tex`. Il reconnaît ses commandes au fait que celles-ci commencent par un backslash (\), et reconnaît certains caractères comme étant spéciaux.

7 Premiers pas

7.1 Éditer ton premier document

Lance **`ved`**, par exemple en cliquant sur son icône. Tapes, par exemple, le document suivant :

```
\documentclass[french,12pt]{article}

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

\begin{document}
Bonjour, Monde!
\end{document}
```

Sauve le sous un nom finissant par `.tex`, mettons `premier.tex`. Ça y est, ton premier document existe.

7.2 Ta première compilation

Il te reste à le compiler. Pour cela, deux choix. À la souris, il te suffit de double-cliquer avec le bouton de gauche sur l'icône de ton fichier. Au clavier, dans un terminal, tu taperas la ligne suivante :

```
latex premier
```

en ayant soin d'être dans le bon répertoire.

A ce stade plusieurs fichiers ont été créés :

`premier.log` C'est la transcription détaillée de tout ce qui s'est passé à la compilation. En particulier des éventuelles erreurs.

`premier.aux` C'est un fichier auxilaire (parmi plusieurs autres pour certains documents plus complexes) très précieux à LaTeX.

`premier.dvi` C'est le résultat de la compilation, celui que tu visualiseras.

7.3 Regarder ton premier document

Pour regarder le résultat de tout cela à l'écran, double-clique sur le fichier `premier.dvi`, ou tape au clavier dans un terminal :

```
xdvi premier
```

7.4 Pour imprimer ...

Si, satisfait de ton travail, tu souhaites l'imprimer, là encore, deux options s'offrent à toi :

- Clique avec le bouton de droite de la souris sur l'icône du fichier `premier.dvi` et choisis l'option *Print_lps20*.
- Tape dans un terminal

```
dvips premier
```

7.5 Dernière fois

Autant te le dire tout de suite, c'est la dernière fois que je perdrais autant de temps et de place à t'expliquer des choses aussi simples et futiles.

8 Structuration du document

Dans LaTeX 2ε on structure les documents de manière différente de l'ancienne version. En effet apparaissent de nouvelles notions comme la notion de classe, d'option, de package…

8.1 Les classes

En fait, tu sais déjà de quoi il s'agit[3], ce sont les sempiternels `report`, `article`, `book`, et `letter`. Avant, c'étaient des fichiers de styles tout bêtes, si ce n'est qu'il en fallait un par document. Et forcément un de ceux-là. Pour marquer leur spécificité, on leur a donné un nom : ce sont des classes de documents, et les fichiers correspondants portent l'extension `.cls`.

Les classes de LaTeX 2ε sont référencées de manière à peu près exhaustive dans le tableau 1 page ci-contre.

8.2 Les options

Tu te souviens sans doute que dans les options du `\documentstyle`[4] on plaçait un peu tout et n'importe quoi : aussi bien les options du style principal (11 pt, par exemple) que les macros d'extension à charger pour la compilation (`eepic`, `epsf`…).

LaTeX 2ε, dans un souci de clarification, a décidé de séparer les choses selon leur sens. Les options sont des options, les extensions (packages) sont des extensions, deux syntaxes différentes sont désormais utilisées pour les appeler.

Conformément aux habitudes sous LaTeX, les options sont placées entre crochets juste après le nom de la commande. Ainsi pour charger la classe de document `report` en 12 points, on tapera en tête de document :

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

Il a aussi été constaté que des versions différentes survivaient selon les endroits où on les cherchait. Ainsi, en recevant un document d'une autre université, il pouvait supposer une version différente de celle actuellement utilisée. Pour cela, on peut spécifier une date de version à partir de laquelle on accepte le fichier :

```
\documentclass[12pt]{report}[1994/06/01]
```

permet de demander à LaTeX 2ε d'utiliser le style `report` avec l'option `12pt` dans une version datant d'après le premier juin 1994. Si une version plus ancienne est trouvée, un message te le signalera lors de la compilation.

8.3 Les packages

Les paquets de macros-commandes à charger avant la compilation sont désormais légion sous LaTeX (plusieurs dizaines, rien qu'à l'ESIEE). Une commande spéciale a donc été prévue.

Ainsi, si tu souhaites utiliser les macros du fichier `a4.sty`, tu placeras cette ligne entre le `\documentclass` et le `\begin{document}` :

```
\usepackage{a4}
```

Cela fait exactement la même chose que lorsque tu plaçais `a4` dans les options de `\documentstyle`. Toutefois, les packages admettent désormais des options et peuvent là encore avoir des dates de version à respecter.

Pour éviter d'avoir à répéter sans cesse cette commande, il est possible de donner une liste de packages à charger, simplement en séparant leurs noms par des virgules. De plus pour les options, toutes les options positionnées dans un `\usepackage` sont passées à tous les packages lus par cette commande. Enfin, les options déclarées dans le `\documentclass` sont globales et inchangeables, elles sont donc passées à tous les packages sans les répéter.

Par exemple, l'en-tête actuel de ce document est :

```
\documentclass[french,10pt]{article}
\usepackage[german,english]{babel}
```

L'option `french` a été placée dans les options du `\documentclass` car tout mon document est en français, et je ne souhaite pas avoir à la répéter sans cesse.

Les différents packages ainsi que les options qu'ils reconnaissent sont décrits tout au long de ce document.

8.4 Mode compatibilité

Les auteurs de LaTeX 2ε étant des gens sérieux, une compatibilité avec l'ancienne version de LaTeX a été prévue. Ainsi, si ton document commence avec l'ancien `\documentstyle` au lieu du nouveau `\documentclass`, LaTeX 2ε passera tout seul en mode « compatibilité 2.09 », c'est à dire qu'il fera comme si on compilait avec LaTeX 2.09.

Quelques remarques rapides :

- Aucune des nouvelles possibilités n'est accessible depuis ce mode.
- La compatibilité n'est garantie que pour l'utilisation des commandes de haut niveau. En particulier pour la gestion des fontes, il peut s'avérer utile de charger le package `rawfonts` si certaines de tes macros font appel à des macros « bas-niveau » pour les fontes.

  Ce cas étant difficile à détecter, parle-m'en avant de dire que le mode compatibilité ne marche pas.

  Certaines extensions faisaient appel à des commandes bas-niveau et ne sont donc plus tolérées. C'est pour cela que nous conservons les deux versions.

[3] Si tu as déjà travaillé avec l'ancienne version, LaTeX 2.09. Sinon tu vas apprendre très vite.

[4] Ancêtre du `\documentclass`

`article`	Inchangé.
`report`	Inchangé.
`book`	Inchangé.
`letter`	Je ne sais toujours pas m'en servir.
`slides`	Remplaçant et successeur de SliTEX.
`refman`	Classe permettant la mise en page de manuels de référence.
`refman-s`	Comme `refman` mais en plus petit.
`proc`	Classe spéciale pour les compte rendus de conférences (proceedings).
`exam`	
`ltxguide`	Classe spéciale utilisée par les concepteurs de LaTeX 2ε pour écrire un mode d'emploi.
`ltxdoc`	Classe de documents utilisée pour fournir un exemplaire très lisible et bien documenté des sources de LaTeX 2ε.
`ltnews`	Classe utilisée pour écrire la lettre d'information diffusée avec LaTeX 2ε.

TABLEAU 1: Classes utilisables par LaTeX 2ε

- Il est fortement recommandé de ne pas utiliser ce mode pour de nouveaux documents car il n'est là que pour éviter les pertes et ne sera probablement plus dans LaTeX3.
- Si le mode compatibilité échoue lamentablement, essaie de passer en LaTeX 2ε pur en corrigeant l'en-tête et les changements de fonte (`\bf`, `\it`,...)
- La majorité des extensions pour LaTeX 2.09 fonctionnent encore avec LaTeX 2ε.

9 Structure de documents types

Je ne te donnerais ici que deux documents types. L'un est un rapport bidon, dont le seul intéret est de te montrer comment cela se présente normalement. Je n'y utilise pas le système d'inclusion de fichiers, mais il est recommandé de le faire en respectant le principe qui dit un chapitre par fichier.

9.1 Rapport type

Ce document est disponible sur le reseau sous le nom :

/user/tex/Exemples/rapport.type.tex

9.1.1 En-tête du document

```
\documentclass[french,twoside,openright]{report}

\usepackage[T1]{fontenc}
\usepackage{babel,indentfirst}

\begin{document}
```

A noter :

- *L'utilisation du package* `fontenc` *qui doit se faire comme cela pour permettre la césure convenable de notre belle langue ou de toute langue ayant des caractères accentués. De toutes façon, ça ne nuit pas dans les autres cas.*
- *L'utilisation du package* `babel`*. C'est lui qui se chargera de faire les traductions utiles (par exemple «chapter» en «chapitre»).*
- *Le package* `indentfirst` *pour que le premier paragraphe suivant un titre de section soit indenté, alors qu'en anglais on ne le fait pas.*
- *L'option* `french` *qui est mise comme globale alors que ce n'est pas une option de* `report`*. Simplement, elle sera passée à tous les packages qui en auront besoin. C'est à dire tous les packages qui sont suceptible de produire du texte.*
- *L'option* `twoside` *qui demande à ce que l'on travaille en recto-verso. Ça a deux conséquences. La première est que les parties commencent sur des pages impaires, et la deuxième est que les marges — et éventuellement les entêtes de pages — seront alternées selon que l'on est sur une page de droite (recto) ou de gauche (verso).*
- *L'option* `openright`*, qui n'a de sens que si la précédente est là, indique que même les chapitres doivent commencer sur des pages impaires (de droite).*

9.1.2 Le titre

```
\begin{document}

\title{Rapport bidon}
\author{Benjamin \textsc{Bayart}
\and Moi \textsc{M\^eme}
\and Moi \textsc{Aussi}
\and Personne \textsc{d'Autre}}
\date{Le \today}
```

```
\maketitle

\strut\thispagestyle{empty}
\vfill
\pagebreak

\setcounter{page}{1}
```

A noter :

- *Le titre, la date, l'auteur et la génération de la page de titre, sans commentaire.*
- *Juste après, c'est à dire pile à l'endroit où se trouvent les références ISBN et autres formalités pour les livres, je met un truc invisible (`\strut`). Ensuite je demande à ce que **cette** page là ne soit pas numérotée. Ensuite je la remplis (`\vfill`). Enfin, je change de page. Le but de toute cette manœuvre est d'insérer une page blanche au dos de la couverture pour que lorsque l'on imprime directement en recto-verso, tout ne soit pas décalé d'une page sous prétexte que la couverture est en recto simple.*
- *Il convient de penser à ramener le compteur de page à 1 sinon toute la manœuvre précédente n'a servi à rien. LaTeX devine si une page est recto ou verso d'après sa parité (page impaire=recto, page paire=verso)[5].*

9.1.3 Le début

```
\tableofcontents

\chapter*{Introduction}
\addcontentsline{toc}{chapter}{Introduction}
\markboth{\uppercase{Introduction}}
{\uppercase{Introduction}}

Voici l'introduction de mon rapport, elle est
tr\`es jolie et tout et tout. Tu noteras que
dans un rapport en \LaTeX{}, il n'y a que ce qui
se trouve \textsc{avant} le
\verb"\begin{document}" qui est d\'ependant de
la machine utilis\'ee.

En effet, le reste est du standard \LaTeX{} \`a
peu pr\`es ind\'ependant de l'installation
effectu\'ee. Par exemple, il est probable que
sur ton site en Grande-Bretagne le fran\c{c}ais
soit convenablement install\'e.

De toutes fa\c{c}ons tu t'en fiches, je pense
que ton rapport sera en anglais. Pour ce faire,
il suffit de virer le \verb"[french]" dans la
premi\`ere ligne, le
\verb"\usepackage[T1]{fontenc}" et le
\verb"babel". Apr\`es, tu te retrouves avec le
standard am\'ericain. L\`a o\`u cela se corse
c'est que si tu met du fran\c{c}ais, il risque
de faire des fautes. Par exemple, avec les
mod\`eles de c\'esure\footnote{coupure des mots
en fin de ligne} am\'ericain, il tol\`ere une
c\'esure entre le <<n>> et le <<c>> de <<donc>>.
Si << et >> donnent des points d'interrogation
bizarres, c'est que tu as vir\'e la ligne sur le
\verb"fontenc".
```

A noter :

- *La table des matières (`\tableofcontents`).*
- *L'introduction. Cette construction complexe sert à produire un chapitre non-numéroté et recensé dans la table des matières. Une commande toute faite existe dans le package **ESIEE**, mais celui-ci n'est pas un standard international[6].*

9.1.4 Le corps du document

Attention : *l'exemple réel contient un texte, mais je n'ai pas souhaité le reproduire ici pour ne pas perdre de place.*

```
\part{\'Etude pr\'eliminaire}

\chapter{On a commenc\'e}

\section{Le commencement}

C'est par l\`a qu'on a commenc\'e, comme
d'habitude.

\section{Juste apr\`es}

Ben on a continu\'e de commencer.
Mais plus fort.

\section{Enfin}

..........................................
[tr\`es long rapport tr\`es intelligent]
..........................................

\chapter{Euthanasie des profs cons}

\chapter{Euthanasie des administratifs cons}

\section{Y sont nombreux en plus}
```

A noter :

- *Les commandes de sectionnement*
- *C'est principalement cette partie là qu'explique le reste de la doc.*

[5] Convention typographique plus que classique. À tel point que les pages de droites sont dites «belles pages» et les pages de gauche sont dites «fausses pages».

[6] C'est déjà pas un standard à l'ESIEE ...

9.1.5 La fin du rapport

```
\chapter*{Conclusion}
\addcontentsline{toc}{chapter}{Conclusion}
\markboth{\uppercase{Conclusion}}
{\uppercase{Conclusion}}

Heureusement que le projet il durait pas
plus longtemps parce que \c{c}a co\^ute
vachement cher de glander \`a la Kfet
toute la journ\'ee.

\part{Annexes}

\appendix

\chapter{R\'esultats chiffr\'es}

\begin{tabular}{|l|r|r|r|}  \hline
          & D\'epenses & Recettes
          & Bilan    \\ \hline
Caf\'e    & 3658,25 F  & 1,75 F
          & -3656,50 \\ \hline
Croissant & 1548,12 F  & 0,00 F
          & -1548,12 \\ \hline
Beignet   &  925,30 F  & 8,25 F
          &  -917,05 \\ \hline
                       \hline
Total     & 6131,67 F  & 10,00 F
          & \textbf{-6121,67}
          \\ \hline
\end{tabular}

\chapter{R\'esultats humains}

Ben je m'est beaucoup amus\'e. Pas vous?

\end{document}
```

A noter :

- *La conclusion qui marche comme l'intro. On utilisera volontier le même système pour les remerciements et la préface. Pour la bibliographie c'est pas la peine, c'est prévu par LaTeX.*
- *L'exemple le plus simple de création de tableau.*
- *Pour passer du corps du document aux annexes, on indique simplement le mot clef `\appendix`. Note qu'il ne produit pas de page avec écris en gros* **Annexes***. C'est à toi de le faire.*
- *Les deux chapitres viendront en fait créer l'«***Annexe A***» et l'«***Annexe B***». C'est le choix fait automatiquement par LaTeX.*

9.2 Lettre type

La lettre type est donnée un peu plus loin. Avant, je vais exposer comment on utilise la classe `letter`.

Il s'agit d'une classe de documents que j'utilise plus que rarement, aussi me bornerais-je à te donner les commandes usuelles en te rappelant que la mise en page choisie est celle «à l'américaine».

Les cinq déclarations globales :

- `\name` pour le nom de l'expéditeur ;
- `\signature` pour sa signature ;
- `\address` pour son adresse ;
- `\location` pour un ajout de précision à l'adresse, par exemple «cellule 1452» ;
- `\telephone` y faut vraiment te faire un dessin ?

Chaque lettre est en fait un environnement `letter`.

Il est jugé comme essentiel que le nom du destinataire soit la première ligne de l'argument, et que chaque ligne de celui-ci soit séparée des autres par un `\\`

Voilà l'exacte étendue de mes connaissances sur le sujet.

Je sais aussi que l'on peut se servir du style `letter` pour générer des lettres automatiquement à partir d'un fichier d'adresses de destinataires, mais je ne souhaite pas développer ici ce genre de cuisine.

Voilà ce que peut être une lettre type. On trouvera le document ci-dessous dans le fichier

`/user/tex/Exemples/lettre.type.tex`

Le source est le suivant et le résultat est illustré figure 1 page 15

```
\documentclass[french,12pt]{letter}

\usepackage[T1]{fontenc}
\usepackage{babel}
\usepackage{ESIEE}

\begin{document}

\name{Benjamin \textsc{Bayart}}
\signature{}
\address{1, rue de Beaumont\\
95 560 Maffliers}
\location{}
\telephone{34~69~88~17}

\begin{letter}{Beno\^it Joly\\
33, rue du Duc de Dantzig\\
77 340 Pontault-Combault}
\opening{Tr\`es cher Beno\^it,}

Depuis ces presque cinq semaines que nous ne
nous sommes vu, bien des choses se sont
produites. Bien des \'ev\'enements ont
\'et\'e c\'el\'ebr\'es.

Il faudra que je te racontes combien
int\'eressant fut ce projet, que je te dise
\`a quel point il est d\'elicat de travailler
sur d'aussi complexes sujets sans avoir de
solides bases th\'eoriques auxquelles se
raccrocher.
```

```
Comme convenu je pense que nous nous
retrouverons tr\`es bient\^ot pour discuter
de tout cela et du reste.

\closing{Tr\`es affectueusement,}
\end{letter}
\end{document}
```

9.3 CV type

Nous devons ce paragraphe à Pierre Le Maguet *qui par son insistance folle à réussi a me faire écrire un petit package répondant au doux nom de* `ESIEEcv` *et qui sert à produire des CVs comme il les aime. Pierre étant à peu près aussi fou que moi, il a tenu à ce que ce package soit ultra-paramètrable. En échange de l'effort que m'a demandé l'écriture du package, je lui ai demandé d'en rédiger la documentation en français. C'est ce texte que vous allez lire. Qu'il soit ici chaleureusement remercié de sa généreuse et grande contribution à la défense et à la promotion de LaTeX à l'ESIEE.*

Un CV type, écris avec ce package est disponible sur le réseau comme étant le fichier suivant :

`/user/tex/Exemple/cv.type.tex`

Libre à toi, lecteur feignant, de recopier ce CV et de le modifier pour faire le tien.

9.3.1 Bonjour

Cher lecteur adepte de LaTeX, je te salue.

Je ne suis pas un spécialiste de LaTeX, mais j'aime bien son rendu. Aussi ai-je tanné Benjamin pas mal de temps pour faire mon CV. Il en est sorti un package : `ESIEEcv.sty` qu'il m'a demandé de présenter.

Le but du package est de proposer des outils facilitant la mise en page de CVs et permettant de changer cette dernière rapidement. Pour cela, on s'est attaché à avoir un fichier LaTeX contenant un CV organisé par aspects, et un fichier `ESIEEcv.sty` assurant la mise en page de ce dernier. Pour changer l'aspect du CV, on modifiera le fichier `.sty`, pour en changer le contenu, on reprendera le fichier `.tex`.

Dans un premier temps, je vais essayer de présenter le fichier `.tex`. Je ne m'attarderai donc pas trop sur les considérations techniques mais plus sur la méthode d'utilisation. Ensuite, si tu veux personnaliser le layout[7], tu peux recopier `ESIEEcv.sty` dans ton répertoire de travail et le changer à souhait[8].

9.3.2 «Rentrons dans le vif du sujet»

un philosophe inconnu,
un jour brumeux de l'automne 1991.

Les pré-requis Votre CV doit inclure le package `ESIEEcv` de la manière suivante :

```
\usepackage{ESIEEcv}
```

N.B. : Dans sa version de base, `ESIEEcv` appelle les packages `tabularx` et `stmaryrd`. `tabularx`, package bien connu, est utilisé pour la création de tableaux. Il est requis car chaque «rubrique» (cf plus loin) déclare un tableau, qui va aider à la mise en page. `stmaryrd` est un package définissant une quantité impressionnante de caracteres spéciaux, notemment celui utilisé en début de champ «Apport» (cf plus loin aussi).

La Rubrique Un CV est en général structuré en grandes rubriques du style état civil, études, expériences, etc ... Nous avons donc créé une fonction permettant de faire une «Rubrique». Les pros s'y retrouveront mieux si je leur dit que Benjamin a créé un environnement.

La rubrique a un titre, sensé résummer son thème : On utilisera donc les commandes suivantes :

```
\begin{rubrique}{titre de la rubrique}
notre & rubrique  \\
est & vide \\
nous & devons maintenant la remplir \\
\end{rubrique}
```

avec un résultat dont la magnifiçance nous échappe encore :

Titre de la rubrique

notre	rubrique
est	vide
nous	devons la remplir

La sous-rubrique A l'intérieur de la rubrique, on trouvera plusieurs sous-rubrique, que nous avons classées comme suit :

- Les sous-rubrique de type «expérience située dans le temps, l'espace, la fonction, etc».
- Les sous-rubrique du type «compétence» telles que les langues, les outils CAO ou les langages informatiques.
- Les autres sous-rubrique, auxquelles nous n'avons pas pensé. Ces sous-rubrique n'ont pas été implémentées, nous n'en parlerons donc pas.

En langage LaTeX, la sous-rubrique est, comme la rubrique, un environnement. On la déclare donc de la même manière, au titre près :

```
\begin{sousrubrique}
.
.
.
\end{sousrubrique}
```

[7] Foutu angliciste de m ... en français, on dit «personnaliser la mise en page» !

[8] Pierre est d'une bestialité sans bornes. Il est tout à fait possible de personnaliser sans corriger le fichier `ESIEEcv.sty`, pas autant, certes, mais quand même pas mal.

1, rue de Beaumont
95 560 Maffliers

28 juin 1995

Benoît Joly
33, rue du Duc de Dantzig
77 340 Pontault-Combault

Très cher Benoît,

Depuis ces presque cinq semaines que nous ne nous sommes vu, bien des choses se sont produites. Bien des évènements ont été célébrés.

Il faudra que je te racontes combien intéressant fut ce projet, que je te dise à quel point il est délicat de travailler sur d'aussi complexes sujets sans avoir de solides bases théoriques auxquelles se raccrocher.

Comme convenu je pense que nous nous retrouverons très bientôt pour discuter de tout cela et du reste.

Très affectueusement,

Benjamin BAYART

FIGURE 1: Résultat de la compilation de la lettre type

Pierre Le Maguet	Né le 28 mai 1973 (22 ans)
68, rue Jacques Prévert	Français
95320 Saint-Leu-la-Forêt	Non dégagé des obligations militaires
Tél: (1) 39-32-05-38	Célibataire
E-mail: lemaguep@esiee.fr	Permis B

Formation

1990-96	ESIEE, ECOLE SUPÉRIEURE D'INGÉNIEURS EN ELECTROTECHNIQUE ET ELECTRONIQUE. Spécialisation en Micro-Electronique et Traitement de Signal.
Langages:	C, VHDL, Pascal, ASM TMS320C50 & 680x0, Signal.
CAO:	Compass, Cadence, Mentor Graphics, PSpice.
Logiciels	Matlab, Mathematica, SPW, LaTeX.

Langues

Anglais	Courant.
Allemand	Courant. Un an en Saxe.

Expérience Technique

1995 (6 mois)	FRAUNHOFER-INSTITUT, DRESDEN. AIDE DE RECHERCHE 10 H/SEMAINE. Conception d'une interface graphique pour la visualisation de résultats de recherche. ▷ *Développement sous Unix/Sun OS.* ▷ *Etablissement d'un cahier des charges en coopération avec les futurs utilisateurs.* ▷ *Projet entièrement réalisé en langue allemande.*
1994 (6 semaines)	ESIEE-PARIS. DÉVELOPPEMENT D'UN MODEM QPSK SELON LA NORME V26. Etude de la correction de dérive de fréquence porteuse. Intégration Full duplex.
1993 (6 semaines)	ESIEE-PARIS. CONCEPTION ET RÉALISATION D'UNE CARTE PC POUR LE TRAITEMENT D'IMAGE. Carte à architecture reconfigurable du fait de l'utilisation d'un Xilinx 3050. ▷ *Implantation de filtres morphiques de prédétection.* ▷ *Première expérience pratique conséquente.*

Expérience Personnelle

1992-94 (2 ans)	LINGUAFORM, PARIS. AUXILIAIRE PÉDAGOGIQUE. Aide à l'encadrement et à l'organisation de voyages linguistiques au Royaume-Uni. ▷ *Travail d'organisation.* ▷ *Sensibilisation aux problèmes déontologiques posés par l'encadrement d'enfants.* ▷ *Contact client.*
1990 (1 mois)	BONDUELLE, BEAUVAIS. STAGE OUVRIER. Travail à la chaine, organisation en trois huit.

FIGURE 2: CV type réalisé avec le package `ESIEEcv`.

Le titre a disparu. En effet, le titre, de même que les autres champs, est optionnel. On trouve 6 champs dans une sous-rubrique.

- 6 champs utiles si la sousrubrique sert à décrire une expérience :
  - la date de l'expérience, introduite par la commande `\Date`,
  - la durée de l'expérience, introduite par la commande `\Duree`,
  - le lieu de l'expérience, introduit par la commande `\Lieu`,
  - le titre de l'expérience, introduit par la commande `\Titre`,
  - une description, introduite par la commande `\Descr`,
  - le champ «Apport», qui permet de mettre en style télégraphique ce qui manque. Contrairement aux autres champs, «Apport» présente la particularité d'être répétable. Il correspond à la commande `\Apport`.
- 2 champs utiles pour la description de comptétences. Le champ «Compétence» (commande `\Competence`), qui sera associé avec le champ «Description» (`\Descr`), déjà présent plus haut.

Voici un exemple :

```
\begin{rubrique}{Formation}
\begin{sousrubrique}
\Titre {\'Ecole du r\^eve}
\Date {Antiquit\'e}
\Duree {quelques si\'ecles}
\Lieu {Paradis}
\Descr {Apprentissage des bases de la th\'eorie
relative \`a l'imaginaire et \`a la mythologie.
Sp\'ecialisation en mythologie juv\'enile.}
\Apport {Acquisition d'une bonne base de
connaissances, qui m'a permis, par la suite,
de m'adapter aux diff\'erentes \'evolutions
culturelles de la sph\`ere chr\'etienne}
\Apport {Maturation de mon projet professionnel}
\end{sousrubrique}
\end{rubrique}
\begin{rubrique}{Langues}
\begin{sousrubrique}
\Competence {Europ\'eennes}
\Descr {Toutes couramment.}
\end{sousrubrique}
\begin{sousrubrique}
\Competence {Autres}
\Descr {Lues, \'ecrites, parl\'ees. Ai
\'et\'e amen\'e, depuis 30 ans et suite \`a
la globalisation du march\'e de No\"el, \`a
voyager de plus en plus fr\'equemment hors
d'Europe.}
\end{sousrubrique}
\end{rubrique}
```

qui, après compilation, donne le résultat suivant :

Formation

Antiquité (quelques siécles)	PARADIS. ÉCOLE DU RÊVE. Apprentissage des bases de la théorie relative à l'imaginaire et à la mythologie. Spécialisation en mythologie juvénile. ⊳ *Acquisition d'une bonne base de connaissances, qui m'a permis, par la suite, de m'adapter aux différentes évolutions culturelles de la sphère chrétienne.* ⊳ *maturation de mon projet professionnel.*

Langues

Européennes	Toutes couramment.
Autres	Lues, écrites, parlées. Ai été amené, depuis 30 ans et suite à la globalisation du marché de Noël, à voyager de plus en plus fréquemment hors d'Europe.

NB : Il est possible de remplir une rubrique sans passer par les sous-rubriques. La rubrique sera alors remplie comme un tableau normal. Ainsi,

```
\begin{rubrique}
{Essai: rubrique sans sous-rubrique}
Mais & ce n'est pas tr\'es propre,\\
donc ne dites pas \`a & Benjamin que
c'est moi qui vous en ai parl\'e\\
\end{rubrique}
```

donne

Essai : rubrique sans sous-rubrique

Mais ce n'est pas très	propre,
donc ne dites pas	à Benjamin que c'est moi qui vous en ai parlé

Pré-nonrequis Avant d'en finir avec le fichier `.tex`, j'ajouterais qu'il reste un élément à y insérer.

Vous aurez compris qu'en fait, chaque rubrique crée un tableau bi-colonne, et chaque sous-rubrique se contente de remplir une ligne de ce tableau. Or, pour obtenir une jolie mise en page, il faut que la première colonne de chacun de ces tableaux aie la meme largeur que ses petites copines des tableaux des autres rubriques.

On a tout d'abord pensé à faire un système qui prendrait la plus grande de ces largeurs, et l'imposerait aux autres. Mais il n'y a pas de commandes LaTeX permettant de faire cela simplement. La plus grande largeur est en effet connue après la compilation, et LaTeX n'est pas vraiment d'accord pour recommencer la compilation. L'utilisateur est donc obligé, s'il ne veut pas de la largeur par défaut (3 cm), de redéfinir lui-même cette largeur.

Elle est rangée dans la variable `\largeurcolonne`. L'utilisateur devra donc — en début de document — écrire quelque-chose dans ce goût là[9] :

```
\setlength {\largeurcolonne}{1.2cm}
ou
\settowidth{\largeurcolonne}{texte de r\'ef\'erence}
```

Voilà. Si vous n'avez pas envie d'en savoir plus, vous pouvez vous débrouiller. Mais pour ceux que cela intéresse, je propose que nous voyons comment changer le layout[10].

9.3.3 La faute n'est pas la faute, la faute, c'est de recommencer

Platon, un jour vraissemblablement peu brumeux de l'antiquité grecque.

Pourquoi ce titre? C'est ce que je me suis dit en commençant cette deuxième partie, et ça m'a coupé toute envie d'aller plus loin. Je dirais : Ouvrez `ESIEEcv.sty` et lisez le. Il est suffisamment petit pour être facile à comprendre.

Mode d'emploi Si vous trouvez la chose indigeste, parlez-en à Benjamin[11]. C'est lui le Dieu[12], il est donc normal que ce soit lui qui écrive la partie technique de sa doc[13]. Si sa réponse vous fait craindre pour votre survie, venez me voir[14]. Il est parfois un peu surmené. Je suis un TEXnicien de seconde zone, mais j'ai la chance d'avoir été formé par un Dieu. J'essaierai donc de vous dépanner du mieux que je pourrai[15].

Bon CV !

Le CV type dont je parle au début fait l'objet de la figure 2 page 15. Voila donc ce à quoi peut mener ce package.

10 Spécificité de l'installation locale — fichier `ESIEE.sty`

10.1 Le fichier `ESIEE.sty`

Lorsque j'ai installé LATEX 2_ε à l'ESIEE, j'ai eu l'occasion d'écrire certaines macros assez simplistes mais qui permettaient de faciliter la vie de pas mal de gens dans des situations relativement fréquentes. J'en ai profité pour corriger certaines erreurs (que j'ai réussi à comprendre, donc grossières) contenues dans quelques packages standard.

Au lieu d'effectuer toutes ces petites choses soit dans le noyau de LATEX 2_ε soit directement dans les packages concernés, j'ai préféré garder un jeu de fichiers standard et placer le tout dans un fichier à part `ESIEE.sty`. Il s'utilise comme tous les autres packages :

```
\usepackage{ESIEE}
```

Il effectue ses corrections d'après ce que vous avez choisi de précharger, par exemple, pour `varioref`[16], la ligne de correction n'était prise en compte que si `varioref` était lu. Il est donc recommandé d'appeler le package `ESIEE` en dernier, sur une ligne à part (un `\usepackage` pour lui tout seul) puisque lorsque deux packages sont dans la même requête, rien ne garantit l'ordre de lecture.

10.2 Gestion des caractères

Comme tu le sais peut-être, les stations de travail de l'ESIEE sont capables de travailler avec un jeu de 256 caractères complet comportant entre autre les accents français. Toutefois, je ne connais aucune bidouille clavier permettant d'atteindre simplement chaque caractère intéressant[17].

Toutefois dans certains éditeurs (`ved`, celui par défaut ; `vuepad`, un des grands classiques) certains

[9] Pierre expose ici des commandes que j'ai toujours refusé de décrire dans mes docs. En effet, ces commandes ne servent généralement qu'à faire de la programmation pour LATEX et je n'ai jamais voulu faire peur à mes lecteurs avec de tels sujets. Je le laisse donc libre d'agir et ne me porte par garant du résultat.

[10] La mise en page pour les mono-francophone comme moi.

[11] Je serais très clair sur ce point, Pierre, je refuse catégoriquement toute explication technique d'un de mes sources à toute personne qui n'aurait pas préalablement lu avec la plus grande attention les ouvrages de références, c'est à dire celui de KNUTH [58] et celui de LAMPORT [61]. Et quiconque aura lu ces livres trouvera mes bidouilles tellement enfantines qu'il n'aura pas la moindre question à poser. Le seul point que j'aborderais est celui de la redéfinition des 6 commandes qui sont définies au début du fichier `ESIEEcv.sty`. Les deux premières, `\PreApport` et `\PostApport` correspondent à ce qui est mis avant et après chaque «Apport» dans une liste d'apport. Les deux suivantes, `\PrePreApports` et `\PostPostApports` correspondent à ce qui est mis avant et après la liste des apports. Enfin les deux dernières, `\FonteApport` et `\TailleApport` permettent de changer la taille et la fonte des apports facilement. Pour redéfinir une commande on utilisera la commande `\renewcommand`, non documentée ici. Par exemple, pour que les apports soient écris en gras au lieu de l'italique par défaut, il suffira de faire :
`\renewcommand{\FonteApport}{\bfseries}`
C'est aussi simple que ça.

[12] Ouais, euh, pas vraiment, mais bon, je vais pas contrarier Pierre sur un détail aussi ridicule. Je ne suis qu'un suppôt des Grands que sont des gens comme Knuth, Mittelbach, Lamport et quelques autres.

[13] Cette partie technique existe bel et bien. Elle est simplement réservée à un usage limité. Toi, lecteur moyen, tu te fous éperdument de savoir comment on écris un package ou comment on gère une liste chaînée en TEX pur et dur.

[14] Ce sera à priori, toujours le cas.

[15] Pierre sera souvent un conseiller suffisant. Il est moins sot qu'il essaye de le faire croire.

[16] Ce package comportait autrefois un bug.

[17] Du moins, je n'ai pas encore trouvé une bidouille qui marche sur toutes les stations de l'école; mais je cherche ...

caractères sont faciles à obtenir. Il s'agit des deux c-cédilles et des guillemets à la française. Ces caractères ont donc été redéfinis dans le fichier `ESIEE.sty` pour être correctement exploités par LaTeX 2ε. Pour obtenir un ç, il faut taper `Alt-c`[18] ; pour le Ç, il faut faire `Alt-Shift-C`, pour le «, on utilisera `Alt-touche en haut à gauche` (celle avec le tilde) et enfin pour le », ce sera pareil avec `Shift`. Si on tape les quatre à la suite, on obtient : Çç«».

Une petite remarque sur l'obtention des guillemets «à la française». Ils ne sont pas disponibles dans les fontes par défaut de LaTeX (connues sous le nom de `cm` comme *Computer Modern*) ce qui fait qu'autrefois on utilisait le symbole mathématique $\ll$ en plus petit, ce qui donnait $\ll$ et $\gg$. Fort heureusement, de nouvelles fontes sont apparues contenant ces symboles, les fontes `dc`. Elles permettent, en plus, de césurer les mots accentués, ce qui n'était pas possible avant. Pour les utiliser, il te faut appeler

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

Pour obtenir des guillemets «à la française», c'est facile, tu tapes `<<` et `>>`[19]. Et ça marche tout seul[20] !

10.3 Tous les accents : `mapcodes`

Tu le sais peut-être, certains caractères accentués sont accessibles directement depuis le clavier de nos chères stations avec une grosse bidouille à faire. Pour ceux qui auraient trouvé comment faire, il ne reste plus qu'à faire manger tous ces beaux accents à LaTeX.

Pour cela, Michael Piotrowski, nous a écrit un très joli package, `mapcodes` [92]. Ce package est assez général puisqu'il prévoit un grand nombre de codages possibles. On lui indique le codage à utiliser par une option qu'on lui passe. Les options valides sont celles du tableau 2 page suivante.

Bon, en gros tu tapes

```
\usepackage[hproman8]{mapcodes}
```

et tu poses pas de questions.

Ou, encore plus simple, le package `ESIEE` appelle automatiquement `mapcodes` avec l'option de codage `hproman8` puisque c'est ce qui correspond aux stations de travail HP que nous avons à l'école.

10.4 Bug dans les packages

Certains packages, lorsque nous les recevons, sont buggés. Il m'arrive, quand on me le fait remarquer ou lorsque je m'en rends compte tout seul en les utilisant, de trouver des corrections pour ces bugs. J'insère alors lesdites corrections dans le package `ESIEE`. Ce qui fait que si tu le charges après le package erroné, les erreurs peuvent être corrigées.

C'est pourquoi je te recommande de charger le package `ESIEE` en dernier, ainsi que de me tenir informé des bugs que te découvres dans LaTeX 2ε, je suis suceptible de les corriger, ou, tout au moins, de les signaler aux auteurs des packages.

Enfin, si tu souhaites expédier ton joli petit document à un pote, et que tu utilises le package `ESIEE`, il vaut mieux m'en causer parce qu'il a probablement pas ça sur sa bécane.

11 Gestion des langues — LaTeX polyglotte

La nouvelle version de LaTeX permet d'utiliser tout plein de langues sans se fouler trop. Plus de problèmes d'options tordues dans le choix du compilateur ou de trucs dans ce genre. Le compilateur LaTeX 2ε standard à l'ESIEE comprend déjà plusieurs langues.

Ainsi, j'ai, arbitrairement, décidé d'installer ce qui me paraissait intéressant, mais je suis près à en rajouter d'autres. Nous disposons donc en standard de : anglais (USA), français, portugais, allemand, italien, anglais (GB), et espéranto. Il parait que les règles de césure de l'anglais ne sont pas les mêmes aux USA et en Grande-Bretagne, ce qui explique la présence de deux langues différentes.

De nombreuses autres langues sont accessibles, mais pas de manière simple, ou alors incomplètement. Par exemple, on peut utiliser près de 20 langues à l'ESIEE mais sans les césures (voir tableau 3 page ci-contre).

Plusieurs méthodes existent pour exploiter les changements de langues sous LaTeX. Celle retenue à l'ESIEE pour le noyau LaTeX 2ε est connue sous le nom de `babel`. C'est la plus courante sur le plan international, mais l'une des plus contestée en France. On lui préfère souvent ce qui est connu sous le nom de «style `french`». Je ne l'ai pas retenu parce qu'il n'offre pas la même souplesse au niveau des changements de langue et de la reconnaissance au plan international. Ce choix est contestable, et a été contesté, principalement par Bernard GAULLE, auteur du «style `french`». Un travail de rapprochement entre ces deux options incompatibles semblait en cours la dernière fois que nous en avons discuté.

Donc, à retenir, à l'ESIEE on utilise `babel`, un point c'est tout. La question ne sera re-débattue que lorsque quelqu'un d'assez compétent pour refaire l'installation de l'école se présentera et que les deux styles différents seront devenus compatibles.

[18] Appuyer sur `Alt`, maintenir la touche enfoncée, puis taper `c`, puis relacher `Alt`

[19] Oui, deux fois le signe inférieur et deux fois le signe supérieur.

[20] La différence avec la méthode exposée plus haut est que ça, ça marche partout dans le monde. La première solution ne marche que sur stations HP, et peut-être même pas tous les modèles ...

Option	Sens
`iso8859-1`	Codage de la norme ISO N°8859 alinéa 1. Utilisé par de nombreux systèmes UNIX, Windows et l'amiga-OS.
`iso8859-2`	Codage de la norme ISO N°8859 alinéa 2. Comme son prédécesseur, mais pour les alphabets cyriliques. Peu usité dans nos contrées.
`latin1`	Synonyme de `iso8859-1`
`latin2`	Synonyme de `iso8859-2`
`ibm850`	Code `ascii` étendu IBM, page 850. Utilisé par MS-DOS.
`ibm852`	Code `ascii` étendu IBM, page 852.
`hproman8`	Tentative malheureuse de standardisation `roman` sur 8 bits. Encore en usage chez Hewlett-Packard.
`macroman`	Codage utilisé par le SYSTEM7 des Macintosh d'Apple.
`atari`	Comme son nom l'indique.

TABLEAU 2: Options disponibles pour le package `mapcodes`

Langue	**Option**	**Césure**
Allemand	`german` ou `germanb`	Oui
Anglais (USA)	`american`	Oui
Anglais (GB)	`english`	Oui
Autrichien	`austrian`	Oui
Brésilien	`brazil`	Non
Catalan	`catalan`	Non
Croate	`croatian`	Non
Danois	`danish`	Non
Espagnol	`spanish`	Non
Espéranto	`esperanto`	Oui
Finnois	`finnish`	Non
Français	`francais` ou `french`	Oui Oui
Galicien	`galician`	Non
Hongrois	`hungarian` ou `magyar`	Non Non
Italien	`italian`	Oui
Néerlandais	`dutsh`	Non
Polonais	`polish`	Non
Portugais	`portuges`	Oui
Roumain	`romanian`	Oui
Slovaque	`slovak`	Non
Slovène	`slovene`	Non
Suédois	`swedish`	Non
Tchèque	`czech`	Oui
Turque	`turkish`	Non

TABLEAU 3: Langues disponibles sous LaTeX 2_ε

11.1 Le Français

Pour l'utilisation des langues sous LaTeX 2ε tel qu'il est installé à l'ESIEE tu devras faire appel au package `babel`, que nous devons à Johannes Braams [9], avec comme option les langues que tu souhaites utiliser, par exemple :

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

Mais tu peux aussi faire comme moi, déclarer les langues[21] dans les options principales et ne pas les rappeler. D'autres packages que `babel` se posent la question de la langue.

Pour sélectionner la langue courante, il te faudra ensuite utiliser :

```
\selectlanguage{francais}
```

Les spécificités du français à l'ESIEE : tous les raffinements n'ont pas été installés à l'heure actuelle, mais c'est appelé à bouger rapidement (au rythme de la demande, probablement).

- Les espacements français sont gérés (une espace fine avant le point-virgule, et une espace pleine après, par exemple) ;
- il n'est plus utile de taper `\^{\i}` pour obtenir î, un simple `\^i` devrait suffire ;
- pour le tréma, c'est idem : `\"i` produira ï ;
- quant au ç il n'a pas changé ;
- la date est réadaptée automatiquement de telle sorte que l'habituel `\today` produise : 18 décembre 1995.

Un tas de petites choses ont été rajoutées, comme notre bien aimé degré : ° qui s'obtient par `\degre`.

Enfin des facilités pour nos numérotations étranges et assez incomprises des anglo-saxons et autres non-francophones. Il est du meilleur goût de pouvoir obtenir : 1° C'est un essai 2° en deuxième 3° en troisième 4° en quatrième.

J'ai pour cela utilisé `\primo`, `\secundo`, `\tertio`, `\quatro`. Et pour aller à 5 ? Facile ! Tu fais :

```
\FrenchEnumerate5,
\FrenchEnumerate{12345}.
```

pour obtenir : 5° , 12345° .

Enfin, et toujours dans la série des numérotations, il y a les 1°) , 2°) , 3°) et 4°) qui sont obtenus par `\fprimo)`, `\fsecundo)`, `\ftertio)` et `\fquatro)` et que l'on peut étendre par :

```
\FrenchPopularEnumerate5,
\FrenchPopularEnumerate{12345}
```

pour obtenir : 5°) , 12345°)

C'est quand même agréable. D'autres modifs «spécial ESIEE» sont étudiables, mais ne compte pas trop sur moi pour obtenir une installation en contradiction totale avec les standards et habitudes liés à LaTeX 2ε .

11.2 L'Allemand

Pour les germanistes, de grandes nouveautés, toutes plus belles les unes que les autres. Les tripartites seront heureux, ils peuvent enfin pondre des documents en anglais, français et allemand mélangés sans avoir trop de problèmes de typographie ou de césure.

L'histoire de l'implantation de l'Allemand dans LaTeX est assez chaotique (tout comme celle du français) ce qui fait que l'on arrive a des choses bizarres. Il semblerait que le plus fréquent dans les universités allemandes soit d'utiliser une installation spécifique datant en majeure partie de LaTeX 2.09 et qui était très poussée et très avancée. Cependant, mon but n'est pas d'obtenir que l'Allemand soit géré parfaitement par une version vieillissante de LaTeX à l'exclusion de toute autre langue. Aussi, je me suis restreint à ce que prévoit le package `babel`, que les allemands qui me lisent et qui connaissent mieux chez eux me le pardonnent.

L'allemand pose de gros problèmes à LaTeX pour de nombreuses raisons, en particulier pour la césure. En anglais, ou en français, quand on veut césurer un mot (mettons bonjour) on rajoute un trait d'union à l'emplacement de la césure (bon-jour), alors qu'en allemand, les lettres peuvent changer. Par exemple «ck» ne se césurera pas «c-k» mais «k-k», et «ff» qui viendra se césurer «ff-f». De plus les germanophones utilisent ce joli petit caractère : ß de manière assez courante, ainsi que l'accent tréma.

En allemand, sous LaTeX 2ε, le guillemet (ou double-quote) devient un caractère spécial, comme le backslash :

Le `"a` donne ä. Le `"s` donne ß, `"S` donne SS[22] pour être utilisé comme étant le caractère associé au précédant en majuscule. `"‘` et `"’` produiront „ et “. De même, en Allemand, des guillemets «à la française» sont accessibles à partir des commandes `"<` et `">` qui viennent produire « et ».

Les césures spéciales devront être indiquées au cas par cas à l'aide de ce même guillemet : `"ck` et `"ff`. Pas forcément évident à exploiter, enfin c'est ce qu'il me semble, à moi qui n'ai jamais fait d'allemand (et qui le regrette, d'ailleurs).

Pour sélectionner le style allemand, il faut compiler en chargeant le package `babel` avec l'option `german` ou `germanb`. Pourquoi les deux ? Parce qu'en Allemagne, `german` c'est le style datant de LaTeX 2.09 qui tend à se remettre à jour, et que `germanb` c'est le style lié à `babel`.

[21] Un document peut en effet être totalement polyglotte.

[22] En effet, il n'existe pas de ß en majuscule, or dans certains titres de chapitre ou de section, les allemands aiment à pouvoir utiliser ce joli caractère, mais ces titres sont reportés en haut des pages dans certains cas, et ils sont alors convertis en majuscule. C'est pour que cette conversion se passe bien que le `"S` a été créé.

11.3 Les autres langues

Les autres langues peuvent êtres appelées de façons similaires, comme étant des options du package `babel`. Les noms courants sont donnés par le compilateur sitôt qu'on le lance, en effet, il affiche :

```
Hyphenation patterns for english, francais,
portuges, german, italian, ukenglish,
esperanto, loaded.
```

Si tu demandes une langue autre et que le compilateur ne te jette pas comme un malpropre, c'est qu'il a compris et qu'il ne fera pas de césures dans ton texte puisqu'il ne maîtrise pas la langue.

12 Gestion des fontes

12.1 Comment changer de fonte

L'une des premières choses que l'on cherchera à faire avec un traitement de texte sera de changer de fonte pour mettre en évidence certains passages. Par exemple en mettant quelques mots en italique ou en gras. Le principe est simple. Pour obtenir **toto** on tape :

```
obtenir \textbf{toto} on ta...
```

On peut bien-évidemment passer plusieurs mots comme paramètre à la commande `\textbf`. Par contre on évitera de lui passer plusieurs paragraphes ou plusieurs pages, ça devient beaucoup plus difficile de compter les accolades et ça complique la vie du compilateur (il doit tout lire jusqu'à l'accolade fermante avant de commencer son travail).

D'autres changements que le gras peuvent être envisagés. Ils sont exhaustivement recensés dans le tableau 4 page suivante.

Certains changements de fontes nécessitent une explication. Par exemple, `\textmd` sert à repasser en normal lorsque l'on est en gras. Deux exemple pour bien comprendre :

```
\textbf{Mot en \textmd{textmd} dans du gras.}
\textit{Mot en \textmd{textmd} dans de l'italique.}
```

Et ça donne ces deux lignes là :

Mot en textmd **dans du gras.**
Mot en textmd dans de l'italique.

Note, au passage, que le choix de la graisse n'a pas d'influence sur l'italicisation.

De même `\textup` sert à «annuler» un `\textit`, un `\textsl` ou un `\textsc`.

`\emph` à un usage très particulier. Il met en évidence un passage dans son contexte. Dans un texte «normal» il passe en italique, dans un passage italique, il remet en «`textup`». C'est assez pratique : si on décide de passer tout un paragraphe en italique, les mots mis en évidence par ce moyen le restent.

12.2 Changements à longue portée

Lorsque l'on souhaite changer de fonte sur une grande portion de texte (plusieurs pages, voire tout le document), il est très désagréable de devoir se trimbaler des accolades partout. Il serait plus simple de dire «c'est du gras» et puis c'est tout.

Il existe des commandes pour cela, la liste fait l'objet du tableau 5 page suivante.

Écrire

```
{\itshape toto}
```

est équivalent[23] à écrire

```
\textit{toto}
```

et presque équivalent à

```
\itshape toto\upshape
```

En effet, la troisième forme ne ramène pas forcément à la fonte de départ, par exemple si l'on est déjà en italique au début, ou si l'on est en PETITES CAPITALES.

