Lycée Camille Guérin Année scolaire 2001-2002

Stage LaTeX
Niveau débutant

Jean-Côme Charpentier
Dernière modification : 4 juillet 2002

Table des matières

1 Logiciel libre, TEX et LATEX **1**
1.1 Historique . . . 1
1.2 Que signifie logiciel libre ? . . . 2
1.3 Avantages et inconvénients de TEX . . . 2

2 Installation d'une distribution **3**
2.1 Que faut-il ? . . . 3
2.2 Distribution TEXlive . . . 3

3 Premiers documents **5**

4 Commandes de base **11**
4.1 Caractères particuliers . . . 11
4.2 Commandes de fonte . . . 13
4.2.1 Taille des caractères . . . 13
4.2.2 Types de fonte . . . 13
4.3 Commandes de paragraphe . . . 15
4.4 Espacements . . . 18
4.5 Listes . . . 19
4.6 Modes mathématiques . . . 21
4.6.1 Opérations élémentaires . . . 22
4.6.2 Structures indispensables . . . 22
4.6.3 Symboles en vrac . . . 24

5 Quelques présentations plus évoluées **27**
5.1 Structuration des documents . . . 27
5.1.1 Commandes pour le plan . . . 27
5.1.2 Références . . . 28
5.2 Table des matières, index . . . 28
5.3 En-têtes, pieds de page, notes . . . 29
5.4 Tableaux . . . 30
5.5 Mathématiques complexes . . . 34
5.5.1 Fontes mathématiques . . . 34
5.5.2 Grands opérateurs . . . 35
5.5.3 Délimiteurs . . . 36
5.5.4 Espacements . . . 36
5.5.5 Empiler verticalement . . . 37
5.5.6 Matrices . . . 39

6 Macros **40**
6.1 Principe des macros . . . 40
6.2 Macros à paramètres . . . 40
6.3 Compteurs et dimensions . . . 42

7 Extensions utiles **45**
7.1 geometry . . . 45
7.2 times et compagnie . . . 45
7.3 amsmath . . . 47
7.4 Tableaux évolués . . . 48
7.5 multicol . . . 51

8 Petit aperçu des possibilités **52**
8.1 Ce manuel . . . 52
8.2 Graphiques . . . 52
8.3 Dessins scientifiques . . . 54
8.4 Jeux . . . 54

A Bibliographie commentée **55**
A.1 Ouvrages sur la typographie . . . 55
A.2 Ouvrages pour débuter . . . 55
A.3 Ouvrages plus difficiles . . . 55
A.4 Références sur internet . . . 56

Liste des tableaux **57**

1 Logiciel libre, TEX et LATEX

1.1 Historique

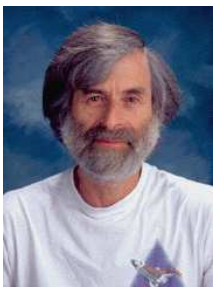
Dans les années 1970, Richard Stallman (photo ci-contre) a commencé à travailler au laboratoire d'intelligence artificielle du MIT. À l'époque, le laboratoire pouvait être qualifié de communauté dans laquelle le logiciel était partagé sans se poser de question (partage qui avait lieu également avec l'extérieur).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

Cette décision est lourde de conséquences : afin de mener son projet à bien, il va devoir démissionner du MIT en 1984 pour éviter que son travail ne tombe automatiquement dans le domaine du logiciel propriétaire. En fait, c'est un de ces premiers programmes — Emacs, un traitement de texte — qui va le faire connaître et attirer vers lui une nouvelle communauté de développeurs.

Le projet va prendre beaucoup d'ampleur. À côté du développement proprement dit, Richard Stallman va devoir mettre sur pied une structure juridique permettant de protéger les logiciels libres. En 1991, avec l'arrivée de Linux, le couple GNU/Linux peut enfin proposer un système complet entièrement libre.

Donald Knuth, un informaticien de renommé mondiale, a entrepris la rédaction d'une véritable bible de la programmation : *The Art of Computer Programming*. Au début, l'ouvrage est composé de façon traditionnelle avec des caractères en plomb et tout allait très bien. Puis, au cours des années 70, les premières photocomposeuses apparaissent et la qualité devient trop médiocre aux yeux de Knuth. Il décide alors de concevoir un logiciel permettant de composer des textes avec une qualité professionnelle. Dès le départ, il se fixe comme but d'arriver à un produit qui devra être parfait (ou presque) et qui devra le rester au cours du temps !

Ce travail va prendre un certain temps et, après la première version sortie à la fin des années 1970, la correction de quelques erreurs amène à la version finale au tout début des années 1980. Cette dernière version a été définitivement gelée et fonctionne ainsi depuis plus de vingt ans sans que le besoin impérieux d'ajouter des fonctionnalités se fasse sentir. Lorsque TEX a été rendu public, de nombreux utilisateurs l'ont enrichi et, au cours du temps, il est devenu de plus en plus facile de réaliser de multiples types de compositions avec cet outil. Citons par exemple la réalisation de formules ou molécules chimiques, de diagrammes physiques, de partitions musicales, de textes arabes, etc. Une autre voie a été de doter les distributions de programmes annexes permettant de gérer le PostScript, le PDF, le langage HTML, etc.