`\normalfont` (comme `\textnormal`) change à la fois la famille, la série, et la forme[24] pour te ramener à la fonte par défaut qui est, a priori, du romain (`\textrm`) en graisse moyenne (`\textmd`) et en forme droite (`\textup`). C'est la seule commande dans le lot à affecter plus d'un paramètre à la fois.

[23] à un microscopique détail près
[24] shape en anglais pour ceux qui comprennent pas bien le grand breton

`\textrm`	Normal (romain)
`\textbf`	**Gras** (bold face)
`\textit`	*Italique*
`\textsc`	PETITES CAPITALES (Small Caps)
`\textsf`	Sans sérifications
`\textsl`	*Penché* (Slanted)
`\texttt`	`Machine à écrire` (Tele Type)
`\textmd`	Graisse normale (Medium)
`\textup`	Droit
`\textnormal`	Fonte par défaut
`\emph`	Mise en *évidence*

TABLEAU 4: Changements de fontes standards

Changement de fonte	
à long terme	à court terme
`\rmfamily`	`\textrm`
`\sffamily`	`\textsf`
`\ttfamily`	`\texttt`
`\bfseries`	`\textbf`
`\mdseries`	`\textmd`
`\itshape`	`\textit`
`\slshape`	`\textsl`
`\scshape`	`\textsc`
`\upshape`	`\textup`
`\normalfont`	`\textnormal`
`\em`	`\emph`

TABLEAU 5: Changements de fontes

13 Gestion des tailles

Ça fonctionne (tout simplement) comme les changements de fonte «à longue portée» (section 12.2 page précédente). En effet, on ne change normalement pas de taille pour un mot en plein milieu d'une phrase.

Les commandes de changement de taille font l'objet du tableau 6 page ci-contre.

On pourra également envisager les changements de taille comme des «environnements», c'est à dire comme ceci :

```
\begin{small}
Passage du texte \'ecrit plus
petit que le reste.
\end{small}
```

Pour obtenir :

Passage du texte écrit plus petit que le reste.

14 Accents, caractères spéciaux et symboles

14.1 Les accents

Toi qui travaille sur ces superbes stations de travail HP, tu auras sans nul doute compris que les caractères accentués n'étaient pas directement accessibles au clavier, ce qui pose problème pour la saisie de textes, en français du moins.

De plus LaTeX a été prévu pour fonctionner sur tous les ordinateurs et dans toutes les configurations, donc il sait produire les accents s'ils n'existent pas sur la machine où il tourne. Il y a des commandes exprès pour cela. Elle sont dans le tableau 7 page suivante.

Note qu'elles fonctionnent de manière très générale, c'est à dire que R-cédille (Ȑ) s'obtient le plus naturellement du monde : `\c{R}`.

14.2 Caractères non-américains

La plupart des caractères non-américains utilisés dans les langues européennes ont été prévus. Les commandes permettant de les obtenir font l'objet du tableau 8 page ci-contre.

\tiny	Exemple
\scriptsize	Exemple
\footnotesize	Exemple
\small	Exemple
\normalsize	Exemple
\large	Exemple
\Large	Exemple
\LARGE	Exemple
\huge	Exemple

Tableau 6: Changements de taille

á	\'a
à	\`a
â	\^a
ä	\"a

ã	\~a
ā	\=a
ȧ	\.a
a̋	\H{a}

ă	\u{a}
ǎ	\v{a}
a͡	\t{a}
ạ	\d{a}

a̲	\b{a}
ç	\c{c}
a̧	\c{a}

Tableau 7: Accents en mode texte

œ	\oe
Œ	\OE
æ	\ae
Æ	\AE
ß	\ss
SS	\SS
å	\aa
Å	\AA
«	<<

ð	\dh
Ð	\DH
đ	\dj
Đ	\DJ
ŋ	\ng
Ŋ	\NG
þ	\th
Þ	\TH
»	>>

ø	\o
Ø	\O
ł	\l
Ł	\L
ı	\i
ȷ	\j
¿	\textquestiondown
¡	\textexclamdown
£	\pounds

«	\guillemotleft
»	\guillemotright
‹	\guilsinglleft
›	\guilsinglright
‚	\quotesinglbase
„	\quotedblbase
“	\textquotedblleft
”	\textquotedblright

Tableau 8: Caractères non-américains

14.3 Caractères spéciaux

Certains caractères sont interprétés de manière spéciale par LaTeX, comme tu t'en seras sûrement rendu compte.

On relèvera :

&	utilisé dans les tableaux ;
~	espace inseccable entre deux mots ;
%	commentaire jusqu'à la fin de la ligne ;
\	début d'un nom de commande ;
_	indice en mode mathématiques ;
$	début ou fin de mode mathématiques ;
{}	délimiteurs.

Pour les obtenir dans du texte, il n'y a que deux solutions. La première est celle que j'utilise pour te montrer les noms de commandes comme `\texttt` : `\verb"\texttt"`. Le premier caractère non lettre qui suit cette commande sert de délimiteur pour marquer la zone sur laquelle elle s'applique. Cette commande indique simplement à LaTeX qu'il ne doit pas traiter la zone de texte concernée.

La méthode plus élégante et moins bourrine est d'utiliser les commandes du tableaux 9 page suivante. En effet, écrire 12% c'est plus joli que 12`%` même si ça fait une touche de plus à taper.

14.4 Ponctuation

Le français a des règles d'espacement de la ponctuation[25] qui sont assez strictes [84] et que très peu de traitements de texte respectent. Par exemple, avant un point d'exclamation, il faut un espace fine insécable. LaTeX respectera ces règles, pour peu que tu aies indiqué que ton document est en français, dans deux cas :

- si tu mets un espace avant les signes de ponctuation qui en ont besoin (! ? ; :), LaTeX le mettra à la bonne taille et le rendra insécable si nécessaire.
- si tu oublies, mais que tu utilises le package `ESIEE`.

Notons au passage que les points de suspension (. . .) s'obtiennent à l'aide de la commande `\ldots`, et pas autrement. En particulier pas en tapant trois points. Ceci pour l'espacement soit le bon (. . .) au lieu de (...). Précisons clairement que les points de suspension (ou ellipse) sont au nombre de trois. Jamais moins, en aucun cas plus, comme le précise d'ailleurs [84].

14.5 Paragraphe

Contrairement aux ignominies que j'ai entendues proférer des centaines de fois à l'ESIEE, il n'y a qu'une seule et unique manière d'indiquer la fin d'un paragraphe : laisser une ligne blanche dans le fichier source.

On peut régler facilement l'alinéa en donnant une nouvelle valeur à la variable `\parindent` :

```
\parindent=15pt
```

La typographie française tolère de 0,5 à 2 fois le corps de la fonte en cours d'utilisation, en préférant une valeur entre 1 et 1,5 fois sa taille [84], c'est à dire

```
\parindent=15pt
```

pour un document en 12pt. Note au passage que le corps par défaut sous LaTeX est 10pt (le plus courant en typographie) et que l'on peut le passer à 11pt ou 12pt en spécifiant `11pt` ou `12pt` en option au `\documentclass`.

De plus il existe une unité spéciale correspondant pile poil au corps en cours d'utilisation, c'est `em`. On pourra donc faire :

```
\parindent=1.25 em
```

15 Constructions simples

15.1 Environnements

Tu as déjà vu certains environnements à la section 13 page 22, ceux qui sont utilisés pour les changements de taille.

Eh bien, il en existe plein d'autres ! Les plus courrants sont ceux permettant d'écrire du texte centré (`center`), au fer à droite (`flushright`), ou au fer à gauche (`flushleft`). Par exemple :

```
\begin{flushleft}
Ceci est un paragraphe d'exemple de
texte mis au fer \'a gauche avec
l'environnement \texttt{flushleft}.
\end{flushleft}
```

produira :

Ceci est un paragraphe d'exemple de texte mis au fer à gauche avec l'environnement `flushleft`.

De même que pour les changements de taille il existe une autre commande pour faire la même chose, ces trois environnements là ont des commandes équivalentes : `\centering` pour centrer du texte, `\raggedright` pour mettre au fer à gauche (si, si, à gauche) et `\raggedleft` pour mettre au fer à droite. En effet, pour l'exemple ci-dessus il eut été équivalent d'écrire :

```
{\raggedright Ceci est\ldots}
```

[25] L'anglais aussi, mais c'est pas les mêmes.

&	`\&`		%	`\%`		_	`\_`		{	`\{`
~	`$\sim$`		\	`$\backslash$`		$	`\$`		}	`\}`

TABLEAU 9: Caractères spéciaux

la preuve, ça produit :
Ceci est ...

On s'en servira, par exemple, dans les tableaux ; comme c'est le cas dans la section 25.2 page 62.

15.2 Les listes

Trois autres environnements sont bons à retenir, ceux qui permettent d'obtenir des listes : `itemize` pour des listes normales, `enumerate` pour des listes numérotées, et `description` pour des listes type dictionnaire. Un exemple vaut toujours mieux qu'un long discours :

```
\begin{itemize}
\item Premier truc \`a savoir
\item Second truc \`a savoir
\item Troisi\`eme truc \`a savoir
\end{itemize}
```

produira :

- Premier truc à savoir
- Second truc à savoir
- Troisième truc à savoir

```
\begin{enumerate}
\item En premier lieu;
\item en second lieu;
\item en dernier lieu.
\end{enumerate}
```

produira

1. En premier lieu ;
2. en second lieu ;
3. en dernier lieu.

Et enfin

```
\begin{description}
\item[itemize] liste normale
\item[enumerate] liste num\'erot\'ee
\item[description] liste comme celle-ci.
\end{description}
```

produira :

itemize liste normale

enumerate liste numérotée

description liste comme celle-ci.

15.3 Tableaux faciles

Obtenir un tableau n'est pas une chose enfantine avec LaTeX. Mais, tu verras, c'est tout à fait faisable.

La première chose à retenir est qu'un tableau est un environnement `tabular` qui prends comme argument le type des différentes colonnes qui doivent apparaître dans le tableau. À l'intérieur de ce tableau, le texte des différentes colonnes est délimité par le caractère `&` et celui des différentes lignes par la commande \\.

Les trois types de colonnes les plus classiques sont :

l colonne de type gauche (*left*)
r colonne de type droite (*right*)
c colonne de type centré (*center*)

Voyons tout de suite un premier exemple simple :

```
\begin{tabular}{lr}
Premier mot & Second mot \\
Troisi\`eme mot & Quatri\`eme mot
\end{tabular}
```

produira

Premier mot	Second mot
Troisième mot	Quatrième mot

La largeur des colonnes sera automatiquement adaptée à la largeur du texte qu'elles contiennent. Cela peut parfois poser problème, par exemple lorsque le texte à mettre dans une case est un peu long. Pour régler ce problème précis, LaTeX prévoit un type de colonnes exprès, le type `p`, auquel on indique la largeur de la colonne à créer. On verra un exemple tout à l'heure.

Le prochain problème à régler est de mettre des filets[26] entre les lignes et entre les colonnes du tableau. Deux choses à retenir à cet effet : un `|` entre deux descripteurs de colonne insérera un filet entre deux colonnes, et un `\hline` entre deux lignes les séparera par un filet.

Voyons maintenant l'exemple promis :

```
\begin{center}
\begin{tabular}{|l|p{4cm}|}
\hline
Descripteur & Sens \\
\hline
\texttt{l}  & Colonne de type gauche \\
\hline
\texttt{c}  & Colonne de type centr\'e \\
\hline
\texttt{r}  & Colonne de type droite \\
\hline
```

[26] traits

```
\texttt{p}  & Colonne d'une largeur
donn\'ee \`a l'avance. Plusieurs unit\'es
peuvent \^etre utilis\'ees pour
sp\'ecifier des distances \`a \LaTeX{}:
\texttt{cm} (centim\`etre), \texttt{mm}
(millim\`etre), \texttt{in} (pouce),
\texttt{pt} (point typographique anglais,
72,27\texttt{pt}=1\texttt{in})\ldots
\\ \hline
\end{tabular}
\end{center}
```

produira :

Descripteur	Sens
`l`	Colonne de type gauche
`c`	Colonne de type centré
`r`	Colonne de type droite
`p`	Colonne d'une largeur donnée à l'avance. Plusieurs unités peuvent être utilisées pour spécifier des distances à LaTeX : `cm` (centimètre), `mm` (millimètre), `in` (pouce), `pt` (point typographique anglais, 72,27`pt`=1`in`) ...

Tu vois bien, c'était pas si terrible que ça, les tableaux. Cependant, un grand nombre de raffinements sont possibles, pour en savoir plus, reporte toi à la section 25 page 62.

16 Chapitres et table des matières

16.1 Le titre

La première chose que tu vas taper dans ton document, c'est son titre. Donc LaTeX a prévu des commandes pour cela. Elles viennent généralement juste après

```
\begin{document}
```

Un exemple suffira :

```
\title{Reconnaissance de phon\`emes par
cartes de Kohonen sans apprentissage
supervis\'e}
\author{Benjamin \textsc{Bayart}
\\ I$_4$PASTI \and Pascal \textsc{Vincent}
\\ I$_4$PASTI}
\date{\today}
\maketitle
```

Selon que tu écris un `report` ou un `article`, LaTeX décidera s'il est nécessaire ou non de réserver une page entière au titre.

16.2 Division du document

Un document classique se divise en plusieurs parties, chapitres, sections, etc. Chacun a un numéro qui lui est propre et une mise en page qui doit être toujours la même au long du document si l'on souhaite rester cohérent. C'est pour ces raisons que LaTeX prévoit des commandes. Elles sont recensées dans le tableau 10 page suivante.

Par exemple :

```
\chapter{Cahiers des charges}
```

Commence un nouveau chapitre en le numérotant convenablement et en lui donnant le titre «Cahiers des charges».

Notons au passage que le niveau de division[27] `\chapter` existe pour les rapports et pas pour les articles. Notons également que les chapitres sont numérotés continuement au travers des parties. C'est à dire que la première partie contiendra les chapitres 1 à i et la seconde les chapitres $i+1$ à n.

16.3 Chapitre d'introduction

L'introduction, dans un rapport, tout comme la conclusion d'ailleurs, est souvent un problème. En effet, elle a valeur de chapitre, mais généralement on ne la numérote pas.

Il te suffit de savoir que pour toutes les commandes de sectionnement, en leur ajoutant une `*` elles viennent créer des parties, sections ... non-numérotées pour croire qu'un simple

```
\chapter*{Introduction}
```

est la solution idéale. Malheureusement tel n'est pas le cas. En effet, un `\chapter*` n'est pas recensé en table des matières et posera, pour ce cas précis et pour certaines mises en page, des problèmes.

La meilleure solution est d'utiliser pour cela la commande `\introchapter` qui est définie dans le package ESIEE.

```
\introchapter{Introduction}
```

Sinon sache simplement que cette commande est équivalente à

```
\chapter*{Introduction}
\addcontentsline{toc}{chapter}{Introduction}
\markboth{INTRODUCTION}{INTRODUCTION}
```

et ne pose surtout pas de question là-dessus sans avoir lu avant [61] et [81].

[27] Ou niveau de sectionnement

Commande	Niveau de sectionnement
`\part`	1
`\chapter`	2
`\section`	3
`\subsection`	4
`\subsubsection`	5
`\paragraph`	6
`\subparagraph`	7

Tableau 10: Commandes de sectionnement

Table des matières	`\tableofcontents`
Liste des tableaux	`\listoftables`
Liste des figures	`\listoffigures`

Tableau 11: Déclarations des trois types de tables les plus courants

16.4 Tables diverses

Comme je te sais impatient de savoir comment on produit une table des matières, je vais te donner la bonne méthode pour en faire une. En premier lieu, sache qu'il existe plusieurs sortes de tables, et donc plusieurs façons de les produire. Les trois principales font l'objet du tableau 11.

Pour obtenir une table n'importe où dans le document, tu insères la commande correspondante. Tu peux donc sans difficulté aucune faire apparaître plusieurs fois la table des matières, si cela te fait plaisir.

Il est à noter que dans un «article» les trois tables figurent sur la même page que le texte qui les suit, alors que pour un «report» LaTeX change de page à la fin de chaque table.

17 Inclusion de fichiers

Toi, lecteur programmeur, tu as l'habitude d'écrire tes programmes dans plusieurs fichiers, avec un système d'inclusion : un fichier principal qui fait appel à plusieurs petits fichiers annexes. C'est plus facile à utiliser pour des programmes, il en serait sûrement de même pour un document LaTeX, non ? Mais oui. Tu n'as qu'à le faire.

Pour cet usage LaTeX, qui est grand, prévoit deux commandes : `\input` et `\include`. Leur usage est expliqué ci-dessous.

17.1 Inclusion simple

`\input` permet d'importer purement et simplement un fichier : `\input toto` agira exactement comme si le contenu de `toto.tex` était dans ton texte.

Cela permet de se créer ses propres bibliothèques de macros.

17.2 Compilation partielle

`\include` marche différemment. Tout fichier importé par lui doit contenir des chapitres complets. Il créera un fichier `.aux` par fichier inclus. Il permet une compilation partielle des documents quand il est utilisé avec `\includeonly`.

Exemple :

```
\include{intro}
\include{chap1}
\include{chap2}
\include{chap3}
\include{chap4}
\include{concl}
```

lira les 6 fichiers comme il faut, puis si tu ajoutes AVANT

```
\includeonly{intro.tex,concl.tex}
```

LaTeX ne compilera que l'introduction et la conclusion, et en gardant les bons numéros de page, en conservant une table des matières complète et tout comme il faut. Toutefois, il faudra qu'au moins une compilation complète ait eu lieu pour qu'il puisse se repérer.

De plus aucun `\include` ne doit apparaître avant le `\begin{document}` et dans les fichiers lus avec `\include`.

Mathématiques

18 Principe, environnements, généralités

18.1 Principe

Comme tu le sais déjà, LaTeX est étudié particulièrement pour la mise en page de textes scientifiques. Il est donc particulièrement étudié pour les équations.

Le principe de base de la saisie de mathématiques sous LaTeX est simple : spécifie-lui que tu passes en mode maths, indique-lui ton équation, indique que tu ressors du mode maths.

La notion de mode maths est importante parce que les conventions typographiques (entre autres) sont différentes de la normale.

18.2 Les environnements

Pour passer en mode maths, il y a six méthodes :

- L'environnement `displaymath`
- L'environnement `math`
- `\[...\]`, équivalent à `displaymath`
- `\(...\)`, équivalent à `math`
- `$$...$$`, équivalent à `displaymath`
- `$...$`, équivalent à `math`

La grande différence entre «displaymath» et «math» est que le premier est prévu pour faire apparaître une équation seule centrée sur une ligne, alors que le second est prévu pour mettre une petite équation dans le texte.

On préferera utiliser, par soucis de lisibilité du fichier `.tex` :

- les deux environnements pour tous les passages longs (5 à 6 lignes, voire plus, dans le fichier source)
- `\[...\]` plutôt que `$$...$$` parce que l'on voit clairement si on ouvre ou si l'on ferme le mode mathématique.
- `$...$` pour citer un nom de variable ou un morceau très court (`$x$` pour x ou `$f(x)$` pour $f(x)$).
- `\(...\)` pour tout passage un peu long (une demi-ligne, par exemple) dans un paragraphe.

De plus, si tu souhaites que les équations «display»[28] soient non pas centrées, mais au fer à gauche, alors indique l'option `fleqn` à `\documentclass`. Tu pourras alors faire

```
\mathindent=1cm
```

pour que ces équations se retrouvent à 1cm de la marge gauche.

18.3 Généralités

Les trucs de base à savoir pour taper la majorité des équations sont peu nombreux et assez simples à retenir. Le premier est que LaTeX tape spontanément les maths en italique, ainsi en disant "on a a qui vaut 12", le deuxième 'a' est en italique, on sait que c'est une variable. C'est plus facile à lire. C'est pour ça qu'on peut passer en mode maths facilement, pour le 'a' j'avais tapé `$a$`. Facile, non ?

Le second truc à savoir est le problème des exposants et des indices. Pour les indices, le caractère magique est `_`, et pour les exposants, c'est `^`. Exemple : `$x_0^2$` donnera x_0^2. Pour mettre plus d'une lettre en indice ou en exposant, il faut faire `$x^{12}$` : x^{12}.

19 Symboles mathématiques

On peut, grossièrement, les regrouper en cinq grandes catégories :

1) Les symboles que LaTeX2e reconnait spontanément.
2) Les symboles que reconnaissait LaTeX 2.09 et que LaTeX 2_ε ne reconnaît plus tout seul[29].
3) Les symboles accessibles par le package `amsmath`, ou plus exactement son sous-ensemble `amssymb`.
4) Les symboles ajoutés par `stmaryrd`.
5) Les raccourcis claviers et extensions offerts par `qsymbols`.

---

[28] Je préfère le mot français de «hors-ligne»

[29] Il s'agit de symboles peu nombreux qui monopolisent une fonte à eux seuls, ce qui gaspille de la place en mémoire si on n'en a pas besoin.

19.1 Package `latexsym`

Le tableau 12 te présente les symboles définis dans le package **`latexsym`**. Ton observation attentive et minutieuse t'amènera très vite à constater que tous ces symboles sont aussi définis dans le package **`amssymb`** qui est un sous-ensemble de **`amsmath`**, mais c'est pas utile de déplacer un mammouth pour écraser une mouche, ou alors si mais une grosse mouche.

19.2 Symboles LaTeX 2_ε

LaTeX 2_ε prévoit en standard à peu près les mêmes symboles que LaTeX 2.09 le faisait. Mais tu as été élevé avec une version que j'avais artificiellement étendue pour qu'elle puisse produire beaucoup plus de symboles. Tous les symboles qui sont dans les tableaux 13 à 22 page suivante ne nécessitent pas de package d'extension.

`\mho`	$\mho$	`\Join`	$\Join$	`\Box`	$\Box$	`\Diamond`	$\Diamond$
`\lhd`	$\lhd$	`\unlhd`	$\unlhd$	`\rhd`	$\rhd$	`\unrhd`	$\unrhd$
`\sqsubset`	$\sqsubset$	`\sqsupset`	$\sqsupset$	`\leadsto`	$\leadsto$		

TABLEAU 12: Symboles définis par **`latexsym`**

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

TABLEAU 13: Lettres Grecques

`\pm`	$\pm$	`\mp`	$\mp$	`\times`	$\times$	`\div`	$\div$
`\ast`	$\ast$	`\star`	$\star$	`\circ`	$\circ$	`\bullet`	$\bullet$
`\cdot`	$\cdot$	`\cap`	$\cap$	`\cup`	$\cup$	`\uplus`	$\uplus$
`\oplus`	$\oplus$	`\ominus`	$\ominus$	`\otimes`	$\otimes$	`\oslash`	$\oslash$
`\bircirc`		`\dagger`	$\dagger$	`\ddagger`	$\ddagger$	`\amalg`	$\amalg$
`\sqcap`	$\sqcap$	`\sqcup`	$\sqcup$	`\bigtriangleup`	$\bigtriangleup$		
`\vee`	$\vee$	`\wedge`	$\wedge$	`\bigtriangledown`	$\bigtriangledown$		
`\setminus`	$\setminus$	`\wr`	$\wr$	`\triangleleft`	$\triangleleft$		
`\diamond`	$\diamond$	`\odot`	$\odot$	`\triangleright`	$\triangleright$		

TABLEAU 14: Opérateurs binaires

`\leq`	$\leq$	`\le`	$\le$	`\geq`	$\geq$		
`\equiv`	$\equiv$	`\models`	$\models$	`\prec`	$\prec$	`\succ`	$\succ$
`\sim`	$\sim$	`\perp`	$\perp$	`\preceq`	$\preceq$	`\succeq`	$\succeq$
`\simeq`	$\simeq$	`\mid`	$\mid$	`\ll`	$\ll$	`\gg`	$\gg$
`\subset`	$\subset$	`\supset`	$\supset$	`\subseteq`	$\subseteq$	`\supseteq`	$\supseteq$
`\sqsubset`	$\sqsubset$	`\sqsupset`	$\sqsupset$	`\sqsubseteq`	$\sqsubseteq$	`\sqsupseteq`	$\sqsupseteq$
`\parallel`	$\parallel$	`\approx`	$\approx$	`\bowtie`	$\bowtie$	`\Join`	$\Join$
`\neq`	$\neq$	`\doteq`	$\doteq$	`\cong`	$\cong$	`\approx`	$\approx$
`\in`	$\in$	`\ni`	$\ni$	`\smile`	$\smile$	`\frown`	$\frown$
`\vdash`	$\vdash$	`\dashv`	$\dashv$			`\asymp`	$\asymp$

TABLEAU 15: Symboles de relation

`\leftrightarrow`	$\leftrightarrow$	`\leftarrow`	$\leftarrow$	`\rightarrow`	$\rightarrow$
`\longleftrightarrow`	$\longleftrightarrow$	`\uparrow`	$\uparrow$	`\downarrow`	$\downarrow$
`\Leftrightarrow`	$\Leftrightarrow$	`\Leftarrow`	$\Leftarrow$	`\Rightarrow`	$\Rightarrow$
`\Longleftrightarrow`	$\Longleftrightarrow$	`\Uparrow`	$\Uparrow$	`\Downarrow`	$\Downarrow$
`\longleftarrow`	$\longleftarrow$	`\mapsto`	$\mapsto$	`\longmapsto`	$\longmapsto$
`\longrightarrow`	$\longrightarrow$	`\nearrow`	$\nearrow$	`\searrow`	$\searrow$
`\Longleftarrow`	$\Longleftarrow$	`\nwarrow`	$\nwarrow$	`\swarrow`	$\swarrow$
`\Longrightarrow`	$\Longrightarrow$				
`\Updownarrow`	$\Updownarrow$	`\updownarrow`	$\updownarrow$		
`\leftharpoonup`	$\leftharpoonup$	`\rightharpoonup`	$\rightharpoonup$		
`\leftharpoondown`	$\leftharpoondown$	`\rightharpoondown`	$\rightharpoondown$		
`\hookleftarrow`	$\hookleftarrow$	`\hookrightarrow`	$\hookrightarrow$		

TABLEAU 16: Flèches

`\ldots`	$\ldots$	`\cdots`	$\cdots$	`\vdots`	$\vdots$	`\ddots`	$\ddots$
`\forall`	$\forall$	`\exists`	$\exists$	`\imath`	$\imath$	`\jmath`	$\jmath$
`\nabla`	∇	`\triangle`	$\triangle$	`\prime`	$\prime$	`\infty`	∞
`\hbar`	$\hbar$	`\emptyset`	$\emptyset$	`\natural`	$\natural$	`\flat`	$\flat$
`\sharp`	$\sharp$	`\neg`	$\neg$	`\wp`	$\wp$	`\partial`	∂
`\clubsuit`	$\clubsuit$	`\diamondsuit`	$\diamondsuit$	`\heartsuit`	$\heartsuit$	`\spadesuit`	$\spadesuit$
`\angle`	$\angle$	`\surd`	$\surd$	`\top`	$\top$	`\bot`	$\bot$
`\Box`	$\Box$	`\ell`	ℓ	`\aleph`	$\aleph$	`\Diamond`	$\Diamond$
`\Re`	$\Re$	`\Im`	$\Im$				

Tableau 17: Divers symboles

`\sum`	$\sum$	`\prod`	$\prod$	`\coprod`	$\coprod$	`\int`	$\int$
`\bigcap`	$\bigcap$	`\bigcup`	$\bigcup$	`\bigsqcup`	$\bigsqcup$	`\oint`	$\oint$
`\bigodot`	$\bigodot$	`\bigotimes`	$\bigotimes$	`\bigoplus`	$\bigoplus$	`\biguplus`	$\biguplus$
`\bigvee`	$\bigvee$	`\bigwedge`	$\bigwedge$				

Tableau 18: Symboles à taille variable

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

Tableau 19: Noms de ¡¡fonctions¿¿ (log-like)

`\uparrow`	$\uparrow$	`\Uparrow`	$\Uparrow$	`\downarrow`	$\downarrow$	`\Downarrow`	$\Downarrow$
`\{`	$\{$	`\}`	$\}$	`\updownarrow`	$\updownarrow$	`\Updownarrow`	$\Updownarrow$
`\lfloor`	$\lfloor$	`\rfloor`	$\rfloor$	`\lceil`	$\lceil$	`\rceil`	$\rceil$
`\langle`	$\langle$	`\rangle`	$\rangle$	`/`	$/$	`\backslash`	$\backslash$
`|`	$\vert$	`\\|`	$\Vert$				

Tableau 20: Délimiteurs

`\rmoustache`	$\rmoustache$	`\lmoustache`	$\lmoustache$	`\rgroup`	$\rgroup$	`\lgroup`	$\lgroup$
`\arrowvert`	$\arrowvert$	`\Arrowvert`	$\Arrowvert$	`\bracevert`	$\bracevert$		

Tableau 21: Grands délimiteurs

`\widetilde{abc}`	$\widetilde{abc}$	`\widehat{abc}`	$\widehat{abc}$
`\overleftarrow{abc}`	$\overleftarrow{abc}$	`\overrightarrow{abc}`	$\overrightarrow{abc}$
`\overline{abc}`	$\overline{abc}$	`\underline{abc}`	$\underline{abc}$
`\overbrace{abc}`	$\overbrace{abc}$	`\underbrace{abc}`	$\underbrace{abc}$
`\sqrt{abc}`	$\sqrt{abc}$	`\sqrt[n]{abc}`	$\sqrt[n]{abc}$
`f'`	f'	`\frac{abc}{xyz}`	$\frac{abc}{xyz}$

Tableau 22: Constructions mathématiques

19.3 Le package amssymb

Par contre ceux des tableaux 23 à 28 page suivante nécessitent d'utiliser soit le package amsmath au complet, soit son sous-ensemble gèrant seulement les symboles, amssymb.

\dashleftarrows		\dashrightarrows	
\leftleftarrows	$\leftleftarrows$	\rightrightarrows	$\rightrightarrows$
\rightleftarrows	$\rightleftarrows$	\leftrightarrows	$\leftrightarrows$
\upuparrows	$\upuparrows$	\downdownarrows	$\downdownarrows$
\twoheadleftarrow	$\twoheadleftarrow$	\twoheadrightarrow	$\twoheadrightarrow$
\Lleftarrow	$\Lleftarrow$	\Rrightarrow	$\Rrightarrow$
\leftarrowtail	$\leftarrowtail$	\rightarrowtail	$\rightarrowtail$
\curvearrowleft	$\curvearrowleft$	\curvearrowright	$\curvearrowright$
\circlearrowleft	$\circlearrowleft$	\circlearrowright	$\circlearrowright$
\loopharpoonleft		\loopharpoonright	
\Lsh	$\Lsh$	\Rsh	$\Rsh$
\leftsquigarrow		\rightsquigarrow	$\rightsquigarrow$
\upharpoonleft	$\upharpoonleft$	\upharpoonright	$\upharpoonright$
\downharpoonleft	$\downharpoonleft$	\downharpoonright	$\downharpoonright$
\leftrightharpoon		\rightleftharpoon	
\multimap	$\multimap$	\leftrightsquigarrow	$\leftrightsquigarrow$

Tableau 23: Flèches (ajout amsmath)

\nleftarrow	$\nleftarrow$	\nrightarrow	$\nrightarrow$
\nLeftarrow	$\nLeftarrow$	\nRightarrow	$\nRightarrow$
\nleftrightarrow	$\nleftrightarrow$	\nLeftrightarrow	$\nLeftrightarrow$

Tableau 24: Flèches négatives (ajout amsmath)

\leqq	$\leqq$	\leqslant	$\leqslant$	\eqslantless	$\eqslantless$	\lll	$\lll$
\geqq	$\geqq$	\geqslant	$\geqslant$	\eqslantgtr	$\eqslantgtr$	\ggg	$\ggg$
\lesssim	$\lesssim$	\lessapprox	$\lessapprox$	\lessdot	$\lessdot$	\lesseqgtr	$\lesseqgtr$
\gtrsim	$\gtrsim$	\gtrapprox	$\gtrapprox$	\gtrdot	$\gtrdot$	\gtreqless	$\gtreqless$
\lesseqqgtr	$\lesseqqgtr$	\gtreqqless	$\gtreqqless$	\risingdotseq	$\risingdotseq$	\fallingdotseq	$\fallingdotseq$
\subseteqq	$\subseteqq$	\supseteqq	$\supseteqq$	\Subset	$\Subset$	\Supset	$\Supset$
\sqsubset	$\sqsubset$	\sqsupset	$\sqsupset$	\trianglelefteq	$\trianglelefteq$	\trianglerighteq	$\trianglerighteq$
\vartriangleleft	$\vartriangleleft$	\vartriangleright	$\vartriangleright$	\curlyeqprec	$\curlyeqprec$	\curlyeqsucc	$\curlyeqsucc$
\precsim	$\precsim$	\succsim	$\succsim$	\precapprox	$\precapprox$	\succapprox	$\succapprox$
\vDash	$\vDash$	\Vvdash	$\Vvdash$	\Vdash	$\Vdash$	\between	$\between$
\smallsmile	$\smallsmile$	\smallfrown	$\smallfrown$	\shortmid	$\shortmid$	\shortparallel	$\shortparallel$
\bumpeq	$\bumpeq$	\Bumpeq	$\Bumpeq$	\eqcirc	$\eqcirc$	\circeq	$\circeq$
\triangleq	$\triangleq$	\thicksim	$\thicksim$	\thickapprox	$\thickapprox$	\pitchfork	$\pitchfork$
\blacktriangleleft	$\blacktriangleleft$	\blacktriangleright	$\blacktriangleright$	\therefore	$\therefore$	\because	$\because$
\varpropto	$\varpropto$	\backepsilon	$\backepsilon$	\backsim	$\backsim$	\backsimeq	$\backsimeq$
\preccurlyeq	$\preccurlyeq$	\succcurlyeq	$\succcurlyeq$				

Tableau 25: Symboles de relation (ajout amsmath)

\nless	$\nless$	\nleq	$\nleq$	\nleqslant	$\nleqslant$	\nleqq	$\nleqq$
\ngtr	$\ngtr$	\ngeq	$\ngeq$	\ngeqslant	$\ngeqslant$	\ngeqq	$\ngeqq$
\lneq	$\lneq$	\lneqq	$\lneqq$	\lvertneqq	$\lvertneqq$	\lnsim	$\lnsim$
\gneq	$\gneq$	\gneqq	$\gneqq$	\gvertneqq	$\gvertneqq$	\gnsim	$\gnsim$
\lnapprox	$\lnapprox$	\nprec	$\nprec$	\npreceq	$\npreceq$	\precnsim	$\precnsim$
\gnapprox	$\gnapprox$	\nsucc	$\nsucc$	\nsucceq	$\nsucceq$	\succnsim	$\succnsim$
\precnapprox	$\precnapprox$	\nsim	$\nsim$	\nshortmid	$\nshortmid$	\nmid	$\nmid$
\succnapprox	$\succnapprox$	\napprox		\nparallel	$\nparallel$	\nshortparallel	$\nshortparallel$
\nvdash	$\nvdash$	\nvDash	$\nvDash$	\ntriangleleft	$\ntriangleleft$	\ntrianglelefteq	$\ntrianglelefteq$
\nvDash	$\nvDash$	\nVDash	$\nVDash$	\ntriangleright	$\ntriangleright$	\ntrianglerighteq	$\ntrianglerighteq$
\nsubseteq	$\nsubseteq$	\subsetneq	$\subsetneq$	\varsubsetneq	$\varsubsetneq$	\subsetneqq	$\subsetneqq$
\nsupseteq	$\nsupseteq$	\supsetneq	$\supsetneq$	\varsupsetneq	$\varsupsetneq$	\supsetneqq	$\supsetneqq$
\varsubsetneqq	$\varsubsetneqq$	\nsubseteqq	$\nsubseteqq$				
\varsupsetneqq	$\varsupsetneqq$	\nsupseteqq	$\nsupseteqq$				

Tableau 26: Symboles de relation négatifs (ajout amsmath)

\dotplus	$\dotplus$	\smallsetminus	$\smallsetminus$	\Cap	$\Cap$	\Cup	$\Cup$
\barwedge	$\barwedge$	\doublebarwedge	$\doublebarwedge$	\veebar	$\veebar$	\boxminus	$\boxminus$
\boxtimes	$\boxtimes$	\boxdot	$\boxdot$	\boxplus	$\boxplus$	\divideontimes	$\divideontimes$
\ltimes	$\ltimes$	\rtimes	$\rtimes$	\leftthreetimes	$\leftthreetimes$	\rightthreetimes	$\rightthreetimes$
\curlywedge	$\curlywedge$	\curlyvee	$\curlyvee$	\circleddash	$\circleddash$	\circledast	$\circledast$
\circledcirc	$\circledcirc$	\centerdot	$\centerdot$	\intercal	$\intercal$		

Tableau 27: Opérateurs binaires (ajout amsmath)

`\hbar`	$\hbar$	`\hslash`	$\hslash$	`\vartriangle`	$\vartriangle$	`\triangledown`	$\triangledown$
`\square`	$\square$	`\lozenge`	$\lozenge$	`\circledS`	$\circledS$	`\angle`	$\angle$
`\measuredangle`	$\measuredangle$	`\nexists`	$\nexists$	`\mho`	$\mho$	`\Finv`	$\Finv$
`\Game`	$\Game$	`\Bbbk`	$\Bbbk$	`\backprime`	$\backprime$	`\varnothing`	$\varnothing$
`\blacktriangle`		`\bigstar`	$\bigstar$	`\eth`	$\eth$		
`\blacktriangledown`	$\blacktriangledown$	`\blacksquare`	$\blacksquare$	`\blacklozenge`	$\blacklozenge$	`\diagdown`	$\diagdown$
`\sphericalangle`	$\sphericalangle$	`\complement`	$\complement$	`\diagup`	$\diagup$		

TABLEAU 28: Divers symboles (ajout **amsmath**)

19.4 Le package stmaryrd

Ce package, écrit par Jeremy Gibbons et Alan Jeffrey [41] donne accès aux symboles de leur propre fonte, celle qu'ils ont appelée «St Mary's Road symbol font[30]».

Ce package définit une tripotée de symboles, en fait tous ceux des tableaux 29 à 33 page suivante. Note en particulier que sur les délimiteurs (tableau 33 page suivante), seuls `\llbracket` et `\rrbracket` sont de taille variable et peuvent être utilisés avec `\left` et `\right`.

`\Ydown`	$\Ydown$	`\Yleft`	$\Yleft$
`\Yup`	$\Yup$	`\Yright`	$\Yright$
`\binampersand`	$\binampersand$	`\bindnasrepma`	$\bindnasrepma$
`\boxbar`	$\boxbar$	`\boxbox`	$\boxbox$
`\boxslash`	$\boxslash$	`\boxcircle`	$\boxcircle$
`\boxbslash`	$\boxbslash$	`\boxast`	$\boxast$
`\boxempty`	$\boxempty$	`\boxdot`	$\boxdot$
`\curlyveedownarrow`	$\curlyveedownarrow$	`\curlyveeuparrow`	$\curlyveeuparrow$
`\curlywedgeuparrow`	$\curlywedgeuparrow$	`\curlywedgedownarrow`	$\curlywedgedownarrow$
`\fatslash`	$\fatslash$	`\fatbslash`	$\fatbslash$
`\fatsemi`	$\fatsemi$	`\interleave`	$\interleave$
`\leftslice`	$\leftslice$	`\rightslice`	$\rightslice$
`\sslash`	$\sslash$	`\nplus`	$\nplus$
`\minuso`	$\minuso$	`\moo`	$\moo$
`\obar`	$\obar$	`\oblong`	$\oblong$
`\obslash`	$\obslash$	`\ogreaterthan`	$\ogreaterthan$
`\owedge`	$\owedge$	`\olessthan`	$\olessthan$
`\ovee`	$\ovee$		
`\talloblong`	$\talloblong$	`\varbigcirc`	$\varbigcirc$
`\varcurlyvee`	$\varcurlyvee$	`\varcurlywedge`	$\varcurlywedge$
`\varobar`	$\varobar$	`\varoast`	$\varoast$
`\varobslash`	$\varobslash$	`\varocircle`	$\varocircle$
`\varoslash`	$\varoslash$	`\varoplus`	$\varoplus$
`\varominus`	$\varominus$	`\varotimes`	$\varotimes$
`\varolessthan`	$\varolessthan$	`\varogreaterthan`	$\varogreaterthan$
`\varowedge`	$\varowedge$	`\varovee`	$\varovee$
`\vartimes`	$\vartimes$		

TABLEAU 29: Opérateurs (ajout **stmaryrd**)

`\bigbox`	$\bigbox$	`\bigsqcap`	$\bigsqcap$
`\biginterleave`	$\biginterleave$	`\bigparallel`	$\bigparallel$
`\bigcurlyvee`	$\bigcurlyvee$	`\bigcurlywedge`	$\bigcurlywedge$
`\bigtriangledown`	$\bigtriangledown$	`\bigtriangleup`	$\bigtriangleup$
`\bignplus`	$\bignplus$		

TABLEAU 30: Opérateurs à taille variable (ajout **stmaryrd**)

[30] Faut pas me demander pourquoi. Peut-être qu'ils l'on dessinée dans un couvent ou devant une église.

\inplus	$\inplus$	\niplus	$\niplus$
\subsetplus	$\subsetplus$	\supsetplus	$\supsetplus$
\subsetpluseq	$\subsetpluseq$	\supsetpluseq	$\supsetpluseq$
\trianglelefteqslant	$\trianglelefteqslant$	\trianglerighteqslant	$\trianglerighteqslant$
\ntrianglelefteqslant	$\ntrianglelefteqslant$	\ntrianglerighteqslant	$\ntrianglerighteqslant$

Tableau 31: Symboles de relation (ajout `stmaryrd`)

\Mapsfrom	$\Mapsfrom$	\Mapsto	$\Mapsto$
\Longmapsfrom	$\Longmapsfrom$	\Longmapsto	$\Longmapsto$
\leftarrowtriangle	$\leftarrowtriangle$	\rightarrowtriangle	$\rightarrowtriangle$
\leftrightarrowtriangle	$\leftrightarrowtriangle$	\leftrightarroweq	$\leftrightarroweq$
\shortleftarrow	$\shortleftarrow$	\shortrightarrow	$\shortrightarrow$
\mapsfrom	$\mapsfrom$	\lightning	$\lightning$
\longmapsfrom	$\longmapsfrom$		
\nnwarrow	$\nnwarrow$	\nnearrow	$\nnearrow$
\sswarrow	$\sswarrow$	\ssearrow	$\ssearrow$
\shortdownarrow	$\shortdownarrow$	\shortuparrow	$\shortuparrow$

Tableau 32: Flèches (ajouts `stmaryrd`)

\Lbag	$\Lbag$	\Rbag	$\Rbag$
\lbag	$\lbag$	\rbag	$\rbag$
\llceil	$\llceil$	\rrceil	$\rrceil$
\llfloor	$\llfloor$	\rrfloor	$\rrfloor$
\llbracket	$\llbracket$	\rrbracket	$\rrbracket$
\llparenthesis	$\llparenthesis$	\rrparenthesis	$\rrparenthesis$

Tableau 33: Délimiteurs (ajout `stmaryrd`)

19.5 Le package qsymbols

Ce package est écrit par Kristoffer H. Rose [102].

Son utilité n'est pas flagrante : aucun nouveau symbole, aucune nouvelle fonctionnalité. Simplement, une syntaxe abrégée pour accéder à de nombreux symboles. Ces symboles sont ceux usuels sous LaTeX, une partie de ceux du package `amsmath`, du package `stmaryrd` ainsi que certaines possibilités offertes par Xy-pic pour les flèches.

19.5.1 Les symboles simples

On les obtient à l'aide du caractère ` comme l'indique le tableau 34 page suivante.

Note qu'on peut encadrer ou encercler même des symboles qui ne sont pas prévus (c'est le cas du a). Le gras d'un symbole s'obtient en le faisant précéder de `", par exemple `"`a donne : $\boldsymbol{\alpha}'$.

Tu peux ajouter un nouveau symbole à la liste :

```
\newqsymbol{`code}{signification}
```

Où `code` peut être une lettre ou une lettre entre (), [], {}.

19.5.2 Symboles de relation

On les obtiendra souvent en utilisant deux ``, comme le montre le tableau 35 page ci-contre.

Relevons également les symboles à taille variable du tableau 36 page suivante.

19.5.3 Flèches standard

Les flèches les plus courantes avec LaTeX sont accessibles avec une syntaxe simplifiée[31] comme indiqué par le tableau 37 page ci-contre.

19.5.4 Flèches étendues

Les tableaux 38 page suivante à 40 page 36 t'indiqueront comment obtenir des flèches de plus en plus complexes avec le même type de syntaxe[32].

19.6 Le package ulsy

Ce petit package, que nous devons à Ulrich Goldschmitt [43] définit 6 nouveaux symboles. Ce ne sont pas des symboles mathématiquesen ce sens qu'ils peuvent être appelés n'importe où dans le texte.

Ces six symboles sont les suivants :

[31] Ça c'est l'avis de l'auteur du package, et pas tout à fait le mien, parce que bon, faut pas pousser, c'est pas toujours intuitif.

[32] Là, j'aime de moins en moins, en particulier les flèches du tableau 40 page 36. Je les trouve ignobles à taper ces trucs, mais c'est mon opinion et elle n'engage que moi.

x	a	b	c	d	e	f	g	h	i	j	k	l	m	n	p	q	r	s
\`x	α	β	χ	δ	ϵ	ϕ	γ	η	ι	ψ	κ	λ	μ	ν	π	θ	ρ	σ
x	t	w	x	y	z			D	F	G	J	L	P	Q	S	W	X	Y
\`x	τ	ω	ξ	υ	ζ			Δ	Φ	Γ	Ψ	Λ	Π	Θ	Σ	Ω	Ξ	Υ
x	+	*	:	;	/	U	-	_	o	O	.	=	~	E	A	!	^	V
\`x	$\pm$	$\times$	$\in$	$\notin$	$\setminus$	$\cup$	$\vdash$	$\perp$	$\circ$	$\emptyset$	$\cdot$	$\equiv$	$\sim$	$\exists$	$\forall$	$\neg$	$\wedge$	$\vee$
x		+	-	\`*	/	|	\`/	\`.	*	\`o	\`^	\`V	<	>	<	!	:-	a
\`(x)	○	⊕	⊖	⊗	⊘	⦶	⃠	⊙	⊛	⊚	⟁	⧨	⧀	⧁	∘	•	☺	ⓐ
\`[x]	□	⊞	⊟	⊠	⧄	◫	⧅	⊡	⧆	⧇	◬	⧩	⧅	⧆	▫	■	☹	🅐

Tableau 34: Symboles simples avec **qsymbols**

∈	∋	\`\`∈	\`\`/∈	\`\`∈=	\`\`/∈=	\`\`\`∈	\`\`\`\`∈	\`\`∋	\`\`/∋	\`\`∋=	\`\`/∋=	\`\`\`∋	\`\`\`\`∋
<	>	<	≮	≤	≰	∧	⋀	>	≯	≥	≱	∨	⋁
(	)	⊂	⊄	⊆	⊈	∩	⋂	⊃	⊅	⊇	⊉	∪	⋃
[	]	⊏	⊏̸	⊑	⋢	⊓	⨅	⊐	⊐̸	⊒	⋣	⊔	⨆
\{	\}	≺	⊀	⪯	⋠	⋏	⋏	≻	⊁	⪰	⋡	⋎	⋎
\<	\>	⊲	⋪	⊴	⋬	▵	△	⊳	⋫	⊵	⋭	▿	∇
~	\~	∼	≁	≃	≄	≀		∽	∽̸	⋍	⋍̸	∫	∫
—	\-	⊢	⊬	⊨	⊭	⊤		⊣	⊣̸	⫤	⫤̸	⊥	

Tableau 35: Symboles de relation avec **qsymbols**

x	U	^	V	S	P
\`\`	⋃	⋀	⋁	∑	∏

Tableau 36: Symboles à taille variable avec **qsymbols**

"<-"	←
"<="	⇐
"<3"	⭅
"<-!"	⟵
"<=!"	⟸
"</-"	↚
"</="	⇍
"<3!"	≡<
"<-'"	↩
"<<-"	↞
"<-<"	↢
"<~"	↜
"<|-"	⇽
"o-"	⟜
"o<-"	o←
"^<-"	↼
"_<-"	↽
"o<<-!"	o↞—

"->"	→
"=>"	⇒
"3>"	⇛
"-!>"	⟶
"=!>"	⟹
"-/>"	↛
"=/>"	⇏
"3!>"	≡>
"\`->"	↪
"->>"	↠
">->"	↣
"~>"	↝
"-|>"	⇾
"-o"	⊸
"->o"	→o
"^->"	⇀
"_->"	⇁
"!->>o"	—↠o

"<->"	↔
"<=>"	⇔
""	—
"<-!>"	⟷
"<=!>"	⟺
"</->"	↮
"</=>"	⇎
""	—
""	—
"<<->>"	↞↠
"<~!>"	⟨∼∼⟩
"<~>"	↭
"<|-|>"	⇿
"o-o"	⧟
"o<->o"	o↔o
"^<->"	⥊
"_<->"	⥋
"o<<-!>>o"	o↞↠o

Table 37: Flèches standard avec **qsymbols**

x	\`-/>	-//>	=> ?	\`->+	|-->*
"x"	↪↛	⇸⇸	⇒?	$\overset{+}{\hookrightarrow}$	$\vdash \overset{*}{-} \rightarrow$

Table 38: Flèches non-standard avec **qsymbols**

x	->	-!>	-!!>	-!!!>	-!!!!>
"x"	$\rightarrow$	$\longrightarrow$	$\longrightarrow$	$\longrightarrow$	$\longrightarrow$

TABLE 39: Flèches longues avec `qsymbols`

<table>
<tr><td>x</td><td>-{_1}></td><td>={^\sin}!!!>></td><td>={|{`b}}!!!|></td><td>(1)></td><td>[2]></td></tr>
<tr><td>"x"</td><td>$\xrightarrow[1]{}$</td><td>$\overset{\sin}{\Rrightarrow}$</td><td>$=\!=\beta\!\!\Rightarrow$</td><td>$-\!(1)\!\rightarrow$</td><td>$-\!\boxed{2}\!\rightarrow$</td></tr>
<tr><td>x
"x"</td><td>-(1)!!>
$-\!(1)\!\longrightarrow$</td><td>-[1]!!>
$-\!\boxed{1}\!\longrightarrow$</td><td>-(1)!!>>
$-\!(1)\!\twoheadrightarrow$</td><td>-[`a]!!!>><br>$-\!\boxed{\alpha}\!\twoheadrightarrow$</td><td><<|=[`c]!!!>>
$\Lleftarrow\!\boxed{\chi}\!\Rrightarrow$</td></tr>
</table>

TABLE 40: Flèches complexes avec `qsymbols`

\odplus	⊖
\blitza	ϟ
\blitzb	ϟ
\blitzc	ϟ
\blitzd	ϟ
\blitze	ϟ

19.7 Le package wasysym

Ce package, que nous devons à Axel **Kielhorn** [55] définit une floppée de nouveaux symboles pour LaTeX dont certains sont des symboles mathématiques et d'autres sont des symboles normaux utilisables dans le texte.

\male	♂	\RIGHTarrow	►
\female	♀	\LEFTarrow	◄
\currency	¤	\UParrow	▲
\phone	☎	\DOWNarrow	▼
\recorder	⌕	\diameter	⌀
\clock	🕒	\invdiameter	⦰
\lightning	ϟ	\varangle	∢
\pointer	⇨	\wasylozenge	⌑
\kreuz	✠	\ataribox	◙
\smiley	☺	\cent	¢
\frownie	☹	\permil	‰
\blacksmiley	☻	\brokenvert	¦
\sun	☼	\wasytherefore	∴
\checked	✓	\Bowtie	⋈
\bell	🔔	\agemO	℧

TABLEAU 41: Symboles généraux ajoutés par `masysym`

\AC	∼		\HF	≈
\photon	〰		\VHF	≋
\gluon	➰			

TABLEAU 42: Symboles de physique et d'électricité ajoutés par **wasysym**

\Square	□		\XBox	⊠
\pentagon	⬠		\CheckedBox	☑
\hexagon	⬡		\hexstar	✳
\varhexagon	⬢		\varhexstar	✳
\octagon	⯃		\davidstar	

TABLEAU 43: Polygônes et étoiles (**wasysym**)

\eighthnote	♪		\quarternote	♩
\halfnote	𝅗𝅥		\fullnote	𝅝
\twonotes	♫			

TABLEAU 44: Notes de musique (**wasysym**)

\Circle	○		\CIRCLE	●
\Leftcircle	◖		\LEFTCIRCLE	◖
\Rightcircle	◗		\RIGHTCIRCLE	◗
\leftturn	↺		\LEFTcircle	◐
\rightturn	↻		\RIGHTcircle	◑

TABLEAU 45: Cercles divers (**wasysym**)

\thorn	þ		\dh	ð
\Thorn	Þ		\Dh	Ð
\openo	ɔ		\inve	ə

TABLEAU 46: Symboles phonétiques (**wasysym**)

\vernal	♈		\astrosun	☉
\ascnode	☊		\mercury	☿
\descnode	☋		\venus	♀
\fullmoon	○		\earth	♁
\newmoon	●		\mars	♂
\leftmoon	☾		\jupiter	♃
\rightmoon	☽		\saturn	♄
\uranus	♅		\neptune	♆
\pluto	♇			

Tableau 47: Symboles d'astronomie (**wasysym**)

\aries	♈		\scorpio	♏
\taurus	♉		\sagittarius	♐
\gemini	♊		\capricornus	♑
\cancer	♋		\aquarius	♒
\leo	♌		\pisces	♓
\virgo	♍		\conjunction	☌
\libra	♎		\opposition	☍

Tableau 48: Symboles d'astrologie (**wasysym**)

\APLstar	⋆	\APLdownarrowbox	⍗
\APLlog	⍟	\APLleftarrowbox	⍇
\APLbox	⎕	\APLrightarrowbox	⍈
\APLup	∆	\notbackslash	⍀
\APLdown	∇	\notslash	⌿
\APLinput	⍞	\APLnot	∼
\APLcomment	⍝	\APLcirc	∘
\APLinv	⌹	\APLcert	∣
\APLuparrowbox	⍐	\APLminus	¯

TABLEAU 49: Symboles APL (`wasysym`)

20 Constructions mathématiques

20.1 Sommes

Et comment qu'on fait pour obtenir

$$\sum_{i=0}^{n} u_i$$

me demanderas-tu, lecteur curieux et impatient.

C'est fastoche :

```
\[\sum_{i=0}^{n}u_i\]
```

De nombreux symboles peuvent être utilisés comme cela, plus exactement, tous les symboles à taille variable recensés dans le tableau 18 page 31.

Relevons au passage la différence entre `math` et `displaymath`. Dans un cas on produit $\sum_{i=0}^{n} u_n$ dans la lignes avec

```
\(\sum_{i=0}^nu_n\)
```

Alors que dans l'autre cas on produit l'exemple précédent.

Sache toutefois que si tu souhaites voir un truc du genre $\displaystyle\sum_{i=0}^{n} u_n$ en plein dans le texte comme ici, alors il faudra faire appel à la commande `\displaystyle` :

```
\(\displaystyle\sum_{i=0}^nu_n\)
```

Les autres commandes du même accabit sont :

- `\textstyle` équation dans le texte $\textstyle\sum_{i=0}^{n} u_n$
- `\scripstyle` équation en indice $\scriptstyle\sum_{i=0}^{n} u_n$
- `\scriptscriptstyle` équation en indice d'indice $\scriptscriptstyle\sum_{i=0}^{n} u_n$

20.2 Opérateurs et fonctions

Ce qu'il est convenu d'appeler les «fonctions» standard (sin, cos, ...) sont plus jolies quand elles sont tapées en texte normal alors que le reste des équations (principalement les variables) est en italique. Pour cela, ces fonctions correspondent à des commandes, parmi lesquelles tu retrouveras aussi la limite, puisque ces commandes sont ce que LaTeX appelle des `mathop` ou opérateurs mathématiques. Tous ces opérateurs mathématiques sont regroupés dans le tableau 19 page 31. Certains fonctionnent comme la limite, à toi de savoir lesquels ...

Cas «particulier» : la limite

$$\lim_{n\rightarrow+\infty} u_n = \ell$$

s'obtient en tapant

```
\[ \lim_{n\rightarrow +\infty}u_n=\ell \]
```

20.3 Fractions, racines et accolades

Allons-y à grands coups d'exemples :

$$\sum_{n=0}^{+\infty} \frac{x^n}{n!} = e^x = \sqrt{e^{2x}}$$

$$\sum_{i=0}^{n} u_i = \underbrace{u_0 + u_1 + \cdots + u_n}_{n+1 \text{ termes}}$$

s'obtiennent à partir de

```
\[ \sum_{n=0}^{+\infty}\frac{x^n}{n!}
   =e^x=\sqrt{e^{2x}} \]
\[ \sum_{i=0}^n u_i =
   \underbrace{u_0+u_1+\cdots+u_n}_{n+1
   \hbox{\scriptsize~termes}} \]
```

Note au passage le `\cdots` qui permet d'aligner les points convenablement, alors que pour $(x_0, x_1, \ldots, x_n)$ on tapera :

```
\((x_0,x_1,\ldots,x_n)\)
```

Terminons avec

```
\[\overbrace{u_0+u_1+\cdots+u_n}^{x}\]
```

qui donne

$$\overbrace{u_0+u_1+\cdots+u_n}^{x}$$

20.4 Délimiteurs

En mathématiques, on aime bien avoir des grandes parenthèses, pour cela, on procèdera comme suit :

```
\[\left(\sum_{i=0}^n u_i\right)\]
```

pour obtenir

$$\left(\sum_{i=0}^n u_i\right)$$

Tous les délimiteurs qui peuvent se trouver derrière `\left` ou `\right` sont recensés dans le tableau 20 page 31. Rappelons le délimiteur vide (un point) qui est bien utile pour les systèmes d'équations (une accolade à gauche et rien à droite), comme, par exemple :

```
\[\left\{ system \right.\]
```

20.5 Les matrices

Pour faire une matrice c'est très simple. Regarde l'exemple là et tu sauras :

```
\[A=\left(
\begin{array}{ccccc}
1 & 2 & 3 & \cdots & n \\
2 & 3 & 4 & \cdots & n+1 \\
3 & 4 & 5 & \cdots & n+2 \\
\vdots & \vdots & \vdots &
\ddots & \vdots \\
n & n+1 & n+2 & \cdots & 2n-1 \\
\end{array}
\right)^{2}\]
```

produira :

$$A=\left(
\begin{array}{ccccc}
1 & 2 & 3 & \cdots & n \\
2 & 3 & 4 & \cdots & n+1 \\
3 & 4 & 5 & \cdots & n+2 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
n & n+1 & n+2 & \cdots & 2n-1 \\
\end{array}
\right)^{2}$$

Fastoche non ? À noter : le 'c' dans l'argument du 'array' indique que l'on souhaite une colonne centrée. On peut préférer indiquer 'l' pour une colonne gauche ou 'r' pour une colonne droite.

20.6 Les accents et les espaces

Si tu voulais taper un texte en mode mathématiques, tu apprendrais très vite que les accents ne sont pas compris et que les espaces sont éjectés. De plus, les ligatures sont sauvagement ignorées, ce qui est normal : fi en mode maths (fi) signifie le produit de f par i et ne doit donc pas être ligaturé puisque ce sont deux entités distinctes.

Toutefois, LaTeX est quand même gentil, il accepte de faire des accents en mode mathématiques, mais il faut alors le demander différemment de ce qui est dit dans le tableau 7 page 23. Il devient nécessaire de se conformer au tableau 50 page ci-contre.

Pour remédier à la suppression des espaces en mode mathématiques, il faut impérativement dire explicitement combien on veut d'espaces, et de quelle nature. Les six sortes d'espaces sont illustrées dans le tableau 51 page suivante.

20.7 Constructions avancées (amsmath)

En premier lieu voyons les intégrales multiples. Un exemple, c'est très parlant :

```
\[ \iint_V\neq\iint\limits_V \]
\[ \iint\neq\iiint\neq\iiiint\neq
\idotsint \]
```

Pour voir les deux équations suivantes :

$$\iint_V\neq\iint\limits_V$$

$$\iint\neq\iiint\neq\iiiint\neq\idotsint$$

Sans commentaire.

Quelques petits jeux sur les flèches (pour les vecteurs par exemple). Juste pour s'amuser.

```
\[\overrightarrow{A_{i,j}B_{k,l}}\]
\[\underrightarrow{A_{i,j}B_{k,l}}\]
```

produira

$$\overrightarrow{A_{i,j}B_{k,l}}$$

$$\underrightarrow{A_{i,j}B_{k,l}}$$

Et ça marche même en indice ou en exposant (même la pointe de la flèche !).

Des accents mathématiques ont été redéfinis pour qu'on puisse en mettre deux sur la même lettre sans trop de difficulté. Pour en savoir plus, voir tableau 52 page ci-contre.

On peut dorénavant encadrer des formules :

$\hat{a}$	`$\hat{a}$`
$\check{a}$	`$\check{a}$`
$\tilde{a}$	`$\tilde{a}$`
$\acute{a}$	`$\acute{a}$`
$\grave{a}$	`$\grave{a}$`
$\dot{a}$	`$\dot{a}$`
$\ddot{a}$	`$\ddot{a}$`
$\breve{a}$	`$\breve{a}$`
$\bar{a}$	`$\bar{a}$`
$\vec{a}$	`$\vec{a}$`

Tableau 50: Accents en mode mathématiques

`\;`	grand espace	$xx\;x$
`\:`	espace moyen	$xx\:x$
`\,`	petit espace	$xx\,x$
`\!`	espace négatif	$xx\!x$
`\quad`	espace d'un cadratin	$xx\quad x$
`\qquad`	espace de deux cadratins	$xx\qquad x$

Tableau 51: Espaces en mode mathématiques

`\Acute`	$\Acute{\Acute{A}}$	`\Bar`	$\Bar{\Bar{A}}$	`\Breve`	$\Breve{\Breve{A}}$	`\Check`	$\Check{\Check{A}}$
`\Ddot`	$\Ddot{\Ddot{A}}$	`\Dot`	$\Dot{\Dot{A}}$	`\Grave`	$\Grave{\Grave{A}}$	`\Hat`	$\Hat{\Hat{A}}$
`\Tilde`	$\Tilde{\Tilde{A}}$	`\Vec`	$\Vec{\Vec{A}}$				

Chaque commande est appelée deux fois dans l'exemple.

Tableau 52: Accents doublés (ajout `amsmath`)

```
\[ \boxed{\sum_iu_i}\neq
\boxed{\sum_iv_i} \]
```

produira

$$\boxed{\sum_i u_i} \neq \boxed{\sum_i v_i}$$

Flèches à rallonge. C'est très pratique pour certaines formules, mais ça parasite pas mal LaTeX dans ses habitudes. Le caractère @ est intensément utilisé[33], ce qui posera problème dans d'autres applications[34].

```
\[\xleftarrow{texte}\]
\[\xleftarrow[texte long]{texte}\]
\[\xrightarrow{texte}\]
\[\xrightarrow[texte long]{texte}\]
```

donnera

$$\xleftarrow{texte}$$

$$\xleftarrow[texte long]{texte}$$

$$\xrightarrow{texte}$$

$$\xrightarrow[texte long]{texte}$$

20.8 Options de chargement de $\mathcal{AMS}$-LaTeX

Les extensions de l'$\mathcal{AMS}$ sont nombreuses et font l'objet de plusieurs packages qui tous ensemble portent le joli nom de $\mathcal{AMS}$-LaTeX.

Un particulier est `amsmath` puisqu'il comprends les trois packages `amstext`, `amsbsy`, `amsopn` sans le dire.

Les packages sont les suivants :

amsmath : Le monstre décrit ci-dessus. Il fait appel à `amstext`, `amsbsy` et `amsopn` et fournit tout un tas de constructions mathématiques que je ne documente pas. Il prend tout un tas d'options possibles :

- intlimits bornes au-dessus et au-dessous des intégrales
- nointlimits bornes à côté des intégrales
- sumlimits pareil pour les sommes
- nosumlimits ben, euh, je te fais un dessin ?
- namelimits comme pour `amsopn`
- nonamelimits comme pour `amsopn`
- leqno numéros d'équations à gauche
- reqno numéros d'équations à droite
- centertags je sais pas
- tbtags je sais pas non plus
- fleqn option globale prise en compte (équations hors-texte sont mises à gauche et non pas centrées)

Par défaut nointlimits, sumlimits, namelimits, et centertags.

amstext : Crée l'alphabet `\text`.

amsbsy : Pour les alphabets `\boldsymbol` et `\pmb`.

amsopn : Retouche les `\lim`, `\max` et autres du même accabit. L'option nonamelimits doit interdire de mettre les commandes du type `\lim` ou du moins rendre innactif le positionnement de l'indice en dessous. Par exemple en produisant $\lim_{n\to+\infty} u_n = \ell$ au lieu de $\lim\limits_{n\to+\infty} u_n = \ell$

amsthm : S'amuse avec les théorème. Le package `theorem` de Frank Mittelbach est meilleur voir à ce sujet la section 27.6 page 76.

amsintx : Pour les choses rigolotes sur les intégrales.

amscd : Pour les diagrammes commutatifs, mais j'en cause pas.

amsxtra : Trop compliqué pour que je te raconte. Pas très utile.

amsgen : Chargement de quelques macros utiles aux autres packages. Sera chargé automatiquement.

Maintenant étudions les symboles

amsfonts : Pour charger les fontes, inintéressant. L'option psamsfonts est reconnue.

amssymb : Pour charger les symboles, appel tout seul `amsfonts`, reconnaît l'option psamsfonts décrite plus loin.

euscript : Pour charger l'alphabet `\EuScript`. Admet trois options :

- psamsfonts utilise les versions PostScript des fontes. On les a pas à l'ESIEE et ça sert grosso-modo à pas grand chose[35].
- mathcal `\mathcal` devient alors équivalent à `\EuScript`.
- mathscr `\mathscr` est créé et est équivalent à `\EuScript`.

[33] Du moins était-ce le cas dans les versions précédentes de ce package ; je ne suis pas tout a fait sur qu'il en soit toujours ainsi, mais, en règle générale, on pourra retenir sans trop se tromper que les packages de l'$\mathcal{AMS}$ ont tendances à parasiter les autres.

[34] Ce n'est *apparemment* plus le cas, mais ça change à chaque remise à jour des packages, alors bon, je laisse la remarque, même si elle est plus très vraie.

[35] Pour nous tout au moins. Il parait que dans certains cas c'est utile, sur MacIntosh entre autre, mais je suis pas sûr.

eufrak : Pour charger l'alphabet `\EuFrak`. Admet l'option psamsfonts.

eucal : Comme `euscript` avec l'option mathcal par défaut.

20.9 Le package vector

Ce package, que nous devons au britanique Nick Efford [38], étend considérablement le traditionnel «accent» mathématique `\vec` — cf. tableau 50 page 41.

Je te rappelle, avant d'aller plus loin, que `$\vec{a}$` produit $\vec{a}$. C'est la notation la plus usuelle en mathématiques, jusqu'à la terminale. D'autres sont prévues, pour les physiciens par exemple :

`\uvec{a}`	$\underline{a}$
`\bvec{a}`	$\mathbf{a}$
`\svec{a}`	a

De plus on note souvent les vecteurs de norme 1 avec un accent circonflexe. C'est en effet ce qui est prévu ici pour les «`uvec`» (unary vector) :

`\uuvec{a}`	$\underline{\hat{a}}$
`\buvec{a}`	$\hat{\mathbf{a}}$
`\suvec{a}`	$\hat{\mathsf{a}}$

D'autres commandes sont prévues, pour taper directement les vecteurs et non plus simplement leurs noms :

```
\[ (\rvec{x}{1}{5}) \]
\[ \left( \cvec{x}{1}{5} \right) \]
```

pour obtenir :

$$(x_1, x_2, x_3, x_4, x_5)$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

On manipulera souvent des vecteurs de taille n :

```
\[ (\irvec{x}) \]
\[ \left(\icvec{x}) \]
```

pour obtenir :

$$(x_1, \dots, x_n)$$

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

On pourra vouloir spécifier l'indice du dernier terme :

```
\[ (\irvec[k]{x}) \]
\[ \left( \icvec[12]{x} \right) \]
```

pour obtenir :

$$(x_1, \dots, x_k)$$

$$\begin{pmatrix} x_1 \\ \vdots \\ x_{12} \end{pmatrix}$$

Ou encore changer l'indice du premier élément :

```
\[ \firstelement{0}
\left( \icvec[12]{x} \right) =
(\irvec[12]{x})^t \]
```

nous donnera :

$$\begin{pmatrix} x_0 \\ \vdots \\ x_{12} \end{pmatrix} = (x_0, \dots, x_{12})^t$$

Voilà, tu devrais tout savoir faire sur les vecteurs maintenant.

21 Alphabets mathématiques

21.1 Définition

Un «alphabet mathématique» est, à peu près, aux mathématiques sous LaTeX ce que les fontes sont aux textes. Il ne s'agit par vraiment de fontes, puisse que dans une fonte donnée on a plusieurs alphabets. Par exemple, dans la fonte usuelle sous LaTeX qui s'appelle Computer Modern, on trouves alphabets différents tels que `\mathbb` pour obtenir les lettres à double barre, ou `\mathcal` pour obtenir des lettres rondes.

Je vais te montrer dans cette section tout ce qui ressemble de près ou de loin à des alphabets pour LaTeX. Ne te fie pas trop à ce que je raconte, il m'arrive de te mentir : certaines des choses présentées ici *ne sont pas* des alphabets, mais simplement des commandes permettant de modifier l'aspect visuel des caractères produits.

21.2 Gras (amsmath)

Si tu souhaites obtenir des symboles en gras dans une formule mathématique, sans pour autant que toute la formule soit en gras, deux choix s'offrent à toi : soit le symbole existe en gras, et alors tu utiliseras :

```
\boldsymbol{...}
```

soit le symbole n'existe pas en gras, et alors tu utiliseras

```
\pmb{...}
\mathop{\pmb{...}}
```

la deuxième forme servant à produire des opérateurs du type `\sum`. Un exemple rapide et c'est bouclé :

```
\[ \sum_{i=1}^nu_i
\neq
 \boldsymbol{\sum}_{i=1}^nu_i
\neq
 \mathop{\pmb{\sum}}_{i=1}^nu_i
\]
```

Ça donne ça :

$$\sum_{i=1}^n u_i \neq \textstyle\sum_{i=1}^n u_i \neq \displaystyle\sum_{i=1}^n u_i$$

21.3 Gras (LaTeX 2ε)

Pour faire du gras dans toute une équation voire dans toutes tes équations, il y a des méthodes moins brutales que celle vue ci-dessus, et en plus elles ne requièrent pas de package d'extension. LaTeX 2ε prévoit une commande

```
\mathversion{...}
```

qui permet de commuter entre `bold` et `normal` en standard. Le package `concrete` offre aussi la possibilité d'utiliser la « version » `euler`, enfin dès qu'il sera installé. Un exemple rapide :

```
\mathversion{bold}
\[ \sum_{i=1}^nu_i \]
\mathversion{normal}
\[ \sum_{i=1}^nu_i \]
```

Pour produire :

$$\boldsymbol{\sum_{i=1}^n u_i}$$

$$\sum_{i=1}^n u_i$$

21.4 Alphabets

Par défaut, LaTeX 2ε ne connait qu'un alphabet mathématique rigolo, c'est `\mathcal`. On peut lui en apprendre d'autres. Deux très bons exemples sont les alphabets Euler Script et Euler Frakture (page 47).

Le package `amsmath` permet d'en créer 5 autres. Etudions les rapidement. `\mathbb` sers pour les lettres à double barre, `\boldsymbol` est expliqué plus haut avec `\pmb`. Ne reste plus que `\frak` qui redéfinit Euler Frakture[36] et `\text` qui permet de placer du texte dans une équation. Le texte sera dans la même fonte que celle valide en dehors de l'équation :

```
\[ \sum_{i=1}^nu_i
\text{un petit mot}
\sum_{i=1}^nu_i \]
```

Produira :

$$\sum_{i=1}^n u_i \text{un petit mot} \sum_{i=1}^n u_i$$

Un exemple de l'alphabet le plus utile, à savoir `\mathbb` qui permet d'obtenir les lettres à double barre pour noter les ensembles :

```
\[ \forall n\in \mathbb{N}
   \quad u_n=v_{n+1} \]
```

Produira :

$$\forall n \in \mathbb{N} \quad u_n = v_{n+1}$$

Un autre alphabet rigolo est offert par le package `oldstyle` (voir section 22.7 page 48 à ce sujet). C'est l'alphabet `\mathos` qui permet de produire ça : 123.

Le tableau 53 page suivante donne un exemple sur les lettres A, B et C de chaque alphabet en rappelant leur lieu de définition.

[36] Très exactement comme le fait le package `eufrak` qui est exposé section 22.2 page 47, sauf que la commande définie ici s'appelle `\frak` au lieu de `\EuFrak`.

Nom de l'alphabet	Package à inclure	Résultat produit
`\mathcal`	aucun	$\mathcal{ABC}$
`\mathbb`	`amsfonts`	$\mathbb{ABC}$
`\mathbf`	`amsfonts`	$\mathbf{ABC}$
`\mathos`	`oldstyle`	123
`\boldsymbol`	`amsbsy` $\subset$ `amsmath`	$\boldsymbol{ABC}$
`\mathfrak`	`amsfonts` ou `eufrak`	$\mathfrak{ABC}$
`\EuFrak`	`eufrak`	$\mathfrak{ABC}$
`\EuScript`	`euscript`	$\mathscr{ABC}$

Attention, `\boldsymbol` n'est pas à proprement parler un alphabet, c'est plus un «modifieur», comme `\textbf` en est un. On peut le combiner avec `\EuScript`, `\mathfrak` ou `\mathcal`. De même il agit sur les symboles et pas que sur les lettres latines.

Tableau 53: Exemples des différents alphabets mathématiques

Utilisation avancée

22 Gestion des fontes avec LaTeX 2ε

Certains packages d'extension donnent accès à d'autres fontes que celle que tu connais déjà. Voyons ensemble les plus courants.

22.1 euscript

Ce package, écrit par Frank Mittelbach et Rainer Schöpf [83], d'une simplicité extrême, permet d'utiliser, en mode maths, une nouvelle collection de fontes, qui sont les «Euler Script» telles qu'elles ont été dessinées par Hermann Zapf. On déclare le package dans le préambule du document et, en suite, en mode maths, on peut faire :

```
\[ \EuScript{ABC} \neq \mathcal{ABC} \]
```

pour comparer avec les fontes calligraphiques standard de TeX. C'est assez différent.

$$\mathscr{ABC} \neq \mathcal{ABC}$$

22.2 eufrak

Ça ressemble beaucoup au précédent, c'est aussi de Frank Mittelbach et Rainer Schöpf [82] et ça donne :

```
\[ \EuFrak{ABC} \neq
\EuScript{ABC} \neq
\mathcal{ABC} \]
```

pour obtenir :

$$\mathfrak{ABC} \neq \mathscr{ABC} \neq \mathcal{ABC}$$

Attention Lorsque tu utilises les packages de l'$\mathcal{A}\mathcal{M}\mathcal{S}$, ils définissent un alphabet mathématique `\mathfrak` qui est le même que `\EuFrak`. Le package `eufrak` pour sa part définit aussi ce même alphabet sous les deux noms. Ça crée d'évidents conflits (re-définition d'une commande déja existante). Il faut donc savoir que si le package `amsmath`[37] est déjà chargé, le package `eufrak` n'est plus utile.

Le «conflit» a été levé dans les versions plus récentes, mais le fait de charger les deux packages peut toujours provoquer, au moins des messages d'avertissement, au plus des problèmes, selon l'ordre de chargement.

22.3 pandora

Ce package de Frank Mittelbach [77] permet, pour sa part, de sélectionner les fontes «pandora» à la place des «Computer Modern» usuelles. Je les trouve un peu moins jolies, mais c'est chacun selon son goût.

Voici un paragraphe rigolo tapé en pandora. Ben oui, finalement, à force de fouiller dans NFSS j'ai fini par trouver comment lui faire avaler cette fonte même en Sans Serif ou en **Gras**, voire en *Italique*. Je n'aime pas particulièrement cette fonte, mais je te la montre, au cas où elle te tenterait.

La suite en Computer Modern, bien entendu.

22.4 beton

Le package `beton`, écrit par Frank Jensen [52] permet d'utiliser les fontes «Concrete» de la même façon que le package `pandora` permettait d'utiliser les fontes «Pandora».

Voici un paragraphe aussi rigolo que tout à l'heure, tapé en Concrete. Mais tout est faisable puisque cette fonte est construite sur la base des Computer Modern, on doit donc pouvoir, à l'aide de quelques paramètres réécrire toutes les variantes utiles. M'enfin, pour l'heure, c'est pas dans mes projets urgents. Un *Italique*, Sans Serif `Tele Text`... Et le tout en Concrete.

22.5 xavier

Ce package te permet de charger et d'utiliser une nouvelle famille de fontes (la famille `xav`) que Xavier m'a demandé d'intaller pour son usage personnel.

Les seule commandes intéressantes sont :

```
\textxav{...}
{\xavfamily ...}
```

Ce texte a été produit par cette fonte. Il faudra faire attention aux apostrophes, c est pas toujours joli. Pour faire mieux il faut appeler la commande `\app` *l'esthétique du résultat est bien plus grande.*

Sinon, les autres fontes ne sont pas touchées par tout cela.

22.6 Le package mflogo

Ce package, que nous devons à Ulrick VIETH [112] nous permet d'utiliser les fontes qui correspondent au logo de METAFONT. En effet, le logo «METAFONT» est traditionnellement écrit dans cette fonte qui comporte d'ailleurs juste assez de caractères pour faire ce logo et celui d'un programme associé METAPOST.

Ce package crée quatre commandes utiles : `\MP` et `\MF` pour produire les deux logos que tu viens de voir,

[37] ou un de ses dérivés qui chargera cette fonte

et `\textlogo` et `\logofamily` pour accéder à la fonte de manière normale.

Ça n'est pas d'un intéret débordant, mais lorsque l'on parle d'un programme, il est bon d'utiliser son logo de manière convenable.

22.7 Le package `oldstyle`

Ce package, que nous devons à Robin Fairbairns [39] permet de produire des nombres comme en utilisaient les anciens typographes, tu sais, des trucs du genre 123456. C'est assez joli.

Pour ce faire, ce package crée trois commandes. La plus utile, qui est celle que je viens d'utiliser à l'instant est `\textos` qui donne accès à ces chiffres. Son pendant directe, qui permet la même chose, mais avec une syntaxe plus proche de LaTeX 2.09 est `\oldstyle`. Elle s'utilise à l'ancienne manière, c'est-à-dire qu'au lieu d'appeler

```
\textos{123456}
```

on appelle

```
{\oldstyle 123456}
```

Enfin, la troisième commande est `\mathos` qui permet d'utiliser ces chiffres dans une équation. C'est un alphabet mathématique comme un autre.

22.8 Le package `ulem`

Le package `ulem` que nous devons à Donald Arseneau [5] est un joli reste de LaTeX 2.09 qui a été sommairement remis à jour pour LaTeX 2_ε. Il permet de gérer de manière plus poussée les soulignement. Je sais, c'est pas en lien étroit avec les problèmes de changement de fontes, mais je voyais pas où le mettre ailleurs.

Ce package remplace les deux commandes `\em` et `\emph` de manière parfaitement transparente pour qu'au lieu de mettre du texte en italique elles le soulignent. C'est là sa première raison d'être, bien que, de mon point de vue, ce soit plutôt un effet secondaire désagréable. Je précise donc tout de suite que ce package admet plusieurs options, en particulier :

normalem lui indique qu'il ne faut pas qu'il touche aux deux commandes `\em` et `\emph`.

ULforem fait le contraire.

normalbf lui indique de ne pas toucher aux commandes `\bfseries` et `\textbf`.

UWforbf lui indique d'utiliser un soulignement ondulé à la place du gras.

Pour ma part, histoire de ne pas perturber mes petites habitudes avec LaTeX j'aurais tendance à préciser les deux options normalem et normalbf. Mais il partait que certains éditeurs attendent du souligné à la place de l'italique, et dans ce cas précis, ça peut être pratique de ne pas avoir à reprendre tout le document.

Une des fonctionnalités intéressantes apportées par ce package est le fait que les commandes qu'il définit pour le soulignement sont capables de gérer convenablement les changement de lignes dans les parties soulignées, ce qui n'est pas le cas pour la commande `\underline` de LaTeX.

Par contre, certaines restrictions sont annoncées dans la « documentation[38] » du package. En effet, le chengement de ligne est géré automatiquement, les commandes de soulignement définies par `ulem` sont incapables d'utiliser les césures, par contre elles sont capables de tenir compte de la commande `\-` qui indique explicitement un point de césure possible (voire à ce sujet la section 23.2.1 page ci-contre), de même ces commandes savent tenir compte de commandes comme `\newline` ou `\linebreak`. Cependant, elles seront très fortement perturbées par les appels de commandes et par les accolades. Ainsi, s'il te venait la fantaisie de mettre un niveau d'accolades de trop lors de l'appel d'une de ces commandes, alors toutes les possibilités de passage à la ligne disparaitraient. De même, le fait de mettre une partie du texte souligné en gras[39] posera problème.

Bien, rentrons un petit peu dans le vif du sujet. Ce package crée, pour l'utilisateur de base que tu es, 5 commandes :

`\uline` pour utiliser du soulignement normal et qui sait se propager convenablement sur les changement de lignes.

`\uuline` le même pour le double soulignement.

`\uwave` le même avec un soulignement ondulé.

`\sout` pour rayer du texte ~~comme ça~~.

`\xout` pour rayer du texte ~~comme ça~~.

Je ne pense pas que cela nécessite beaucoup plus d'explication pour un usage simple, le reste sera donc consacré à un usage avancé, voir très avancé.

Donald Arseneau donne deux exemple intéressant pour un usage avancé de son package, je vais les reprendre tels quels et les commenter ensuite :

```
L'exemple simple:

\noindent 'Twas {\em brillig\/} and the {\em slithy~toves\/}
did {\em gyre\/} and {\em gim\-ble\/} in the {\em wabe,\\[2pt] }
All {\em mim\-sey\/} were the {\em boro\-goves\/} and
the {\em mome raths outgrabe}.

L'exemple compliqu\'e:

No, I did {\em not} act in the movie {\em \emph{The} %    <<<<<<< Nested
\emph{Persecu}\-\emph{tion} \emph{and} \emph{Assassination} \emph{of}
\emph{Jean-Paul} \emph{Marat}, as per\-formed by the Inmates
of the Asylum of Charenton under the Direc\-tion of the
Marquis de~Sade!} But I {\em did} see it.
```

[38] On peut difficilement parler de documentation puisqu'il s'agit du fichier source du package, cependant, ce fichier contient toute une partie explicative sur le fonctionnement du package à la fin.

[39] Par exemple, parce que, en italique ou en plus petit ou en plus grand, ou ... ça pose les mêmes problèmes bien entendu.

Ce qui produit le résultqt suivant : L'exemple simple :
'Twas brillig and the slithy toves did gyre and gimble in the wabe,
All mimsey were the borogoves and the mome raths outgrabe.

L'exemple compliqué :

No, I did not act in the movie The Persecution and Assassination of Jean-Paul Marat, as performed by the Inmates of the Asylum of Charenton under the Direction of the Marquis de Sade ! But I did see it.

Dans le cas de l'exemple compliqué, on relèvera que l'imbrication de passages soulignés[40] n'est pas simple à obtenir. En particulier, les espaces sont laissés en dehors de l'imbrication pour leur permettre de changer de taille[41] et pour autoriser le changement de ligne. De même, le `\-` est laissé en dehors de l'imbrication pour permettre à la césure de se faire en cas de besoin.

L'exemple simple permet de tester une bonne partie des commandes de changement de ligne et de gestion de l'espacement que tolère ce packages, comme par exemple le `~` ou le `\\`.

23 Mise en page

23.1 Style de page

Trois styles de page classiques existent pour LaTeX 2_ε :

1) `empty` : rien. Ni en-tête, ni pieds de page. Le numéro de page ne figure donc nulle part. Classiquement employé pour les lettres ou les couvertures de rapport.

2) `plain` : c'est le style de page par défaut. Pas d'en-tête, et le numéro de page centré en pieds de page.

3) `headings` : avec en-tête. Le numéro et le nom du chapitre en cours sont reportés en en-tête à gauche des pages de droite, et le numéro et le nom de la section sont reportés en en-tête à droite des pages de gauche. Le numéro de page est en haut à droite des pages de droite et en haut à gauche des pages de gauche (à l'«extérieur», pour aider à la recherche d'une page précise). Un document «recto» est composé exclusivement de pages de droite. On demande un document «recto-verso» en passant l'option `twoside` au `\documentclass`. Les pages de droite se reconnaissent au fait qu'elles ont des numéros impaires, comme l'imposent les règles élémentaires de la typographie classique [84].

   Avec ce style de page, les premières pages de chapitre seront composées en `plain` pour éviter qu'un en-tête ne se retrouve au dessus du titre du chapitre. On pourra souhaiter que les chapitres commencent toujours sur des pages de droite[42], cela s'obtient en mettant l'option `openright` du `\documentclass`.

Deux commandes existent pour changer de style de page :

- `\thispagestyle` : pour changer le style de la page en cours.
- `\pagestyle` : pour changer le style des pages jusqu'à nouvel ordre.

23.2 Changement de ligne, changement de page, espacement

23.2.1 Changement de ligne, césure

Plusieurs méthodes existent pour changer de ligne.

La première est de changer de paragraphe (cf 14.5 page 24). La seconde est d'imposer un changement-brutal comme
ici avec la commande `\newline`.

Enfin, il est possible d'autoriser un changement de ligne bizarre à l'aide de la commande `\linebreak[`*n*`]` où *n* peut avoir les valeurs suivantes :

1 ⟶ changement de ligne autorisé ;

2 ⟶ changement de ligne acceptable ;

3 ⟶ changement de ligne bon ;

4 ⟶ changement de ligne préférable ;

5 ⟶ équivalent à un `\linebreak` sans paramètre ; changement de ligne imposé. Attention, contrairement au `\newline`, le `\linebreak` laisse la justification du paragraphe se faire.

On peut interdire un changement de ligne avec `\nolinebreak` qui admet les mêmes options que `\linebreak`.

Dans certains cas précis, LaTeX ne trouve pas un point de césure dans un mot, on peut alors les lui indiquer :

```
fonc\-tion\-nai\-re
```

NB : le troisième, entre le `i` et le `r`, n'est toléré par la typographie française que dans les cas extrêmes puisqu'il renvoit seulement 2 lettres à la ligne.

On peut aussi, pour des mots fréquents, les ajouter à son dictionnaire d'exceptions avec la commande

```
\hyphenation{fonc-tion-nai-re}
```

[40] ou plus précisément l'appel de `\emph` dans `\em`, puisque cet exemple utilise le fait de remplacer `\em` par du souligné
[41] Pour pouvoir faire la justification
[42] aussi appelées «belles pages» par les typographes

Attention, LaTeX différencie majuscules et minuscules, il faudra donc plus vraissemblablement entrer :

```
\hyphenation{fonc-tion-nai-re Fonc-tion-nai-re}
```

23.2.2 Espacement

Un espacement par défaut est ajouté entre deux paragraphes, on le trouvera parfois insuffisant, par exemple pour séparer deux séries de paragraphes sur deux idées différentes (thèse et anti-thèse dans une dissertation). On aura alors recours à l'une des trois commandes suivantes :

- `\smallskip` laisse une espace étroite.
- `\medskip` laisse une espace moyenne.
- `\bigskip` laisse une grande espace.

Tu voudras parfois, pour d'autres raisons, laisser plus de blanc encore. Tu auras alors recours à une commande plus générale :

```
\vspace{5cm}
```

Commande à laquelle on peut spécifier la distance à laisser en `cm`, `mm`, `in` (pouce), `pt` (72,27`pt`=1`in`=2,54`cm`), `bp` (big point, ou point PostScript, 72`bp`=1`in`) et bien d'autres unités encore.

`\hspace` permettra d'obtenir un espacement horizontal d'une taille choisie.

Note bien que le blanc produit par `\hspace` et `\vspace` est de la même nature que celui qui est entre deux mots (ou entre deux lignes), c'est à dire qu'il diaparaît en fin de ligne (pour `\hspace`) ou en fin de page (pour `\vspace`). Pour qu'il en soit autrement, il te faudra utiliser `\hspace*` et `\vspace*`.

23.2.3 Changement de page

De même que pour le changement de ligne, il existe plusieurs commandes. `\newpage`, `\pagebreak` et `\nopagebreak` sont similaires à leurs homologues pours les lignes étudiés plus haut.

Il existe en sus `\clearpage` qui fait l'équivalent d'un `\newpage` puis s'assure que tous les flottants en attentes soient placés. `\cleardoublepage` fait la même chose en veillant en plus — quitte à laisser une page blanche — à ce que l'on reparte bien sur une page de droite.

23.3 Le package `indentfirst`

Un petit

```
\usepackage{indentfirst}
```

te permettra de résoudre ce difficile problème avec LaTeX qui est qu'il n'indente pas le premier paragraphe après un titre alors qu'on voudrait pas toujours qu'il en soit ainsi. C'est très bourrin, mais ça marche impeccable, y'a qu'à voir ce document.

Ce package nous est offert par David Carlisle [16].

23.4 La package `a4`

Le package `a4` qui est un des grands standard pour LaTeX a été réécrit pour LaTeX 2_ε par Nico Poppelier et Johannes Braams [93]. Il a l'air assez joli et semble prévoir tous les cas. On peut lui passer une option `widemargins` qui permet d'obtenir de grandes marges.

Fais gaffe, je n'ai pas corrigé le style, aussi les marges sont déjà trop grandes à mon goût. Il n'est pas exclu que je rajoute à la main une option non-standard qui sera `shortmargins` pour gagner un peu de place sur les pages.

23.5 Le package `afterpage`

Ce package est une vaste bidouille écrite par David Carlisle [20] (décidément, un habitué) qui permet des petits trucs assez cools au niveau de la mise en page de documents un peu bizarres. L'idée est de stocker des commandes qui seront appliquées juste après la fin de la page courante. Par exemple, tu as un tableau dont tu sais pertinemment qu'il sera grand. Tu as la possibilité de couper la page pour insérer ton tableau puis de reprendre ton texte après, mais ça a le désavantage de laisser une page non-pleine avant le tableau. C'est pas beau. Tu vas alors décider de mettre ton tableau en flottant. Oui, mais Dieu seul sait quand il va apparaître, peut-être bien à des pages et des pages de là.

Alors que si tu intimes l'ordre à LaTeX 2_ε de le placer juste après la fin de la page courante, tu pourras aisément dire dans ton texte : «le tableau en haut de la page suivante vous indique... », ce qui est très joli.

On peut aussi, pour éviter que tous les flottants d'un chapitre ne viennent s'entasser à la fin du chapitre, indiquer un `\clearpage` juste après la fin de la page. Ça ne changera pas ta page, mais ça viendra vider tous les flottants encore en mémoire avant de continuer dans le texte. C'est parfois pratique.

Comment on utilise ce package ? Facilement :

```
\afterpage{
\framebox{En haut de la page suivante}
}
```

Attention! Ce package est une version béta. La commande n'est pas particulièrement robuste et pourrait probablement introduire des erreurs.

De plus, ce package pose problème quand il est utilisé avec `multicol` puisque `multicol` détourne la gestion des pages pour gérer ses colonnes. Il ne fonctionnera donc que dans les passages en une seule colonne, ou plus exactement le texte apparaîtra au prochain changement de page se faisant en une seule colonne.

23.6 Changement de page amélioré

L'une des grandes difficultés avec LaTeX 2.09 était de lui expliquer où il avait le droit de changer de page. En effet, dans un texte un peu compliqué (contenant des tableaux ou des blocs de texte insécables[43]) LaTeX ne savait jamais vraiment bien où couper sa page, et il avait tendance à faire des pages pleines de vide alors qu'une petite ligne de plus aurait bien arrangé ses affaires.

Il y a maintenant une solution

```
\enlargethispage{longueur}
\enlargethispage*{longueur}
```

qui permettent d'allonger la page de la longueur spécifiée pour faire tenir une ligne de texte en plus ou pour en retirer une (longueur négative) de manière à arranger un peu les choses.

La forme avec une `*` impose à LaTeX de réduire au strict minimum toutes les longueurs élastiques mises en œuvre dans l'espacement vertical, c'est à dire qu'il comprime au maximum la page pour en faire tenir le plus possible dessus.

23.7 Format de page : `vmargin`

Nous devons à la générosité de Volker Kuhlmann, un Néerlandais, un joli package du nom de `vmargin` qui permet sans trop se fouler de changer les marges d'un document.

Il fournit quelques commandes dont voici la syntaxe :

```
\setpapersize{A4}
```

La première permet de sélectionner le format de papier dans une liste relativement impressionnante : tous les $\mathtt{A}_n$ et tous les $\mathtt{B}_n$ pour n entre 1 et 9. Ton poly sur du papier A1 ? Bien sûr... Et puis si tu le souhaites en landscape (à l'italienne pour les francophones puristes comme moi), il te suffira de le dire en option et d'en parler tendrement à ton driver préféré :

```
\setpapersize[landscape]{A1}
```

Et puis, pour les gens comme toi, il reste la possibilité de créer de nouveaux formats de papier. Pour positionner les marges ? Eh bien voici la méthode la plus simple qui soit :

```
\setmarginsrb{a}{b}{c}{d}{e}{f}{g}{h}
```

Avec la signification suivante : `a` est la marge de gauche, `b` est la marge haute, `c` la marge de droite et `d` la marge basse. Il ne reste plus que quelques paramètres à préciser. `e` est la hauteur de l'en-tête, `f` est la distance entre l'en-tête et le texte, `g` la hauteur du pied-de-page et `h` la hauteur totale entre le bas de la page et le bas du pied de page.

Facile, non ?

[43] théorèmes, définitions...

23.8 Obtenir du landscape

On cherche à utiliser du landscape généralement pour trois classes de problèmes différents. Ils y a donc trois types de réponses :

1. Tu cherches à faire tout ton document en landscape, par exemple pour des transparents pour une soutenance.
2. Tu cherches à alterner régulièrement entre landscape et portrait selon les passages du document.
3. Tu cherches à mettre un tableau ou un schéma en landscape parce qu'il est trop grand pour la page.

Ces trois problèmes sont relativement différents. On relèvera surtout le troisième qui est plus proche des problèmes de flottants que des problèmes de mise en page. Donc il sera traité à la section 24.4 page 58 qui traite du package `rotating`.

Le premier problème est relativement simple. Il suffit de spécifier à LaTeX que tu fais du landscape, soit avec l'option prévue spécialement dans le `\documentclass`, soit avec le package `vmargin`. Attention, l'option de `\documentclass` ne change pas les marges, c'est à toi de le faire. C'est pourquoi je te recommande le package `vmargin`. Vois à ce sujet la section qui lui est consacrée (23.7 page précédente).

Il ne reste plus alors qu'à expliquer à `dvips` que tu imprimes en landscape. Pour ce faire, il suffit d'appeler `dvips` au clavier et de lui donner les options suivantes :

```
dvips -t landscape -t a4 toto
```

pour imprimer, ou en rajoutant encore l'option `-o` pour obtenir le fichier PostScript associé.

Le second problème, à savoir alterner entre landscape et portrait dans un même document, est relativement simple. Il te suffit de connaître le package `lscape` que nous devons à David Carlisle [17]. Ce package fonctionne en collaboration avec `graphics`. Il crée un environnement `landscape` dans lequel toutes les pages sont en landscape. Enfantin. Pour obtenir un document principal en landscape avec certaines pages en mode portrait, tu auras deviné tout seul, il faut combiner les deux. Tu utilises la première méthode (correction de la largeur du texte et des marges, et appel de `dvips` avec les bonnes options) pour spécifier que tout le document est en landscape et tu fais appel à l'environnement fourni par le package `lscape` pour repasser en mode portrait pour certaines pages.

Ce passage en deux colonnes et en mode landscape à été produit par l'environnement du package `lscape`.

23.9 En-tête : fancyheadings

Ce package est un vieux reste de LaTeX2.09 écrit par Piet van Oostrum [111, 100] mais qui fonctionne encore tout à fait bien.

Regardons rapidement la structure d'une page. On s'intéresse à l'en-tête et au pied de page. Chacun de ces morceaux est composé de trois parties : droite, gauche, et centre. Ce package est capable, avec une aisance hors du commun, d'intervertir la droite et la gauche, selon que l'on soit sur une page paire ou une page impaire. Très joli, comme effet. De plus, il prévoit que certaines pages sont spéciales, comme les premières pages de chapitres. Les habitudes veulent que pour ces pages spéciales, on spécifie

```
\thispagestyle{plain}
```

Cette habitude est respectée, quand on le souhaite. Il reste à s'y retrouver.

Il existe trois «`pagestyles`» pour ce package :

```
fancy
plain
fancyplain
```

où `fancy` fait en sorte que toutes les pages soient comme tu les définis, et respecte le style `plain` de TeX. Ainsi, si ton document est déclaré avec

```
\pagestyle{plain}
```

et que tu tentes

```
\thispagestyle{fancy}
```

alors seule une page serait construite par les définitions de ce package, toutes les autres utiliseraient le format de TeX.

Le dernier de ces formats (`fancyplain`) vient redéfinir le format `plain` d'après tes instructions, ainsi tu auras deux types de mise en page : les pages «normales» et les plages «`plain`» qui sont généralement celles de début de chapitre.

A retenir, les noms des macros définissant l'apparence des en-têtes et pieds de pages :

```
\lhead[Texte gauche paire]
      {Texte gauche impaire}
\rhead[Texte droit paire]
      {Texte droit impaire}
\chead{Centre}
\lfoot[Texte gauche paire]
      {Texte gauche impaire}
\rfoot[Texte droit paire]
      {Texte droit impaire}
\cfoot{Centre}
```

Pour spécifier ce qui doit apparaître si la page est «`plain`» au lieu de «normale», il faut utiliser la commande

```
\fancyplain{Texte plain}
           {Texte normal}
```

Ainsi, une définition pourra ressembler à

```
\lhead
  [\fancyplain
    {Texte gauche paire plain}
    {Texte gauche paire normal}
  ]
  {Texte gauche impaire}
```

On pourra, simplement, souhaiter mettre une ligne entre l'en-tête et le texte ou entre le texte et le pied. Pour cela `fancyheadings` prévoit que tu puisses utiliser quatre longueurs :

```
\headrulewidth
\footrulewidth
\plainheadrulewidth
\plainfootrulewidth
```

Reste à effectuer le choix des textes à afficher, alors, juste pour mémoire je te rapelle, lecteur perdu dans ces méandres de complexité, que `\thepage` est une macro qui est remplacée par le numéro de la page en cours. De plus `\leftmark` contient généralement le titre du chapitre dans un rapport (ou de la section dans un article) et que `\rightmark` contient le titre de la section dans un rapport (ou de la sous-section dans un article).