Dès le début des années 1980, un de ces développeurs, Leslie Lamport, a créé une surcouche à TEX – LATEX– permettant une approche plus intuitive du langage et qui en a standardisé l'utilisation. Le principe qui a guidé l'élaboration de LATEX était de faire en sorte que l'utilisateur se concentre plus sur la structure du document que sur les détails de composition. LATEX propose pour cela toute une gamme de commandes de haut niveau construites à partir de TEX mais dont les détails n'ont pas obligatoirement besoin d'être connus de l'utilisateur. Actuellement, il y a beaucoup plus d'utilisateurs de LATEX que de TEX, cela a pour conséquence que de nombreuses extensions du langage ont été développées en LATEX et qu'elles ne fonctionnent pas toutes sous TEX.

1.2 Que signifie logiciel libre ?

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

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

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

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

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

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

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

1.3 Avantages et inconvénients de TEX

Inconvénients.

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

1. Il s'agit d'un langage de programmation. Pour ceux qui sont habitués à travailler sous un traitement de texte plus classique, le mieux à faire est d'oublier à peu près tout ce qu'ils ont appris : la suite sera plus facile !
2. À ma connaissance, il n'y a qu'une seule distribution (payante et qui ne fonctionne que sous Macintosh) permettant de voir le résultat au fur et à mesure de la frappe. Il y a également le logiciel Lyx qui n'est pas vraiment wysiwyg mais qui s'en rapproche, on peut aussi citer le logiciel Scientific Workplace. Avec une distribution classique, il faut taper le source (décrire le document), le compiler puis visualiser ou imprimer le fichier de sortie.
3. La prise en main du logiciel demande un effort non négligeable en raison du nombre de commandes à connaître. Les non anglophones seront particulièrement désavantagés !
4. Certaines mises en forme sont assez lourdes à gérer. En particulier, on pourra citer les tableaux.

Avantages.

Comment arrive-t-on à concevoir un logiciel qui est utilisé depuis vingt ans sans que des mises à jour ne soient nécessaires ? Je ne peux pas expliquer toutes les idées de l'auteur de TEX dans ce manuel et je ne les

connais d'ailleurs pas toutes. Pour en avoir une petite idée, voici quelques caractéristiques de TEX qui font sa force :

1. La qualité typographique est impressionnante. Il faut faire des efforts pour obtenir un document peu esthétique.
2. Le source étant un fichier texte, il n'y a aucun problème de format propriétaire. D'autre part, la taille des fichiers est très petite (1 000 fois plus qu'un texte contenant beaucoup de formules mathématiques fait sous Word) ce qui constitue un avantage énorme en cas de travail via internet ou d'échange par disquettes.
3. TEX a été porté sur toutes les plates-formes possibles et imaginables. En fait, je ne connais aucune machine avec laquelle il soit impossible de travailler avec TEX.
4. TEX est libre.
5. TEX a été très rapidement gelé, ce qui fait que tous les documents sont lisibles sur n'importe quelle installation. Par exemple, le manuel de TEX que KNUTH a écrit en 1982 est toujours lisible à l'heure actuelle. Inversement, un fichier écrit sur la dernière distribution pourra être compilé correctement sur une machine disposant d'une très vieille distribution.
6. Le fichier de sortie créé par TEX est indépendant du périphérique. Cela fait que, hormis les problèmes de qualité, le résultat sera strictement identique quel que soit l'imprimante utilisée, que se soit une jet d'encre à 180 dpi datant de 10 ans ou la dernière laser couleur à 2 400 dpi.
7. Le fichier construit avec TEX place les différents éléments avec une unité de longueur appelée le point d'échelle qui correspond exactement à 2^{-16} point typographique, ce qui nous donne $5,36 \times 10^{-6}$ mm. À titre indicatif, la plus petite longueur d'onde visible (violet) vaut environ 400×10^{-6} mm. Autrement dit, même en concevant des périphériques d'impression de plus en plus perfectionnés, TEX sera toujours capable de gérer des différences de tailles très largement invisibles à l'œil ; tout au plus, pourrait-on regretter que TEX devienne insuffisant pour fabriquer des microfilms lisibles au microscope électronique !

2 Installation d'une distribution

2.1 Que faut-il ?

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

Que trouvons-nous à côté de ces deux programmes ? En fait, tout plein de choses, des fichiers décrivant les fontes de caractères, des programmes permettant d'organiser les index, les bibliographies, des bibliothèques de commandes préprogrammées, des formats (fichiers décrivant les grandes lignes de la présentation d'un document) et d'autres choses plus ou moins exotiques.

Le tout fait plusieurs centaines de fichiers et une distribution classique peut très bien atteindre 100, 200 ou 300 Mo. L'installation complète de la distribution TEXlive occupe 1,5 Go du disque dur.