Enfin, et pour finir, rappelons aux plus courageux que les titres des chapitres, sections... sont transmis via des commandes du type

```
\chaptermark{Titre du chapitre}
```

et que l'on peut fixer les valeurs de `\leftmark` et de son petit frère avec les commandes suivantes :

```
\markboth{gauche}{droite}
\markright{droite}
```

Aussi il suffit de déclarer :

```
\renewcommand{\chaptermark}[1]
{\markboth{Chapitre \thechapter.}{#1}}
```

pour faire apparaître dans la marque du côté gauche le numéro du chapitre et dans celle du côté droit le titre de celui-ci.

Voila, tu as maintenant toutes les clefs en main, lecteur courageux et à la limite du téméraire.

23.10 Le package ssqquote

Ce petit package, écrit par Ulrik Vieth [113] permet de faire des citations en fin de chapitre comme il fait Knuth dans le TeXbook et même que je trouve que c'est joli. Alors je vais faire une citation, là, tout de suite.

```
\begin{chapterquotes}
\nextquote
Y fait B\^o\^o\^o\^o\^o\^o-\^o\^o\^o\^o-\^o\^o\^o\^o
\author P. Vincent (Un jour de printemps)
\nextquote
Nous allons \'a jamais vers demain.
\author J.R.R. Tolkien,
\title{The Lord of the Rings}
(Bilbo Baggins)
\end{chapterquotes}
```

Prends garde à un fait gênant : la fonte utilisée n'existe pas en codage T1, il te faut repasser en encodage OT1 avant d'utiliser ce package. Ce défaut est très temporaire et devrait disparaitre dans l'une des futures «release» des fontes.

Le passage de `T1` en `OT1` se fait avec la ligne

```
\renewcommand{\encodingdefault}{OT1}
```

et le retour en sens inverse est assuré par

```
\renewcommand{\encodingdefault}{T1}
```

Y fait Bôôôôôô-ôôôô-ôôôô

— P. Vincent (Un jour de printemps)

Nous allons à jamais vers demain.

— J.R.R. Tolkien,
The Lord of the Rings
(Bilbo Baggins)

24 Les flottants

24.1 La base

«C'est très joli tout ça, me diras-tu, mais toi tu arrives à savoir quel numéro portent tes tableaux, et où ils se trouvent dans le document. En plus, ils sont jamais là où ça ferait tache, par exemple dépassant en bas de la page. Passerais-tu donc ton temps à recompiler ta doc pour voir où précisément mettre les tableaux?»

Que neni. C'est LaTeX qui se charge de les positionner. En fait, je lui dit d'en faire des «flottants», c'est-à-dire qu'il les garde en mémoire jusquà ce qu'une place adéquate leur soit trouvée.

Les différents types de flottants font l'objet du tableau 54 page suivante.

Ils sont bien pratiques puisqu'ils te permettront de les référencer comme c'est dit en 26 page 71. Pour donner un titre et un numéro à un flottant, on utilise la commande `\caption` n'importe où dans le flottant. De plus le titre sera repris dans la liste correspondante. La syntaxe de `\caption` est la suivante :

```
\caption[titre1]{titre2}
```

Le `titre1` est celui pris pour la table des matières, et le `titre2` est celui pris pour le flottant (parfois beaucoup plus long). Si le `titre1` n'est pas spécifié, c'est le `titre2` qui sera reporté dans la table des matières comme c'est le cas pour la majorité des flottants de ce document. La syntaxe devient alors :

```
\caption{titre2}
```

Tu peux, avec certaines options, imposer des contraintes sur la postion du flottant, en fait en offrant certaines possibilités à LaTeX parmis les quatre admises par défaut. Elles sont `h` comme «here» pour autoriser le flottant à apparaître exactement là où il est déclaré, `t` comme «top» pour autoriser à le mettre en haut d'une page, `b` comme «bottom» pour autoriser à le mettre en bas d'une page, et enfin `p` pour autoriser à le mettre sur une page spéciale qui sera composée uniquement de flottants. Ainsi

```
\begin{table}[h]
```

indique un tableau qui ne pourra se trouver qu'en haut ou en bas d'une page mais nulle part ailleurs. Joli, non?

24.2 Le package `endfloat`

Ce package a été écris par MM. McCauley et Junker [68] et permet d'indiquer à LaTeX que tous les flottants d'un document soient placés à la fin. C'est une chose qui complique quelque peu la lecture d'un article (devoir se reporter aux dernières pages pour lire des données concernant le début, c'est pas top) mais qui est régulièrement demandé par certaines revues scientifiques. Ça peut aussi être agréable pour des rapports dans lesquels il y a trop de flottants. En effet, avoir quelques lignes de texte sur chaque page et des dessins partout, ça n'aide pas à la lecture. Ramener toutes les images en fin de rapport ça peut être une bonne solution.

Ce package ne fournit aucune commande, il suffit de l'appeler avec les bonnes options, et il fera son travail tout seul sans rien demander à personne.

Les options? Elles sont décrites dans le tableau 55 page suivante.

Quelques petits détails pour conclure sur ce package :

- L'adaptation à `babel` est possible, mais n'a pas été faite. Je pense la suggérer aux auteurs par `mail`, et, si j'ai le courage de le faire, leur envoyer les corrections à apporter à leur package. Il n'est pas impossible non plus que les corrections pour les langues usuelles à l'ESIEE (français, allemand et espagnol) soient faites très rapidement.

- Quand ce package déplace une figure ou un tableau vers la fin du document, il laisse dans le texte une marque du type [Figure 4 about here]. Si tu sais le faire tu peux redéfinir les commandes `\figureplace` et `\tableplace` qui produisent le texte. A cet effet, je te donne le code source de ces commandes :

```
\newcommand{\figureplace}{%
   \begin{center}
     [\figurename~\thepostfig\ about here.]
   \end{center}}
\newcommand{\tableplace}{%
   \begin{center}
     [\tablename~\theposttbl\ about here.]
   \end{center}}
```

  Évite tout de même de jouer aux apprentis sorciers si tu ne sais pas bien manipuler les déclarations de commandes.

- De même une adaptation des titres des sections spéciales «Figures» et «Tables» peut être faite juste selon tes goûts et tes besoins, mais là aussi il faut passer par des redéfinitions.

24.3 Flottants améliorés: `float`

Ce package, d'Anselm Lingnau [65], te permettra une manipulation des flottants d'un niveau que tu n'as jamais espéré atteindre.

En fait, que peut-on demander de plus que ce qui est déjà prévu par LaTeX? Eh bien des choses savantes. Je vais prendre un ou deux exemples simples de problèmes qui sont résolus par ce package.

Dans un rapport que j'ai été amené à rendre et qui parlait de traitement d'images, il y avait, c'est la spécialité du genre, pleins d'images (original, retraitée par tel filtre, retraitée par tel autre...), il y avait aussi une représentation des filtres par des diagrammes,

Début	Fin	Pour déclarer ...
`\begin{table}`	`\end{table}`	Tableau sur une colonne
`\begin{table*}`	`\end{table*}`	Tableau sur toute la largeur de la page
`\begin{figure}`	`\end{figure}`	Figure sur une colonne
`\begin{figure*}`	`\end{figure*}`	Figure sur toute la largeur de la page

TABLEAU 54: Les quatres principaux types de flottants

Option	Sens
`nofiglist`	Supprime la liste des figures.
`notablist`	Supprime la liste des tableaux.
`nolists`	Equivalent aux deux précédentes.
`figlist`	Demande une liste des figures. Cette option est prise si on ne spécifie rien quant à la liste des figures. Il est évidemment contradictoire de demander à la fois `figlist` et l'une des trois premières options.
`tablist`	Demande une liste des tableaux. Mêmes remarques que ci-dessus.
`lists`	Demande les deux listes. Equivalent aux deux précédentes.
`nofighead`	Supprime la section «Figures» qui peut être ajoutée en fin de document. Oui, une section, parce que c'est étudié pour les articles. A l'heure actuelle, le nom de la section est imposé à «Figures» ce qui est convenable à la fois pour le français et l'anglais. Il n'est pas impossible que je fasse une adaptation à `babel` un de ces jours. Cette adaptation se fera alors dans le package `ESIEE` en attendant qu'il y en ait une «officielle». Cette option est prise si rien n'est spécifié quant à la section en question.
`notabhead`	La même chose, mais pour les tableaux. Dans ce cas, le nom de la section est «Tables» ce qui est déja moins convenable puisqu'en français on souhaiterait plus volontiers trouver «Tableaux». Peut-être dans une adapatation `ESIEE`...
`noheads`	Equivalent à choisir les deux précédentes options.
`fighead`	C'est le contraire de `noheadfig`. Bien évidemment, on évitera, là aussi de faire des demandes contradictoires.
`tabhead`	La même chose, mais pour les tableaux.
`heads`	Equivalent à choisir les deux options précédentes.
`markers`	Impose de laisser les marques dans le texte. Option prise par défaut si rien n'est spécifié quant aux marques.
`nomarkers`	Supprimes les-dites marques.
`tablesfirst`	Place les tableaux avant les figures.
`figuresfirst`	Place les figures avant les tableaux. Option prise par défaut si rien n'est spécifié quant à la position respective des tableaux et des figures.

TABLEAU 55: Options de chargement du pakage `endfloat`

ainsi que, pour la partie théorique, des courbes sorties de MATHEMATICA ou de GNUPLOT. Dans la version de ce rapport que j'ai rendue, tout était recensé comme figure et apparaissait dans la liste des figures. Si j'avais eu la possibilité de faire une liste des images, une liste des diagrammes et une liste des courbes, c'eut été plus esthétique. Mais à l'époque, ce package n'était pas encore installé à l'ESIEE.

L'autre exemple simple de problèmes qui peuvent se poser, c'est l'ancienne version de cette doc. Il y avait beaucoup de flottants qui venaient se positionner en haut ou en bas de chaque colonne, et, bien souvent, on avait l'impression de continuer à lire le texte, alors que pas du tout, on avait commencé à lire un bout d'explication donné dans une figure. Il eut en fait été très agréable que les figures soient séparées du texte par un trait ou quelque chose du genre. C'est maintenant prévu par LaTeX, mais en plus, c'est étendu par ce package.

Voyons les choses dans l'ordre. Tout d'abord ce package introduit la notion de **style** de flottant. Un style, pour un flottant, ça peut être :

plain Le style habituel sous LaTeX.

boxed Le flottant est encadré, et le `\caption` est placé dessous.

ruled On voit apparaître, dans cet ordre, un trait, le `\caption`, un trait, le flottant, et un trait pour finir. L'épaisseur des différents traits n'est pas toujours partout la même. C'est ce tyle de flottant que Knuth a utilisé dans son livre «Concrete Mathematics».

D'autres choses que le style permettent de caractériser un flottant. Par exemple son nom, qui est le texte qui apparaît juste avant son numéro. Les habitudes liées à la programmation avec LaTeX veulent que les noms des deux types de flottants habituels soient déclarés dans des variables : `\figurename` et `\tablename` pour qu'un package comme `babel` puisse les redéfinir en fonction de la langue. Le jour où je prévoirai la déclaration de tout un tas de flottants «standard» dans le package `ESIEE`[44] je prévoirai aussi la traduction de leurs titres par `babel`, mais pour l'instant là n'est pas la question. Une autre caractéristique d'un flottant est son placement par défaut. En effet, LaTeX prévoit que l'on puisse éventuellement indiquer les positions possibles pour un flottant (`t`, `b`, `p` ou `h`). Et puis, si on ne spécifie rien, il faut bien qu'il prenne une valeur par défaut. C'est elle qui caractérise le flottant.

Entrons maintenant dans les syntaxes offertes par ce package :

- `\newfloat{type}{placement}{ext}[compt]`
- `\floatstyle{style}`
- `\floatname{type}{nom}`
- `\floatplacement{type}{placement}`
- `\restylefloat{type}`
- `\listof{type}{Titre}`

Dans la description ci-dessus on a :

- `type` le nouveau type de flottant. Les types standard sont `table` et `figure`.
- `style` l'un des trois styles enumérés ci-avant.
- `placement` une règle de placement telle qu'on a le droit de la spécifier dans l'option de déclaration d'un flottant.
- `nom` le nom qui devra être pris par ce type de flottant.
- `ext` l'extension du fichier qui sera utilisé pour réaliser la liste correspondant à ton nouveau type de flottants. Pour les flottants habituels, c'est `lof` pour les figures et `lot` pour les tableaux.
- `compt` le nom d'un compteur.

Expliquons un peu les syntaxes, maintenant, et les habitudes d'utilisation. La première commande sert à déclarer un nouveau type de flottant `type` en lui attribuant une règle de placement par défaut `placement`. Le fichier utilisé pour dresser la liste de ces flottants se terminera par l'extension `ext`, de plus le compteur de ces flottants sera remis à zéro à chaque fois que le compteur `compt` sera incrémenté. Par exemple :

```
\newfloat{prog}{tph}{lop}[section]
```

Attention cette commande ne peut apparaître qu'avant le `\begin{document}`. Le nouveau type de flottant aura comme style le style courant. Pour qu'il en aille autrement, il faut changer le style courant, là aussi dans le préambule. Ça marche un peu comme la définition de nouveaux théorèmes, qui, elle, est expliquée à la section sur le package `theorem` (section 27.6 page 76). La commande pour changer de style courant est la deuxième, à savoir `\floatstyle`.

On précise le nom d'un type de flottant par un `\floatname` judicieusement appelé. De même on peut changer en cours de document les règles de placement par défaut d'un type de flottant.

Ne reste plus qu'à pouvoir dresser la liste des flottants d'un type donné, c'est le rôle de `\listof`.

Enfin, la moins utile des commandes de ce package est `\restylefloat` qui permet de changer le style des flottants standard. Cela ne peut se faire que dans le même contexte que la création de flottants, c'est à dire dans le préambule et sous l'influence de `\floatstyle`.

Dernière nouveauté, la règle de placement supplémentaire. Une nouvelle règle de placement est introduite par ce package, il s'agit de `H`, oui, un h majuscule. Si tu t'en souviens encore, le `h` indiquait à

[44] Très bientôt

LaTeX qu'il devait essayer autant que possible de placer le flottant à l'endroit même où celui-ci apparaissait, mais que s'il y arrivait pas il pouvait le déplacer. Avec `H` on lui impose de le mettre strictement et obligatoirement là où il est apparu. Même si ça doit faire une mise en page toute moche. Mais alors, quel est l'intérêt d'avoir des flottants ? Simplement que ton truc (tableau, image, figure...) soit référencé dans la liste avec ses copains et qu'il soit numéroté comme eux et qu'il soit mis en page (filets, cadres...) comme eux.

24.4 Flottants en landscape : `rotating`

Un problème fréquent dans un rapport est de faire tenir un tableau un poil trop large dans une page. Face à de tels problèmes, tu n'as que deux solutions. Soit tu écris ton tableau en fonte microscopique, et ça fait très moche dans ton rapport, soit tu es très fort, et tu joues pendant une heure ou deux à essayer de faire tenir ton tableau à l'aide de la commande `\rotatebox`. Dans un cas comme dans l'autre, c'est pas fabuleux.

Pour résoudre ce difficile problème, Leonor Barroca nous a écrit exprès pour nous le package `rotating` [6]. Ce package prévoit deux nouveaux types de flottants, les `sidewaysfigure` et les `sidewaystable`. Ces deux type de flottants seront mis en landscape dans un document en mode portrait. De plus, leur titre sera, lui aussi, mis en landscape. Enfin, ils seront recensés, numérotés, et listés dans les diverses tables, comme les flottants traditionnels `figure` et `table`.

Il est à noter que les flottants produits par ces deux environnements seront mis sur des pages entières. En effet, il n'est pas trop surprenant que LaTeX refuse de mélanger des bouts de page en mode portrait avec d'autres bouts en mode landscape. Par contre, on aurait pu souhaiter que deux petits flottants en mode landscape puissent se retrouver sur la même page. Et bien ce n'est pas le cas. Dans la pratique, c'est assez peu gênant puisque l'on a rarement des tableaux très larges (trop pour une page) et très peu long (suffisament peu pour que l'on puisse mettre deux flottants, titre y-compris, dans une même page). Cette restriction n'est donc pas dramatique.

24.5 Les titres des flottants : `caption`

La seule chose que tu ne saches pas encore paramétrer dans les flottants, ce sont leurs titres. Tu as la possibilité de choisir le texte, mais pas sa mise en page. Ben dès que tu auras lu les lignes qui suivent tu sauras.

Pour le néophyte que tu es on peut considérer que ce package, écrit et offert par Harald Axel Sommerfeldt [109] ne fournit pas de commande, et que ses options de chargement seules sont intéressantes. On peut spécifier quatre sortes d'options :

1°) Les options de taille. Elles permettent de spécifier la taille du texte dans les `\caption`. Les options reconnues sont :

   `scriptsize`
   `footnotesize`
   `small`
   `normalsize`
   `large` et
   `Large`

2°) Les options de style. Elles permettent de choisir quelle fonte doit être utilisée pour taper le nom du type de flottant (Tableau, Figure...). Les valeurs admises sont : `up`, `it`, `sl`, `sc`, `md`, `bf`, `rm`, `sf`, et `tt`.

3°) Les options de mise en page. Elles spécifient comment doit être tapé le texte du `\caption`.

   - `normal` Fait comme d'habitude
   - `hang` Prévoit une indentation, c'est à dire que si le texte est sur plusieurs lignes, alors la première sera normale, et les suivantes seront indentées de la largeur du nom du type de flottant, on aura donc un résultat similaire à ce que l'on obtient dans les listes comme celle-ci.
   - `isu` Identique à la précédente.
   - `center` Toutes les lignes sont centrées.
   - `centerlast` Toutes les lignes sont justifiées sauf la dernière qui est centrée.

4°) L'option `ruled` nécessaire pour que ce package fonctionne convenablement avec le package `float` quand on choisit le style `ruled`. Dans ce cas là, seules la taille et la fonte seront prises en compte.

Le seul truc autre qui peut être franchement utile à retenir est le fait qu'il existe une longueur `\captionmargin` qui permet d'imposer le rajout d'une marge à droite et à gauche du texte. Ça fait parfois joli.

Ça s'utilise comme toutes les longueurs :

```
\setlength{\captionmargin}{2cm}
```

Ce document a été tapé avec dans son préambule la ligne suivante :

```
\usepackage[normalsize,normal,sc]{caption}
```

24.6 Découpages rigolos : `floatfig`

L'idée de ce package, écrit par Thomas Kneser [56] est tout à fait simple. Généralement, une figure ne prend pas toute la largeur de la page, et c'est dommage de gâcher du papier qui coûte si cher et que le blanc en grande quantité ça fait pas toujours très joli dans un livre sérieux. Dans un rapport de stage ça rallonge, mais il y a des cas où c'est trop encombrant.

Ce genre d'exploit, à savoir fondre une figure dans le texte était autrefois réservé aux traitements de texte wysiwyg qui, se souciant peu des exigences typographiques rigides, pouvaient effectuer ce genre de placement assez simplement. L'algorithme implémenté est efficace, prévoit beaucoup de cas d'exceptions qui sont indiqués à la compilation par des erreurs et des warnings. Donc à recommander.

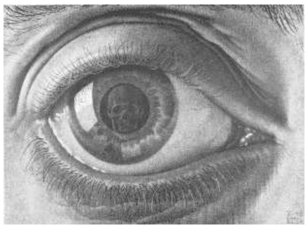


FIGURE 3: Œil

C'est une chose assez jolie, mais peu recommandable en deux colonnes, c'est pour cela que je l'ai utilisé en une colonne. La documentation de ce package spécifie clairement qu'il ne doit pas être utilisé avec l'option `twocolumn` mais ne dit rien au sujet de `multicols`.

En fait, il y a conflit entre `floatfig` et `multicol` pour des raisons bêtes et simples : l'un crée des flottants (`floatfig`) tandis que l'autre ne les gère pas encore complètement. En effet, les seuls flottants tolérés par `multicol` sont les notes de pied de page et les flottants occupant toute la largeur de la page.

De plus ce package est prévu pour LaTeX2.09 et n'a visiblement pas été réadapté à LaTeX 2_ε. Donc, à n'utiliser que pour du texte sur une colonne.

Le principe est simple : juste après le préambule on initialise le package :

```
\begin{document}
\initfloatingfigs
```

et ensuite, n'importe où entre deux paragraphes, on insère :

```
\begin{floatingfigure}{12cm}
Dessin
\caption{titre}
\end{floatingfigure}
```

et ça marche très bien.

Lorsque ce type de flottants est utilisé dans `multicols`, alors il amène des effets bizzares. Par exemple, le fait que l'indentation des paragraphes destinée à laisser de la place à la figure est systématique et irréversible, ou dure trop longtemps, ou se répercute sporadiquement jusqu'à la fin de l'environnement. C'est fâcheux. De plus si une de ces figures était incluse dans le `multicols` lui même le résultat serait en premier lieu de faire hurler le compilateur, mais cela n'est pas dramatique, en second lieu de faire commencer l'indentation là où la figure commence et de la faire se poursuivre jusqu'à la fin du `multicols`, et, enfin, `multicols` ne gèrant pas les flottants, si la figure commence trop bas dans la colonne, elle dépassera en bas de la page. Bref, rien que des bêtises.

Donc jusqu' à nouvel ordre, une figure incluse par ce moyen doit être entièrement noyée dans du texte sur une seule et unique colonne. C'est ton devoir d'y veiller. Fais attention aux plaintes du compilateur, elles sont toujours justifiées sur ce genre de problème.

Peut-être qu'un jour `multicol` gèrera les flottants, et alors aura-t-on droit à des choses magnifiques, mais pas pour aujourd'hui de toutes façons.

24.7 Découpages rigolos, version étendue : `floatflt`

Tu as étudié tout à l'heure la possibilité de mettre une figure «dans» un paragraphe. Tu as lu, puisque je l'ai écrit, que le package qui faisait ça était un poil vieux (il date de LaTeX2.09). Ben, en fait, une version spéciale LaTeX 2_ε à été écrite par Thomas Kneser et Mats Dahlgren [57], elle s'appelle `flotflt`.

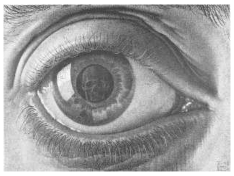


FIGURE 4: Œil

Ce paragraphe un peu long et totallement inutile sers juste à remplir la page pour éviter à la figure de venir s'afficher sur le `verbatim` qui arrive juste après. En effet, l'environnement `verbatim` étant extrêmement rudimentaire (il supprime toutes ses fonctionnalités à TeX), il n'est pas pris en compte dans la majorité des traitements de la mise en page. Par exemple, ici, le texte mis en `verbatim` ne peut pas être décalé sur le côté pour faire de la palce à une figue. Ce n'est pas un cas dramatique puisque dans la majorité des rapport et documents que tu pourrais être amené à utiliser, tu ne feras normalement pas appel à `verbatim` dont l'utilité la plus flagrante est permettre de montrer directement du code source LaTeX.

Tout d'abord, un rappel de syntaxe :

```
\begin{floatingfigure}[option]{largeur}
La figure...
\end{floatingfigure}
```

Notons les options possibles :

- `l` La figure sera placée du côté gauche du paragraphe.
- `r` Idem mais à droite.
- `p` Selon la page. Sur les pages impaires (pages de droite) la figure devra apparaître à droite ; et sur les paires (pages de gauche) la figure devra apparaître à gauche. De cette manière, elle sera toujours à l'«extérieur» du paragraphe, et jamais dans la pliure.
- `v` La figure sera placée selon l'option précisée lors du chargement du package. C'est de plus la valeur prise par défaut.

Puisqu'il y a des options au chargement du package, regardons les :

- `rflt` Les flottants (figures ou tableaux) positionnés avec l'option `v` apparaîtront à droite de leurs paragraphes respectifs.
- `lflt` Idem, mais à droite.
- `vflt` C'est l'option prise par défaut, qui dit que les flottants positionnés avec l'option `v` seront considérés comme positionnés avec l'option `p`.

Voyons maintenant le nouvel environnement, celui qui permet d'utiliser les tableaux :

```
\begin{floatingtable}[option]{
\begin{tabular}{...}
...
\end{tabular}
}
\caption{Si on veut...}
\end{floatingtable}
```

Note bien que le tableau est passé comme un argument au `\begin{floatingtable}`.

Les options sont les mêmes que précédemment.

Notons enfin que ce package est censé[45] être compatible avec `multicol` ce qui est un immense atout. Toutefois il est précisé dans la doc originale, mais ça relève du simple bon sens, que si on veut que LaTeX s'en sorte pas trop mal, il faut que le flottant soit étroit : en multicolonnage, les lignes sont déjà pas longues, alors si en plus on rajoute un flottant elles deviennent franchement ridicules. Et puis LaTeX il aime pas manipuler des colonnes franchement ridicules.

Enfin, il te faudra retenir le fait qu'il n'est plus utile, lorsque l'on utilise ce package de faire appel à la commande `\initfloatingfigs` en tout début de document puisque LaTeX 2_ε prévoit une syntaxe spéciale pour automatiser ce genre d'appel.

24.8 Le package `window`

Le package `window`, de Elmar Schalück [105], permet d'insérer une figure quelconque dans un paragraphe. Ce package a été écrit pour LaTeX 2.09, et fonctionne encore pas mal du tout pour LaTeX 2_ε. Toutefois, certaines restrictions sont à noter. En particulier pour ce qui est des incompatibilités avec les autres packages ou avec l'environnement `multicols`. Comme d'habitude des conflits ont étés détectés avec le package `babel`[46], en particulier, `window` fait appel au caractère « : » qui est redéfini par ailleurs par `babel`. Donc, si `window` est lu avant `babel`, alors il fonctionne, mais attend le « : » normal, alors que s'il est lu après il attend le « : » modifié par `babel`.

D'où la conclusion suivante : si `window` est lu avant `babel`, il doit être utilisé dans une langue ne modifiant pas le « : », c'est à dire à peu près n'importe laquelle sauf le français. Si par contre `window` est lu alors que `babel` l'a déjà été et qu'il est sorti configuré en français (c'est à dire que le français est la dernière des options qu'il ait reçu) alors ce sont les définitions particulières qui seront attendues.

Pour ce qui est de l'incompatibilité avec le package `multicol`, il suffit de savoir que le paragraphe dans lequel on inclut l'image est monobloc et ne peut être divisé sur deux colonnes par `multicol`. Il faudra donc veiller à ce que ce paragraphe soit assez long pour contenir l'image, mais pas trop, pour ne pas trop fortement pénaliser la mise en page.

Le principe de la commande `\windowbox` est assez simple. Cette commande prends trois options, toutes entre crochets (étonnant et inhabituel) qui spécifient, dans cet ordre :  le nombre de lignes à laisser avant de faire apparaître le dessin, le dessin à faire apparaître, et le ratio de centrage.

Le formalisme à respecter est très strict, la première option doit être de la forme « `[toplines :`

[45] Mon expérience en la matière tendrait plutôt à prouver que non, puisque dans la présente doc, j'ai dû supprimer le multicolonnage pour faire marcher ce package. Ceci dit, les problèmes que j'ai rencontrés sont probablement liés au fait que j'utilise la nouvelle *et* l'ancienne version dans le même document, ce qui est tout à fait déconseillé.

[46] En effet, la version de `babel` qui était installée à l'ESIEE jusqu'au début septembre 1995 comportait un bug très sérieux sur la gestion des caractères spéciaux tels que le deux-points en français.

n]», la seconde «[inwindow: *image*]», et la troisième «[ratio: *n1 n2*]». Deux points semblent bons à préciser. Tout d'abord le fait que si des crochets apparaissent dans l'*image*, alors il semble nécessaire de mettre celle-ci entre accolade ; ensuite, les deux nombres du «ratio» indiquent la proportion de place à laisser à droite et à gauche de la figure. Par exemple, un ratio «0 1» placera l'image à gauche du texte, «1 1» au centre et «1 0» à droite. Toutes sortes de variations semblent possibles !

A titre d'exemple, la façon dont a été tapé le paragraphe précédent :

```
\windowbox[toplines: 3][inwindow: {
\includegraphics[width=1cm]{eye.ps}}]%
[ratio: 1 1]
Le principe de la commande
...
```

Facile, non ?

24.9 Le package endnotes

Ce package, que nous devons à Dominik Wujastyk et John Lavagnino [118] est un vieux reste de LaTeX 2.09 qui permet de gérer les notes de fin de document.

Ce package fournit en gros trois commandes utiles à l'utilisateur de base. La commande `\endnote` qui fonctionne comme `\footnote` permet de créer une note en fin de document, par exemple[1] ici. La commande `\theendnotes` indique où doivent apparaître les commandes (pas forcément tout à la fin du document) et la commande `\addtoendnotes` permet d'ajouter du texte aux notes comme `\addtocontents` permet d'ajouter du texte dans une table des matières. Un petit exemple :

```
\addtoendnotes{Ce texte \'a \'et\'e ajout\'e
artificiellement aux notes en fin de document
produites par le package \texttt{endnotes}.}
```

Pour faire apparaître[2] les notes de fin de document ici, je vais utiliser :

```
\theendnotes
```

Et ça donne ça :

Notes

[1] Cette note inutile sers d'exemple au package endnotes.

Ce texte à été ajouté artificiellement aux notes en fin de document produites par le package endnotes.

[2] Ceci est un deuxième exemple de note en fin de document gérée par le package endnotes

Pour une utilisation un peut plus avancée, quelques autres commandes sont accessibles, en particulier `\endnotemark` qui permet de positionner une marque de renvoi en note de fin de documents. Par exemple, pour faire deux appels à la même note. Par exemple, ici[2], j'ai ré-inséré un appel à ma deuxième note d'exemple de tout à l'heure. Pour ce faire, j'ai fait appel à

```
\endnotemark[2]
```

Ceci dit, ça marche plus par hasard que pour toute autre raison. On pourra aussi relever que le texte «Notes» au début de la liste des notes est produit automatiquement par ce package et qu'il ne change pas selon la langue utilisée. Ça peut être gènant, par exemple, pour un document en allemand.

De même, d'autres commandes existent, mais je ne souhaite pas les documenter ici, leur usage n'étant pas toujours sans risque.

Attention : il y a conflit entre ce package et le suivant (fn2end) puisque les deux font appel à la commande `\theendnotes`, mais normalement je suis le seul assez débile pour vouloir les appeler tous les deux en même temps vu qu'ils font la même chose.

24.10 Le package fn2end

Ce package fort intéressant que nous devons à K.C. Border [8] permet de transformer les notes de pieds de page d'un document en notes de fin de document.

Ce package fournit trois commandes intéressantes pour l'utilisateur. La plus «évidente» est `\theendnotes` qui fait apparaître les notes en attente. Ainsi, on peut faire apparaître les notes en fin de chapitre en faisant appel à cette commande à la fin de chaque chapitre.

Les deux autres commandes permettent de basculer entre les modes «notes de pied de page» et «notes en fin de document» sont `\makeendnotes` et `\restorefootnotes`.

Donc les trois notes[47] de pied de page[48] qu'contient cette phrase[49] devraient se retrouver un petit peu plus loin.

Notes

47 Ceci est une note.

48 Justement pas, puisque j'utilise le package

49 Il m'en fallait une troisième, alors je l'ai mise là

Attention il y a conflit entre ce package et le précédent (endnotes) puisque tous les deux définissent la commande `\theendnotes`. Normalement c'est pas gênant, je devrais être le seul type assez idiot pour charger deux packages différents qui font la même chose.

25 Tableaux avec LaTeX 2_ε

25.1 Quelques précisions

Comme tu l'auras deviné, lorsque j'évoque les tableaux dans la section 15.3 page 25, je suis un petit peu succint. Voire, carrément bref.

Il y a trois détails qui font partie du noyau de base de LaTeX et que je passe sous silence.

25.1.1 Le multicolonnage

La commande `\multicolumn` permet d'écrire une case sur plusieurs colonnes, par exemple :

```
\begin{tabular}{|l|c|c|} \hline
& \multicolumn{2}{c|}{Ventes} \\
& 1993 & 1994 \\ \hline
Chaussures & 850 & 737 \\ \hline
Tables     &  18 &  25 \\ \hline
Chaises    & 175 & 127 \\ \hline
Manteaux   &  2  & 175 \\ \hline
\end{tabular}
```

produira :

	Ventes	
	1993	1994
Chaussures	850	737
Tables	18	25
Chaises	175	127
Manteaux	2	175

Pour la redéfinition des colonnes, le truc utile à retenir c'est que le séparateur qui se trouve APRÈS une colonne lui appartient, et que celui qui est avant revient à la précédente. La seule exception étant bien évidemment la première colonne.

25.1.2 Les filets horizontaux

Le tracé de filets horizontaux partiels est possible en remplaçant le `\hline` par un — ou plusieurs — `\cline` (comme «column line»).

```
\begin{tabular}{|l|c|c|} \cline{2-3}
\multicolumn{1}{l|}{} &
\multicolumn{2}{c|}{Ventes} \\
\multicolumn{1}{l|}{} &
1993 & 1994 \\ \hline
Chaussures & 850 & 737 \\ \hline
Tables     &  18 &  25 \\ \hline
Chaises    & 175 & 127 \\ \hline
Manteaux   &  2  & 175 \\ \hline
\end{tabular}
```

produira :

	Ventes	
	1993	1994
Chaussures	850	737
Tables	18	25
Chaises	175	127
Manteaux	2	175

25.2 Le package array

Ce gigantesque et magnifique package nous est offert par deux grand noms : Frank Mittelbach et David Carlisle [79].

Tout d'abord de nouvelles options pour les descriptions de colonnes. J'ai bien dit NOUVELLES, les anciennes demeurent inchangées. La première, `m{largeur}` permet d'obtenir une colonne de largeur donnée, tout comme `p{...}` mais le contenu centre verticalement sur la ligne. `b{largeur}` fait pareil en plaçant le texte en bas de la ligne.

Pour insérer systématiquement un texte au début ou à la fin d'une colonne, deux options ont été prévues : `>{declaration}` et `<{declaration}`.

De plus la barre verticale (|) sert toujours à la même chose, mais en venant ajouter son épaisseur à l'espacement entre les colonnes, ce qui est plus joli. Dans le même esprit, pour les fous qui se souviennent de `@{...}` il y a maintenant `!{...}` qui a le bon goût de laisser les espaces d'origine et de rajouter ce que tu souhaites rajouter. C'est un peu moins casse gueule.

Un joli exemple vaut mieux qu'un long discours. Le source ci-après produira le tableau 56 page 64.

```
\begin{center}
\begin{tabular}
{|>{\raggedright}p{1in}
 |>{\raggedleft}p{1in}
 |>{\centering}p{1in}
 |>{(\raggedright}p{1in}<{)}
 |l|!{texte}|l|}
\hline
%
Paragraphe court. &
Paragraphe nettement plus long
que le pr\'ec\'edent se trouvant
dans la seconde colonne. &
%
Paragraphe excessivement long
puisqu'\'etudi\'e pour cr\'eer
une ligne tout \`a fait haute
dans un tableau pour bien
mesurer l'effet de certaines
commandes.&
Colonne rigolote.
& Un & Deux \\ \hline
\end{tabular}
%
\begin{tabular}
{|>{\raggedright}m{1in}
 |>{\raggedleft}m{1in}
 |>{\centering}m{1in}
 |>{(\raggedright}m{1in}<{)}
 |l|!{texte}|l|}
Paragr...
\\ \hline
\end{tabular}
```

```
%
\begin{tabular}
{|>{\raggedright}b{1in}
 |>{\raggedleft}b{1in}
 |>{\centering}b{1in}
 |>{(\raggedright}b{1in}<{)}
 |l|!{texte}|l|}
Paragr...
\\ \hline
%
\end{tabular}
\end{center}
```

D'autres possibilités nouvelles sont offertes, comme, par exemple, celle d'ajouter systématiquement une longueur donnée entre deux lignes d'un tableau. Cette longueur s'appelle `\extrarowheight` et se redéfinit comme toute longueur avec :

```
\setlength{\extrarowheight}{longueur}
```

Enfin, une commande nouvelle permet de définir des types de colonne pour éviter de se répéter trop souvent quand on commence à avoir ses petites habitudes :

```
\newcolumntype{A}{>{\{}m{3cm}<{\}}}
```

permet de déclarer l'en-tête du tableau donné en exemple de manière plus simple.

On peut même faire du paramétrique assez facilement :

```
\newcolumntype{a}[1]{>{\{}m{#1}<{\}}}
```

où dans le `[1]` tu placeras le nombre de paramètre (9 maxi).

25.3 Le package `delarray`

Ce package, écrit par David Carlisle [22] nécessite l'emploi du package `array`, aussi se charge-t-il de le lire lui-même si tu ne l'as pas fait. Il suffit donc de spécifier

```
\usepackage{delarray}
```

pour que les deux soient chargés.

Ce package vient redéfinir l'environnement des tableaux en mode maths, à savoir `array`. Il ne corrige pas `tabular`. Il permet de mettre automatiquement des délimiteurs autour d'un tableau, ce qui est un gain de temps pour l'écriture des matrices, par exemple. Un petit exemple vaut mieux qu'un long discours :

```
$\displaystyle
\begin{array}({cc})
a & b \\
c & d \\
\end{array}$
```

produira $\left(\begin{array}{cc} a & b \\ c & d \end{array}\right)$

Pense toutefois bien que tu peux ne pas mettre de délimiteur pour réobtenir la même chose qu'avant et que, comme d'habitude, si tu souhaites un délimiteur d'un seul côté, il te faudra, bien entendu, mettre un point comme délimiteur de l'autre côté. Deux petits exemples pour bien comprendre :

```
$\displaystyle
\begin{array}{cc}
a & b \\
c & d \\
\end{array} \quad\hbox{et}\quad
\begin{array}\{{rl}.
f(x) & = x(x+1) \\
     & = x^2+x\\
\end{array}$
```

produiront $\begin{array}{cc} a & b \\ c & d \end{array} \quad\text{et}\quad \left\{\begin{array}{rl} f(x) & = x(x+1) \\ & = x^2+x \end{array}\right.$

Toi qui es un grand connaisseur, tu n'ignores pas l'option de l'environnement `array`, aussi tu te souviens de petites choses comme

$$\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)$$

qu'on obtenait en tapant

```
\[\left(\begin{array}[t]{c}
1\\2\\3 \end{array}\right)
\left(\begin{array}[c]{c}
1\\2\\3 \end{array}\right)
\left(\begin{array}[b]{c}
1\\2\\3 \end{array}\right)\]
```

Et bien maintenant on obtient

$$\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)$$

en tapant

```
\[
\begin{array}[t]({c}) 1\\2\\3 \end{array}
\begin{array}[c]({c}) 1\\2\\3 \end{array}
\begin{array}[b]({c}) 1\\2\\3 \end{array}
\]
```

ce qui est plus joli mais n'empêche pas de garder l'ancienne manière.

25.4 Le package `dcolumn`

Ce package, encore un, nous est offert par David Carlisle [21].

Là encore, il est nécessaire de précharger `array` pour utiliser ce package, mais il s'en charge tout seul comme un grand, si tu oublies.

Ce package relativement simpliste vient créer un type de colonne encore manquant sous LaTeX qui est celui des colonnes alignées sur le point décimal, et comme M. Carlisle (l'auteur) est très gentil, il a pensé aux petits français qui utilisent une virgule.

La syntaxe est la suivante :

Paragraphe court.	Paragraphe nettement plus long que le précédent se trouvant dans la seconde colonne.	Paragraphe excessivement long puisqu'étudié pour créer une ligne tout à fait haute dans un tableau pour bien mesurer l'effet de certaines commandes.	(Colonne rigolote.)	Un	texte	Deux
Paragraphe court.	Paragraphe nettement plus long que le précédent se trouvant dans la seconde colonne.	Paragraphe excessivement long puisqu'étudié pour créer une ligne tout à fait haute dans un tableau pour bien mesurer l'effet de certaines commandes.	(Colonne rigolote.)	Un	texte	Deux
Paragraphe court.	Paragraphe nettement plus long que le précédent se trouvant dans la seconde colonne.	Paragraphe excessivement long puisqu'étudié pour créer une ligne tout à fait haute dans un tableau pour bien mesurer l'effet de certaines commandes.	(Colonne rigolote.)	Un	texte	Deux

Tableau 56: Exemple de tableau avec `array`

```
\begin{tabular}{|D{.}{,}{2}|}
```

déclare que le séparateur recherché est le (.), qu'il est remplacé pas une virgule dans la sortie et que l'on ne s'autorise que deux chiffres derrières la virgule. Si on met -1 chiffres derrière la virgule, LaTeX 2_ε cherchera à ajuster la colonne tout seul, mais ce sera moins joli, comme le montrent les deux exemples ci-dessous, le premier avec l'en-tête ci-avant et le second pareil mais avec `-1` au lieu de `2`.

12345678901,1
,12
1,1
12
125,2589

12345678901,1
,12
1,1
12
125,2589

Si l'on reprend le premier en respectant les 2 décimales indiquées, voici le résultat :

12345678901,1
,12
1,1
12
125,25

Comme quoi il est absolument nécessaire de respecter ce qui est dit pour obtenir un bon résultat.

Ce package est particulièrement utile pour mettre en page des tableaux de résultats générés par des programmes, car on peut dans ce cas fixer facilement le nombre de décimales.

Tu pourra t'amuser à redéfinir, par exemple :

```
\newcolumntype{,}{D{,}{,}{2}}
```

25.5 Le package tabularx

Ce superbe package nous est offert, encore une fois, par David Carlisle [29].

Pour comprendre de quoi il retourne, il faut se souvenir de l'environnement `tabular*` (te l'avais-je décrit ?) qui permet d'ajuster la largeur d'un tableau en ajustant l'espace entre les colonnes :

```
\begin{tabular*}{250pt}{...
```

Pour sa part, ce package viendra régler la largeur de certaines colonnes en fonction de la taille du tableau (souviens toi des exemples que je donnais dans ma doc sur les tableaux). Ces colonnes sont définies sous le nom `X`. En fait, en interne, toute colonne `X` sera remplacée par une colonne `p{l}` où `l` est une longueur qui sera ajustée selon les besoins. C'est bien pratique. De plus, on peut continuer à utiliser les entrées sur plusieurs colonnes. Une petit exemple rapide pour bien comprendre.

Sur deux		3	Un texte à largeur variable
1	Un autre texte à largeur non-prévisible	3	Toujours pas facile à prédire
1	Plus court	3	Un roman fleuve pour faire paniquer un peu LaTeX 2_ε au niveau des calculs de largeurs de colonnes, faut qu'il en chie un max.

A été produit par :

```
\begin{center}
\begin{tabularx}{\linewidth}{|l|X|l|X|}
\hline
\multicolumn{2}{|c|}{Sur deux} & 3 &
Un texte \`a largeur variable \\ \hline
1 & Un autre texte \`a largeur
non-pr\'evisible & 3 & Toujours pas
facile \`a pr\'edire \\ \hline
1 & Plus court & 3 & Un roman fleuve
pour faire paniquer un peu \LaTeXe{}
au niveau des calculs de largeurs de
colonnes, faut qu'il en chie un max.
\\ \hline
\end{tabularx}
\end{center}
```

Il avait comme largeur imposée la largeur de la ligne, *ie* `\linewidth`. Mettons seulement les trois-quarts de cette largeur.

Sur deux		3	Un texte à largeur variable
1	Un autre texte à largeur non-prévisible	3	Toujours pas facile à prédire
1	Plus court	3	Un roman fleuve pour faire paniquer un peut LaTeX 2_ε au niveau des calculs de largeurs de colonnes, faut qu'il en chie un max.

Tu auras constaté que toutes les colonnes `X` sont de la même largeur.

Quelques détails pour que tu saches vraiment tout. En interne, toute colonne `X` est transformée en colonne `p` par une macro dont le définition est

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

tu pourras aisément, pour les besoins de la cause redéfinir cette commande selon ton bon vouloir, par exemple

```
\renewcommand{\tabularxcolumn}[1]%
{>{\small}m{#1}}
```

25.6 Le package hhline

C'est un package assez complet et complexe qui nous est offert, pour changer, par David Carlisle [24].

Ce machin est assez imbitable et m'a l'air relativement affreux à utiliser, bien qu'il soit très complet et visiblement très puissant dans sa gestion. Il vient remplacer le traditionnel `\hline` par un `\hhline` beaucoup plus difficile à utiliser mais offrant des possibilités immenses. Il reste donc à en comprendre la syntaxe. Il prend un paramètre qui n'est pas sans rappeler celui des environnements type `tabular`.

Tout d'abord les truc compréhensibles : `=` correspond à une ligne double, `-` à une ligne simple et `~` à pas de ligne du tout. Pour ce qui est du vertical, `|` indique que la ligne verticale coupe la ligne horizontale *ie* `===||===` indique six colonnes, une séparation entre la ligne avant et la ligne après par un trait double et que le trait double qui sépare les colonnes 3 et 4 vient interrompre le double trait de séparation des lignes. Pas clair ? Comment ça pas clair ? Bon, un petit dessin pour te faire plaisir :

1	2	3	4	5	6
1	2	3	4	5	6

qui s'obtient à partir de :

```
\begin{tabular}{||c|c|c||c|c|c||} \hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hhline{===||===}
1 & 2 & 3 & 4 & 5 & 6 \\ \hline
\end{tabular}
```

Le : indique le contraire, à savoir que c'est le double trait horizontal qui est interrompu par le double trait vertical, comme dans l'exemple ci-après.

1	2	3	4	5	6
1	2	3	4	5	6

dont voici le source

```
\E

\begin{tabular}{||c|c|c||c|c|c||} \hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hhline{===::===}
1 & 2 & 3 & 4 & 5 & 6 \\ \hline
\end{tabular}\F
```

On complique un peu la sauce. Quand on fait une première ligne, il faut que le double trait soit à moitié coupé, ou plus exactement que le trait du bas soit coupé et que le trait du haut reste intacte. «Haut» c'est «top» en anglais, alors pour ce cas de figure, c'est `t`. Pour le cas opposé, à savoir le bas du tableau, c'est `b`. Comme dans l'exemple ci-après :

1	2	3	4	5	6
1	2	3	4	5	6

obtenu avec

```
\E

\begin{tabular}{||c|c|c||c|c|c||}
\hhline{t:===:t:===:t}
1 & 2 & 3 & 4 & 5 & 6 \\
\hhline{===||===}
1 & 2 & 3 & 4 & 5 & 6 \\
\hhline{b:===:b:===:b}
\end{tabular}\F
```

Alors que si on se plante entre les : et les | on obtient un truc plus rigolo

1	2	3	4	5	6
1	2	3	4	5	6

avec comme code source

```
\E

\begin{tabular}{||c|c|c||c|c|c||}
\hhline{:t|===|t|===|t:}
1 & 2 & 3 & 4 & 5 & 6 \\
\hhline{===||===}
1 & 2 & 3 & 4 & 5 & 6 \\
\hhline{|b:===|b|===:b|}
\end{tabular}\F
```

Là où ça devient carrément immonde, c'est que le « : » en français il a une définition spéciale, c'est plus un caractère normal (pour des histoires d'espacements à respecter) alors il faut repasser en anglais (ou toute autre langue n'utilisant pas le : pour un but particulier) avant tout tableau contenant un `\hhline` utilisant le :. J'ai définit deux macros à cet effet, `\E` pour passer en anglais et `\F` pour revenir au français. Joli non ? Bon, en fait, pas tant que ça, parce que j'ai même été obligé de jouer avec les codes de catégories de certains caractères pour contourner une mésentente entre `babel` et `hhline` mes deux macros `\E` et `\F` sont donc les suivantes :

```
\def\E{\selectlanguage{english}\catcode`:=12}
\def\F{\catcode`:=13\selectlanguage{francais}}
```

Bon, continuons la torture. Il reste un symbole à voir, c'est le `#` qui indique que les lignes se croisent complètement. Enfin, pour simplifier (sic) l'écriture, il existe là aussi la commande `*` qui permet de répèter un certains nombre de fois un motif ainsi `*{3}{==#}` donnera `==#==#==#`. Ça peut être pratique.

Je te refile l'exemple qui est dans la doc originale et te renvoie à elle si t'as un pépin. M'enfin, `t:|=#~=-#::=-b:=`, c'est tout à fait compréhensible. C'est des gros mots dans une BD.

```
\E

\begin{tabular}{||cc||c|c||}
\hhline{|t:==:t:==:t|}
a&b&c&d\\
\hhline{|:==:|~|~||}
1&2&3&4\\
\hhline{#==#~|=#}
```

```
i&j&k&l\\
\hhline{||--||--||}
w&x&y&z\\
\hhline{|b:==:b:==:b|}
\end{tabular}\F
```

a	b	c	d
1	2	3	4
i	j	k	l
w	x	y	z

Bon, ben j'ai pas tout compris ce que j'ai expliqué et je trouve ça très immonde, mais ça l'air puissant et pas souple du tout. Ce n'est pas sans rapeler certaines syntaxes unix, genre awk. Visiblement, David Carlisle (l'auteur) c'est pas un pro du plug-and-play.

25.7 Gros tableaux : `supertab`

Ceci n'est pas à proprement parler un package de LaTeX 2$_\varepsilon$, c'est en fait un beau reste de LaTeX2.09, écrit par un illustre inconnu qui n'a pas donné son nom [1]. L'idée est la même que celle du suivant (`longtable`) mais un peu moins subtile, et donc un peu plus stable.

La magouille de base pour mettre un tableau sur plusieurs pages est de le couper à la main. C'est pas cool, il faut le refaire dés qu'on change le texte avant (ben oui, ça décale tout vers le haut ou vers le bas). La deuxième ruse de sioux, déjà plus évoluée, c'est de prévoir de terminer le tableau et le recommencer entre chaque ligne :

```
\def\A{\end{tabular}
       \begin{tabular}{|p{5cm}|p{5cm}|}}
\begin{tabular}{|p{5cm}|p{5cm}|} \hline
Titre 1 & Titre 2 \\ \hline
Ligne 1 & Ligne 1 \A \hline
Ligne 2 & Ligne 2 \A \hline
Ligne 3 & Ligne 3 \\ \hline
\end{tabular}
```

Mais, primo, c'est très lourd, deusio, ça remet pas l'en-tête du tableau (Titre 1 & Titre 2) en début de page suivante. En plus, ça serait chouette que ça dise «à suivre. . . » en bas de la première partie du tableau. Il faudra donc prévoir que `\A` détecte les changements de page et agisse en fonction. Ça finit par faire une macro assez énorme qu'on peut tout à fait appeler \\ pour retrouver une écriture traditionnelle.

C'est là le principe de fonctionnement de `supertab`. On fixe l'en-tête de la première partie et de chaque suivante, ainsi que le bas de la dernière partie et de chaque précédente. Ensuite l'environnement `supertabular` viendra redéfinir \\ pour fonctionner.

Quelques commandes :

```
\tablefirsthead{...}
\tablehead{...}
\tabletail{...}
\tablelasttail{...}
```

<u>A retenir :</u>

- Le bas de chaque «tail» doit contenir son `\hline`
- Le haut de chaque «tail» ne doit PAS contenir son `\hline` : il viendra du tableau. Si y en a pas dans le tableau à l'endroit de la coupure, t'es dans la mouise.
- Le haut de chaque «head» doit contenir son `\hline`
- Le bas de chaque «head» doit contenir son `\hline`, vu que celui entre les deux lignes du tableau a servi pour le bas de la partie précédente.
- Tout «head» doit se finir par le \\ parce que sinon, l'en-tête est pas sur une pleine ligne.

Tu peux considérer que chaque sous tableau est construit comme suit :

Première partie $\left\{\begin{array}{l}\text{tablefirsthead}\\ \text{corps coupé n'importe où}\\ \text{tabletail}\end{array}\right.$

Ennième partie $\left\{\begin{array}{l}\text{tablehead}\\ \text{corps coupé n'importe où}\\ \text{tabletail}\end{array}\right.$

Dernière partie $\left\{\begin{array}{l}\text{tablehead}\\ \text{corps coupé n'importe où}\\ \text{tablelasttail}\end{array}\right.$

Quelques remarques : le titre du tableau (produit avec `\caption`) est lui aussi réglable : soit en haut (avant la première partie), soit en bas (après la dernière partie).

Tu as la possibilité d'utiliser l'environnement `supertabular*` qui fonctionne comme `tabular*`, c'est à dire en espaçant les colonnes.

Si je suis courageux, il y aura un jour un `supertabularx`, qui permettra d'utiliser le type de colonne `X` qui est tout de même joli tout plein.

Comme ce package fait violemment appel à `tabular` qui est redéfini dans le package `array`, toutes les nouveautés sont utilisables.

Enfin, et pour conclure, un petit exemple :

```
\bottomcaption{Titre du bas}
\tablefirsthead{\hline Titre 1 & Titre 2 & Titre 3 \\ \hline}
\tablehead{\hline
\multicolumn{3}{|c|}{Suite$\ldots$} \\ \hline
```

```
Titre 1 & Titre 2 & Titre 3 \\}
\tabletail{\hline \multicolumn{3}{|c|}{A suivre$\ldots$} \\ \hline}
\tablelasttail{\hline}
\begin{supertabular}{|c|c|c|}
Ligne 0 & ligne 0 & ligne 0 \\ \hline
Ligne 1 & ligne 1 & ligne 1 \\ \hline
Ligne 2 & ligne 2 & ligne 2 \\ \hline
...
Ligne 50 & ligne 50 & ligne 50 \\
\end{supertabular}
```

Titre 1	Titre 2	Titre 3
Ligne 0	ligne 0	ligne 0
Ligne 1	ligne 1	ligne 1
Ligne 2	ligne 2	ligne 2
Ligne 3	ligne 3	ligne 3
Ligne 4	ligne 4	ligne 4
Ligne 5	ligne 5	ligne 5
Ligne 6	ligne 6	ligne 6
Ligne 7	ligne 7	ligne 7
Ligne 8	ligne 8	ligne 8
Ligne 9	ligne 9	ligne 9
Ligne 10	ligne 10	ligne 10
Ligne 11	ligne 11	ligne 11
Ligne 12	ligne 12	ligne 12
Ligne 13	ligne 13	ligne 13
Ligne 14	ligne 14	ligne 14
Ligne 15	ligne 15	ligne 15
Ligne 16	ligne 16	ligne 16
Ligne 17	ligne 17	ligne 17
Ligne 18	ligne 18	ligne 18
Ligne 19	ligne 19	ligne 19
Ligne 20	ligne 20	ligne 20
Ligne 21	ligne 21	ligne 21
Ligne 22	ligne 22	ligne 22
Ligne 23	ligne 23	ligne 23
Ligne 24	ligne 24	ligne 24
Ligne 25	ligne 25	ligne 25
Ligne 26	ligne 26	ligne 26
Ligne 27	ligne 27	ligne 27
Ligne 28	ligne 28	ligne 28
Ligne 29	ligne 29	ligne 29
Ligne 30	ligne 30	ligne 30
Ligne 31	ligne 31	ligne 31
Ligne 32	ligne 32	ligne 32
Ligne 33	ligne 33	ligne 33
Ligne 34	ligne 34	ligne 34
Ligne 35	ligne 35	ligne 35
Ligne 36	ligne 36	ligne 36
Ligne 37	ligne 37	ligne 37
Ligne 38	ligne 38	ligne 38
Ligne 39	ligne 39	ligne 39
Ligne 40	ligne 40	ligne 30
Ligne 41	ligne 41	ligne 41
Ligne 42	ligne 42	ligne 42
Ligne 43	ligne 43	ligne 43
A suivre...		

Suite...		
Titre 1	Titre 2	Titre 3
Ligne 44	ligne 44	ligne 44
Ligne 45	ligne 45	ligne 45
Ligne 46	ligne 46	ligne 46
Ligne 47	ligne 47	ligne 47
Ligne 48	ligne 48	ligne 48
Ligne 49	ligne 49	ligne 49 plus longue pour compliquer.
Ligne 50	ligne 50	ligne 50

TABLEAU 57: Titre du bas

25.8 Gros tableaux : `longtable`

Ce package, que nous devons à David Carlisle[50] [27], dans le même esprit le précédent, permet de gérer de gros tableaux. Mais contrairement au précédent, il garantit que sous certaines conditions (relativement strictes, tout de même) les différentes pages d'un même tableau auront la même largeur. Pour cela, il est amené à prendre des «notes» dans le fichier `.aux` et donc, en cas de modification du tableau, il est recommandé de supprimer ce fichier avant de recompiler le document. De plus pour que ces notes soient productives, il faudra compiler jusqu'à 3 fois le document.

Toutefois, lorsque tu prépares ton rapport, tu t'en fous un peu de ce problème de largeur des colonnes, alors le truc tout bête c'est que cet ajustement n'est pas fait automatiquement, il faut le demander en début de document par un

```
\setlongtables
```

Il suffira de le mettre durant les deux ou trois dernières compilations pour que tout se passe très bien, même pendant les moments où on modifie le tableau.

La syntaxe est légèrement différente de la précédente, mais suit le même esprit. On commence par ouvrir un environnement `longtable` qui se présente exactement comme un environnement `tabular`. Ensuite, on placera le contenu du premier en-tête, des en-têtes suivants, des bas de tableaux et du bas du dernier tableau. Pour cela, quatre commandes sont prévues :

```
\endfirsthead
\endhead
\endfoot
\endlastfoot
```

Le plus simple pour bien comprendre est de regarder le bout de source suivant et le tableau produit avec :

```
\begin{longtable}[c]{|*{3}{p{0.3\linewidth}|}}
\hline
Colonne 1 & Colonne 2 & Colonne 3 \endfirsthead
\hline
Colonne 1 & Colonne 2 & Colonne 3 \\
\textit{(suite)} & \textit{(suite)} & \textit{(suite)} \endhead
\multicolumn{3}{|c|}{\textit{A suivre$\ldots$}} \\ \hline \endfoot
\endlastfoot
\hline
Ligne 0 & ligne 0 & ligne 0 \\ \hline
Ligne 1 & ligne 1 & ligne 1 \\ \hline
.......................................
Ligne 50 & ligne 50 & ligne 50 \\ \hline
\end{longtable}
```

Et ça produit le tableau suivant :

Colonne 1	Colonne 2	Colonne 3
Ligne 0	ligne 0	ligne 0
Ligne 1	ligne 1	ligne 1
Ligne 2	ligne 2	ligne 2
Ligne 3	ligne 3	ligne 3
Ligne 4	ligne 4	ligne 4
A suivre...		

[50] Ça nous change, hein.

Colonne 1 *(suite)*	Colonne 2 *(suite)*	Colonne 3 *(suite)*
Ligne 5	ligne 5	ligne 5
Ligne 6	ligne 6	ligne 6
Ligne 7	ligne 7	ligne 7
Ligne 8	ligne 8	ligne 8
Ligne 9	ligne 9	ligne 9
Ligne 10	ligne 10	ligne 10
Ligne 11	ligne 11	ligne 11
Ligne 12	ligne 12	ligne 12
Ligne 13	ligne 13	ligne 13
Ligne 14	ligne 14	ligne 14
Ligne 15	ligne 15	ligne 15
Ligne 16	ligne 16	ligne 16
Ligne 17	ligne 17	ligne 17
Ligne 18	ligne 18	ligne 18
Ligne 19	ligne 19	ligne 19
Ligne 20	ligne 20	ligne 20
Ligne 21	ligne 21	ligne 21
Ligne 22	ligne 22	ligne 22
Ligne 23	ligne 23	ligne 23
Ligne 24	ligne 24	ligne 24
Ligne 25	ligne 25	ligne 25
Ligne 26	ligne 26	ligne 26
Ligne 27	ligne 27	ligne 27
Ligne 28	ligne 28	ligne 28
Ligne 29	ligne 29	ligne 29
Ligne 30	ligne 30	ligne 30
Ligne 31	ligne 31	ligne 31
Ligne 32	ligne 32	ligne 32
Ligne 33	ligne 33	ligne 33
Ligne 34	ligne 34	ligne 34
Ligne 35	ligne 35	ligne 35
Ligne 36	ligne 36	ligne 36
Ligne 37	ligne 37	ligne 37
Ligne 38	ligne 38	ligne 38
Ligne 39	ligne 39	ligne 39
Ligne 40	ligne 40	ligne 30
Ligne 41	ligne 41	ligne 41
Ligne 42	ligne 42	ligne 42
Ligne 43	ligne 43	ligne 43
Ligne 44	ligne 44	ligne 44
Ligne 45	ligne 45	ligne 45
Ligne 46	ligne 46	ligne 46
Ligne 47	ligne 47	ligne 47
Ligne 48	ligne 48	ligne 48
Ligne 49	ligne 49	ligne 49
Ligne 50	ligne 50	ligne 50

26 Les références

Maintenant tu dois avoir compris comment je fais pour numéroter mes tableaux et mes figures sans me tromper et comment je procède pour les positionner. Mais tu dois[51] te demander comment je fais pour parler avec tant d'aisance du tableau 7 page 23 qui te décrit les différents accents possibles, alors je ne sais pas du tout quel numéro il porte ni à quelle page il se trouve. Eh bien c'est très simple : c'est LaTeX qui se charge de le savoir !

26.1 Les bases

Pour pouvoir faire référence à un point quelconque du texte, il faut spécifier ce point précis en lui donnant un nom[52] à l'aide de la commande `\label` :

```
\label{refs-bases}
```

Ensuite on pourra connaître la valeur du dernier compteur incrémenté avant l'étiquette (26.1) :

```
\ref{refs-bases}
```

Ainsi que la page où se trouve l'étiquette (71) :

```
\pageref{refs-bases}
```

C'est ultra facile.

26.2 Le package `showkeys`

Ce petit quelque chose offert par David Carlisle [28] est assez agréable à utiliser. Cela permet de voir toutes les références, par exemple, lorsque l'on fait

```
\label{toto}
```

un petit «toto» encadré est imprimé dans la marge, et, de plus, à chaque fois qu'on y fera référence, juste au-dessus de la référence, se trouvera le texte «toto» en petit.

C'est très pratique pour se souvenir des noms utilisés, ou pour localiser les fautes de frappe à la relecture. Personnellement, j'utilise désormais systématiquement ce package. En plus, comme il est assez bien étudié, les textes qu'il ajoute ne viennent en rien perturber la mise en page du document. Pratique et joli.

Attention : Un problème d'utilisation surgit lorsque l'on utilise ce package avec le suivant (`varioref`) car ce dernier crée automatiquement de nombreux labels automatiques qu'il utilise généralement au même endroit, ce qui rend le document particulièrement difficile à lire. Ce n'est pas a proprement parler un bug, ni même une restriction d'utilisation, c'est juste que le document qui sort est une grosse m... et que c'est pas beau, mais tant pis.

26.3 Le package `varioref`

Ce petit package, écrit par Frank Mittelbach [78] permet de faire de jolies choses avec les références. En lisant l'ancêtre de cette doc, à savoir celle sur LaTeX 2.09, tu, lecteur à l'œil aussi vif qu'exercé, auras remarqué sans la moindre difficulté que j'avais une forte tendance à t'indiquer qu'un tableau se trouvait à la page 15, alors que tu lisais la page 15, et que si j'avais pu dire «sur cette page» ça aurait pas été plus mal que si ça avait été pire. Mais, voila, je n'étais pas maître du positionnement des tableaux.

Maintenant, je n'en suis toujours pas maître, mais LaTeX 2_ε a appris à faire tout seul, et en français, des petits bouts de phrase assez agréables, dans le genre «tableau 18 à la page précédente» ou «tableau 12 sur cette même page». Pour cela, il faut utiliser `\vref` au lieu de `\ref` et il n'est plus besoin de faire référence à la page. Pour simplement indiquer au lecteur d'aller voir une page donnée, genre «rendez-vous page 157» dans les livres dont on est le héros, il suffit d'utiliser `\vpageref`. Avec un peu d'entraînement on se fait très bien aux tournures à utiliser, tu verras.

A titre d'exemple, je place un `\label` dans le texte de la présente page et je te rappelle que la section concernant les tableaux commence page 62.

Pour produire le texte ci-dessus, j'ai tapé :

```
... le texte \vpageref{ici-meme} et je...
```

26.4 Le package `lastpage`

Ce package, écrit par Jeff Goldberg [42] permet de faire référence à la dernière page, même si celle-ci n'est pas gérée par l'utilisateur comme c'est parfois le cas.

En effet, dans certains cas, par exemple si on utilise des notes en fin de document[53] alors les dernières pages sont générées automatiquement par LaTeX et on ne peut plus y mettre de `\label`. Ce package se chargera donc de mettre un `\label`[54] sur la dernière page du document.

Toutefois, il est bon de prendre garde à ce que l'on fait, en effet, les package qui travaillent sur la fin du document, comme `endfloat` (section 24.2 page 55) ou `endnotes` (section 24.9 page 61) ou encore `fn2end` (section 24.10 page 61), ont tendance à ne pas s'entendre très bien. En effet, ils fonctionnent en disant à LaTeX quelque chose comme «lorsque tu n'auras plus rien d'autre à faire, juste avant de finir le document, fais ça pour moi». Et LaTeX exécute ces trucs dans l'ordre où ils sont donnés. Donc, en cas de pépin, changer l'ordre d'appel des packages pourra avoir des effets intéressants.

---

[51] ou tu devrais

[52] On parlera aussi volontiers d'«étiquette»

[53] Voire à ce sujet le package `endnotes` à la section 24.9 page 61 et le package `fn2end` à la section 24.10 page 61

[54] Ou plus exactement quelque chose qui agit comme un `\label` bien que ce n'en soit pas un

Le label créé par le package `lastpage` s'appelle `LastPage`[55]. Donc en appelant

```
\pageref{LastPage}
```

Je n'ai pas de difficultés particulières pour te dire que cette doc fait 143 pages.

27 Environnements

Certains environnements ont été étendus par des packages dans LaTeX 2_ε, d'autres totalement nouveaux ont été créés. Voici un petit aperçu de ceux des packages que je connais.

27.1 La package `enumerate`

Que voilà un joli petit package que David Carlisle [23] il a écrit pour nous. Bien bô. Ça sert à énumérer de manière un poil plus poussée que d'habitude. C'est bête comme chèvre à utiliser.

Tu prends un `enumerate` normal, tu lui colles en option la façon dont tu souhaites numéroter des items et puis voilà. Mais faut quand même faire gaffe, les caractères `A`, `a`, `I`, `i` et `1` sont spéciaux, ils indiquent le type de numérotation et doivent être protégés dans l'option. Un petit exemple ou deux et puis c'est marre.

Exemple 1 premier item
Exemple 2 second item
Exemple 3 troisième item

Liste I numéro 1
Liste II numéro 2
Liste III numéro 3
Liste IV numéro 4

Ces exemples ont été produits par

```
\begin{enumerate}[Exemple 1]
\item premier item
\item second item
\item troisi\`eme item
\end{enumerate}

\begin{enumerate}[L{i}ste I]
\item num\'ero 1
\item num\'ero 2
\item num\'ero 3
\item num\'ero 4
\end{enumerate}
```

27.2 `description`

27.2.1 L'environnement étendu (`expdlist`)

Le joli package `expdlist`, qui nous est offert par Rainer Hülse et Wolfgang Kasper [46], deux Allemands, fournit un nouvel environnement qui porte le même nom que l'ancien, mais offre un argument optionnel dans lequel on peut ajouter plein de réglages rigolos.

Le premier est la marge, que l'on peut fixer à la main, avec

```
\setleftmargin{taille}
```

ou automatiquement, avec

```
\setlabelphantom{texte}
```

Si on spécifie les deux, c'est la deuxième variante qui a priorité.

On a aussi la possibilité de spécifier comment se comporter si le 'label' est plus large que la marge prévue. Le comportement normal est de continuer sur la même ligne, mais on peut, à l'aide de `\breaklabel` demander à ce que le texte soit alors passé à la ligne, comme dans l'exemple ci-après, pas très loin.

Une autre chose, encore, est réglable, c'est le style des labels. Par défaut, ils sont en gras. On pourra vouloir les mettre en italique avec un

```
\setlabelstyle{\itshape}
```

Comme c'est le cas dans l'exemple ci-après :

Toto : héros de l'histoire.

Belge : nationalité de Toto. Utilisé aussi pour dénoter l'aspect histoire voire absurde de ce récis.

Zéro : élément neutre de l'addition et absorbant de la multiplication dans les environnements mathématiques habituels. Utilisé ici en tant que neutre de l'addition, il ne sert à rien.

Plus : nom signifiant la loi de composition interne première des structures de corps dans le vocabulaire enfantin ou plus globalement non-mathématicien.

Qu'on obtient facilement à partir de l'en-tête suivant :

```
\begin{description}[\breaklabel
\setlabelstyle{\itshape}
\setlabelphantom{Belge:}]
\item[Toto:] h\'eros de l'histoire.
...
  globalement non-math\'ematicien.
\end{description}
```

Enfin, il y a une option, `\compact`, pour virer les blancs entre les items successifs. Style genre que ça donnerait ça :

Toto :
héros de l'histoire.

[55] Les utilisateurs chevronnés de LaTeX remarqueront que ce label comporte des majuscules et auront peut-être à l'idée que ce n'est pas très conforme au schéma de nommage recommandé par l'équipe du projet LaTeX 3. Je leur rapellerais simplement qu'il s'agit d'un label et non d'une commande et donc qu'il échappe au règles souhaitées par l'équipe LaTeX 3.

Belge :
: nationalité de Toto. Utilisé aussi pour dénoter l'aspect histoire voire absurde de ce récit.

Zéro :
: élément neutre de l'addition et absorbant de la multiplication dans les environnements mathématiques habituels. Utilisé ici en tant que neutre de l'addition, il ne sert à rien.

Plus :
: nom signifiant la loi de composition interne première des structures de corps dans le vocabulaire enfantin ou plus globalement non-mathématicien.

En utilisant l'en-tête suivant :

```
\begin{description}[\breaklabel
\setleftmargin{5mm}\compact]
\item[Toto:] h\'eros de l'histoire.
...
  globalement non-math\'ematicien.
\end{description}
```

27.2.2 Commentaire

Le dernier truc apporté par ce package est la commande `\listpart` qui permet d'insérer un paragraphe dans une liste sans en boulverser la numérotation ou la structure. Par exemple, voici ma liste de courses :

Indispensable

1. 8 paquets de café
2. 2 boîtes de filtres
3. 3 kg de sucre
4. 2 cartouches de cigares

Miam miam

5. 1 poulet
6. 2 boîtes de p'tits pois
7. 1 pot de bolognaise
8. 500g de spaghetti

Pour offrir

9. 45 l de jus d'orange
10. 1 télé
11. 1 Plantu
12. 2 MC Escher
13. 1 bouteille d'Armagnac

A lire

14. Dernier SVM
15. Le Canard
16. Le Monde.

Qui en forme cryptée donne :

```
\begin{enumerate}[1.]
\listpart{Indispensable}
\item 8 paquets de caf\'e
\item 2 bo\^ites de filtres
\item 3 kg de sucre
\item 2 cartouches de cigares
\listpart{Miam miam}
\item 1 poulet
\item 2 bo\^ites de p'tits pois
\item 1 pot de bolognaise
\item 500g de spaghetti
\listpart{Pour offrir}
\item 45 l de jus d'orange
\item 1 t\'el\'e
\item 1 Plantu
\item 2 MC Escher
\item 1 bouteille d'Armagnac
\listpart{A lire}
\item Dernier SVM
\item Le canard
\item Le monde.
\end{enumerate}
```

27.3 verbatim

Ce package, écrit par Rainer Schöpf, Bernd Raichle et Chris Rawley [106] permet d'utiliser plusieurs environnements sans grande nouveauté, si ce n'est le `comment` qui est assez agréable et pas trop mal fichu. Il étend les deux environnements `verbatim` et `verbatim*` de manière à ne plus saturer bêtement. En effet, avant, tout le texte devait être mémorisé par TEX avant de commencer la mise en page, alors un `verbatim` de plusieurs pages pouvait poser problème pour peu que le document en cours soit déjà chargé (c'est le cas de celui-ci) par tout un tas de macros compliquées. Pour mémoire, un petit exemple de `verbatim*`

```
C'est␣un␣texte␣en␣‘verbatim*’
avec␣tout␣les␣caract\‘eres␣indiqu\’es
comme␣il␣faut␣et␣sans␣se␣tromper.
Si␣on␣tape␣un␣truc␣pas␣ascii,␣genre
un␣c-cedille␣‘ç’␣je␣sais␣pas␣ce␣que
ca␣donne.␣Logiquement␣rien,␣ou␣plus
exactement␣selon␣la␣fonte.
```

L'environnement `verbatim` ne se termine que sur la chaîne de caractère `\end{verbatim}`. Il en va de même pour la forme avec `*`. On ne pouvait pas laisser d'espace entre le `\end` et le nom de l'environnement, maintenant, on peut. Ce qui veut dire qu'on ne peut plus laisser ce nom avec un espace dans un `verbatim`. C'est pas clair ? Tant pis.

L'environnement `comment` marche pareil, mais au lieu d'imprimer le texte sans le comprendre, TEX l'éjecte sans réfléchir. On peut donc virer n'importe quoi, alors que les bidouilles usuelles du type

```
\long\def\A#1{}
\A{texte a virer}
```

n'étaient pas satisfaisantes : on pouvait prendre plus d'un paragraphe (grâce au `\long`), mais pas une accolade fermante, puisqu'elle venait fermer l'argument, et pas non plus une accolade ouvrante non refermée, alors que maintenant on peut sans difficulté.

Une jolie commande a été prévue pour montrer facilement le source d'un fichier en respectant l'indentation (pas de tabulations !), c'est le `\verbatiminput` qui prend comme argument le nom du fichier à charger.

Par exemple

```
\verbatiminput{manuel2ep.ltx}
```

Viendra charger le fichier source de ce document à chaque compilation. En toute logique, il devrait-être relativement court.

```
\documentclass[francais,french,twoside]{article}

%\batchmode

\vfuzz 3pt

%\def\dest{pascal}
\def\dest{all}
\def\Index{yes}
%\def\Index{no}
\def\Optionnel{yes}
%\def\Optionnel{no}

\input preambule
\begin{document}

\selectlanguage{francais}

%\let\OldIat\Iat
%\gdef\Iat{\string@}
%\let\NewIat\Iat

\input initials
\input couverture

\Part{Introduction}

\input introP
\input geut

\Part{Overview}

\input presentation.generale
\input premier.pas
\input structure.du.documentP
\input doc.types
\input ESIEE
\input languesP
\input fontes.overview
\newpage
\input taille
\input accents
\input constructions
\input tables.overview
\input inclusion

\Part{Math\'ematiques}

\input math.principes
\catcode`"=12
\input math.symboles
\catcode`"=13
\input math.constructions
\input math.alphabets
\Part{Utilisation avanc\'ee}

\input fontes.precisions
\input mise.en.page
\input flottants
\input tableauxP
\input references
\input environnementsP
\input mathematica.olivier.tex

\Part{Images}

\let\bar\NewDrawingBar
\input dessins.pur
\let\bar\OldMathBar
\input dessins
\input dessins.prod
\input tronique

\Part{Bibliographie, index}

\input biblio
\input index

\begin{optionnel}

\Part{Configuration et installation}

\input kpathsea
\input tex.config
\input mf.config
\input xdvi.config
\input dvips.config
\input makeindex.config
\input bibtex.config
\input package.install
\input install.linux

\end{optionnel}

\choice{\Part{Programmation}}{}

\input programmation

\begin{optionnel}

\Part{Divers}

\input diversP
\input incompatibilites

\cleardoublepage

\nocite{*}

\Part{Annexes}

\input{remerciements}

\printindex{Gal}{Index g\'en\'eral}
\printindex{Com}{Index des commandes}
\printindex{Env}{Index des environnements}
\printindex{Pak}{Index des packages}
\printindex{Sym}{Index des symboles math\'ematiques}

\end{optionnel}

\begin{multicols}{\NbCols}
\bibliographystyle{frplain}
\bibliography{manuel2ep}
\end{multicols}

\end{document}
```

27.4 moreverb

Ce package, que nous devons à un illustre inconnu, est une très jolie extension construite sur **`verbatim`**. Plusieurs commandes ont été prévues. On retiendra quatre environnements :

1) **`listing`** qui permet de numéroter les lignes d'un environnement de type verbatim. Le paramètre obligatoire est le numéro à attribuer à la première ligne, et le paramètre optionnel est celui indiquant de combien en combien on numérote.

   Un exemple de syntaxe :

```
\begin{listing}[5]{1}
Contenu...
\end{listing}
```

2) **`boxedverbatim`** qui produit du texte comme **`verbatim`**, mais encadré. Le seul petit problème est que cet environnement vient générer une grosse boîte insécable.

3) **`verbatimtab`** gère les tabulations convenablement, ce qui est très utile pour les listings indentés.

4) **`verbatimcmd`** qui est en verbatim, sauf pour les trois caractères `\`, `{` et `}`, ce qui permet de mettre en page comme du verbatim, mais en ayant la possibilité de faire de la mise en page. Intéressant pour récupérer de l'ASCII pur déjà mis en page, mais où l'on souhaite régler des choix de fonte ou des clowneries comme ça.

Il existe aussi pour deux de ces environnements des versions pour inclure directement depuis un fichier, il s'agit de **`\listinginput`** et **`\verbatimtabinput`**. Les quelques lignes ci-dessous donnent les deux exemplaires de code source qui suivent :

```
{\tiny
\listinginput[5]{10}{voisinage.h}
}

{\tiny
\verbatimtabinput{voisinage.h}
}
```

```
#if !defined(__voisinage)
#define __voisinage

/*

*******************************************************
*******************************************************
**                                                   **
**                      PROJET                       **
**                                                   **
**  Reconnaissance de phonemes en parole continue    **
**                                                   **
**              par cartes de Kohonen                **
**                                                   **
*******************************************************
**                                                   **
**     (c)1995 Benjamin BAYART & Pascal VINCENT      **
**                                                   **
**                     I4, PASTI                     **
**                                                   **
*******************************************************
*******************************************************

*/

#include <iostream.h>
#include "set.h"

class Voisinage
        {
        public:
                Voisinage()
                        {}

                virtual set apply(int,baseset*) const=0;
        };

#endif
```

```
#if !defined(__voisinage)
#define __voisinage

/*

*******************************************************
*******************************************************
**                                                   **
**                      PROJET                       **
**                                                   **
**  Reconnaissance de phonemes en parole continue    **
**                                                   **
**              par cartes de Kohonen                **
**                                                   **
*******************************************************
**                                                   **
**     (c)1995 Benjamin BAYART & Pascal VINCENT      **
**                                                   **
**                     I4, PASTI                     **
**                                                   **
*******************************************************
*******************************************************

*/

#include <iostream.h>
#include "set.h"

class Voisinage
        {
        public:
                Voisinage()
                        {}

                virtual set apply(int,baseset*) const=0;
        };

#endif
```

27.5 alltt

Ce package, que nous devons à Johannes Braams [12] peut, au premier abord, apparaître comme une restriction du précédent, mais il est plus que cela. En effet, il fourni un environnement similaire à **`verbatimcmd`**, mais semble se passer du package **`verbatim`** et est très court. L'avantage ? Ça va plus vite à la compilation ! Par exemple, pour compiler ce document, TEX passe autant de temps à lire les packages d'extension qu'a compiler les pages ![56]

Histoire d'être plus complet, fournissons un rapide exemple :

```
\begin{alltt}
Une \'equation en <<verbatim>>:
$f(x)=x^2, f'(x)=2x$
La m\^eme en mieux:
\( f(x)=x\sp{2}, f\sp{\prime}(x)=2x\)
\end{alltt}
```

Produira :

`Une équation en <<verbatim>>:`
`$f(x)=x^2, f'(x)=2x$`
`La même en mieux:`
$f(x) = x^2, \;\; f'(x) = 2x$

[56] Je t'accorde que ce document comporte plus de packages — beaucoup plus — que le rapport moyen.

`plain`	C'est l'équivalent du «`theorem`» de LaTeX de base.
`break`	L'en-tête du théorème est séparé du corps par un changement de ligne.
`marginbreak`	Comme `break` mais le numéro du théorème est dans la marge.
`changebreak`	Comme `break` mais en échangeant le titre et le numéro.
`change`	Comme `plain` mais en échangeant le titre et le numéro.
`margin`	Comme `plain` mais le numéro est mis dans la marge.

TABLEAU 59: Styles de théorèmes acceptés par le package `theorem`

On notera donc les commandes `\sp` et `\sb` pour exposant (superscript) et indice (subscript) puisque les caractères `^` et `_` ont perdu leur sens spécial.

27.6 theorem

L'environnement standard LaTeX étant limité ou assez complexe à utiliser, Franck Mittelbach [72] a concocté un petit package d'extension qui, bien que ne permettant pas de réaliser plus simplement ce que j'aime utiliser comme environnement pour les théorèmes, permet toutefois un paramétrage assez sympathique et devrait pouvoir être étendu par mes bons soins le jour où j'en aurai le courage et le besoin.

L'idée forte de ce package est qu'un théorème (ou tout ce qui s'en rapproche : définition, lemme, proposition...) c'est *un énoncé portant un nom généralement mis en évidence par des espaces et des changements de fonte.*

Pour permettre de spécifier la façon dont on doit mettre en page un tel type de «`theorem`», un package a été écrit (celui-ci) qui contient quelques commandes de réglages simples.

En premier lieu, tu dois régler le style du «`theorem`» en choisissant dans la liste qui figure sur le tableau 59. Pour cela tu utiliseras la commande

```
\theoremstyle{style a utiliser}
```

Cette commande effectue un changement global, c'est à dire que tous les types de `theorem` qui seront définis après seront dans ce style là, jusqu'à la prochaine apparition de cette commande.

Une autre commande permet de choisir la fonte du corps du théorème. Attention, il faut indiquer la fonte à l'ancienne manière et non pas avec les commandes du NFSS. Cette commande est

```
\theorembodyfont{\upshape}
```

pour décider que les prochains `theorem` seront en caractères droits au lieu d'être en italique par défaut. Attention, le «chemin» de fonte est pris depuis la fonte par défaut, tu peux donc être amené à spécifier des choses un poil plus complexes. Si un argument vide lui est passé, on revient à la fonte par défaut.

Une troisième commande permet de choisir la fonte de l'en-tête du `theorem`. Elle fonctionne comme la précédente :

```
\theoremheaderfont{\scshape}
```

Une telle déclaration ne doit apparaître que dans le préambule et ne saurait en rien être modifiée. Si tu souhaites que tes lemmes et tes définition n'aient pas la même fonte pour leur en-tête, il te faudra revenir aux anciennes méthodes qui sont toujours valables.

Un exemple :

```
Dans le preambule:
\theoremheaderfont{\scshape}
\theoremstyle{break}
\theorembodyfont{\upshape}
\newtheorem{Def}{D\'efinition}[subsection]

Dans le texte:
\begin{Def}[Suite convergente]
$\forall(u_n)\in\mathbb{R}^\mathbb{N}\;\;\;
\forall\ell\in\mathbb{R}$ si
\[ \forall\varepsilon\in\mathbb{R}_+ \;\;\;
\exists n_0\in\mathbb{N} \;\;\;
\forall n\in\mathbb{N} \;\;\;
n\geqslant n_0\Rightarrow
|u_{n}-\ell|\leqslant\varepsilon \]
On dira que $u_n$ converge vers $\ell$

\noindent On notera:
\[ \lim_{n\rightarrow +\infty}u_n=\ell \]
\end{Def}
```

DÉFINITION 27.6.1 (SUITE CONVERGENTE)
$\forall(u_n)\in\mathbb{R}^\mathbb{N}\;\;\;\forall\ell\in\mathbb{R}$ si

$$\forall\varepsilon\in\mathbb{R}_+\;\;\;\exists n_0\in\mathbb{N}\;\;\;\forall n\in\mathbb{N}\;\;\;n\geqslant n_0\Rightarrow|u_{n}-\ell|\leqslant\varepsilon$$

On dira que u_n converge vers ℓ
On notera :

$$\lim_{n\rightarrow +\infty}u_n=\ell$$

27.7 Le package multicol

Ce somptueux package est un petit bijou offert par Frank Mittelbach [71]. Il te permettra sans aucune difficulté de placer des textes sur une, deux, trois ou $n \leq 9$ colonnes dans un texte et même de changer en cours de document. C'est, à mon sens, l'un des plus beaux packages pour LaTeX 2_ε.

Faisons un petit test.

Faux titre pour faire semblant

Ce package fonctionne assez simplement, on passe le nombre de colonnes comme paramètre à l'environnement et éventuellement un titre à mettre sur toute la largeur et ensuite tout travail tout seul. On a aussi la possibilité de spécifier si les colonnes doivent être balancées (de même longueur sur la dernière page) ou non ainsi que quelques autres détails sur la mise en page.

L'une des options de ce package est la possibilité de faire apparaître un texte en-travers des colonnes comme ci-avant «Faux titre pour faire semblant». On a même la possibilité de mette des vrais instructions LaTeX utiles et tout.

Une autre option de l'environnement est la hauteur minimale pour commencer les colonnes, pour éviter de se retrouver avec une ligne sur quatre colonnes en bas de page toute seule, alors qu'elle aurait très bien pu aller batifoler dans les pages suivantes avec ses copines.

Ça se présente comme ça :

```
\begin{multicols}{3}[coucou][6cm]
```

pour faire du texte sur 3 colonnes, écrire «coucou» au dessus des trois colonnes et prévoir au moins six centimètres de libre pour commencer les colonnes (en comptant dans les 6cm la hauteur du «coucou»).

27.8 `subeqnarray`

Ce petit package, que nous devons à Johannes Braams [11] permet bien des jolies choses pour la manipulation des équations en mode mathématique.

Un environnement (`equation`, que j'ai peu l'habitude d'utiliser) permettait déjà de numéroter les équations. Comme par exemple :

$$f(x) = \sin(x^2) \tag{1}$$

Il existait même un environnement, `eqnarray`, permettant de mettre plusieurs équations dans un tableau :

$$\begin{aligned} f(x) &= \sin(x^2) && (2) \\ &= \frac{e^{ix^2} - e^{-ix^2}}{2i} && (3) \\ &= \frac{e^{ie^{x\log(2)}} - e^{-ie^{x\log(2)}}}{2i} && (4) \end{aligned}$$

Mais, si tu regardes avec une attention soutenue ce que tu viens de lire, tu te rendras compte bien vite qu'il ne s'agit que d'une seule équation en trois volets, et qu'il n'y a pas de raison de donner de nouveau numéro à chaque volet de cette unique équation. On peut alors indiquer qu'il s'agit d'une seule équation et la rentrer sous forme d'un tableau, mais alors il n'y a pas de moyen simple de référencer chaque volet... Bref un problème insoluble.

Enfin presque. Il y a maintenant le package `subeqnarray` qui permet d'entrer une équation en plusieurs volets en les sous-numérotant. Un exemple rapide.

$$\begin{aligned} f'(x) &= (x^2)'\cos(x^2) && (5a) \\ &= 2x\cos(x^2) && (5b) \\ &= 2x\frac{e^{ix^2} + e^{-ix^2}}{2} && (5c) \end{aligned}$$

On peut, très facilement, parler du volet 5b de l'équation 5. D'ailleurs voici le source de l'équation 5 :

```
\begin{subeqnarray}
f^\prime(x) & = & (x^2)^\prime\cos(x^2)
                  \label{eq}\slabel{seq1} \\
          & = & 2x\cos(x^2) \slabel{seq2}\\
    & = & 2x\frac{e^{ix^2}+e^{-ix^2}}{2}
                  \slabel{seq3}
\end{subeqnarray}
```

`\slabel` est une variante de `\label` qui au lieu de sauvegarder bêtement le dernier compteur incrémenté sauvegarde le numéro de la dernière sous-équation (ou du dernier volet). C'est bête comme chèvre, mais il fallait y penser.

27.9 Le package `acronym`

Ce package, que nous devons à Tobias Oetiker [86] permet d'utiliser des acronymes[57] de manière relativement rationnelle.

L'idée de base est relativement simple. Dans un article ou un rapport où on utilise plein d'acronymes barbares (genre un truc sur les GSM ou sur les réseaux) il est de bon ton de donner la définition complète de l'acronyme une fois de temps en temps, par exemple la première fois qu'on l'utilise. De même, lorsque leur sens n'est pas évident et intuitif (ce qui est fréquent), on aime à avoir une liste des acronymes avec une explication sur chaque quelque part dans le document.

Ce package permet de réaliser ce travail automatiquement et assez joliment. Il repose sur un environnement et cinq commandes. L'environnement est celui où l'on définit le sens de chaque acronyme et c'est lui qui produira la liste complète. Généralement on le place à la fin du document (en annexe, par exemple) ou au début, dans une introduction ou un chapitre zéro (genre «vocabulaire utile» ou «pré-requis»). Les commandes permettront de rappeler les acronymes sous forme complète (avec le rappel du sens), sous forme abrégée (sans le rappel du nom), de faire un choix automatique entre les deux, de définir un nouvel

[57] Pour ceux qui savent pas ce que c'est, SNCF est un acronyme pour «Société Nationale des Chemins de Fer». Dans des articles scientifiques, on en utilise souvent plein.

acronyme dans l'environnement, ou définir un acronyme en dehors de l'environnement (pour qu'il ne soit pas dans la liste).

La commande `\ac` déterminera automatiquement si l'acronyme doit apparaître sous forme longue comme ceci Société Nationale des Chemins de Fer (SNCF) parce que c'est la première fois qu'il est appelé ou comme cela SNCF parce que ce n'est pas la première fois. On peut tout à fait forcer le choix du système, par exemple en utilisant `\acs` pour obtenir la forme courte comme SNCF ou une forme longue en utilisant `\acf`, où le `f` signifie «full» comme ici Régie Autonome des Transports Parisiens (RATP).

Pour définir les acronymes utilisables, il y a deux approches. Soit on les définis dans l'environnement `acronym` et ils sont dans la liste des acronymes, comme SNCF ou bien on les définit en dehors comme c'est le cas pour Régie Autonome des Transports Parisiens (RATP). Dans l'environnement `acronym`, on utilise la commande `\acro` pour définir chaque nouvel acronyme, en dehors, on utilise la commande `\acrodef`. Leurs syntaxes sont les suivantes :

\acro{acro}{nom complet}{explication}
\acrodef{acro}{nom complet}

Évidemment, pour `\acrodef`, il n'y a pas d'explication puisque l'explication n'apparaît que dans la liste des acronymes.

Pour définir mes deux acronymes de test, j'ai fait appel aux commandes suivantes :

```
\subsection*{Acronymes utilis\'es}
\begin{acronym}
\acro{SNCF}{Soci\'et\'e Nationale des Chemins
de Fer}, organisme publique dont le but officiel
est de transporter des voyageurs et dont le
but officieux est de bloquer l'Ile-de-France en
cas, fr\'equent, de gr\`eve. La \acs{SNCF} est
en effet l'un des rares organismes publiques \`a
\^etre r\'eguli\`erement en gr\`eve.
\end{acronym}
\acrodef{RATP}{R\'egie Autonome des Transports
Parisiens}
```

Et ça produit ça :

Acronymes utilisés

SNCF Société Nationale des Chemins de Fer, organisme publique dont le but officiel est de transporter des voyageurs et dont le but officieux est de bloquer l'Ile-de-France en cas, fréquent, de grève. La SNCF est en effet l'un des rares organismes publiques à être régulièrement en grève.

27.10 Le package `parallel`

Ce package, que nous devons à Matthias Eckermann [37], permet de résoudre un problème fréquemment posé par des utilisateurs de LaTeX : écrire un texte en deux langues différentes sur deux colonnes en parallèle, de manière à ce que les traductions soient en vis-à-vis quasiment ligne à ligne.

On peut cependant imaginer d'utiliser ce package pour synchroniser deux textes quelconques.

Ce package est relativement simple d'emploi. Il repose sur un environnement et trois commandes. L'environnement s'apelle `Parallel` et indique que l'on est en train de synchroniser deux textes. La première commande, `\ParallelLText` indique le texte qui devra apparaître dans la colonne de gauche, la seconde `\ParallelRText` indique celui qui doit apparaître dans la colonne de droite, et la troisième `\ParallelPar` doit[58] permettre d'indiquer les points de synchronisation importants.

L'environnement `Parallel` prends deux arguments : la largeur désirée pour chacune des colonnes. En effet, si l'un des textes est plus long que l'autre (par exemple peu de commentaires et un long texte à commenter) alors il est plus prudent de lui allouer une plus grande largeur de colonne pour éviter de perdre de la place. Les deux commandes `\ParallelLText` et `\ParallelRText`, pour leur part, prennent un argument : le texte sur lequel elles s'appliquent.

Voici, par exemple, un texte agrémenté de son commentaire :

```
\begin{Parallel}{0.49\linewidth}{0.49\linewidth}
\ParallelLText{Pour bien \'etudier la synchronisation
des textes, il convient d'\'etudier plusieurs cas. Le
premier, et de loin le plus simple, est celui o\`u les
deux textes ont une longueur voisine.}
\ParallelRText{Pour commenter et ilustrer ce cas pr\'ecis,
il me faut faire un commentaire d'une longueur voisine
de celle du texte pr\'ec\'edent, c'est-\`a-dire quatre
ligne et quelques.}
\ParallelPar
\ParallelLText{Le second cas int\'eressant est celui du
d\'es\'equilibre total, par exemple, comme ici, un texte
fort long, sur plusieurs paragraphes d'ailleurs, et
tr\`es sobrement comment\'e, puisque c'est le jeu.
```

[58] La doc est en allemand, ce qui fait que j'ai pas tout compris les détails, vu que je parle pas un mot d'allemand.

```
Le second paragraphe se doit, lui aussi, d'\^etre relativement
long, au moins par rapport au commentaire qui en sera fait.
Le commentaire ne comptant que quelques mots, quelques
lignes devraient suffir a ma d\'emonstration.}
\ParallelRText{Rien \`a redir sur le premier paragraphe.

Ni sur le second.}
\ParallelPar
\ParallelLText{Enfin, le cas extr\^eme qui m'int\'eresse
est celui o\`u l'un des deux textes est vide. Pour ce qui
nous concerne, il s'agira du texte de droite. Donc rien
dans la colonne de commentaires.}
\ParallelRText{}
\ParallelPar
\end{Parallel}
```

Pour bien étudier la synchronisation des textes, il convient d'étudier plusieurs cas. Le premier, et de loin le plus simple, est celui où les deux textes ont une longueur voisine.	Pour commenter et ilustrer ce cas précis, il me faut faire un commentaire d'une longueur voisine de celle du texte précédent, c'est-à-dire quatre ligne et quelques.
Le second cas intéressant est celui du déséquilibre total, par exemple, comme ici, un texte fort long, sur plusieurs paragraphes d'ailleurs, et très sobrement commenté, puisque c'est le jeu. Le second paragraphe se doit, lui aussi, d'être relativement long, au moins par rapport au commentaire qui en sera fait. Le commentaire ne comptant que quelques mots, quelques lignes devraient suffir a ma démonstration. On notera cependant que les paragraphes ne s'alignent pas spontanément.	Rien à redir sur le premier paragraphe. Ni sur le second.
Enfin, le cas extrême qui m'intéresse est celui où l'un des deux textes est vide. Pour ce qui nous concerne, il s'agira du texte de droite. Donc rien dans la colonne de commentaires.	

28 Interaction LaTeX Mathematica

Nous devons ce paragraphe à la précieuse et amicale collaboration d'Olivier Gutknecht. *Qu'il en soit vivement remercié, et que des générations d'ESIEEéns lui vouent un culte sans faille pour cette précieuse aide.*

La collaboration entre Mathematica et LaTeX peut même aller encore plus loin[59]. En effet, si on le lui demande gentiment, Mathematica peut parler TeX. Imaginons qu'on veuille prendre une belle fonction, et utiliser Mathematica pour la dériver, puis inclure les deux fonctions dans son rapport préféré.

Commençons par définir notre première fonction F :

```
In[1]:= F = Sin[x]/x

         Sin[x]
Out[1]= ------
           x
```

Et maintenant, dérivons-la par rapport à x en une fonction G :

```
In[2]:= G = D[F,x]

         Cos[x]   Sin[x]
Out[2]= ------ - ------
           x        2
                   x
```

[59] Que la simple exportation de graphiques en PostScript vue à la section 31.3 page 89

Comment demander la forme TEX de ces expressions? Facile, il n'y a qu'à utiliser la commande `TeXForm`, qui va te traduire ton expression en code TEX que tu n'auras plus qu'à inclure dans le source de ton document LATEX :

```
In[3]:= TeXForm[F]

Out[3]//TeXForm= {{\sin (x)}\over x}

In[4]:= TeXForm[G]

Out[4]//TeXForm= {{\cos (x)}\over x} -
{{\sin (x)}\over {{x^2}}}
```

Et ça va donc te donner au final ceci :

$$F = \frac{\sin(x)}{x}$$

$$G = \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2}$$

Magique, non[60]?

[60]En fait pas tant que ça. En effet, un problème grave, détecté sur le rapport d'un élève (oui, JC, le tien) est que Mathematica génère du code TEX, et fait donc appel à la primitive `\over`, alors que, officiellement, LATEX préfèrerait que l'on utilise `\frac`. En fait, à la base, LATEX reconnait encore très bien la primitive `\over`, mais le package `amsmath` à le mauvais goût de la redéfinir pour un autre usage. D'où l'on conclut que le code TEX produit par Mathematica ne pourra pas être inclus directement dans un document utilisant le package `amsmath`. Toutefois il me semble que le sous ensemble le plus intéressant d'`amsmath` (celui sur les symboles, `amssymb`) ne pose pas de problème. Dans la pratique, on prendra garde à l'utilisation conjointe des produits de l'$\mathcal{AMS}$ et du code Mathematica.

Images

29 Dessins avec LaTeX

29.1 Le plus bestial : LaTeX pur

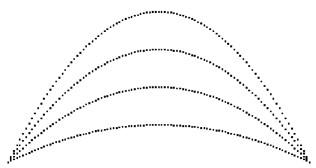


Figure 5: Petit dessin, merci Bézier !

Tu le sais, LaTeX sait aussi faire des petits dessins, mais comment lui demande-t-on de le faire, te demandes-tu, en me le demandant ? Il suffit de lui demander POLIMENT. Avant tout, lui donner la taille de ton dessin, puis les coordonnées du point de référence (le coin en bas à gauche de ton dessin) avant de lui dire ce qu'il faut dessiner.

Tu as la possibilité de dessiner plusieurs choses : des lignes dans quasiment tous les sens et de presque toutes les longueurs, des cercles, mais pas trop grands, des courbes de Bézier discrètes (figure 5), des boites avec ou sans texte ...

29.1.1 L'environnement `picture`

Voyons les diverses déclarations. Tout d'abord le début de l'environnement `picture` : il commence comme tous les autres par un `\begin` et se termine par `\end` :

```
\begin{picture}(x1,y1)(x2,y2)
...
\end{picture}
```

Ici, le couple `(x1,y1)` représente la dimension du dessin dans l'unité courante et le couple `(x2,y2)` représente les coordonnées du point en bas à gauche de ton dessin. C'est à dire que LaTeX va te prévoir une boîte de `x1` sur `y1` avec des coordonnées variant de `x2` à `x2+x1` en abscisse et de `y2` à `y2+y1` en ordonnée.

En règle générale, tout objet figurant dans un dessin y est introduit par

```
\put(x,y){objet}
```

29.1.2 Lignes

Le premier objet que nous verrons est la ligne, qui se demande par `\line` avec la syntaxe suivante :

```
\put(x,y){\line(a,b){n}}
```

Où `(x,y)` est le point de départ de la ligne et `(a,b)` le taux d'accroissement relatif, c'est à dire que lorsque l'on avance de `a` unités sur l'abscisse, on avance de `b` unités sur l'ordonnée. Ainsi un accroissement relatif de `(0,1)` indique une ligne verticale croissante à partir du point d'origine, et un accroissement relatif de `(-1,-1)` indique une diagonale partant vers le bas et vers la gauche depuis le point d'origine. Attention : toutes les pentes ne sont pas possibles, et de plus il faut indiquer les accroissements avec des entiers premiers entre eux[61]. Pour obtenir des pentes bizarres, se reporter aux courbes de Bezier. La valeur `n`, pas forcément entière, indique la longueur de la ligne d'une façon assez spéciale : pour les diagonales, `n` indique la longueur projetée sur l'abscisse (longueur toujours positive), alors que pour les lignes horizontales ou verticales, c'est la longueur normale.

29.1.3 Cercles

Le second objet que nous verrons est le cercle. Sa syntaxe d'appel est

```
\put(x,y){\circle{a}}
```

Où le point de référence `(x,y)` est le centre et `a` est le rayon. Là encore toutes les dimensions ne sont pas possibles, des cercles de plus de deux centimètres de diamètre ne sont généralement faisables qu'avec des courbes de Bézier.

29.1.4 Textes

Le troisième objet que nous allons voir est la boîte rectangulaire de texte (avec cadre). Sa syntaxe d'appel est :

```
\put(x,y){\framebox(a,b){Texte}}
```

Où `(a,b)` est la dimension du cadre dans lequel le texte vient apparaître. Il existe des options pour régler à sa convenance le centrage du texte dans le boite, mais je te les donne pas.

Le quatrième objet que je vais survoler est la boîte de texte sans cadre : c'est tout comme `\framebox` sauf que ça s'appelle `\makebox`.

[61] Explications: mail `weberj`

29.1.5 Flèches

Le cinquième objet que nous allons survoler est le `\vector` (flèche). Une flèche s'obtient de façon tout à fait similaire à celle dont on obtient les lignes, si ce n'est que l'on tape `\vector` au lieu de `\line`.

29.1.6 Courbes de Bézier

Enfin, voyons les courbes de Bézier. La syntaxe est assez différente de celle des autres objets :

```
\bezier{n}(x1,y1)(x2,y2)(x3,y3)
```

Tout d'abord il faut savoir que LaTeX ne sait pas, spontanément, tracer de courbe de Bézier. Par contre il sait tracer un point. Alors une macro commande lui a été ajoutée pour placer un nombre `n` de points sur la courbe de Bézier définie par les trois points dont les coordonnées sont passés en paramètres. Le point `(x1,y1)` est le point de départ de la courbe et `(x3,y3)` est le point d'arrivée. Le point `(x2,y2)` que je vais appeler «point de contrainte» joue un rôle décisif : c'est lui qui indique comment la courbe passe du point de départ au point d'arrivée. Il est l'intersection le la tangente au départ avec la tangente à l'arrivée. Voir un exemple sur la figure 6 dont le code source est le suivant :

```
\unitlength 1mm
\begin{picture}(80,100)(0,0)
\bezier{400}(0,0)(40,100)(80,0)
\put(0,0){\line(2,5){40}}
\put(40,100){\line(2,-5){40}}
\end{picture}
```

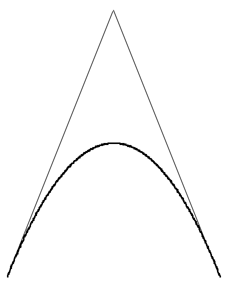
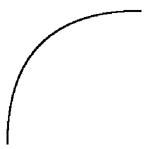


FIGURE 6: Courbe de Bézier : exemple

Ce que tu viens de voir, ce sont les courbes de Bézier selon LaTeX 2.09. Une nouvelle mouture des courbes de Bézier est arrivée avec LaTeX 2_ε. Tout d'abord cette macro est inclue dans le noyau LaTeX 2_ε et ne nécessite donc plus de chargement indépendant.



A été produit par

```
\qbezier(0,0)(0,50)(50,50)
```

On peut, en option, passer le nombre de points à placer pour réobtenir la même chose qu'avec l'ancienne commande. En plus, l'ancienne commande avec sa vieille syntaxe[62] est toujours valide[63].

Deux applications bizarres des courbes de Bézier sont souvent utilisées : obtenir une droite de pente quelconque, ou obtenir un cercle de rayon quelconque. Pour la droite, c'est simple : il suffit de placer le point de contrainte dans l'alignement et de demander suffisamment de points pour obtenir l'impression de la continuité lors de la visualisation. Pour le cercle, par contre, il faudra biaiser un peu plus. Un quart de cercle approximatif est obtenu en demandant un angle droit entre les deux tangentes. Cependant en réunissant quatre quarts de cercle, on obtient quelque chose qui ne ressemble plus tellement à un cercle : il est «tassé» aux quatre points de jonction. La solution ? Si tu la trouves, tu me la files.

29.1.7 Exemple

Un exemple assez complet de toutes les commandes est en figure 7 page ci-contre. Le code complet pour l'obtenir est :

```
\unitlength=1mm
\begin{picture}(80,100)(-10,-10)
\put(-10,0){\vector(1,0){80}}
\put(0,-10){\vector(0,1){100}}
\put(30,30){\framebox(40,10){ Origine }}
\put(30,30){\vector(-1,-1){29}}
\put(30,57){\makebox(15,5){axe}}
\put(30,57){\vector(-1,0){29}}
\end{picture}
```

29.2 Le package `bar`

Le très joli package `bar` qui nous est offert par Joachin Bleser et Edmund Lang [81] te permettra de réaliser directement sous LaTeX 2_ε de très beaux diagrammes. Il y a pas mal de commandes, et je ne les explique pas toutes.

Commençons par la plus simple : `\bar`[64] c'est elle qui viendra générer chacune des barres de l'histogramme. Elle demande deux arguments obligatoires, à savoir sa hauteur et le type de hachuré qu'on veut voir apparaître dessus parmi les huit possibles. Un argument optionnel permet d'indiquer le texte attaché à la barre de l'histogramme.

La commande `\hlineon` indique que le fond du dessin devra être quadrillé.

[62] Celle exposée en premier.

[63] La preuve : je l'ai utilisée.

[64] Il est à noter que cette commande écrase et remplace l'accent mathématiques `\bar`. C'est assez désagréable. Le package `ESIEE` prévoira très vite une solution de remplacement optimale. Reste à savoir laquelle ...

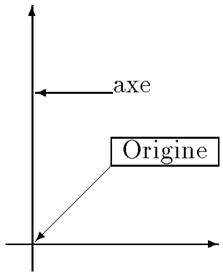


FIGURE 7: Exemple de dessin

La commande `\legend` permet de générer des petits carrés contenant le hachuré indiqué et d'écrire à côté un texte. Ça sert à faire des légendes comme son nom l'indique.

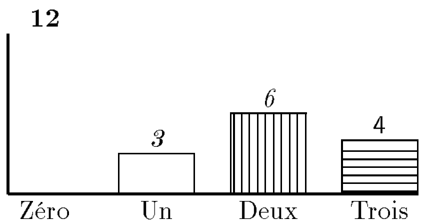
Pour les graphes en 3D, la commande `\setdepth` permet d'indiquer la profondeur du graphe sous la forme d'un entier supérieur à dix. `\sethspace` permet d'indiquer l'espacement entre les barres. C'est exprimé comme une partie de la largeur de la barre, ainsi `0.5` indiquera que si les colonnes ont une largeur de 1, LaTeX devra laisser un espace de 0,5 entre elles. `\setlinestyle` permet de choisir le type de ligne qui formeront le quadrillage en fond de figure. Deux valeurs sont possibles `solid` et `dotted`.

La largeur des barres est donnée en nombre de points en argument à la macro `\setwidth`. L'échelle de représentation est à priori fixe mais on peut facilement lui appliquer un facteur multiplicatif (en hauteur) en utilisant `\setstretch`. Si on veut changer de fonte en cours de graphe, il faut utiliser `\setstyle` pour cela.

Un premier exemple pour se faire la main et pour saliver un peu :

```
\begin{barenv}
\sethspace{0.5}
\setwidth{28}
\setstretch{5}
\setstyle{\bfseries}
\bar{12}{0}[Z\'ero]
\setstyle{\itshape\bfseries}
\bar{3}{1}[Un]
\setstyle{\itshape}
\bar{6}{2}[Deux]
\setstyle{\sffamily}
\bar{4}{3}[Trois]
\end{barenv}
```

donnera l'histogramme suivant.



Note que le style de texte n'est valable que pour le prochain texte produit.

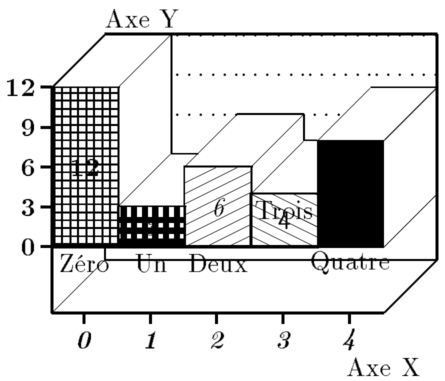
D'autres options? Oui mon lapin, régale toi. `\setdepth` te permettra de régler la profondeur du graphe (il sera alors en 3D). La valeur qui lui est passée doit au moins valoir dix. Pour graduer les axes `\setxaxis` avec trois arguments : le début, la fin et le pas. `\setyaxis` réalise le même exploit pour l'axe des ordonnées. Toutefois un premier argument, optionnel, permet d'indiquer un offset par exemple pour que le bas de l'axe ne soit pas gradué. `\setxname` permet de donner un nom à l'axe des x et son homologue `\setyname` fait la même chose pour l'axe des y.

Enfin, ultime détail, `\setnumberpos` permet d'indiquer la position du nombre associé à la barre de l'histogramme. Les valeurs admises sont «empty», «axis», «down», «inside», «outside», et «up».

Un dernier exemple et c'est marre :

```
\begin{barenv}
\setwidth{25}
\setstretch{5}
\setdepth{20}
\setnumberpos{inside}

\setstyle{\bfseries\itshape}
\setxaxis{0}{4}{1}
\setstyle{\large}
\setxname{Axe X}
\setstyle{\bfseries}
\setyaxis[5]{0}{12}{3}
\setyname{Axe Y}

\hlineon

\setstyle{\bfseries}
\bar{12}{4}[Z\'ero]
\setstyle{\itshape\bfseries}
\bar{3}{5}[Un]
\setstyle{\itshape}
\bar{6}{6}[Deux]
\setstyle{\sffamily}
\bar{4}{7}[Trois]
\setstyle{\ttfamily}
\bar{8}{8}[Quatre]

\end{barenv}
```

Pour obtenir

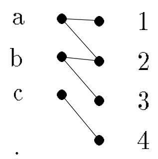


C'est-y clair dans ta tête maintenant ? LaTeX sait faire du dessin. Et en plus il n'y a rien en PostScript là dedans, ce qui rend l'exercice bien plus périlleux.

Attention : Ce package vient remplacer l'accent mathématiques `\bar`, il conviendra d'y faire attention (cf. note 64 page 82 à ce sujet).

29.3 Le package `eclbip`

Ce petit package, écrit par Hideki Isozaki [47], un ricain, permettra aux plus matheux de taper les cours de Weber puisqu'il permet de produire ça :



qui, si mes souvenirs sont bons, n'est ni injectif, ni surjectif.

Le principe est simple, il repose sur :

1°) Un environnement : `bipartite`

Il demande 5 arguments :

- `wll` *width of labels left*. C'est la largeur prévue des étiquettes pour la partie gauche.
- `wg` *width of gap*. C'est la largeur de l'espace entre les deux parties du graphe.
- `wlr` *width of labels right*. Faut vraiment te faire un dessin ?
- `hg` *height of gap*. Hauteur maximale entre deux nœuds successifs.
- `wnl` *width between a node and its label*. Y'a qu'a traduire.

On déclare :

```
\begin{bipartite}{wll}{wg}{wlr}{hg}{wnl}
```

2°) Trois commandes :

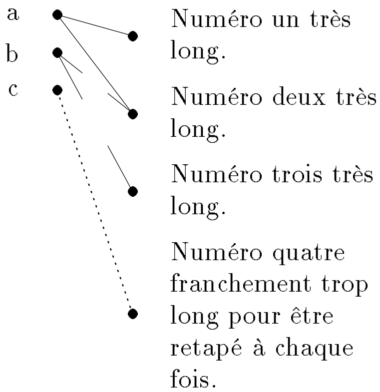
- `\leftnode[na]{nom}` ajoute un nouveau nœud à gauche dont l'étiquette sera `nom` et auquel on se référera par son nom ou par son nom abrégé (`na`) s'il est donné.
- `\rightnode[na]{nom}`. J'aime pas me répéter.
- `\match{left}{right}`. Trace une ligne entre l'étiquette `left` du côté gauche et l'étiquette `right` du côté droit. On utilisera volontiers des noms abrégés si les étiquettes sont trop longues.

3°) Les commandes de base :

- `\brush{outil}` pour indiquer avec quel outil il faut tracer les prochaines lignes. Tous les outils de `epic` peuvent être utilisés.
- Petits rappels : quelques outils de `epic`

  `\drawline` : ligne pleine.
  `\dashline[n1]{n2}` : ligne «tirets».
  `\dottedline{n}` : ligne pointillée.

Vite fait, un exemple :



Le source, pour que tu puisses bien tout comprendre :

```
\begin{bipartite}{1cm}{2cm}{3cm}{3mm}{5mm}
\leftnode{a}
\leftnode{b}
\leftnode{c}
\rightnode[1]{Num\'ero un tr\`es long.}
\rightnode[2]{Num\'ero deux tr\`es long.}
\rightnode[3]{Num\'ero trois tr\`es long.}
\rightnode[4]{Num\'ero quatre franchement
trop long pour \^etre retap\'e \`a chaque
fois.}
\brush{\dashline{50}}
\match{b}{2}
\match{b}{3}
\brush{\dottedline{3}}
\match{c}{4}
\brush{\drawline}
\match{a}{1}
\match{a}{2}
\end{bipartite}
```

Ce package fait partie des beaux restes de LaTeX2.09, et n'a pas encore été adapté à LaTeX 2_ε (au jour où j'écris). Il faut donc charger *préalablement* les packages `epic` et `eepic`. Le jour où `eclbip` sera adapté à LaTeX 2_ε, il s'en chargera tout seul. Ça vaut peut-être le coup de vérifier en commençant par ne *pas* charger les packages dont il a besoin.

29.4 Le package `ecltree`

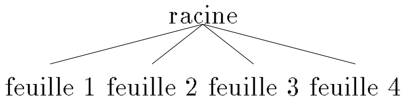
Ce petit package, lui aussi écrit par Hideki Isozaki [48] a visiblement été développé en même temps que le précédent. Il sert à dessiner des arbres n-aires. Comme le précédent, ce package n'a pas encore été adapté aux normes LaTeX 2_ε, d'où ton attention foudroyante alliée à ton esprit prodigieux déduit qu'il faut charger au préalable les deux packages `epic` et `eepic`, dans cet ordre.

Le principe de fonctionnement est bête comme chèvre. Il repose sur un environnement (`bundle`) et une commande (`\chunk`). L'environnement décrit un nœud de l'arbre, alors que la commande décrit une branche. Soyons plus clair. Entre le `\begin{bundle}` et le `\end{bundle}` se trouvent des `\chunk` qui correspondent à autant de branches partant du nœud. Le nom du nœud est le paramètre de l'environnement et le paramètre de `\chunk` est le nom de la feuille.

Un exemple tout de suite pour comprendre :

```
\begin{bundle}{racine}
\chunk{feuille 1}
\chunk{feuille 2}
\chunk{feuille 3}
\chunk{feuille 4}
\end{bundle}
```

devrait en toute logique produire ceci :

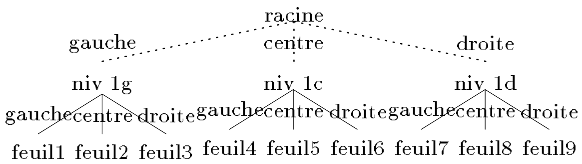


Si en guise de nom de feuille tu redonnes un nœud d'arbre, tu obtiens une structure beaucoup plus rigolote que je te laisse le soin de découvrir plus loin.

L'étape suivante est de mettre des noms non plus sur les nœuds et les feuilles, mais aussi sur les branches, par exemple pour indiquer la signification d'une relation. Pour ce faire, on indique le nom de la branche comme paramètre optionnel (entre crochets). Gardons notre souffle pour l'étude d'un cas simple ultérieurement.

Enfin, l'ultime raffinement, le choix du type de tracé pour les branches. Le truc s'appelle `\drawwith` et permet d'utiliser les types de tracé fournis par `epic` et `eepic`. C'est à dire, les mêmes que pour `eclbip`.

Voyons tout de suite l'exemple un peu tordu que je te promets depuis fort longtemps. Il s'agit d'un bête petit arbre ternaire. D'abord le résultat :



Regardons tout de suite le code source correspondant :

```
\begin{bundle}{racine}
\chunk[gauche]{\begin{bundle}{niv 1g}
               \chunk[gauche]{feuil1}
               \chunk[centre]{feuil2}
               \chunk[droite]{feuil3}
               \drawwith{\drawline}
               \end{bundle}}
\chunk[centre]{\begin{bundle}{niv 1c}
               \chunk[gauche]{feuil4}
               \chunk[centre]{feuil5}
               \chunk[droite]{feuil6}
               \drawwith{\drawline}
               \end{bundle}}
\chunk[droite]{\begin{bundle}{niv 1d}
               \chunk[gauche]{feuil7}
               \chunk[centre]{feuil8}
               \chunk[droite]{feuil9}
               \drawwith{\drawline}
               \end{bundle}}
\drawwith{\dottedline{3}}\end{bundle}
```

Tu me feras très vite remarquer quelques petits points délicats, comme par exemple que l'on spécifie le type du tracé en dernier. En fait on peut le spécifier où l'on veut, mais il faut savoir qu'il n'est évalué qu'à la fin, ce qui est logique, avant de pouvoir tracer les branches, il faut connaître la taille des feuilles. Donc en déclarant les `\drawwith` en fin de nœud, on est sûr qu'il n'y aura pas de problème de redéfinition en cours de route. Oui, me diras-tu, toi qui est toujours aussi perspicace que tout à l'heure, mais ça me dit toujours pas comment qu'on fait pour que deux branches du même nœud soient tracées avec des styles différents, vu que si je spécifie un changement de style entre deux nœuds, seul le dernier sera pris en compte et évalué à la fin. Tout d'abord sache que tu es fort intelligent d'avoir trouvé tout seul le problème et que je suis très bon de te donner la bidouille.

Lorsque LaTeX va évaluer le `\drawwith` au moment du tracé des branches, il va exécuter les ordres qui sont dedans. Si on met un ordre de changement de tracé ET un `\drawwith` alors LaTeX exécute les deux, c'est à dire qu'il change de tracé et qu'il enregistre qu'il DEVRA changer de tracé après.

Transformons l'arbre précédent en arbre binaire, avec les deux relations *dessous* et *coté*. On obtient la figure 8 page suivante.

Certes, j'en conviens l'arbre obtenu n'est pas des plus discrets, mais il correspond à ce qu'indique la théorie pour convertir un arbre n-aire en arbre binaire. Là où cela tourne au gag, c'est sur la longueur du code, mais bon, ça c'est pas grave.

Bon, le code de l'exemple, pour que tu ne te sentes pas frustré :

```
\setlength{\GapDepth}{1.5cm}
\setlength{\GapWidth}{0.5cm}
\drawwith{\drawwith{\drawline}\dottedline{3}}
\begin{bundle}{racine}
\chunk[dessous]{
  \begin{bundle}{niveau 1g}
  \chunk[dessous]{
    \begin{bundle}{feuille1}
      \chunk[dessous]{}
      \chunk[cot\'e]{
        \begin{bundle}{feuille2}
          \chunk[dessous]{}
          \chunk[cot\'e]{feuille3}
        \end{bundle}}
    \end{bundle}}
  \chunk[cot\'e]{
```

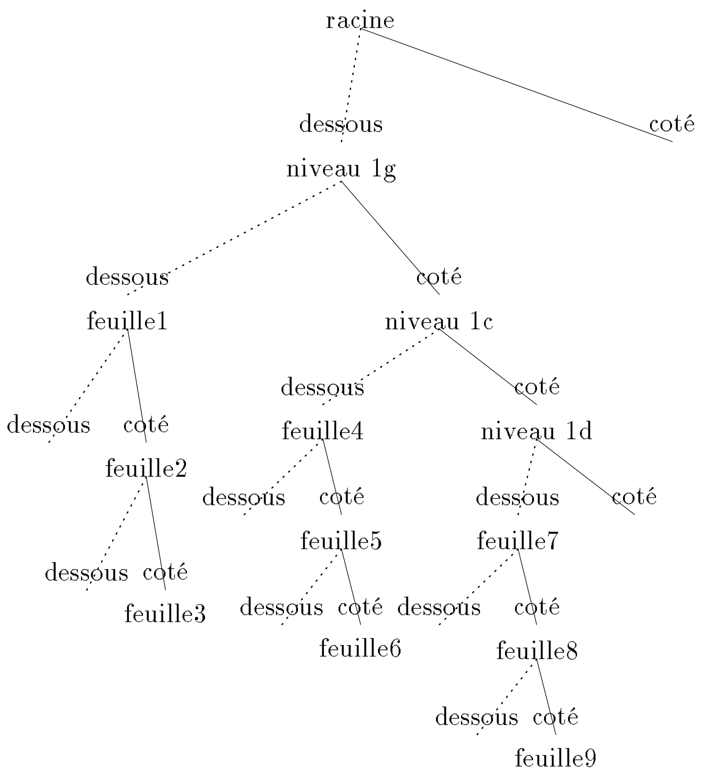


FIGURE 8: Exemple d'arbre complexe avec `ecltree`

```
\begin{bundle}{niveau 1c}
  \setlength{\GapDepth}{1cm}
  \chunk[dessous]{
    \begin{bundle}{feuille4}
      \chunk[dessous]{}
      \chunk[cot\'e]{
        \begin{bundle}{feuille5}
          \chunk[dessous]{}
          \chunk[cot\'e]{feuille6}
        \end{bundle}}
    \end{bundle}}
  \chunk[cot\'e]{
    \begin{bundle}{niveau 1d}
      \chunk[dessous]{
        \begin{bundle}{feuille7}
          \chunk[dessous]{}
          \chunk[cot\'e]{
            \begin{bundle}{feuille8}
              \chunk[dessous]{}
              \chunk[cot\'e]{feuille9}
            \end{bundle}}
        \end{bundle}}
      \chunk[cot\'e]{}
    \end{bundle}}
\end{bundle}}
  \end{bundle}}
\chunk[cot\'e]{}
\end{bundle}}
```

Une petite remarque : si le `\drawwith` avait été placé à la fin comme je le préconisais précédemment, alors il n'aurait pas été transmis aux différents nœuds de l'arbre, seule la racine aurait eu deux tracés différents pour ses deux branches.

Saches, enfin, que pour que cet arbre ne soit pas trop crado, j'ai du ré-adapter certains paramètres d'espacement. Ces paramètres d'espacement sont au nombre de trois :

GapDepth Hauteur de l'espace entre deux nœuds consécutifs.

GapWidth Largeur de l'espace entre deux nœuds adjacents.

EdgeLabelSep Hauteur du nom de la branche, mesurée depuis le bas de la branche.

30 Dessins à inclure

Le plus souvent, on préferera ne pas coder tous ses dessins en LaTeX bestialement, parce que ça demande quand même pas mal de temps. C'est pourquoi il est possible d'inclure des dessins fait avec d'autres logiciels plus adaptés.

Pour cela, deux solutions sont à envisager.

30.1 Inclusion de code LaTeX

Certains logiciels sont directement capables de produire du code compréhensible par LaTeX. C'est, entre autres, le cas de Xfig et gnuplot.

On utilisera alors les commandes prévues par LaTeX en standard pour inclure le fichier ainsi produit :

```
\begin{figure}
\input nom_du_fichier
\caption{Titre de la figure}
\end{figure}
```

30.2 Inclure du PostScript (graphics)

Pour l'inclusion des dessins PostScript encapsulés, il est des choses à retenir, comme la disparition (ou plus exactement l'incompatibilité) de l'ancien `epsf` qui permettait tant de choses de manière assez simple. Toutefois, un ramplaçant est arrivé.

Il s'appelle `graphics`. Ce package nous est offert par David Carlisle [18]. Il offre quelques commandes rigolotes, selon le driver qui lui est imposé. Le driver de l'ESIEE est `dvips`. On aura donc soin d'inclure le package comme cela :

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

La première commande importante est celle permettant d'inclure un graphique dans un document LaTeX 2$_\varepsilon$.

```
\includegraphics{fichier.ps}
```

viendra placer à l'endroit courant le dessin contenu dans le fichier qui est passé en paramètre.

30.3 Extensions rigolotes

Le premier réglage, mais aussi probablement l'un des plus intéressant, est celui permettant de définir en interne le chemin de recherche des fichiers contenant les images. Par défaut, LaTeX 2$_\varepsilon$ cherchera ses images aux mêmes endroits que les fichiers sources TeX et LaTeX. C'est un peu restrictif. En indiquant :

```
\graphicspath{{/EPSF/}{images/}{./}}
```

tu indiques à LaTeX 2$_\varepsilon$ trois voies de recherche possibles : le répertoire `EPSF` se trouvant à la racine, le répertoire `images` se trouvant dans le répertoire courant et le répertoire courant (celui-ci est tout à fait inutile, il est toujours pris en compte). A priori la liste est balayée dans l'ordre indiqué.

On peut aussi indiquer des extensions par défaut, par exemple, si je spécifie

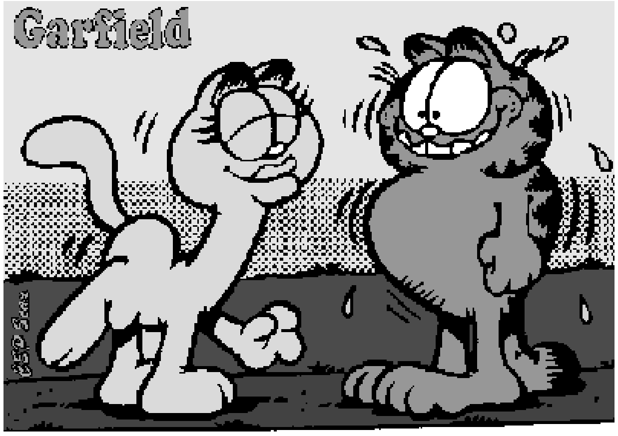
```
\DeclareGraphicsExtensions[.]{toto,ps,eps}
```

indique au système qu'il doit chercher dans cet ordre, si aucune extension n'est indiquée, les fichier `.toto`, puis `.ps`, puis enfin `.eps`. Si aucun n'est présent, il indiquera un fichier absent. .]toto,ps,eps

Ainsi, le petit dessin ci-dessous est inclus par

```
\includegraphics{Garf_02c}
```

et LaTeX se chargera tout seul de savoir que c'est un `.ps`.



comme me le montre le fichier `.log` que je ne souhaite pas te montrer.

Le package `graphics` est d'ores et déjà prévu pour être étendu, par exemple en lui apprenant a gérer d'autres types de fichiers qui seront à inclure avec des directives spécifiques. A priori rien d'autre que le PostScript n'est prévu par les drivers (que ce soient des drivers d'écrans ou d'imprimantes) de l'ESIEE qui sont les plus standard (`dvips` et `xdvi`). Cette déclaration n'a donc que peu d'intérêt, si ce n'est de permettre de futures extensions. Je t'en donne tout de même la syntaxe :

```
\DeclareGraphicsRule{ext}{type}{readfile}{command}
```

Le système utilisera cette règle de lecture pour tous les fichiers qui porteront l'extension `ext` (éventuellement détectée avec les commandes vues précédemment). Ces fichiers seront rattachés au type générique `type`. Par exemple, si `ps` et `eps` ne sont pas forcément gérés exactement de la même façon, ils sont toujours indiqués pareillement au driver final, seul leur prétraitement sera changé (présence ou non d'une BoundingBox, par exemple). Le troisième paramètre, `readfile` indique l'extension du fichier à lire réellement lors de la compilation LaTeX pour obtenir les informations. Par exemple, ton fichier `toto.ps` qui ne contient pas de BoundingBox, tu as la possibilité d'en mettre une dans `toto.bb` (faite à la main ou interpolée par ailleurs) et d'indiquer à LaTeX `bb` en guise de `readfile` pour qu'il lise `toto.bb` et qu'il trouve tout seul ses infos. Le driver (qui lira le `.dvi`) pour sa part ne verra que `toto.ps`. C'est assez pratique comme tu le verras.

Le dernier paramètre, à savoir `command` permet d'indiquer le texte qui sera passé au driver (via le `.dvi`) en fonction du nom qui sera passé à la routine d'inclusion du graphique.

Un exemple bien sympathique :

```
\DeclareGraphicsRule{ps.Z}{eps}{ps.bb}{`zcat #1}
```

fait en sorte que

```
\includegraphics{toto.ps.Z}
```

vienne bien inclure le bon fichier (équivalent de `toto.ps`) si sa BoundingBox est bien dans `toto.ps.bb`. On peut tout a fait écrire le joli petit script suivant sous Unix, et alors ça devient carrément beau :

`angle`	Permet de spécifier l'angle de la rotation.
`width`	Permet d'imposer la largeur.
`height`	Impose la hauteur.
`scale`	Indique un facteur de zoom qui devra être le même en horizontal et en vertical.
`clip`	Indique si on doit clipper le graphique ou non. On peut spécifier `clip=true` ou `clip` tout seul.
`draft`	Indique un mode brouillon. Ça marche comme `clip`.

Tableau 60: Clefs utilisables par `graphicx`

```
#!/bin/tcsh
set nom=‘dirname $1‘/‘basename $1 .ps‘
grep %%BoundingBox $nom.ps > $nom.ps.bb
compress $nom.ps
```

C'est bête comme chèvre mais ça marche souvent assez bien et c'est pas mal du tout à utiliser.

30.4 Rotation (PostScript)

Le package `graphics` permet d'effectuer la rotation de n'importe quelle boîte de texte (ou de dessin) sous LaTeX 2_ε.

C'est vachement dur à utiliser :

```
\rotatebox{angle}{boite}
```

où `angle` est en degrés dans le sens trigo.

Un exemple : exemple obtenu avec :

```
\rotatebox{45}{exemple}
```

Facile.

Théoriquement le point de rotation est le point à gauche de la ligne porteuse de la boîte.

30.5 Zoom (PostScript)

Deux méthodes : soit on indique le(s) facteur(s) de zoom et LaTeX calcule la taille finale, soit on indique la taille finale et LaTeX calcule le(s) facteur(s) de zoom. Il n'y a aucune raison que le zoom soit le même horizontalement et verticalement.

Tout d'abord indiquons le facteur de zoom :

```
\scalebox{hscale}[vscale]{boite}
```

Si `vscale` n'est pas spécifié alors sa valeur par défaut est celle de `hscale`.

Ensuite regardons le cas où l'on spécifie la dimension :

```
\resizebox{largeur}{hauteur}{boite}
```

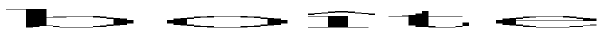
Si l'une des deux longueurs est remplacée par « ! » alors elle est calculée de façon à ne pas déformer l'objet (respect du rapport d'échelle). Si on spécifie `\resizebox*` comme nom de commande, alors c'est la hauteur totale (hauteur plus profondeur) qui sera prise en compte et non plus la hauteur seule.

Comme d'habitude lors de la manipulation des boîtes, les longueurs `\height`, `\depth`, `\width` et `\totalheight` désignent respectivement la hauteur, la profondeur, la largeur et la hauteur totale de la boîte avant zoom. Ainsi

```
\resizebox{\linewidth}{\height}{bo\^ite}
```

viendra étirer la boîte sur toute la ligne sans en changer la hauteur.

Juste comme ci-dessous :



30.6 Une syntaxe plus intuitive encore : le package `graphicx`

Les commandes définies dans `graphics` aussi puissantes qu'elles soient ne sont toutefois pas toujours très simples à utiliser. Le package `graphicx` se charge de simplifier la syntaxe de bien des commandes.

L'idée de base est d'affecter des valeurs à des mots clef. Par exemple pour que le dessin soit de la même largeur que la ligne où il se trouve :

```
\includegraphics[width=\linewidth]{fichier.ps}
```

On peut aussi demander la largeur d'une demi-ligne, mais centré :

```
\begin{center}
\includegraphics[width=0.5\linewidth]{fichier.ps}
\end{center}
```

D'autres clefs que `width` peuvent être utilisées. Elles sont référencées dans le tableau 60.

Attention, l'ordre dans lequel on spécifie les choses peut influer fortement sur le résultat. Par exemple

```
[height=1in,angle=90]
```

indiquera une LARGEUR de un pouce puisque le zoom est effectué avant la rotation, alors que

```
[angle=90,height=1in]
```

fera le contraire, et l'image aura donc un pouce de HAUT.

31 Production des dessins à inclure

31.1 Xfig

Plusieurs formats sont possibles pour récupérer un dessin fait depuis Xfig :

1. Le code LaTeX à inclure. Pour l'obtenir, dans le menu `Export` de Xfig, choisit le format «`LaTeX picture + eepic macros`» dans le menu «`Export`», indique un nom de fichier, puis exporte.
2. Le code PostScript. Pour l'obtenir, c'est pareil, sauf qu'il faut choisir «`Encapsulated Postscript`».

La grande différence entre les deux est que PostScript permet tous les dessins là où LaTeX en interdit certains (les grisés, par exemple). Donc, par prudence, si tu dois simplement travailler à l'école (où tu disposes d'une bonne imprimante PostScript), utilise de préférence l'encapsulé PostScript. Par contre, si tu es amené à travailler sur ton ordinateur personnel, le code LaTeX sera accepté par toutes les imprimantes.

31.2 Gnuplot

C'est un logiciel gratuit (offert par GNU, tout comme Xfig) qui permet de tracer tout un tas de courbes à partir d'une fonction ou d'un fichier de valeurs. Il fonctionne sur à peu près tous les types d'ordinateurs.

Pour obtenir une sinusoïde[65], c'est facile :

```
plot sin(x)
```

Pour plus d'informations :

```
help plot
```

Ou plus généralement

```
help
```

Là encore, le choix du type de fichier à produire et donc de la façon de l'inclure après s'offre à toi.

1. Le PostScript

   ```
   set term postscript portrait
   set output "mon_fichier.ps"
   plot sin(x)
   set term X11
   ```

   La dernière ligne servant à repasser en mode normal, pour visualiser d'autres courbes à l'écran, par exemple.

2. Le code LaTeX

   ```
   set term latex
   set output "mon_fichier.tex"
   plot sin(x)
   set term X11
   ```

   Qui fonctionne similairement.

31.3 Mathematica

Commençons par produire le graphique (juste un exemple, pas plus) :

```
A={{1,9,38,79,102,79,38,9,1}}
B=Transpose[A].A
ListPlot3D[B]
```

Et on voit à l'écran le joli dessin de la figure 9 page suivante. Maintenant que nous obtenons la courbe, générons le fichier PostScript à inclure dans le document LaTeX :

1. Traçons un graphique. Par exemple :

   ```
   In[15]= ListPlot3D[B]
   Out[15]= -- Surface Graphics --
   ```

2. Exportons-le en pseudo PostScript, ce que j'appelle le `mps` :

   ```
   Display["fichier.mps",%15]
   ```

3. Une fois sorti de Mathematica, convertissons en PostScript :

   ```
   psfix -epsf fichier.mps > fichier.ps
   ```

4. Reste plus qu'à inclure pour obtenir la figure 9 page suivante.

31.4 Khoros

Pour obtenir un fichier depuis Khoros qui fonctionne avec LaTeX, procéder comme suit :

1. Choisir l'option `Print image` dans le menu `Output`.
2. Indiquer comme commande d'envoi à l'imprimante

   ```
   | grep -v initgraphics > /.../nom.fic.ps
   ```

3. Penser à mettre sur `NO` l'option `Force page output`.

Le résultat de tout ce travail pourra être l'image de la figure 10 page suivante.

31.5 xv

Ce logiciel est capable de lire à peu près tous les formats d'images classiques (`gif`, `jpg`, `tiff`, et bien d'autres). Il est capable de sauvegarder en PostScript avec tout un tas d'options et de réglages possibles. Le seul point important est de rester en PostScript, le reste n'est qu'une question de goût.

[65] Des fois que t'aurais pas tout bien compris, les commandes données ici concernent gnuplot et non pas LaTeX. Donc elles sont à utiliser avec gnuplot, pas avec LaTeX.

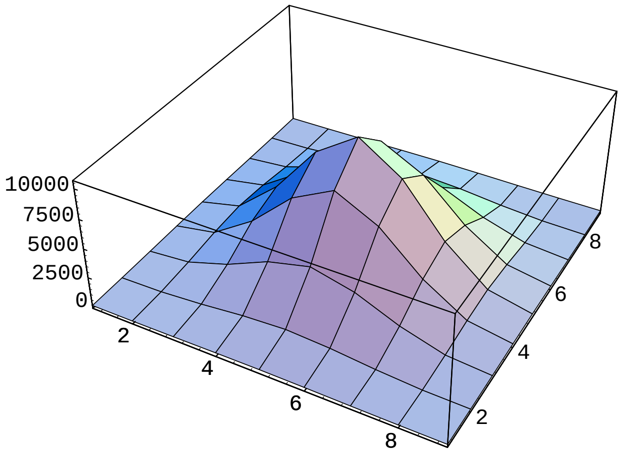


Figure 9: Graphique Mathematica

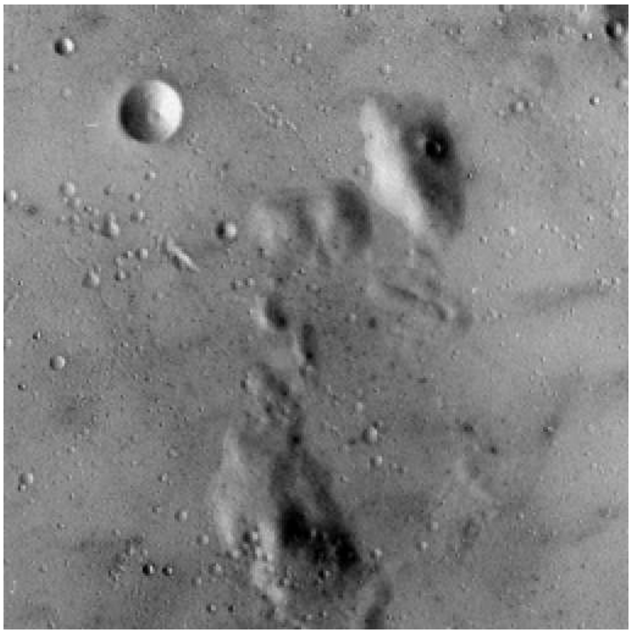


Figure 10: Image exportée depuis Khoros

32 Schémas électroniques

Il existe un vieux reste de LaTeX 2.09, écrit autrefois pas Dieter Jurzitza [54] qui fonctionne presque encore avec LaTeX 2_ε et qui permet de faire des schémas électriques basés sur les diodes, les transistors, les résistances, les bobines et quelques autres petites choses. Je donne juste un exemple pour montrer que ça marche. De toute façon c'est assez lourd à utiliser et personnellement je préfère de très loin dessiner tout ça avec Xfig. De plus il existe des choses plus récentes et plus efficaces pour faire le même travail. Simplement, j'ai pas eu le temps de les explorer, alors c'est pas dans cette doc.

Si un jour il y a un autre fascicule racontant d'autres extensions de LaTeX 2_ε, alors ça sera sûrement dedans.

Toujours est-il que pour utiliser ce truc il te suffit d'inclure le fichier `e_symbol` par exemple avec un `\input e_symbol`

L'exemple, c'est la figure 11 page 93 et le code source c'est le suivant. Oui, moi aussi j'ai trouvé ça rigolo que les noms des symboles soient donnés en allemand...

```
\unitlength8mm
\begin{picture}(15,14)
     \thicklines
     \put(0,0){\makebox(10,14){}}
     \put(2.5,0.5){\line(1,0){7.0}}
     \put(9.6,0.5){\circle{0.2}}
     \put(9.0,0.8)%
          {%
          $-V_B$%
          }%

     \put(1.5,13.5){\line(1,0){8.0}}
     \put(9.6,13.5){\circle{0.2}}
     \put(9.0,13.0)%
          {%
          $+V_B$%
          }%

     \put(2.5,0.5){\line(0,1){0.5}}

     \vwiderstand{2.5}{1.0}
     \put(2.8,1.5)% Widerstand R1
          {%
          $R_1$%
          }%

     \put(2.5,3.0){\line(0,1){0.5}}

     \biptrans{2.5}{3.5}{l}{n}{n}
     \put(3.3,3.5)%
          {%
          $T_3$%
          }%

     \put(2.5,5.0){\line(0,1){2.5}}
     \put(2.5,7.5){\circle*{0.1}}

     \put(1.5,7.5){\line(1,0){2}}
     \put(1.5,7.5){\line(0,1){1}}

     \fet{0.5}{8.5}{r}{n}
     \put(0.8,8.0)%
          {%
          $T_1$%
          }%

     \put(0.2,9){\line(1,0){0.3}}
     \put(0.1,9.0){\circle{0.2}}
     \put(1.5,9.5){\line(0,1){4}}
     \put(0.0,9.3)%
          {%
          $-E$%
          }%

\put(3.5,7.5){\line(0,1){1.0}}

\fet{3.5}{8.5}{l}{n}
\put(3.8,8.0)%
     {%
     $T_2$%
     }%

\put(3.5,9.5){\line(0,1){1.5}}

\vwiderstand{3.5}{11}
\put(3.8,11.5)%
     {%
     $R_2$%
     }%

\put(3.5,13){\line(0,1){0.5}}
\put(3.5,13.5){\circle*{0.1}}

\put(3.5,10.5){\circle*{0.1}}
\put(3.5,10.5){\line(1,0){2.5}}

\put(4.5,13.5){\circle*{0.1}}
\put(4.5,13.5){\line(0,-1){1.0}}

\vkondensator{4.5}{11.5}
\put(4.8,11.5)%
     {%
     $C_1$%
     }%

\put(4.5,10.5){\circle*{0.1}}
\put(4.5,10.5){\line(0,1){1.0}}

\put(5.5,13.5){\circle*{0.1}}
\put(5.5,13.5){\line(0,-1){0.5}}

\vwiderstand{5.5}{11}
\put(5.8,11.5)%
     {%
     $R_3$%
     }%

\put(5.5,11.0){\line(0,-1){8.25}}

\zdiode{5.5}{1.25}{o}
\put(5.8,1.5)%
     {%
     $ZD_1$%
     }%

\put(5.5,0.5){\circle*{0.1}}
\put(5.5,0.5){\line(0,1){0.75}}
\put(5.5,4.25){\circle*{0.1}}
\put(5.5,4.25){\line(-1,0){2.0}}

\put(4.5,9.0){\line(0,-1){2.5}}
\put(4.5,6.5){\line(-1,0){4.3}}
\put(0.1,6.5){\circle{0.2}}
\put(0.0,6.0)%
     {%
     $+E$%
     }%

\put(7.0,13.5){\circle*{0.1}}
\put(7.0,13.5){\line(0,-1){0.5}}

\vwiderstand{7}{11}
\put(7.3,11.5)%
     {%
     $R_4$%
     }%

\biptrans{6.0}{9.75}{r}{p}{i}
\put(7.3,10.0)%
     {%
     $T_4$%
     }%

\put(7.0,9.75){\line(0,-1){0.5}}

\diode{7}{9.25}{u}
\put(7.3,8.25)%
     {%
```

```
      $D_1$%
      }%

\diode{7}{7.75}{u}
\put(7.3,6.75)%
      {%
      $D_2$%
      }%

\diode{7}{6.25}{u}
\put(7.3,5.25)%
      {%
      $D_3$%
      }%

\put(7.0,4.75){\line(0,-1){2.25}}

\vwiderstand{7.0}{1.0}
\put(7.3,1.5)%
      {%
      $R_5$%
      }%

\put(7.0,1.0){\line(0,-1){0.5}}
\put(7.0,0.5){\circle*{0.1}}

\put(8.5,13.5){\circle*{0.1}}
\put(8.5,13.5){\line(0,-1){3.25}}

\biptrans{7.5}{8.75}{r}{n}{n}
\put(8.8,9.0)%
      {%
      $T_5$%
      }%

      \put(7.0,9.5){\circle*{0.1}}
      \put(7.0,9.5){\line(1,0){0.5}}

      \vwiderstand{8.5}{7.0}
      \put(8.8,7.5)%
            {%
            $R_6$%
            }%

      \put(8.5,7.0){\circle*{0.1}}
      \put(8.5,7.0){\line(1,0){0.9}}
      \put(9.4,7.0){\circle{0.2}}

      \vwiderstand{8.5}{5.0}
      \put(8.8,5.5)%
            {%
            $R_7$%
            }%

      \biptrans{7.5}{3.75}{r}{p}{i}
      \put(8.8,4.0)%
            {%
            $T_6$%
            }%

      \put(7.0,4.5){\circle*{0.1}}
      \put(7.0,4.5){\line(1,0){0.5}}
      \put(8.5,3.75){\line(0,-1){3.25}}
      \put(8.5,0.5){\circle*{0.1}}
\end{picture}
```

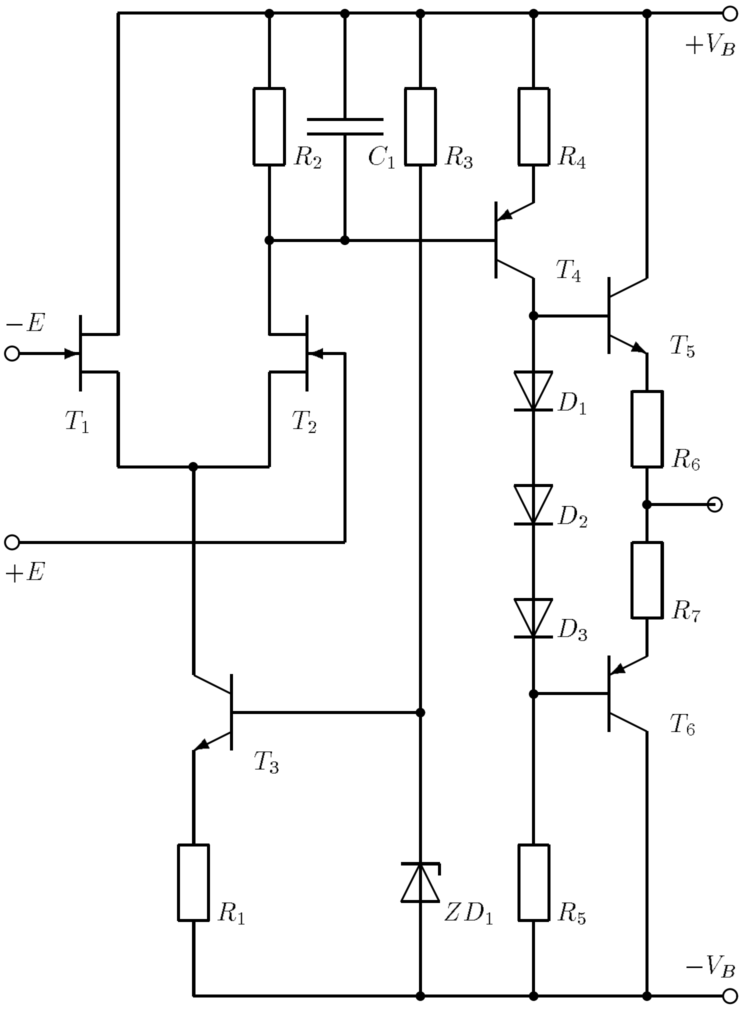


FIGURE 11: Premier essai de symboles électroniques avec LaTeX

Bibliographie, index

Toute la partie sur la bibliographie sera sur une seule colonne pour permettre aux lignes d'exemple parfois longues d'être écrites dans une taille lisible. Il eut été désagréable d'utiliser une police de corps 4 ou 5 pour des exemples importants.

33 Introduction

Bon, pour rien te cacher, tu viens d'attaquer un monstre. Produire une bibliographie a toujours été un casse-tête. Ça l'est plus encore avec LaTeX. Si tu es flemmard, c'est la seule intro dont tu aies besoin; sinon le reste n'est pas dénué d'intérêt.

Déjà les typographes d'antan n'aimaient pas les bibliographies. D'abord parce que c'est un jeux foireux de références. En effet, si dans le texte tu parles de [45] comme d'un article sur la sexualité des mouches et qu'en bibliographie la référence 45 t'indique le petit Larousse illustré, c'est que quelque chose a merdé quelque part. Pour cela, LaTeX, en générant lui-même toutes les références, t'évite tout problème.

Ensuite, chaque livre — quasiment — a sa mise en page de la bibliographie. Si tu regardes deux articles dans deux revues scientifiques différentes, tu trouveras deux présentations différentes, et c'était déjà le cas le siècle dernier. C'est ce merdier ambiant qui justifie la foultitude de styles différents.

En fait, le but ouvertement visé par la communauté des utilisateurs de LaTeX est de gérer les bibliographies comme LaTeX gère les documents, c'est à dire que tu indiques la logique du truc, et le soft se charge d'en déduire une mise en page que tu pourras changer en retouchant juste quelques détails, par exemple en redéfinissant une commande ou deux, ou plus typiquement en chargeant un autre fichier de style.

34 Approche manuelle

Avant toute chose, je la déconseille, mais tu n'es pas obligé de m'écouter. Si tu me fais confiance, passe à la section suivante[66].

L'approche manuelle est assez enfantine, elle repose sur une commande et un environnement. L'environnement : `thebibliography`, et la commande `\bibitem`. L'environnement prend un argument : la plus large des références; et la commande, elle, prend un argument optionnel (la référence) et un argument obligatoire (la clef).

Un exemple rapide :

```
\begin{thebibliography}{99}
\bibitem{Irving1} J. Irving, \textit{Le monde selon Garp}, Seuil, 1986.
\bibitem{Irving2} J. Irving, \textit{Un mariage poids moyen}, Seuil, 1988.
\end{thebibliography}
```

donnera :

Références

[1] J. Irving, *Le monde selon Garp*, Seuil, 1986.

[2] J. Irving, *Un mariage poids moyen*, Seuil, 1988.

Un doigt d'explication :

1°) J'ai pris 99 comme référence la plus large parce que j'ai supposé que ma bibliographie contiendrait moins de 100 références et qu'elles y sont numérotées depuis 1.

2°) Je n'ai pas spécifié la référence à produire parce qu'une version numérotée me satisfaisait.

3°) J'ai décrété qu'il fallait mettre le titre en italique parce que c'est l'usage.

Tu me diras bien vite qu'une biblio à laquelle on ne fait pas référence c'est débile. J'en conviens. Parlons donc de [1] et [2], ou plus brièvement de [1, 2] qui sont les deux livres de John Irving que je cite en bibliographie. Pour ce faire, trois commandes :

[66] C'est un «manuel dont vous êtes le héros».

```
\cite{Irving1}
\cite{Irving2}
\cite{Irving1,Irving2}
```

Maintenant, si tu préfères l'autre grand classique qui consiste à mettre les trois premières lettres du nom de l'auteur et l'année de parution comme référence, libre à toi :

```
\begin{thebibliography}{WWW99}
\bibitem[IRV86]{Irving3} J. Irving, \textit{Le monde selon Garp}, Seuil, 1986.
\bibitem[IRV88]{Irving4} J. Irving, \textit{Un mariage poids moyen}, Seuil, 1988.
\end{thebibliography}
```

donnera :

Références

[IRV86] J. Irving, *Le monde selon Garp*, Seuil, 1986.

[IRV88] J. Irving, *Un mariage poids moyen*, Seuil, 1988.

Sitôt que tu citeras le second livre, tu obtiendras [IRV88].

Si j'ai pris des clefs différentes c'est parce qu'il est nécessaire de différencier les deux références puisque tel est mon propos !

Voilà, tu sais tout faire. C'est à dire que tu as toute latitude pour faire ce que tu souhaites, et si jamais ton but était de respecter une mise en page imposée (par une revue, un éditeur, un chef . . .) tu es libre de te tromper !

35 Les commandes LaTeX

La première commande à retenir est celle permettant de faire une citation :

```
\cite[commentaire]{clef1,clef2,...}
```

fait référence au(x) livre(s) ayant comme clef d'accès `clef1`, `clef2` . . . avec le commentaire donné, par exemple, `\cite[page 30]{texbook}` donnera [58, page 30].

La seconde commande à retenir est celle indiquant les fichiers de la base de données bibliographiques à exploiter :

```
\bibliography{fichier1,fichier2...}
```

indique que les données utilisées sont dans les fichiers `fichier1.bib`, `fichier2.bib` . . .

La troisième commande est celle indiquant le style de mise en page et de formatage à utiliser :

```
\bibliographystyle{plain}
```

Attention, dans un même document il ne peut y avoir :

1. qu'un seul `\bibliographystyle`
2. qu'un seul `\bibliography`

chacun devant se trouver après le

```
\begin{document}
```

La commande indiquant les fichiers de base de données bibliographiques, à savoir `\bibliography`, produit aussi la biblio elle-même, c'est à dire que c'est là où se trouve cette commande que la bibliographie se trouvera.

On la trouve d'habitude en fin de document, tout près de `\bibliographystyle`.

La quatrième commande est celle permettant de faire apparaître un ouvrage en bibliographie sans le citer explicitement dans le texte. En effet, le système de génération automatique de bibliographies ne met dans ladite bibliographie que les ouvrages cités pour permettre, par exemple, l'utilisation de fichiers `.bib` beaucoup plus généraux et beaucoup plus vastes que les stricts besoins du document en cours.

Cette commande est :

```
\nocite{clef1,clef2...}
```

elle admet une variante rigolote :

```
\nocite{*}
```

qui permet de voir apparaître tous les ouvrages de tous les fichiers de la base de donnée. Un détail cependant : dans certains styles les ouvrages sont classés dans la biblio dans l'ordre où ils sont cités. Un `\nocite{*}` fait semblant de citer tous les ouvrages des fichiers `.bib` dans l'ordre où ils y apparaissent, donc si un `\nocite{*}` apparaît en début de document, les ouvrages seront classés dans l'ordre où ils sont dans les fichiers `.bib`. Alors que si le `\nocite{*}` apparait en fin de document, les ouvrages apparaîtront dans l'ordre où ils sont cités, ceux qui ne le sont pas ne venant qu'après.

36 Production de la bibliographie

Le programme clef est BibTEX. Il lit plusieurs fichiers :

1. Le fichier `.aux` pour savoir
   - les livres cités
   - l'ordre des citations
   - le style de la bibliographie à générer
   - les fichiers `.bib` à utiliser
2. Les fichiers `.bib` pour avoir les données à utiliser
3. Un fichier `.bst` définissant le style de la bibliographie.

Il produit deux fichiers :

1. un fichier `.bbl` qui contient la bibliographie finale mise en page convenablement et qui sera inclus par `\bibliography`.
2. un fichier `.blg` (`bib-log`) qui contient un listing des éventuelles erreurs trouvées (similaire au fichier `.log` produit par TEX).

Il se lance comme suit :

```
bibtex nom
```

où `nom.aux` est le fichier `.aux` principal, c'est à dire que (à priori) `nom.tex` est le fichier que tu compiles pour produire ton document.

37 Le fichier de bibliographie

Tout d'abord comprendre le principe : on travaille en indiquant à BibTEX le type du document, puis toutes les infos dont on dispose, ou, plus raisonnablement, les infos qu'on souhaite lui donner.

Un exemple pour comprendre :

```
@BOOK(texbook,
      author="Knuth, Donald Ervin",
      title="The \TeX{}book",
      publisher="Addison-Wesley",
      year=1984
     )
```

Un doigt de terminologie :

`BOOK` est un type de document, ou type d'entrée dans le fichier,
`author` est un champ de cette entrée,
`texbook` est la clef,
`"The \TeX{}book"` est le contenu d'un champ.

Quelques chiffres pour faire peur :

14 types différents,
24 champs possibles,
3 sortes de champs.

38 Les champs

Ils sont donc 24 possibles :

`address`	`edition`	`month`	`school`
`annote`	`editor`	`note`	`series`
`author`	`howpublished`	`number`	`title`
`booktitle`	`institution`	`organization`	`type`
`chapter`	`journal`	`pages`	`volume`
`crossref`	`key`	`publisher`	`year`

Certains semblent redondants (`institution` et `organization`, par exemple) ou idiots (`howpublished`). D'autres peuvent laisser dubitatif (`booktitle` et `title`) quant au choix à faire.

Débroussaillons un peu : dans un rapport de conférence, il y a plusieurs papiers ayant chacun un titre propre, alors que le livre qui les contient tous (le rapport[67]) a un titre à lui. C'est aussi dans ce cas là que le champ «`pages`» peut être utile.

Selon le type d'entrée dont il fait partie, un champs peut être :

Oligatoire : il est jugé obligatoire, et généralement on imagine mal une entrée de ce type là qui n'ait pas ce type de renseignement à fournir. Par exemple, l'auteur d'un livre. Au pire, c'est le célèbre «`anonyme`».

Optionnel : il a un sens, mais n'est pas exigé. Si on lui attribue une valeur alors il sera peut-être utilisé pour produire le texte final, mais pas forcément. Tout dépendra du type de bibliographie qui sera demandée.

Ignoré : il n'est ni obligatoire ni optionnel. On le met que si on est très rigoureux, ou pour s'en servir comme aide mémoire, genre notes de lecture[68].

Plusieurs points sont à retenir :

La saisie d'un nom : de manière rapide, disons que l'on doit respecter la forme suivante : «Nom(s), Jr., Prénoms».

BibTEX se chargera, à partir de là, de faire la mise en page nécessaire (ne mettre que les initiales du prénom, changer l'ordre ...). Quand il y a plusieurs auteurs, le mot `and` les sépare, par exemple, pour le LATEX Companion :

```
author="Mittelbach, Frank and Samarin, Alexander and Goossens, Michel"
```

La virgule est importante. En effet, il eût été équivalent d'écrire :

```
author="Frank Mittelbach and Samarin, Alexander and Michel Goossens"
```

Elle prend en fait tout son sens dans les noms longs, complexes, ou à particule. Si le `and` est entre accolades (`{and}`) alors il perd son aspect de mot clef.

Enfin, si la liste des noms est trop longue, on peut toujours contourner le problème :

```
author="Benjamin Bayart and Pascal Vincent and others"
```

Pour voir le résultat produit, il te suffirait de te référer à [7] dans la bibliographie générale de cette doc si cette fonctionnalité éxistait en français. Mais ça n'est pas le cas. En effet, en français, il est considéré comme étant de bon goût de donner la liste complète des auteurs, alors qu'en anglais on juge plutôt de bon goût de perdre les auteurs mineurs dans un générique «et al.» qui vient du latin et qui signifie «et autres». Donc cette construction est à proscrire en français puisqu'elle produit [7], ce qui n'est pas le résultat que l'on pouvait légitimement espérer.

Les accents peuvent poser problème ! Ne serait-ce que pour le classement alphabétique. Retiens simplement que, pour la typo comme pour le classement,

```
{\commandequelconque{lettre}}
```

est équivalent à cette lettre. Par exemple :

[67] Ce que les anglais appellent proceedings.

[68] Il me semble que dans certains cas extrême, il est parfois imprimé. Le cas extrême type, c'est quand on demande à LATEX de donner (via un programme et un style spéciaux) tout le contenu (toutes les infos de toutes les entrées) de la base de données bibliographiques.

```
J{\'{e}}r{\^{o}}me
```

est la façon la plus idéale de saisir le prénom Jérôme comme dans l'exemple [15]

Cette technique est aussi utilisée pour fausser les classements. Par exemple, un livre en deux tomes, le premier, paru en 1960 à été ré-édité en 1975, alors que le second (1962) ne l'a pas été. Dans ton document, tu parles de l'édition la plus récente (1975 et 1962). Si ta bibliographie est classée par auteur et par date de publication (ce qui est fréquent), le tome 2 va être avant le tome 1, ce qui n'est pas drôle. Un coup d'œil sur l'astuce ci-dessous :

```
date="{\rien{a}}1975"
date="{\rien{b}}1962"
```

en admettant que la commande `\rien` prenne un argument et n'en fasse rien, par exemple, on peut la définir avec :

```
\newcommand{\rien}[1]{}
```

Ne cherche pas à comprendre ce que ça veut dire, ça te concerne très peu. Ce n'est pas moi qui t'apprendrais à programmer des macros LaTeX, tel n'est pas mon but.

Les abréviations. En particulier, les noms d'organismes (souvent repris plusieurs fois dans une même base de données), pour être toujours écris pareillement, peuvent être abrégés, par exemple :

```
@STRING(ESIEE="\'Ecole Sup\'erieure d'Ing\'enieurs en \'Electronique et \'Electrotechnique")
```

On utilisera ensuite, le plus simplement du monde :

```
school=ESIEE
```

ou

```
organization="Groupe "#ESIEE
```

où le `#` sert d'opérateur de concaténation entre les deux chaînes. Certaines abréviations standard sont prévues : `jan`, `feb` ... pour les mois. Cela garantit qu'ils soient écris comme le demande le style de la bibliographie.

39 Les différents types d'entrée

Il y a en tout 14 types d'entrée dans un fichier de base de données bibliographique :

`article`	`inbook`	`masterthesis`	`techreport`
`book`	`incollection`	`misc`	`unpublished`
`booklet`	`inproceedings`	`phdthesis`	
`conference`	`manual`	`proceedings`	

Le tableau 61 page suivante te donne, type par type, les champs obligatoires et les champs optionnels.

En plus de ceux cités dans le tableau 61 page suivante, chaque type admet un champ `key` qui est utilisé, selon le style de la bibliographie, pour classer, faire des références croisées, ou produire la référence qui apparaît dans le texte. Par exemple, quand il n'y a pas d'auteur, on pourra prendre un acronyme à la place. Exemple :

```
@BOOK(lex-typo,
     title="Lexique des r\`egles typographiques en usage \`a l'imprimerie nationale",
     editor="Imprimerie Nationale",
     year=1994,
     edition="Troisi\`eme",
     key="IN"
     )
```

sera référencé [IN94] dans les style ou on doit (théoriquement) prendre le début du nom de l'auteur.

Le champs `note` sera utilisé, en particulier, pour les bibliographies dites «annotées[69]». Ces `notes` contiennent généralement des remarques et commentaires sur l'œuvre en question.

Relevons, pour n'y plus revenir, le type `conference` qui n'est là que par soucis (louable) de compatibilité avec un autre système appelé *Scribe*. Je ne sais pas dans quel sens fonctionne la compatibilité (BibTeX comprend *Scribe*, ou le contraire, ou les deux), et ne connais pas *Scribe*.

[69] Annotated-bibliography pour nos amis anglo-saxons

Type	Champs	
	obligatoire	optionnel
`article`	`author, title, journal, year`	`volume, number, pages, month, note`
`book`	`author` ou `editor, title, publisher, year`	`volume` ou `number, series, address, edition, month, note`
`booklet`	`title`	`author, howpublished, address, month, year, note`
`conference`	Synonyme inutile de `inproceedings`	
`inbook`	`author` ou `editor, title, chapter` et/ou `pages, publisher, year`	`volume` ou `year, series, type, address, edition, month, note`
`incollection`	`author, title, booktitle, publisher, year`	`volume` ou `year, series, type, address, edition, month, note, chapter, pages, editor`
`inproceedings`	`title, year, author, booktitle`	`editor, volume` ou `number, series, pages, address, month, organization, publisher, note`
`manuel`	`title`	`author, organization, address, edition, month, year, note`
`masterthesis`	`author, title, school, year`	`type, address, month, note`
`misc`		`author, title, howpublished, month, year, note`
`phdthesis`	`author, title, school, year`	`type, address, month, note`
`proceedings`	`title, year, author, booktitle`	`editor, volume` ou `number, series, address, month, organization, publisher, note`
`techreport`	`author, title, institution, year`	`type, number, address, month, note`
`unpublished`	`author, title, note`	`month, year`

TABLEAU 61: Champs obligatoires et optionnels pour chaque type d'entrée dans une base de données bibliographiques

40 Saisie d'une entrée

La saisie d'une entrée dans un fichier de données bibliographiques devra, généralement, suivre les étapes suivantes, même si dans leur ensemble elles représentent moins de 10 secondes de réflexion.

40.1 Choix du type d'entrée

Il n'est pas aussi évident qu'il y paraît, en effet, il ne faut pas confondre `article` et `inproceedings`. Ils ont souvent la même forme, mais l'un est tiré d'un magazine ou d'une revue scientifique, l'autre de minutes[70] de conférence.

De même, par soucis de rigueur, on différenciera `masterthesis` (équivalent du mémoire de maîtrise en France en plus gros) et `phdthesis` (thèse de doctorat) même si les deux types d'entrée ont la même structure.

Retenons aussi la différence entre `book` et `inbook` ainsi que entre `proceedings` et `inproceedings`. Dans un cas on parle de la globalité de l'ouvrage et dans l'autre part d'un extrait. En particulier, on se réfère rarement à un ouvrage très long dans son entier, ou à l'intégralité d'un compte rendu de conférence. Par exemple, on fera référence à un tome de *The Art of Computer Programming* (7 volumes, sauf erreur de ma part), ou à l'intervention d'une personne lors d'une conférence.

40.2 Recherche des références croisées

Lorsque l'on fait référence à plusieurs parties d'un même ouvrage global, par exemple plusieurs interventions à une même conférence, ressaisir toutes les informations, outre le surcoût de saisie, représente un risque majeur d'incohérence. C'est pour ce cas précis qu'est prévu le champs `crossref`. C'est BibTEX qui se chargera de gérer tout seul les références de ce type.

Relevons tout de même que les entrées qui sont ainsi référencées doivent se trouver dans le même fichier que les entrées qui y font référence, et avant celles-ci.

[70] proceedings en anglais

40.3 Choix des champs

En règle générale, on remplira, autant que faire se peut, tous les champs possibles pour une même entrée.

Une des ambiguïtées fréquentes est celle entre `editor` et `publisher`. En effet, un compte rendu de conférence IEEE édité par IEEE, n'aura pas d'auteur, et pourra être publié par Addison-Wesley ou Sybex.

On pourra aussi éviter la redondance de certaines informations, comme par exemple l'année. Un exemple (un classique du genre) :

```
@BOOK(almanach79-gnu,
      title="The 1979 GNU Almanach",
      publisher="Addison Wesley",
      editor="Free Software Foundation"
     )
```

On ne tiendra alors pas compte des protestations de BibTEX.

De même, quand les champs `editor` et `publisher` devraient contenir strictement le même texte, ce qui est très rare, on pourra envisager de supprimer l'un des deux.

40.4 Choix de la clef de référence

Elle doit absolument être unique et suffisament claire pour que l'on puisse s'en souvenir facilement, donc éviter les clowneries du genre `IEEE-nn-1991-12-1825` pour indiquer un extrait de conférence IEEE sur les réseaux de neurones[71] en 1991 (12e extrait, pages 1825 et suivantes). Il vaudra mieux retenir le thème et le nom de l'auteur, ou le titre.

[71] nn=neural network

41 Principe de réalisation d'un index

Le principe de la réalisation d'un index est dans le fond assez simple. La commande fondamentale a retenir c'est `\index`. Elle prend un argument, le mot qui doit figurer en index.

Par exemple

```
\index{index}
```

pour spécifier que c'est ici que je parle des index[72]. D'ailleurs tu n'as qu'a vérifier, le numŕo de cette page (102) figure bien au mot clef «index» dans l'index général du présent volume.

Cette commande viens ajouter une ligne au fichier `.idx` associé à ton document, cette ligne à tête suivante :

```
\indexentry{mot clef}{page}
```

Ensuite, il reste bien peu de chose à savoir pour pouvoir finaliser la réalisation de l'index, en fait, juste la mise en page.

41.1 Approche manuelle

C'est de loin la méthode la plus idiote et la plus mauvaise que je connaisse, mais bon, ca peut interesser du monde.

C'est une bidouille toute simple qui consiste a classer le fichier `.idx`, par exemple à l'aide de la commande UNIX `sort` :

```
sort < toto.idx > toto.ind
```

puis à définir la commande `\indexentry` pour qu'elle fasse la mise en page que tu attends, par exemple[73] :

```
\newcommand{\indexentry}[2]{#1, #2}
```

Enfin, il ne reste plus qu'à inclure le fichier classé à l'endroit où tu souhaites voir apparaître l'index[74] :

```
\introchapter{Index}
\begin{multicols}{2}
\input FICHIER.ind
\end{multicols}
```

Ceci dit, je n'aime pas cette approche parce qu'elle ne permet quasiment aucun rafinnement. En plus elle demande de définir soi-même des commandes plus ou moins complexe, par exemple si tu as besoin de faire un traitement particulier sur l'index, tu seras obligé de le programmer en TEX, ce qui est tout sauf simple. C'est pourquoi je préfère l'approche automatisée.

41.2 Approche automatisée, `makeindex`

Ça fonctionne un peu comme la bibliographie, mais en plus simple. En fait l'idée est la même : des macros LATEX et l'utilisation d'un programme externe pour effectuer le tri et quelques fonctions avancées de mise en page de l'index. Le principe est simple :

```
latex toto.tex
makeindex toto
latex toto.tex
```

Le programme `makeindex` permet d'effectuer le classement des entrées ainsi que quelques autres travaux. Par exemple, lorsque qu'une entrée de l'index apparaît sur les pages 11, 12, 13 et 14, au lieu de faire une entrée avec 4 numéros de page, il en produira une avec «11-14», ce qui est beaucoup plus esthétique. Ce programme produit, à partir d'un fichier `.idx`, deux nouveaux fichiers : un `.ind` qui est l'index en lui-même, et un fichier `.ilg` qui est le fichier contenant la liste des éventuelles erreurs survenues durant l'exécution[75].

De plus `makeindex` règlera la mise en page conformément à un fichier de style donné, et accepte un certain nombre d'options rigolotes. Je relèverais simplement les plus intéressantes : `-s` pour fixer le style de mise en page a utiliser, `-o` pour indiquer dans quel fichier l'index préparé doit être écrit, `-t` pour indiquer dans quel ficheir `makeindex` doit écrire la transcription des erreurs qu'il rencontre (équivalent du `.log` de LATEX).

En effet, pour générer un glossaire, le principe est très proche de celui de la génération d'index[76], et donc le même programme de classement est utilisé, à savoir `makeindex`. Mais, pour différencier le glossaire de l'index, le fichier associé, au lieu de l'extension `.idx`, porte l'extension `.glo`. Et la version classée de ce dernier sera un fichier `.cls`, donc la suite de commande pour un document toto avec un glossaire pourra ressembler à ceci :

```
latex toto
makeindex toto
makeindex -o toto.gls -t toto.glg toto.glo
latex toto
```

En effet, invoquer quelque chose comme

```
latex toto
makeindex toto
makeindex toto.glo
latex toto
```

[72] L'index du présent document est réalisé de manière un poil plus complexe puisque cette doc comporte 5 index différents. C'est plus loin, dans la section 42.4 page 104 sur le multi-indexage que j'aborderais ce type de méthode.

[73] Non, je ne répondrais pas aux questions sur «comment qu'on fait pour définir une commande ?» ou «Dis, Benjamin, ça veut dire quoi ce qu'y y a écris là ?». Je ne donne pas de cours de programmation en LATEX. Seuls les utilisateurs avertis pourront donc utiliser ce genre de bidouille, et c'est pas un mal.

[74] Pour la commande `\introchapter`, voir la section 10 page 17 à propos du package `ESIEE`, pour l'environnement `multicols`, voire la section 27.7 page 76 qui lui est consacrée

[75] `ilg`=Index LoG

[76] En gros, l'idée est d'utiliser la commande `\glossary` au lieu de `\index`, ainsi que toutes les commandes associées, c'est à dire `\makeglossary` et `\printglossary`.

pourrait bien écrire le glossaire classé dans un fichier `.ind` ce qui serait du plus mauvais effet puisque, non seulement ça écraserait le précédent, mais ça ferait apparaître le glossaire à la palce de l'index.

41.3 Insertion de l'index

Ben oui, mais bon, y'a l'index il est dans le fichier `.ind`, soit après un tri «manuel» soit après un tri «automatique», mais il est pas encore dans mon document, me diras-tu. Y fô donc l'inclure. Et pour cela une seule commande que l'on place là où l'on souhaite avoir l'index :

```
\printindex
```

Encore que, pas tout à fait. Il nous manque une commande, celle qui indique à LaTeX que durant cette compilation là, il doit générer le fichier `.idx` qui est la base de tout. En effet, la correction et la mise en page de l'index sont deux des tâches tout à fait finales lorsque l'on travail sur un document. Donc il n'est pas utile de générer les fichiers correspondant à chaque compilation. Il vaut mieux le générer à la demande. La commande indiquant qu'il est temps de générer ce fichier (donc que l'on incluera — avant le `\begin{document}` — durant les dernières phases du travail sur le document) est

```
\makeindex
```

Ces deux commandes ne sont pas toujours accessibles, en effet, il n'est pas utile d'encombrer la mémoire le LaTeX avec tout ce qui touche à l'index alors même que l'on n'en fait pas. Donc pour ajouter les deux commandes `\makeindex` et `\printindex` il faudra inclure le package `makeidx`. Ce n'est pas une extension en ce sens que ce package est livré systématiquement avec le noyau LaTeX.

41.4 Changement de fonte

Un des trucs qui fonctionne bizarement dans les index, c'est les changement de fonte. Par exemple, si je souhaite faire apparaître en index le mot «`makeindex`» dans cette typographie là, puisque c'est celle que j'ai utilisé dans tout le document, la première idée est de faire appel à :

```
\index{\texttt{makeindex}}
```

ce qui marche très peu puisque c'est alors le texte passé en index est classé comme commençant par le caractère \, ce qui n'est pas tout à fait faux ...

Il faudrait donc lui dire que le mot est à classer sous une identité, mais à faire apparaître sous une autre. Par exemple comme ceci :

```
\index{makeindex@\texttt{makeindex}}
```

Ce qui se trouve à gauche du @ est le mot utilisé pour déterminer le classement, ce qui se trouve à droite du @ est l'étiquette qu'il faut faire apparaître à l'emplacement déterminé. C'est costaud[77].

42 Conception avancée d'un index, multi-indexage

42.1 Utilisation avancée

Le plus simple des raffinements que l'on peut apporter à un index, après le changement de fonte, est la notion de sous-entrée dans un index.

Par exemple, dans l'index général du présent document, tu trouveras deux entrées (entrée1 et entrée2) chacune pourvue de trois sous-entrées (sous-entrée1, sous-entrée2, sous-entrée3 et sous-entrée4), ainsi que par-ci par-là quelques sous-sous-entrées.

Pour saisir cet exemple j'ai fait[78] :

```
\index{entr\'ee1@\textsf{entr\'ee1}}
\index{entr\'ee1@\textsf{entr\'ee1}!sous-entr\'ee1@\textsf{sous-entr\'ee1}}
\index{entr\'ee1@\textsf{entr\'ee1}!sous-entr\'ee1@\textsf{sous-entr\'ee1}!sous-sous}
\index{entr\'ee1@\textsf{entr\'ee1}!sous-entr\'ee2@\textsf{sous-entr\'ee2}}
\index{entr\'ee2@\textsf{entr\'ee2}}
\index{entr\'ee2@\textsf{entr\'ee2}!sous-entr\'ee3@\textsf{sous-entr\'ee3}}
\index{entr\'ee2@\textsf{entr\'ee2}!sous-entr\'ee4@\textsf{sous-entr\'ee4}}
```

Une seconde utilisation avancée est de préciser que l'on parle d'un mot sur toute une portion du texte sans avoir à remettre un `\index` toutes les 5 lignes. Pour cela on placera au début de la zone en court :

```
\index{mot|(}
```

et à la fin :

```
\index{mot|)}
```

C'est très pratique, par exemple pour indiquer qu'on parle d'index dans toute une partie d'une documentation.

[77] La preuve en est que le mot `makeindex` est pas trop mal classé dans l'index général du présent document.

[78] Enfin bon, pas tout à fait parce que je travaille en multi-indexage, ce qui complique un poil la manœuvre.

42.2 Renvoi d'une entrée à l'autre

Une chose relativement classique dans un index, est d'avoir deux ou trois synonymes. Générallement, on en désigne un comme étant le «vrai» mot, et les autres renvoient sur celui-là. Par exemple, on pourra prévoir qu'une entrée «Operating System» nous renvoie à l'entrée plus francophone «Système d'exploitation». Pour ce faire on utilisera le `|see` :

```
\index{Operating System|see{Syst\`eme d'exploitation}
```

42.3 Encore plus avancée

L'idée est de, maintenant, cherche à obtenir ce que j'ai mis dans l'index du présent document, c'est à dire les lettres écrites en gras (va voir et tu comprendras).

Pour ça c'est très simple, je me suis définit un fichier de style d'index et ensuite pour compiler mon index j'ai utilisé[79] :

```
makeindex -s manuel2ep toto
```

Tu souhaites écrire ton propre style d'index ? Libre à toi, tu n'as qu'à regarder ceux qui sont dans le répertoire par défaut[80].

42.4 Multi-indexage

Là c'est carrément la dimension au-dessus. Il s'agit de faire, comme moi dans cette doc, plusieurs index. En fait c'est relativement simple. Ça repose sur un package (qui n'est d'ailleurs plus maintenu) qui s'appelle `multind`. D'ailleurs en plus de n'être plus maintenu, il n'est pas documenté non plus ...

En fait l'idée directrice est bonne, et relativement simple à utiliser ou à implémenter. Le jeu consiste à utiliser autant de fichiers que d'index, à appeler `makeindex` une fois par fichier, et à préciser aux différentes commandes traitant des index sur lequel elles doivent s'appliquer.

Par exemple, dans le présent document, le fait que je souhaite réaliser 5 index est précisé dans l'en-tête :

```
\makeindex{Com}
\makeindex{Env}
\makeindex{Gal}
\makeindex{Pak}
\makeindex{Sym}
```

où ce que je spécifie comme argument est le nom du fichier dans lequel seront stockées les informations sur l'index.

Par la suite, pour ajouter une entrée à mon index général je faisais dans le document :

```
\index{Gal}{entr\'ee}
```

Enfin, pour imprimer les différents index j'ai fait appel, par exemple, à la commande :

```
\printindex{Gal}{Index g\'en\'eral}
```

Attention : Il y a un bug dans ce package sur une commande répondant au doux nom de `\see`. Pour la corriger il est bon de charger le package `ESIEE` après `multind`. En effet, de longues secondes (30 ou 40) de travail attentif m'ont permis, non pas de corriger le bug, mais de le contourner sans avoir à toucher au fichier original `multind.sty`.

[79] Enfin presque, parce que je travail en multi-index.

[80] Celui qu'utilise `makeindex` pour aller chercher ses fichiers. Je ne le précise pas ici puisqu'il est succeptible de ne pas rester à la même place.

Configuration et installation

43 La librairie kpathsea

43.1 Trouver un fichier

Savoir où TEX et ses acolytes doivent aller chercher leurs fichiers à toujours été un grand problème. En effet, KNUTH spécifie clairement que c'est lors de l'adaptation de TEX à un nouveau système d'exploitation que ce problème doit être résolu. Or, au fil des années, ces adaptations ont été faites par de nombreuses personnes qui, au début du moins, ne se sont pas concertées.

De plus, un minimum de cohérence doit être garanti si l'on souhaite s'y retrouver. Par exemple, si tous les fichiers concernant les fontes sont à un endroit de l'arborescence, tous les softs doivent le savoir :

1. TEX pour les fichiers `.tfm`,
2. METAFONT pour les fichiers `.mf`,
3. `xdvi` pour les fichiers `.tfm` et `.pk`[81]
4. `dvips` pour les fichiers `.tfm` et `.pk`.

Et je ne parle ici que des deux drivers `dvi` classiques sous UNIX. Il doit en exister une petite centaine, tous systèmes confondus.

Sous UNIX, pour espérer rendre le système cohérent, une librairie à été écrite dont le rôle est de trouver un fichier dans l'arborescence d'un réseau d'après son nom.

Cette librairie s'appelle `kpathsea`. Son utilisation garantie que toutes les applications vont bien aller chercher les fichiers au même endroit et de la même manière.

43.2 Configuration

L'idée est simple, un certain nombre de constantes sont déclarées dont les noms évoquent des types de fichiers, et leur valeur indique où les chercher.

Par exemple, la constante `PKFONTS` indique où trouver les bitmaps de fontes au format `.pk`. Si l'un des programmes à besoin du fichier `cmr10.300pk` la librairie parcourera le contenu de tous les répertoires désignés par `PKFONTS` jusqu'à trouver le bon fichier.

Pour assigner des valeurs à ces constantes, trois moyens sont possibles :

1. À la compilation, en allant mettre les doigts dans le Makefile, ou de toute autre manière similaire (ajout d'une option à `make`, par exemple).
2. Via un fichier (`texmf.cnf`) qui contient une liste des variables et de leurs valeurs. S'il existe, les valeurs déclarées dedans viennent remplacer celles spécifiées à la compilation.
3. Via une variable d'environnement portant le nom de la constante à redéfinir. Si une telle variable existe, son contenu est pris de préférence à tout autre. On réservera volontiers ce type de configuration à des corrections hyper-ponctuelles du genre «j'ai des fontes sur mon compte et je souhaite les utiliser». Remarque cependant qu'en rajoutant un : à la fin de ta variable d'environnement, son contenu sera pris avant celle par défaut et non pas à la place de celle par défaut. C'est assez agréable.

Une liste des constantes est donnée au tableau 62 page suivante.

43.3 Syntaxe

Je vais juste te décrire la syntaxe du fichier `texmf.cnf`, pour les deux autres variantes, la seule différence est qu'elles sont mono-application.

La forme générale de chaque ligne du fichier est la suivante :

```
variable[.application] = chemin1:chemin2:chemin3
```

Les variables utilisées par les applications ont, par convention, des noms tout en majuscule. Mais on peut en utiliser d'autres de manière interne au fichier comme tu le verras. La liste des variables[82] utilisées par les applications fait l'objet du tableau 63 page suivante.

On peut spécifier des contenus de variables selon l'application, par exemple, pour que `dvips` et `xdvi` ne cherchent pas leurs fichiers au même endroit. Lorsqu'une application lit le fichier, elle retient, pour une variable donnée, la première qui lui convienne. Par exemple :

```
PKFONTS.xdvi = chemin1
PKFONTS.dvips = chemin2
PKFONTS = chemin3
```

Ceci réalise bien ce à quoi on s'attend : `xdvi` fouillera chemin1, `dvips` fouillera chemin2 et toute autre application cherchant un fichier `.pk` fouillera chemin3.

Par contre

[81] Plus généralement les bitmaps (`.pk`, `.gf` et `.pxl`), mais de nos jours on ne rencontre plus guère que des `.pk`.

[82] Oui, on parle plus souvent de variable que de constantes, bien que cela soit un abus de langage. Mais bon, on s'interesse ici au moyen de fixer une valeur au bidulle, donc il se comporte comme une variable. Ceci dit, durant l'exécution d'une application donnée, il ne change pas de valeur, le bidulle, alors il se comporte comme une constante. D'où l'ambiguïté.

Variable	Sens
`TEXMF`	Position globale des librairies TeX. Utile pour changer d'un seul coup toute l'installation en cours d'utilisation.
`TEXINPUTS`	Position des fichiers de macros pour TeX
`MFINPUTS`	Position des fichiers de macros pour METAFONT
`TEXFORMATS`	Position des formats pour TeX
`MFBASES`	Position des bases pour METAFONT
`TEXPOOL`	Position du fichier `tex.pool`
`MFPOOL`	Position du fichier `mf.pool`
`VFFONTS`	Position des fontes virtuelles (`.vf`)
`TFMFONTS`	Position des fichiers de métriques de fontes (`.tfm`). Utilisé par quasiment toutes les applications liées à TeX.
`PKFONTS`	Position des fontes `.pk`
`GFFONTS`	Position des fontes `.gf`
`GLYPHFONTS`	Position du fichier `texfonts.map`
`BIBINPUTS`	Position des bases de données bibliographiques utilisées par BibTeX.
`BSTINPUTS`	Position des fichiers de style de bibliographie utilisés par BibTeX.
`TEXCONFIG`	Position des fichiers de configuration de `dvips`.
`DVIPSHEADERS`	Position des fichiers d'en-tête de `dvips`.
`TEXMFLOG`	Nom du fichier où, éventuellement, `kpathsea` listera tous les fichiers utilisés avec l'heure de la recherche.
`TEXMFCNF`	Position du fichier `textmf.cnf`.

TABLEAU 62: Liste des constantes définies pour `kpathsea`

Application	Variables utilisées
TeX	`TEXINPUTS`
	`TEXFORMATS`
	`TEXPOOL`
	`TFMFONTS`
	`TEXMFLOG`
METAFONT	`MFINPUTS`
	`MFBASES`
	`MFPOOL`
	`TEXMFLOG`
BibTeX	`BIBINPUTS`
	`BSTINPUTS`
	`TEXMFLOG`
`xdvi`	`TFMFONTS`
	`VFFONTS`
	`PKFONTS`
	`GFFONTS`
	`TEXMFLOG`
`dvips`	`TFMFONTS`
	`VFFONTS`
	`PKFONTS`
	`GFFONTS`
	`TEXCONFIG`
	`DVIPSHEADERS`
	`TEXMFLOG`

TABLEAU 63: Constantes de `kpathsea` utilisées par chaque application

```
PKFONTS = chemin1
PKFONTS.xdvi = chemin2
PKFONTS.dvips = chemin3
```

ne fera pas ce à quoi on s'attend. Toutes les applications se referreront à chemin1.

L'utilisation d'une variable se fait en mettant un $ devant son nom comme pour les variables d'environnement. Par exemple :

```
PKFONTS.xdvi = chemin1
PKFONTS.dvips = chemin2
PKFONTS = $PKFONTS.xdvi:$PKFONTS.dvips:chemin3
```

Avec une telle configuration, toute autre application que **xdvi** et **dvips** cherchera ses fichiers **pk** dans chemin1, puis dans chemin2, puis dans chemin3.

L'expression d'un chemin ressemble à celle que l'on a sous UNIX, à savoir une suite de noms de répertoires séparés par des «/». Un «//» indiquera une suite quelconque de sous-répertoires. C'est une syntaxe puissante. Par exemple on peut indiquer à TEX que ses fichiers sont quelque part sous un répertoire :

```
TEXINPUTS = /user/tex/lib/texmf/tex//
```

C'est relativement pratique. On peut même utiliser cette fonctionnalité au beau milieu d'un nom de chemin :

```
PKFONTS = /user/tex/lib/texmf/fonts//pk/
```

Cela indique de fouiller dans fonts/tmp/pk et dans fonts/public/cm/pk et dans fonts/pk, mais pas dans fonts/tmp/pk/oldies. C'est on ne peut plus puissant.

L'utilisation de variables permet d'alléger certaines syntaxes, par exemple :

```
Fontes = /user/tex/lib/texmf/fonts
PKFONTS = $Fontes//pk/
TFMFONTS = $Fontes//tfm/
GFFONTS = $Fontes//gf/
VFFONTS = $Fontes//vf/
```

Ce qui, en plus, te permet d'indiquer explicitement la logique interne de ton arborescence. C'est assez agréable. Je l'ai déjà utilisé, par exemple, pour changer d'installation TEX en passant une seule variable de /usr/esiee/tex à /user/tex.

43.4 Le programme kpsewhich

Lorsque tu souhaiteras vérifier la présence d'un fichier sur le réseau, par exemple pour savoir si tu as une fonte donnée, **find** te permettra de vérifier que le fichier existe, ou éventuellement qu'il est en double. Par contre, il ne te permettra pas de vérifier que TEX ou **xdvi** le trouvera. Pour celà, il faudrait un programme qui utilise le même algorithme de recherche. Ce programme s'appelle **kpsewhich**.

Je te donne juste un exemple d'utilisation à titre de mode d'emploi :

```
(info6) /: kpsewhich cmr10.tfm
/tmp/lib/texmf/fonts/public/cm/tfm/cmr10.tfm
(info6) /: kpsewhich graphics.sty
kpsewhich: Can't guess format for graphics.sty, using tex.
/tmp/lib/texmf/tex/latex2e/packages/graphics/graphics.sty
```

44 TEX

44.1 Introduction, formats

Je vais ici supposer que tu connais TEX, que tu sais rudimentairement t'en servir, et que tu es vivement animé par l'envie de l'installer et de le configurer. Comme tu connais déjà TEX, tu sais surement que ce que l'on utilise le plus souvent ce sont des macros-commandes et non des primitives. Tu te doutes bien que pour apprendre ces macros, TEX ne relit pas les définitions à chaque fois, ce serait trop lent.

En fait, TEX les lit une fois, puis fait une copie de sa mémoire dans un gros fichier qu'il re-lira ensuite à chaque compilation. Ce fichier s'appelle un format et porte l'extension **.fmt**. Celui que choisit TEX par défaut s'appelle **plain.fmt**. Quand on travaille avec ce format, on dit souvent que l'on travail en «plain TEX» ou encore en «TEX pur». Un autre format assez célèbre est LATEX qui est généralement dans le fichier **latex.fmt**.

La méthode officielle pour compiler un fichier avec un format donné est la suivante :

```
tex "&latex" toto.tex
```

Notons tout de suite que les guillemets servent à protéger le & qui est un caractère spécial sous UNIX. Lorsque l'on tape

```
tex toto.tex
```

la «norme» indique que l'on doit travailler en «plain TEX». Générallement, cela signifie que le fichier **plain.fmt** est recherché.

Sous UNIX, les choses sont légèrement plus subtiles. Le nom du fichier de format par défaut est déterminé par le nom du moteur appelé. Ainsi, si tu fais une copie de **tex** en **latex**, alors en tapant

```
latex toto.tex
```

TEX cherchera tout seul, spontanément, **latex.fmt**. Ce qui fait que lorsque l'on tape

```
tex toto.tex
```

il a tendance à chercher **tex.fmt**.

Note tout de suite qu'une copie n'est pas nécessaire, un simple lien symbolique suffit à faire son bonheur. Le «vrai» moteur TEX s'appelle normalement **virtex**. **tex**, **latex** et les autres, ne sont normalement que des liens symboliques vers **virtex**.

Il ne nous reste plus qu'à savoir créer un format. C'est fort simple, la primitive **\dump** rencontrée lors de la compilation de **toto.tex** entraînera une copie immédiate de la mémoire de TEX dans **toto.fmt**. À un détail près : cette primitive ne fait pas partie du dialecte TEX standard. En effet, certaines tâches ne sont effectuées que lors de la création d'un format. Par exemple, la primitive `\dump` et le chargement des motifs de césure. Pour ne pas encombrer le programme TeX avec tous ces morceaux inutiles, une version «spéciale génération de formats» existe, elle s'appelle `initex` (initialise TeX).

44.2 Génération de `latex.fmt`

Le premier gros problème qui va se poser ici est le choix des motifs de césure et la façon de les utiliser. Je décrirais ici simplement ce que j'ai l'habitude d'installer, à savoir `babel`. Pour installer autre chose, cherches les informations ailleurs, par exemple dans [40] pour le `french` de Bernard GAULLE.

Commençons par récupérer LaTeX sur un sîte `ftp`, par exemple sur

```
ftp.loria.fr/pub/ctan/macros/latex/base.tar.gz
```

Ensuite, récupérons les packages standard fournis par l'équipe du projet LaTeX3 :

```
ftp.loria.fr/pub/ctan/macros/latex/packages.tar.gz
```

Décompactons tout cela et commençons l'installation. Tout d'abord LaTeX. On se place dans le bon répertoire (`latex/base`) et on lance

```
tex unpack.ins
```

qui va, en gros, générer plein de trucs (des fichiers `.sty`, `.cfg`, `.def`, `.cls`, `.clo`, `.fd`, ...) et quelques fichiers `.ltx` dont le très joli `latex.ltx` qui contient la définition du format LaTeX.

Passons ensuite à `babel`. Pour cela, va dans son répertoire (`latex/packages/babel`) et lance l'extraction des nombreux fichiers :

```
tex babel.ins
```

On se retrouve alors avec plein de fichiers (des `.sty`, `.def`, `.ldf`, `.cfg`, ...) et en particulier `language.dat` et `lthyphen.cfg`. Le premier indiquera à `babel` quelles langues il doit apprendre, et le second indiquera à LaTeX qu'il doit utiliser `babel`. Le fichier `language.dat` de l'ESIEE ressemble à ça :

```
% File    : language.dat
% Updated at ESIEE by B. BAYART for latex2e & babel
% Purpose : specify which hypenation patterns to load
%           while running iniTeX
english  ushyph1.tex
francais  fr8hyph.dc
portuges portug.hyphen
german ghyph31.tex
italian lahyph.tex
ukenglish ukhyphen.tex
esperanto eshyph.tex
```

Fais tout de même attention, les moteurs TeX normaux ne sont pas capables d'apprendre autant de langues. Celui de l'ESIEE à été dopé tout particulièrement. Les fichiers `fr8hyph.tex` et autres `portug.hyphen` sont, eux aussi, à récupérer sur les mirroirs CTAN traditionnels, par exemple :

```
ftp://ftp.loria.fr/pub/ctan/hyphenation/...
```

Une fois que ton `language.dat` personnalisé est prêt, mets-le dans `latex/base`, ainsi que `lthyphen.cfg` en le renommant `hyphen.cfg`. Le format LaTeX est alors prêt à être généré. Dans `latex/base`, lance

```
initex latex.ltx
```

Et admire le tout joli fichier `latex.fmt`. Il ne te reste plus qu'à tout ranger à sa place.

44.3 Génération de `tex.fmt`

Alors là, c'est vachement très simple :

```
initex plain.tex
```

puis, lorsque TeX te rends la main (il te montre une `*`) tapes

```
\dump
```

et puis c'est tout. Tu récupères alors `plain.fmt` que tu n'as plus qu'à renommer en `tex.fmt`. Sans commentaire.

44.4 Arborescence des fichiers

TeX s'intéresse de près à 4 types de fichiers :

1. Les fichiers de métriques de fontes (ou `.tfm`) avec la variable `TFMFONTS` :
   `TFMFONTS=$HomeTeX/fonts//tfm/`
2. Les fichiers de formats (`.fmt`) avec la variable `TEXFORMATS` :
   `TEXFORMATS=$HomeTeX/ini/`
3. Le fichier `tex.pool` avec la variable `TEXPOOL` :
   `TEXPOOL=$HomeTeX/ini/`
4. Les fichiers de macros (`.tex`, `.sty`, `.cls`, `.clo`, `.def`, `.cfg`, ...) avec la variable `TEXINPUTS`. Note que les macros ayant un sens pour plain TeX n'en ont pas forcément pour LaTeX et réciproquement. De plus, si tu gardes un vieux LaTeX(2.09) alors ses fichiers de macros ne sont pas les mêmes que pour le nouveau. On retiendra donc :

```
latex209=$HomeTeX/tex/latex209//
latex2e=$HomeTeX/tex/latex2e//
texgeneric=$HomeTeX/tex/generic//
TEXINPUTS.latex209=.:$latex209:$texgeneric
TEXINPUTS.latex2e=.:$latex2e:$TEXINPUTS.latex209
TEXINPUTS.tex=.:$HomeTeX/tex//
```

De plus, `initex` aura besoin des sources des formats et des motifs de césure, que TeX ne saurait pas utiliser, alors autant les stocker à part :

```
TEXINPUTS.initex=.:$HomeTeX/tex//:$HomeTeX/initex//
TEXINPUTS=$TEXINPUTS.tex
```

La dernière ligne indiquant que le comportement par défaut est celui de plain TeX, ce qui est la «norme» la plus usuellement admise.

Remarque tout de suite que j'ai créé la variable HomeTeX pour que ce soit elle qui change d'une installation à l'autre (par exemple sur les stations HP de l'ESIEE, elle est positionnée à /user/tex/lib/texmf, alors que sur mon PC sous LINUX elle est positionnée à /usr/lib/texmf.

La structure d'arborescence que j'ai retenue pour les fichiers de macros de LaTeX à l'ESIEE est simple. Tout se passe dans les deux répertoires tex/latex2e et tex/latex209[83]. J'ai essayé de respecter la structure des archives CTAN le plus possible, et d'appliquer l'idée simple : un package par répertoire. On obtient alors la structure suivante :

tex/latex2e/kernel
: Pour tout ce qui touche le noyau LaTeX 2ε. Il y a ensuite plusisuers sous-répertoires : inputs pour les classes de documents et autres fichiers de config et fd pour les descripteurs de fontes (`.fd`).

tex/latex2e/packages
: Pour tous les packages «officiels» qui sont fournis par l'équipe du projet LaTeX3. On notera plusieurs sous-répertoires, un par package en gros : babel, tools (`tabularx`, `multicol`, ...), mfnfss, ...

tex/latex2e/contribs
: Pour tous les packages «non-officiels» que l'on peut trouver dans macros/latex/contrib/supported, c'est-à-dire ceux qui font l'objet d'une maintenance par leur auteur. On y trouve aussi, à l'ESIEE, les packages que j'ai développé (`ESIEE` et `ESIEEcv` par exemple) puisque j'en assure la maintenance.

tex/latex2e/othercontribs
: Pour tous les packages qui ne font l'objet d'aucune maintenance.

tex/latex209
: Pour tout ce qui touche à LaTeX 2.09. Donc, plus en détail :

tex/latex209/contrib
: Tous les packages d'extension de LaTeX 2.09 que je récupère de temps en temps sur les archives CTAN. Enfin, plus exactement ceux que j'arrive à faire marche. Pas de différence entre maintenus et non-maintenus puisque l'énorme majorité est non-maintenue au profit de LaTeX 2ε.

tex/latex209/dvips
: Les packages livrés avec `dvips` et qui sont maintenant plus ou moins remplacés par `graphics`.

tex/latex209/kernel
: Les sources pour générer les formats, en gros.

tex/latex209/musictex
: Bin, tout musictex.

Pour ce qui est de savoir installer un package, il te faudra te référer à la section 50 page 112.

45 METAFONT

45.1 Introduction, bases

METAFONT est un logiciel permettant de générer des fontes de très grande qualité, juste comme les aime TeX. Il fonctionne sur des principes très très similaires à ceux de TeX. Comme pour TeX, il y a deux versions du programme (`virmf` et `inimf`), la seconde servant à faire du précompilé pour la première. Comme pour TeX, la base (on parle ici de bases et plus de formats) à charger peut être spécifiée par un & ou par un lien symbolique.

45.2 Génération de `plain.base`

`plain.base` est la base par défaut de METAFONT. Normalement elle est créée automatiquement lors de la compilation de METAFONT, ou fournie avec l'exécutable. La procédure de création doit beaucoup ressembler à cela :

```
inimf "\input plain.mf; dump"
```

Ça peut tout de même varier un peut selon les systèmes d'exploitation (rôle des ", du point-virgule et du \).

45.3 Génération de `cmmf.base`

Bin c'est tout pareil :

```
inimf "\input cmmf.mf; dump"
```

`cmmf` est la base de référence des fontes Computer Modern, c'est-à-dire les plus utilisée. Ça permet de recompiler très rapidement ces fontes là.

45.4 Arborescence des fichiers

METAFONT ne s'intéresse qu'à deux types de fichiers : ses bases et ses sources. Les bases sont généralement au même endroit que les formats de TeX (variable `MFBASES`) et ses sources sont de deux types : ceux décrivant des macros et des bases et ceux décrivant réellement des fontes :

```
Fontes=$HomeTeX/fonts
MFBASES=$HomeTeX/ini/
MFINPUTS=$HomeTeX/mf//:$Fontes//src//
```

[83] Je parle ici de répertoires relatifs à $HomeTeX, comme tu l'auras compris.

46 xdvi

46.1 Arborescence des fontes

Les fontes utilisées par TEX, le plus souvent, sont des fontes liées à METAFONT. Il arrive, parfois, que l'on décide d'utiliser des fontes autres, par exemple PostScript ou TrueType. Ce cas d'école ne m'intéresse pas ici puisque, souvent, il revient à simuler le cas «METAFONT» par des biais plus ou moins simples.

Une fonte est, avant tout, un fichier source (`.mf`) qui sera compilé par METAFONT pour produire *un* fichier de métriques et *des* fichiers de description bitmap. Le fichier de métrique d'une fonte donne les règles d'espacement à utiliser pour la fonte, les dimensions et règles de positionnement de chaque caractère et les ligatures[84] à utiliser pour cette fonte. Un fichier de bitmap d'une fonte contient le tracé en point par point de la fonte à une résolution donnée. Le fichier de métriques porte l'extension `.tfm` alors que les fichiers de bitmap *peuvent* porter des noms finissant par `pk`, `gf` ou `pxl`. Le plus courrant de nos jours est le `pk` parce qu'il prend moins de place. Le plus souvent l'extension exacte du nom de fichier bitmap est `.résolutionpk`, par exemple `.300pk` pour un fichier à 300 dpi. Sous MS-DOS, le schéma de nommage différera légèrement. De plus une même fonte à 300 dpi n'aura pas exactement le même bitmap selon l'imprimante à laquelle il sera destiné. Par exemple les imprimates à laser et à jet d'encre n'ont pas exactement le même rendu d'un même motif. METAFONT sait en tenir compte.

Chaque ensemble de fontes est donc structuré en trois sous répertoires : `src` qui contient les sources METAFONT, `tfm` qui contient les fichiers de métriques, et `pk` qui contient les bitmaps au format `pk`. Le répertoire `pk` contiendra un sous-répertoire par mode METAFONT (par imprimante différente, en gros), ce sous-répertorie porte le nom du mode.

`xdvi` cherchera seulement deux types de fichiers sur les fontes : les `tfm` et les `pk`. À vrai dire, les `tfm` ne sont pas trop utiles puisqu'ils ne servent qu'à vérifier que la fonte trouvée par `xdvi` est la même que celle que TEX avait utilisée lors de la compilation, par exemple pour signaler les différences entre deux installations distinctes qui échangent des fichiers `dvi`. Les trois variables utiles seront donc :

```
PKFONTS.xdvi=$HomeTeX/fonts//pk/LeModePourXdvi/
GFFONTS.xdvi=$HomeTeX/fonts//gf/LeModePourXdvi/
TFMFONTS=$HomeTeX/fonts//tfm/
```

46.2 Fichier de ressources

Comme tous les programmes pour XWindows, `xdvi` fait appel à un fichier de ressources pour trouver certains de ses règlages par défaut. Pour ma part, j'ai pris l'habitude de m'en passer et de faire tous ces règlages via la ligne de commande. Ça m'évite d'avoir à demander l'accord des ingénieurs systèmes et ça permet les mêmes choses. À ce sujet, la page de `man` sur `xdvi` est tout à fait claire, je ne tiens pas à paraphraser.

47 dvips

47.1 Arborescence des fontes

`dvips` utilise la même arborescence que `xdvi`, mais pas forcément les mêmes bitmaps, ce qui fait que, souvent, on sépare la variable `PKFONTS` en deux parties :

```
PKFONTS.xdvi=$HomeTeX/fonts//pk/ModePourXdvi
PKFONTS.dvips=$HomeTeX/fonts//pk/ModePourDvips
PKFONTS=$HomeTeX/fonts//pk//
```

On procèdera de même pour `GFFONTS`.

Enfin, `dvips` cherchera ses fichiers d'en-tête PostScript et ses fichiers de config dans les deux variables suivantes :

```
TEXCONFIG=$HomeTeX/dvips
DVIPSHEADERS=$HomeTeX/dvips:$HomeTeX/fonts//type1
```

47.2 Fichier `config.ps`

`dvips` admet un très grand nombre d'options. Un grand nombre d'entre-elles sont positionnées dans le fichier `config.ps`.

Quelques options utiles à tous :

1. Mode METAFONT à utiliser pour générer les bitmaps manquants. Cette option est sur une ligne commençant par `M` :

   ```
   M CanonCX
   ```

2. Résolution de l'imprimante :

   ```
   D 300
   ```

3. Autres résolutions à chercher, faute de mieux :

   ```
   R 300,600,900
   ```

4. Offset de centrage de la page, à règler avec le document LATEX `testpage.tex`

   ```
   O -17pt,45pt
   ```

5. Fichier de sortie standard par défaut

   ```
   o fichier
   ```

   Si l'on ne spécifie rien, alors `dvips toto` créera `toto.ps` à partir de `toto.dvi`. On peut aisément spécifier un programme, comme c'est le cas à l'ESIEE :

   ```
   o |lp -dlps20
   ```

[84] Une ligature c'est, par exemple, f+i=fi

6. Mémoire de l'imprimante. **dvips** en tiendra compte pour la création de ses fichiers PostScript

   ```
   m 3000000
   ```

Le fichier `config.ps` permet aussi de spécifier les formats de papier que l'on souhaite pouvoir utiliser. En règle générale, tous les formats utiles sont déjà définis. Si ça t'intéresse de rajouter d'autres formats de papier, reporte toi à [101].

47.3 Options de ligne de commande

Le truc à savoir absolument c'est que de lancer `dvips` sans aucune option lui fait afficher la liste des options qu'il comprend. Je ne donnerais donc ici que les options les plus fréquemment utilisées :

`-f` pour le mode filtre, c'est-à-dire que le fichier PostScript généré est envoyé sur la sortie standard. Ainsi, par exemple, à l'ESIEE, pour imprimer en recto-verso :

```
dvips toto -f | lpspr -K2 | lp -dlp20
```

`-o` pour spécifier le nom du fichier PostScript à générer. Par exemple :

```
dvips toto -o fichier.ps
```

Si on ne spécifie pas le nom du fichier à générer, alors il est déduit du nom du fichier à imprimer :

```
dvips toto -o
```

créera donc `toto.ps`.

`-t` pour indiquer le format de papier que l'on va utiliser, par exemple pour du A4 en landscape :

```
dvips -t a4 -t landscape toto -o
```

La liste des formats de papier disponibles et leurs descriptions est dans le fichier `config.ps`

`-p` pour indiquer le numéro de la première page à imprimer, par exemple pour tout imprimer de la page 12 à la fin du document :

```
dvips -p 12 toto
```

`-n` pour indiquer le nombre de pages à imprimer, par exemple pour imprimer les pages 12 à 20 incluses :

```
dvips -p 12 -n 9 toto
```

Fais gaffe, c'est `-n 9` parce que de 12 à 20 ça fait 9 pages, et pas 8.

48 Makeindex

48.1 Syntaxe d'appel

Bien que cela fasse un peu redite avec la section 41.2 page 102, je te rappelle que `makeindex` est un programme utilisé pour classer les index, glossaires et autres babiolles du même accabit. Le principe est simple : tu appelle `makeindex` en lui disant quel fichier il doit classer :

```
makeidnex toto.idx
```

Parfois, comme c'est le cas dans cette doc, on souhaite que l'index ait une autre gueule que celle par défaut. Fastoche :

```
makeidnex -s fichier.ist toto.idx
```

où `fichier.ist` est un «index style[85]».

À l'ESIEE, un monstrueux script-shell a été écrit pour remplacer `fichier.ist` par le nom complet du fichier de style parce que j'ai pas trouvé la variable d'environnement à positionner pour obtenir que `makeindex` aille chercher tout seul les fichiers standard.

48.2 Configuration

Par soucis de cohérence, j'ai, pour ma part, rangé les styles d'index dans le répertoire $HomeTeX/makeindex. Je ne sais pas ce qu'indiquent les habitudes officielles en la matière.

À mon sens, il doit y avoir deux niveaux de configuration pour `makeindex` :

1. indiquer le répertoire par défaut où se trouvent les fichiers de style,
2. écrire un fichier de style.

Pour le premier point, ne voulant pas attaquer violament le source de `makeindex`, j'utilise le script suivant :

```
#!/bin/tcsh

set TeXHome=/user/tex
set istname = ' '

restart:

if ( AW$1 == AW-s ) then
  echo $1
  shift
  echo $1
  set istname = "$istname -s $TeXHome/lib/texmf/makeindex/$1"
  shift
endif

if ( AW$1 == AW-s ) then
  goto restart
endif

$TeXHome/GEUT/bin/makeindex $istname $*
```

Pour le second point, c'est relativement simple. Il faudrait voir ça dans [81].

[85] Style d'index pour les mono-francophones simplets.

49 BibTEX — arborescence des fichiers

Le «mode d'emploi» de BibTEX étant donné à la section 36 page 97, je ne parlerais ici que de son arborescence de fichiers.

BibTEX ne s'interesse qu'à deux types de fichiers[86] à savoir les `.bst` ou «bibliography style[87]» et les `.bib` qui sont les bases de données bibliographiques.

Pour trouver les `.bst` il se réfèrera à la variable `BSTINPUTS` de la librairie `kpathsea`, et, pour trouver les `.bib` à la variables `BIBINPUTS`. À l'ESIEE, ça donne ça :

```
BIBINPUTS=.:$HomeTeX/bibtex/bst//
BSTINPUTS=.:$HomeTeX/bibtex/bib//
```

50 Installer un package LATEX

50.1 Automatique dernier cri

Un des grands problèmes des programmeurs c'est de commenter leurs programmes et d'en écrire une documentation. Si c'est un problème pour toi, élève qui rends un source, sache que ça l'est pour tout le monde. Une grande tradition dans le petit monde de TEX est de voir listings et documentation partager le même fichier.

La tradition à été reprise par LATEX 2_ε. Les packages les plus modernes sont livrés sous la forme de deux fichiers : un fichier `.dtx` qui contient le package et sa documentation, et un fichier `.ins` qui permet l'installation. Ainsi, pour obtenir le package dans sa forme exploitable, à savoir un fichier `.sty` et éventuellement d'autres il faut appeler

```
latex toto.ins
```

et récupérer les fichiers utiles qui auront été générés. Moi, je les range dans un répertoire, avec comme habitude un répertoire par packages, mais c'est chacun comme il a envie.

Pour récupérer la doc, c'est pas franchement plus compliqué :

```
latex toto.dtx
```

Ça peut devenir un tout petit peu plus complexe dans certains cas, par exemple lorsqu'il y a un index et un historique des changements. Tout ou partie des lignes suivantes peut être util :

```
makeindex -s gind.ist toto.idx
makeidnex -s gglo.ist toto.glo
latex toto.dtx
```

Pour compiler automatiquement toutes les docs de ce type là, j'utilise ce script shell sous UNIX :

```
#!/bin/tcsh

foreach i ( $* )

set nom = `basename $i .dtx`

latex "\batchmode \input $i"
latex "\batchmode \input $i"

if (-f $nom.glo) then
   makeindex -s gglo.ist -o $nom.gls $nom.glo
endif
if (-f $nom.idx) then
   makeindex -s gind.ist $nom.idx
endif

latex "\batchmode \input $i"
latex "\batchmode \input $i"

end
```

Pour certains packages, le fichier intéressant sera un fichier `.drv` et non pas `.dtx`. C'est selon ...

50.2 Semi-automatique

Il s'agit de packages que je pourrais qualifier de pré-moderne. Ce sont en fait les premiers à avoir utilisé le système permettant, à partir du même fichier, d'obtenir la doc et le package et ... Ce système se nomme «`docstrip`».

En fait, ces packages supposent que tu sais appeler à la main ce «programme» d'extraction écrit en TEX.

La seule fois que j'ai eu à y fait appel, ça a été pout installer le package `bnf`. Je vais donc t'indiquer comment j'ai procédé pour ça.

Tout d'abord, j'ai lancé `docstrip` :

```
latex docstrip
```

Ensuite, je lui ai

bnf Il faut lancer docstrip a la main comme suit : latex docstrip Puis indiquer comme extension de depart 'doc' d'arrivee 'sty' comme option 'style' et comme fichier 'bnf'. Ensuite il faut refaire pareil avec les indications doc, tex, driver et bnf pour produire bnf.tex qui est la documentation.

Voici en clair, tout ce que TEX m'a raconté pendant l'utilisation de `docstrip` pour le packages `bnf` :

```
This is TeX, Version 3.1415 (C version 6.1)
(/tmp/lib/texmf/tex/latex2e/kernel/inputs/docstrip.tex
LaTeX2e <1995/06/01> patch level 3
Hyphenation patterns for english, francais, portuges, german, italian, ukenglis
h, esperanto, loaded.
Utility: `docstrip' 2.2i <1994/12/15>
English documentation    <1994/06/09>

**********************************************************
* This program converts documented macro-files into fast *
* loadable files by stripping off (nearly) all comments! *
**********************************************************


****************************************************
* First type the extension of your input file(s):  *
\infileext=doc
****************************************************

****************************************************
* Now type the extension of your output file(s)  : *
\outfileext=sty
****************************************************
```

[86] Ormis, bien entendu, les fichiers `.aux` qui lui sont donnés en argument

[87] Style de bibliographie pour toi qui ne cause pas le grand breton aisément

```
*****************************************************
* Now type the name(s) of option(s) to include   : *
\Options=style
*****************************************************

*****************************************************
* Finally give the list of input file(s) without   *
* extension seperated by commas if necessary     : *
\filelist=bnf
*****************************************************

. . . . . .
Processing File bnf.doc (style) -> bnf.sty

. % . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . <+driver % % %
% % % % % % % % % % % % % % % % % % % > <*style % % % . % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % . . . . . . . % % % % % % . . . % % % % . . . . % % % % % % % . . . .
. . . . . . . . . . . . . . . . . . . . . % % % . % % % % % % . % % % % % % % % % %
% % % % % % . . . % % % % % % % . . . . . . . . . . . % % % % % . % % % % % .
% % % % % . . . . . . . . % % % . . % % % % . . . % % % . % % % % . . . . . .
% % % % . . . . . . . % % % % . . . . . . . % % % % % . . . . . . . % % % . . .
. . . % % % % . . . . . . . . . . % % . % . . . . . . . . .
Lines  processed: 528
Comments removed: 354
Comments  passed: 39
Codelines passed: 112

. . More files to process (y/n)?
\answer=y

*****************************************************
* First type the extension of your input file(s):  *
\infileext=doc
*****************************************************

*****************************************************
* Now type the extension of your output file(s)  : *
\outfileext=tex
*****************************************************

*****************************************************
* Now type the name(s) of option(s) to include   : *
\Options=driver
*****************************************************

*****************************************************
* Finally give the list of input file(s) without   *
* extension seperated by commas if necessary     : *
\filelist=bnf
*****************************************************

. . . . . .
Processing File bnf.doc (driver) -> bnf.tex

. % . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . <+driver . . .
. . . . . . . . . . . . . . . . . . . . > <*style % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % . . . . . . . . . .
Lines  processed: 528
Comments removed: 354
Comments  passed: 39
Codelines passed: 23

. . More files to process (y/n)?
\answer=n
Overall statistics:
Files  processed: 2
Lines  processed: 1056
Comments removed: 708
Comments  passed: 78
Codelines passed: 135
 )
No pages of output.
Transcript written on docstrip.log.
```

50.3 Antiquité

Lorsque LaTeX était très jeune, les macros étaient livrées directement sous formes de fichiers `.sty` rarement lisibles (en tous cas pas par un humain), et parfois documentés, mais pas souvent.

Bref, à l'époque ça s'installait tout seul (il suffisait de mettre le bon fichier dans le bon répertoire) mais personne savait faire marcher les packages[88].

51 Mon installation pour Linux

Comme jedispose, chez moi, d'un vulgaire PC et pas d'une jolie station de travail HP, j'ai fais une installation de LaTeX aussi performante sur mon PC que sur les HPs de l'ESIEE, mais comme je suis pas complètement fou, je l'ai faite sous Linux et pas sous DOS. Cette version est disponible à l'ESIEE à l'URL suivante[89] :

`http://www.esiee.fr/~tex/forlinux/`

Ça fait cinq disquettes et c'est mis à jour relativement souvent.

Cette section indique simplement les quelques petits réglages qui peuvent être utiles, ou plus exactement que j'ai pu rencontrer en l'installant à droite et à gauche chez plein de copains.

51.1 TeX

Bin, c'est la version dopée comme sur nos HPs pour permettre de charger autant de motifs de césure et pour pouvoir compiler cette doc (en effet, il est nécessaire d'augmenter certaines des dimensions de la mémoire de TeX pour y parvenir).

51.2 METAFONT

Bin, rien à signaler.

51.3 dvips

Retenons simplement quelques réglages intéressant dans `config.ps`.

En particulier :

- Le mode pour les Canon BJ est `bjtenex`.
- Le mode pour les 9 aiguilles est `epsonhi`.
- L'offset pour les BJs est `o -17pt 45pt` si ma mémoire est bonne.

51.4 HomeTeX

Sous Linux, j'ai placé toute l'archive dans ce répertoire :

`/usr/lib/texmf/...`

[88] Si, leurs auteurs, des fois. Mais pas toujours.
[89] Ou sur disquettes si tu me croises dans les couloirs de l'ESIEE

Divers

52 Divers packages et macros pour LaTeX 2ε

Un bon paquet de macros ne sont pas classables dans les catégories que j'ai prévues, aussi j'ai créé une classe «divers» dans laquelle tu trouveras tout et n'importe quoi dans un ordre a priori quelconque. Si je me décide à classer tout, ça disparaîtra, mais le contenu sera retrouvé ailleurs.

52.1 `ftnright`

Ce package, offert par Frank Mittelbach [74], strictement incompatible avec le joli `multicol` permet de placer les notes de pieds de page sur la colonne de droite dans des documents en `twocolumns` standards. Il suffit de l'invoquer sans autre complication pour qu'il se mette au travail. C'est joli et efficace. Peut-être, un jour futur, Frank Mittelbach (l'auteur) nous gratifiera-t-il d'une version compatible avec `multicols`, mais il ne faut pas rêver.

Ce package est particulièrement agréable a utiliser quand on se sert souvent des notes de pieds de page et qu'on en a ras le bol de voir des places réservées pour les notes qui ne soient pas de la même hauteur dans chaque colonne. C'est assez disgracieux, et en plus, ça gène la lecture.

52.2 Le package `xspace`

Pour ceux d'entre mes lecteurs qui ont appris a écrire une macro en LaTeX, David Carlisle à écris un petit package assez utile : `xspace`. Il l'a même documenté dans [19].

Imaginons que tu crée unecommande qui produise du texte, par exemple `\TeX` qui produise TeX. Tu remarqueras que si tu fais `\TeX` en plein milieu ça fait TeXsans mettre d'espace derrière. Pour que ça en mette un, il faut mettre une paire d'accolades derrière l'appel à la commande, comme par exemple `\TeX{}` qui produit TeX avec un espace.

Pour éviter ce petit désagrément, si j'ajoutais un appel à la commande `\xspace` à la fin de la définition de `\TeX`, alors `\TeX` serait capable de détecter automatiquement s'il faut un espace ou pas.

Par exemple, la commadne `\cpp` définie ci-dessous détectera toute seule s'il faut un espace derrière ou pas.

```
\newcommand{\cpp}{\textsc{c}{\small++}\xspace}
```

53 Incompatibilités

À plusieurs endroits dans cette doc j'ai évoqué des problèmes de compatibilité entre certains packages. Ces problèmes devenant plus subtiles à maîtriser au fur et à mesure que cette doc augmente, j'ai décidé de les regrouper en une section à par entière qui sera probablement celle qui sera le plus susceptible d'évoluer entre deux versions de cette doc.

Plusieurs sortes d'incompatibilités sont à envisager :

Écrasement C'est le cas où deux packages différents définissent la même commande. Cela crée un conflit évident qui se manifestera de deux façons. Soit «proprement» par une erreur à la compilation si les deux packages sont conformes aux normes de développement de LaTeX 2ε (c'est rare). L'erreur en question dira en gros «Vous re-définissez une commande déjà définie, c'est pas bien. Je garde la définition précédente.». L'autre manifestation, totalement silencieuse, sera que c'est le package chargé en dernier qui fonctionnera et pas l'autre. Il n'y a pas de remède simple à ce problème.

Conflit au chargement C'est une incompatibilité qu'à peu près seul `babel` est capable d'engendrer. La solution est généralement de changer l'ordre de chargement des packages.

Caractères actifs C'est le cas le plus subtil d'incompatibilité. Bien que j'y ai été confronté de très nombreuses fois, il m'arrive encore de mettre très longtemps à trouver d'où viens le problème. La solution peut être soit enfantine, soit extrêmement complexe. C'est le genre de problème que j'essaye d'intercepter dans le package `ESIEE`.

Redéfinition du noyau C'est le cas le plus brutal et souvent le plus incontournable d'incompatibilité. Généralement c'est celui qui caractérise les packages prévus pour LaTeX 2.09 qui ne fonctionnent plus avec LaTeX 2ε. Ce problème est dû à certains packages qui redéfinissent certaines fonctions du noyau de LaTeX (souvent `\output`) pour faire leur travail, mais ne prennent pas les précautions nécessaires. Le résultat est alors immanquablement une catastrophe.

53.1 Le package `bar`

C'est le cas le plus bénin de **redéfinition du noyau** puisque la commande redéfinie en l'occurrence n'est pas vitale du tout. Il s'agit de la commande

\bar. Normalement, il s'agit d'un accent mathématiques. Après lecture du package, il s'agit d'une commande de dessin ...

La solution que j'ai adopté dans la doc est un peu brutale, mais efficace. L'idée est de «stocker» la définition originale de la commande \bar quelque part pour pouvoir ensuite l'utiliser à mon gré. Pour charger le package bar j'utilise donc la méthode suivante :

```
\let\OldMathBar\bar
\usepackage{bar}
\let\NewDrawingBar\bar
\let\bar\OldMathBar
```

Et ensuite, je me retrouve avec le \bar originel. Sitôt que j'ai besoin du \bar de dessin, je le remet en place :

```
\let\bar\NewDrawingBar
```

Et vice-versa pour l'accent mathématiques.

53.2 Le package eufrak et les packages de l'$\mathcal{AMS}$

Nous avons là l'incompatibilité la plus propre et la moins gênante possible. C'est fort simple : eufrak et les packages de l'$\mathcal{AMS}$ définissent tous les deux la commande \mathfrak qui définit dans les deux cas le même alphabet mathématiques. Si l'on souhaite faire appel aux deux packages, il suffit de ne faire appel qu'à la partie due à l'$\mathcal{AMS}$ puisqu'eufrak ne contient rien qu'elle ne contienne.

53.3 Le package multind

C'est le cas typique de redéfinition du noyau. En effet, ce package écrase la commande \see, et la remplace par une définition ne supportant par l'internationalisation. Pour rétablir les choses convenablement, il suffit de charger le package ESIEE *après* multind.

53.4 Les packages varioref et showkeys

Il ne s'agit pas d'une incompatibilité, simplement d'un comportement étrange. Explication à la section 26.2 page 71.

53.5 Le package babel

Lui, c'est le gros morceau. Son principal problème, c'est qu'il rend certains caractères actifs, et que forcément après ça pose des problèmes à tout le monde. Par exemple, la caractère ! est rendu actif pour gérer convenablement l'espacement spécial de ce signe de ponctuation en français. C'est à dire que, a chaque fois que LaTeX croisera ce caractère, il le remplacera par une commande choisie. Or, un package comme array utilise ce caractére comme descripteur de colonne dans les tableaux. Ce qui fait que forcément, lorsque array croise la macro-commande qui est associée au caractère, il se fâche tout rouge. Et des exemples comme ça, j'en ai a la pelle.

53.5.1 Le package array

Le conflit entre array et babel n'a lieu qu'au chargement et se résout simplement en chargeant babel après avoir chargé array. Bien entendu, lorsque j'indique array ici, tous ses dérivés sont concernés (tabularx, par exemple).

Il semble que la version de babel annoncée pour décembre 1995 corrige cette déficience.

53.5.2 Le package graphics

Le package graphics, ainsi que son avatar graphicx pose exactement le même problème que array, à savoir que l'ordre d'inclusion des packages est primordiale (textttgraphicx ou graphics puis babel).

De même que pour array, le problème devrait disparaître dans la version de décembre 1995. Du moins, les patchs que j'ai reçus de J. Braams fonctionnaient convenablement.

53.5.3 Le package window

Pour lui le problème c'est l'utilisation du :. La solution n'est pas enfantine. En fait il faut réunir plusieurs conditions pour que cela fonctionne bien :

1. que babel soit chargé avec une dernière option qui ne soit pas le français,
2. que window soit chargé après babel,
3. que la commande \windowbox soit utilisée dans une langue où le : n'est pas utilisé (pas le français).

Je n'ai pas vérifié si la dernière version de babel pose encore ce problème. Je pense que non.

53.5.4 Le package qsymbols

Le caractère à problème est cette fois-ci le " qui est utilisé non pas en français mais en Allemand. Il semble que le problème soit carrément un bug dans babel et pas une simple incompatibilité.

En effet, babel devrait permettre cette utilisation du caractère mais sa gestion interne se termine en une boucle infinie, c'est à dire que LaTeX ne signale pas d'erreur, mais ne termine pas la compilation. Il boucle.

Le seul moyen de contournement que j'ai pu trouver est de changer localement le code de catégorie du caractère. C'est un contournement efficace pour cette doc où les particuliarités du style germanique ne sont utiles que dans un seul paragraphe, et pas dans le reste. Dans un vrai rapport en allemand, il serait plus sage de ne pas faire appel à ces raccourcis. Ou de les utiliser «à la hache» comme je le fis moi-même.

Pour changer le code de catégorie des caractères, on procède comme suit :

```
\catcode`"=12
Utilisation de qsymbols:
\[ A "<=>" B \]
\catcode`"=13
```

Un autre caractère à problème avec ce package est le point d'exclamation. En effet celui-ci est utilisé pour allonger les flèches, alors que le style «français» de `babel` l'utilise pour régler des problèmes d'espacement. Le problème est alors que `babel` remplace un point d'exclamation en mode mathématiques par un point d'exclamation suivi d'un espace. Ce qui est un tors incontestable. On utilisera volontiers la même bidouille pour résoudre le problème. Ou, encore plus volontiers, le package `ESIEE` qui redéfinit (un peu violemment ...) les raccourcis claviers de `babel` pour ne pas ajouter d'espace en mode mathématiques. Cela devrait permettre au système de fonctionner[90].

La solution consistant à utiliser les flèches de `qsymbols` «à la hache» est la suivante. On remplace `"XXX"` par `\ArrXXX` et comme on à la chance que pour les flèches le `XXX` ne soit jamais formé de lettres cela se passe très bien. Si cela n'était pas le cas, il suffirait d'utiliser `\Arr{}XXX`. En effet, LaTeX devine qu'un nom de commande est fini au fait qu'il croise un caractère qui ne soit pas une lettre. On laissera par contre volontiers un espace après le `XXX` correspondant à la flèche que l'on souhaite produire pour éviter toute confusion.

Exemple :

```
\[ A \Arr<=> B \]
```

53.5.5 Le package `hhline`

Le caractère problématique est ici le : qui est assez intensément utilisé dans ce package. Là où cela pose de vrais problèmes c'est que contrairement à `array`, une simple correction de l'ordre de lecture ne semble pas régler le problème. Il faudra, là encore, jouer sur les codes de catégorie. Pour cela on repassera en caractère normal avant tout tableau utilisant `\hhline` puis en caractère actif après.

Pour passer en caractère normal, c'est comme ci-dessus :

```
\catcode`:=12
```

Pour revenir en caractère actif, c'est très similaire :

```
\catcode`:=13
```

Ou pourra s'amuser à changer de code de catégorie à chaque `\hhline`, ça permet de rester conforme à la typographie française dans un tableau qui contient du texte. Par contre ça fait un peu plus de frappe à faire.

Une petite bidouille non testée qui pourrait permettre de détourner habilement le problème :

```
\let\Originalhhline\hhline
\def\hhline{\catcode`:=12\Internalhhline}
\def\Internalhhline#1{%
\Originalhhline{#1}\catcode`:=13}
```

Pour ceux d'entre mes lecteurs qui ne savent pas programmer en TeX pur, c'est à dire la très grande majorité, les questions ne seront admises sur ce bout de code qu'après une lecture attentive et détaillée du livre de Leslie LAMPORT ([61]) et de celui de Donald Ervin KNUTH ([58]). Pour ceux qui ont déjà lu avec attention ces deux ouvrages, ils n'ont pas besoin d'explications.

[90] Assertion non vérifiée.

Annexes

54 Remerciements

La rédaction de cette doc n'eut pas été possible sans un certain nombre de personnes.

En premier lieu les gens qui m'ont directement aidés. Olivier Gutknecht qui m'a guidé sur Mathematica[91], Xavier Bénech sur Khoros, Jérôme Breton qui m'a relu …

Je dois aussi remercier les amis qui m'ont soutenu, et hebergé parfois, quand je bossais sur cette doc. Citons Pascal Vincent, Pierre-Jérôme Gauriat, Xavier Bénech, et Benoit[92] Joly.

Il me faut aussi remercier tous les gens qui m'emmerdent à longueur de journée, alors que j'ai mieux à faire, avec des questions à la con. Ils m'indiquent, sans le savoir, les points à développer et à étendre.

Je ne saurais oublier mes professeurs, qui, en me demandant sans cesse d'innombrables rapports, me poussent à me perfectionner. Citons Bernadette Miara (TP Ananum), Olivier De Cambry (projet maths, projet RdF), Mohamed Akil (projet Signal, projet PTAH), Michel Couprie (projet compilation), Jean-Marie Rifflet (TP Unix), René Natowicz (premier module I4, projet Prolog, second module I4), et j'en oublie forcément.

Enfin, je tiens à remercier très chaleureusement Frank Bonnet qui m'a toujours soutenu et n'a jamais laissé passer une occasion de m'aider.

Merci tout de même à ceux qui ne m'ont pas aidé, pas soutenu, qui n'ont pas lu la version précédente, qui s'en foutent éperdument. Merci même aux autres.

[91] La majuscule à Mathematica a été ajoutée à la demande explicite d'Olivier, qui m'a promis en échange d'écrire un bout de doc sur les liens que l'on peut établir entre Mathematica et TEX ou LATEX. En particulier sur l'utilisation des modes TEX de Mathematica.

[92] Benoit, c'est bien pour te faire plaisir que je met pas l'accent sur le i.

Index général

— N —

— O —

— P —

— R —

— S —

— T —

— V —

— X —

Index des commandes

— G —

— H —

— I —

— L —

— M —

— N —

— O —

— P —

— U —

— V —

— X —

Index des environnements

— V —

Index des packages

Index des symboles mathématiques

— C —

— D —

— E —

— F —

— G —

— H —

— I —

— J —

— K —

— L —

— M —

— N —

— O —

— P —

— Q —

— R —

— S —

— T —

— U —

— V —

— W —

— X —

— Y —

— Z —

Références

[1] « The Supertabular ». En anglais.

[2] « Essential Mathematical LaTeX ». En anglais, septembre 1989.

[3] « $\mathcal{AMS}$-LaTeX Version 1.2 User's Guide ». En anglais avec bibliographie, octobre 1994.

[4] « Technical notes on $\mathcal{AMS}$-LaTeX 1.2 ». En anglais, décembre 1994.

[5] Donald ARSENEAU. « Fichier `ulem.sty` ». En anglais, 405 lignes, bien commenté pour un fichier source, août 1994.

[6] Leonor BARROCA. « A style option for rotated objects in LaTeX ». En anglais, avril 1995.

[7] Benjamin BAYART, Pascal VINCENT, et OTHERS. « Architecture pour le traitement d'images — réalisation de deux filtres ». Rapport Technique, École Supérieur d'Ingénieurs en Électronique et Électrotechnique, décembre 1994.

[8] K. C. BORDER. « The `fn2end.sty` style ». En anglais, caché à la fin du fichier source du package, mai 1995.

[9] Johannes BRAAMS. « Babel, a multilingual package for use with LaTeX's standard document classes ». Version courte et compréhensible, `J.L.Braams@research.ptt.nl`, en anglais avec bibliographie, juin 1994.

[10] Johannes BRAAMS. « Creating a mailing ». En anglais, septembre 1994.

[11] Johannes BRAAMS. « Numbering individual lines of equations array's ». `J.L.Braams@research.ptt.nl`, en anglais, février 1994.

[12] Johannes BRAAMS. « the `alltt` environment ». `J.L.Braams@research.ptt.nl`, en anglais, octobre 1994.

[13] Johannes BRAAMS, David CARLISLE, Alan JEFFREY, Chris ROWLEY, et Rainer SCHÖPF. « Producing proceedings articles with LaTeX 2_ε ». En anglais, mai 1994.

[14] Johannes BRAAMS et Kent MC PHERSON. « Displaying page layout variables ». En anglais, octobre 1994.

[15] Jérôme BRETON, Pierre RZEKIECKI, et François CHAUMARTIN. « Matrice de commutation vidéo ». Rapport Technique, École Supérieur d'Ingénieurs en Électronique et Électrotechnique, mai 1995.

[16] David CARLISLE. « The `indent` package ». `carlisle@cs.man.ac.uk`, en anglais.

[17] David CARLISLE. « The `lscape` package ». `carlisle@cs.man.ac.uk`, en anglais.

[18] David CARLISLE. « Packages in the '`graphics`' bundle ». `carlisle@cs.man.ac.uk`, en anglais, octobre 1994.

[19] David CARLISLE. « The `xspace` package ». en anglais, novembre 1994.

[20] David CARLISLE. « The `afterpag` package ». `carlisle@cs.man.ac.uk`, en anglais, mai 1994.

[21] David CARLISLE. « The `dcolumn` package ». `carlisle@cs.man.ac.uk`, en anglais, mars 1994.

[22] David CARLISLE. « The `delarray` package ». `carlisle@cs-man-ac-uk`, en anglais, mars 1994.

[23] David CARLISLE. « The `enumerate` package ». `carlise@cs.man.ac.uk`, en anglais, janvier 1994.

[24] David CARLISLE. « The `hhline` package ». `carlisle@cs.mac.ac.uk`, en anglais, mai 1994.

[25] David CARLISLE. « The `ifthen` package ». `carlisle@cs.man.ac.uk`, en anglais, novembre 1994.

[26] David CARLISLE. « The `keyval` package ». `carlisle@cs.man.ac.uk`, en anglais, février 1994.

[27] David CARLISLE. « The `longtable` package ». `carlisle@cs.man.ac.uk`, en anglais, décembre 1994.

[28] David CARLISLE. « The `showkeys` package ». `carlisle@cs.man.ac.uk`, en anglais, septembre 1994.

[29] David CARLISLE. « The `tabularx` package ». `carlisle@cs.mac.ac.uk`, en anglais, mai 1994.

[30] David CARLISLE. « The `trig` package ». `carlisle@cs.mac.ac.uk`, en anglais, mars 1994.

[31] David CARLISLE. « The `xr` package ». `carlisle@cs.man.ac.uk`, en anglais, mai 1994.

[32] Mats DAHLGREN. « Wecome to `deleq.sty` ! ». `matsd@physchem.kth.de mats@tenk.wav.nl`, en anglais, octobre 1994.

[33] Michael DOWNES. « Differences between $\mathcal{AMS}$-LaTeX version 1.1 and $\mathcal{AMS}$-LaTeX 1.2 ». En anglais avec bibliographie, novembre 1994.

[34] Peter DYBALLA. « The `charter` package ». En anglais, juin 1994.

[35] Peter DYBALLA. « The `nimbus` package ». En anglais, juin 1994.

[36] Peter DYBALLA. « The `utopia` package ». En anglais, juin 1994.

[37] Matthias ECKERMANN. « Paralleles Setzen längerer Texte ». `u311aa@sunmail.lrz-muenchen.de`, en allemand avec bibliographie, 1994.

[38] Nick EFFORD. « The `vector` package ». `nde@dcre.leeds.ac.uk`, en anglais, septembre 1994.

[39] Robin FAIRBAIRNS. « The file `oldstyle.dtx` for use with LaTeX 2_ε ». En anglais, mai 1995.

[40] Bernard GAULLE. « Notice d'utilisation du style french multilingue ». En français, avec bibliographie, octobre 1995.

[41] Jeremy GIBBONS. « The St Mary's Road symbol font ». En anglais, mars 1994.

[42] Jeff GOLDBERG. « The `lastpage` package ». `goldberg@nytud.hu`, en anglais, juillet 1994.

[43] Ulrich GOLDSCHMITT. « The `ulsy` package ». en anglais, février 1995.

[44] Norman GRAY. « The `showlabel` package ». `norman@astro.gla.ac.uk`, en anglais, octobre 1994.

[45] Aloyius G. HELMINCK. « The `mathtime` package ». En anglais, juin 1994.

[46] Rainer HÜLSE et Wolfgang KASPAR. « `EXPDLIST` — an Expanded `description` Environment ». `kaspar@dmswwu1a.uni-muenster.de`, en anglais, septembre 1992.

[47] Hideki ISOZAKI. « A bipartite graph macro ». En anglais, novembre 1990.

[48] Hideki ISOZAKI. « A tree macro ». En anglais, novembre 1990.

[49] Alan JEFFREY. « A font sampler ». En anglais, octobre 1994.

[50] Alan JEFFREY. « The `rawfonts` package ». Version 0.01, en anglais, mai 1994.

[51] Alan JEFFREY. « The `somedefs` toolkit package ». En anglais, juin 1994.

[52] Frank JENSEN. « The `beton` package ». en anglais, avec bibliographie, janvier 1995.

[53] Frank JENSEN et Frank MITTELBACH. « The `euler` package ». En anglais, avec bibliographie, janvier 1995.

[54] Dieter JURZITZA. « Schalkbilder mit LaTeX ». En allemand.

[55] Axel KIELHORN. « The `wasy`symbol fonts for use with LaTeX 2_ε ». en anglais, mai 1994.

[56] Thomas KNESER. « LaTeX-Paragraphes Floating around Figures ». En anglais, août 1990.

[57] Thomas KNESER et Mats DAHLGREN. « Welcome to the `floatflt` package! ». `matsd@physchen.kth.de`, en anglais, octobre 1994.

[58] Donald Ervin KNUTH. *The TeXbook*. Addison-Wesley, 1984.

[59] Tobias KUIPERS. « A first look at `harpoon.sty` ». `kuipers@fwi.uva.nl`, en anglais, novembre 1994.

[60] Leslie LAMPORT. « MakeIndex : An Index Processor For LaTeX ». En anglais, février 1987.

[61] Leslie LAMPORT. *LaTeX : A document preparation system*. Addison-Wesley, 1994. Second edition.

[62] Leslie LAMPORT, Frank MITTELBACH, et Rainer SCHÖPF. « Standard Letter Document Class for LaTeX 2_ε ». En anglais, mars 1995.

[63] « LaTeX 2_ε font selection ». En anglais avec bibliographie, juin 1994.

[64] « LaTeX 2_ε for class and package writers (DRAFT) ». En anglais avec bibliographie, juin 1994.

[65] Anselm LINGNAU. « An Improved environment for Floats ». `lingnau@math.uni-frankfurt.de`, en anglais, juin 1994.

[66] van der LOAN, C. G. « Typesetting Bridge via LaTeX ». En anglais avec bibliographie.

[67] I. L. MACLAINE-CROSS. « Curves in LaTeX Pictures. A manual for `CURVES.STY` and `CURVESLS.STY` ». `S7301164@csdvax.csd.unsw.edu.au`, en anglais, juin 1994.

[68] James Darrell MC CAULEY. « The `endfloat` package ». En anglais, avec bibliographie, novembre 1994.

[69] van der MECR, Hans. « The `exam` package ». `hansm@fwi.uva.nl`, en anglais, novembre 1994.

[70] Michael MEHLICH. « Fichier `readme.fp` ». Distribué avec le package `fp`, `mehlich@informatik.uni-muenchen.de`, en anglais, 163 lignes, octobre 1994.

[71] Frank MITTELBACH. « An environment for multicolumn output ». En anglais, novembre 1994.

[72] Frank MITTELBACH. « An Extension of the LaTeX `theorem` environment ». En anglais, février 1994.

[73] Frank MITTELBACH. « File not found error ». En anglais, décembre 1994.

[74] Frank MITTELBACH. « Footnotes in a multi-column layout ». En anglais, février 1994.

[75] Frank MITTELBACH. « Producing slides with LaTeX 2_ε ». En anglais, décembre 1994.

[76] Frank MITTELBACH. « The `doc` and `shortvrb` Packages ». En anglais, avec bibliographie, octobre 1994.

[77] Frank MITTELBACH. « The `pandora` fonts for use with LaTeX ». En anglais, avec bibliographie, mai 1994.

[78] Frank MITTELBACH. « The `varioref.sty` package ». En anglais, septembre 1994.

[79] Frank MITTELBACH et David CARLISLE. « A new implementation of LaTeX's `tabular` and `array` environment ». `carlisle@cs.man.ac.uk`, en anglais, avec bibliographie, décembre 1994.

[80] Frank MITTELBACH et Yannis HARALAMBOUS. « The `oldgerm` package for use with LaTeX 2_ε ». En anglais, avec bibliographie, juin 1994.

[81] Frank MITTELBACH, Alexander SAMARIN, et Michel GOOSSENS. *The LaTeX companion*. Addison-Wesley, 1994.

[82] Frank MITTELBACH et Rainer SCHÖPF. « The `eufrak` package for use with LaTeX 2_ε ». En anglais, mai 1994.

[83] Frank MITTELBACH et Rainer SCHÖPF. « The `euscript` package for use with LaTeX 2_ε ». En anglais, mai 1994.

[84] Imprimerie NATIONALE, éditeur. *Lexique des règles typographiques en usage à l'imprimerie nationale*. Troisième édition, 1994.

[85] Christianne NOTARMARCO et Rod MULVEY. « Formal Aspects of Computing : LaTeX Style Guide for Authors ». En anglais avec bibliographie, 1994.

[86] Tobias OETIKER. « An Acronym Environment for LaTeX 2_ε ». En anglais, juin 1995.

[87] Thorsten OHL. « `mcite` : Multiple Citations on One Key ». En anglais avec bibliographie, août 1994.

[88] Thorsten OHL. « `feynMF` : Drawing Feynman Diagrams with LaTeX and METAFONT ». `Thorsten.Ohl@Physik.TH-Darmstadt.de`, en anglais avec bibliographie, janvier 1995.

[89] Hubert PARTL et Axel KIELBORN. « Document Class `refman` for LaTeX 2_ε ». En anglais, octobre 1994.

[90] Mike PIFF. « Backus Naur Form in LaTeX ». en anglais, septembre 1992.

[91] Mike PIFF. « Text merges in TeX and LaTeX ». En anglais avec bibliographie, paru dans TUG Boat, vols 13, numéro 14, 1992, p. 518-523, 1992.

[92] Michael PIOTROWSKI. « Using 8-bit Caharcter Sets with `mapcodes` ». `mlpiotro@linguistik.uni-erlangen.de`, en anglais et allemand, décembre 1994.

[93] Nico POPPELIER et Johannes BRAAMS. « A style option to adapt the standard LaTeX document styles to A4 paper ». `Poppelier@elsevier.nl`, ou `J.L.Braams@research.ptt.nl`, en anglais, avec bibliographie, mars 1994.

[94] Sebastian RAHTZ. « Notes on setup of PostScript font for LaTeX 2_ε ». `spqr@ftp.tex.ac.uk`, en anglais, juin 1994.

[95] Sebastian RAHTZ. « The `lucida` package ». `spqr@ftp.tex.ac.uk`, en anglais, juin 1994.

[96] Sebastian RAHTZ. « The `psfonts` package ». `spqr@ftp.tex.ac.uk`, en anglais, juin 1994.

[97] Sebastian RAHTZ. « The `textures` package ». `spqr@ftp.tex.ac.uk`, en anglais, juin 1994.

[98] Sebastian RAHTZ et Leanor BARROCA. « A package for making sticky labels LaTeX ». En anglais, octobre 1994.

[99] Sebastian RAHTZ et Phil TAYLOR. « The `textfit` package for scaling up text to a desired size ». En anglais, avril 1994.

[100] . « Fichier `fancyheadings.doc` ». 201 lignes, en anglais.

[101] Tomas ROKICKI. « Dvips : A DVI-to-PostScript Translator ». en anglais, janvier 1995.

[102] Kristoffer H. ROSE. « Summary of `qsymbols` ». `kris@diku.dk`, ou `http://www.diku.dk/users/kris/`, en anglais, avec bibliographie, novembre 1994.

[103] Kristoffer H. ROSE. « XY-pic User's Guide ». `Kris@diku.dk`, en anglais avec bibliographie, version 2.12, octobre 1994.

[104] Kristoffer H. ROSE et Ross MOORE. « XY-pic Reference Manual ». `Kris@diku.dk` `ross@mpce.mq.edu.au`, en anglais avec bibliographie, version 2.12/3β, octobre 1994.

[105] Elmar SCHALÜCK. « Fichier `window.sty` ». En anglais, avril 1991.

[106] Rainer SCHÖPF, Bernd RAICHLE, et Chris ROWLEY. « A New Implementation of LaTeX's `verbatim` and `verbatim*` Environments ». `schoepf@Uni-Mainz.DE`, ou `raichle@azu.Informatik.Uni-Stuttgart.DE`, ou `C.A.Rowley@open.ac.uk`, en anglais, octobre 1994.

[107] Martin SCHRÖDER. « The `everyshi` package ». `MS@Dream.HB.North.de`, en anglais, janvier 1995.

[108] Axel SOMMERFELDT. « The `rplain` package ». `axel@ang-physik.uni-kiel.de`, octobre 1994.

[109] Harald Axel SOMMERFELDT. « The `caption` package ». `Harald-Sommerfeldt@ki.maus.de`, en anglais avec bibliographie, novembre 1994.

[110] Harald Axel SOMMERFLEDT. « The `umlaute` package ». `Harald-Sommerfledt@ki.maus.de`, en anglais avec bibliographie, novembre 1994.

[111] Piet van OOSTRUM. « Page headers and footers in LaTeX ». En anglais, janvier 1995.

[112] Ulrick VIETH. « The `mflogo` Package for LaTeX 2_ε ». En anglais, 1995.

[113] Ulrick VIETH. « The `ssqquote` package ». En anglais, mars 1995.

[114] Martin WARD. « Fichier `program-demo.tex` ». `Marint.Ward@durham.ac.uk`, en anglais, 307 lignes, source de «A demonstration of the `program` environment», novembre 1994.

[115] Martin WARD. « Fichier `program.sty` ». `Martin.Ward@durham.ac.uk`, en anglais, 901 lignes, novembre 1994.

[116] Nigel WARD, Dan JURAFSKY, et Jean-Pierre DRUCLET. « The `minitoc` style option ». En anglais, avec FAQ.

[117] Martin WRAD. « A demonstration of the `program` environment ». `Martin.Ward@durham.ac.uk`, en anglais, novembre 1994.

[118] Dominik WUJASTYK et John LAVAGNINO. « Fichier `endnotes.sty` ». En anglais, 325 lignes, sobrement commenté, écrit par LAVAGNINO d'après un source de Leslie LAMPORT, modifié par WUJASTYK, septembre 1991.