2.2 Distribution TEXlive

Durant ce stage, nous utiliserons la distribution TEXlive distribuée, entre autre, par l'association GUTenberg. Il s'agit d'une distribution classique du système TEX permettant une installation sous Unix, Windows ou Macintosh. Plus exactement, nous utiliserons la distribution prévue pour Windows et c'est la procédure d'installation sous ce système qui est décrit ci-dessous. Une installation sous Linux est beaucoup plus simple :

pour ceux qui dispose d'un environnement Linux, en général, l'installation d'un système TEX se fait lors de l'installation du système d'exploitation lui-même.

Pour installer cette distribution sur votre ordinateur, vous devez vous trouver devant le clavier, l'ordinateur allumé et le système Windows attendant votre bon vouloir. Introduisez le Cédérom.

Si l'ordinateur n'est pas prévu pour lancer automatiquement les Cédéroms, allez dans le menu `démarrer`, item `Exécuter...` et recherchez sur le Cédérom le fichier `Autorun.exe`. Lancez ce programme et vous serez revenu dans le même chemin.

- une page d'accueil vous souhaite la bienvenue, cliquez sur le bouton `Install TeX on Hard Disk` (le premier en haut) ;
- le programme d'installation proprement dit démarre (TeXlive Setup Wizard) et propose de cocher la case `Quick install` ; comme on souhaite contrôler le processus, on laissera cette case non cochée, cliquez donc directement sur `Suivant` ;
- la fenêtre suivante permet de spécifier la source servant à l'installation ; tout est très bien (normalement, la seule case cochée est `CDROM/Flat directory` et la zone de texte doit correspondre à la lettre du lecteur de Cédérom), cliquez sur `Suivant`.
- la fenêtre suivante permet de spécifier les différents répertoires principaux qui seront gérés par le système TEX ; là aussi, c'est très bien comme cela et cliquez sur `Suivant`

La procédure de création des fichiers temporaires est effectuée (très rapide) et la fenêtre sur laquelle il va falloir ne pas aller trop vite apparaît. Il s'agit d'indiquer ce qu'on veut réellement installer. Si on ne veut pas se poser de question, on peut toujours cocher la case ☐ `texlive` qui procédera à l'installation de ... tout ! C'est bien de ne pas se poser de question mais c'est un peu lourd à supporter pour le disque dur puisque cela produira une copie de 1,5 Go de données. Dans ce manuel, on va essayer d'être plus subtil !

Cliquez sur la case ⊞ pour développer l'arborescence. En cliquant sur le texte du package, on a accès à une brève description (en anglais) dans le cadre en bas à droite. Il peut être intéressant d'étudier attentivement ces packages si vous avez besoin de choses particulières (création de partitions musicales, de diagrammes d'échecs, de texte en japonais, coréen, chinois, grec, ..., de diagrammes physiques complexes, de structures moléculaires, etc. La suite du texte ne fera que charger les packages particulièrement utiles et plus ou moins nécessaires à tout le monde. Si on veut, ce serait tous les packages utiles pour une série de trois stages (débutant, perfectionnement, expert).

L'item `tex-basic` est déjà coché. Il s'agit de tout ce qui est absolument indispensable au bon fonctionnement d'une plate-forme TEX-LATEX. Cochez de plus la case `tex-langfrench`, la case `tex-latex` qui doit d'ailleurs être déjà cochée car essentielle, la case `tex-latexextra` bien que nombre des packages qui en dépendent ne soient pas forcément très utiles, la case `tex-mathextra` indispensable pour un travail sérieux en édition scientifique, la case `tex-pdftex` pour pouvoir produire des documents PDF, la case `tex-pictures` pour pouvoir manipuler tout plein de jolis dessins, la case `tex-psfonts` pour disposer de quelques fontes PostScript et la case `ps-utils` pour manipuler les fichiers PostScript. Pour la dernière case – `win32-support` – déroulez l'arborescence en cliquant au niveau de la case ⊞ afin de choisir les packages :

- `gs-free` qui permet de visualiser les documents PostScript ;
- `TeXShell` qui sera l'éditeur orienté TEX-LATEX servant tout au long de ce stage.

Une fois que LATEX ne vous posera plus de problème, il sera peut être utile de charger plutôt `ntemacs`, sans doute plus rebutant mais ô combien plus puissant.

Avant de valider cette étape en cliquant sur `Suivant`, fouillez bien dans toute l'arborescence pour voir l'étendue des possibilités et charger les extensions qui vous seront utiles. Normalement, la case `Document Files` est cochée. C'est très bien comme cela, la documentation disponible sur toute distribution est une vraie mine d'or.

Une fois que vous êtes certain d'avoir coché les cases indiquées ci-dessus, cliquez sur le bouton `Suivant`. Une fenêtre indiquera le récapitulatif de tout ce qui a été demandé jusqu'ici : ce qui est très long. Il faut avoir confiance, cliquez sur `Suivant` ce qui débutera l'installation.

Celle-ci commence par la procédure d'installation de GhostScript que vous validez en cliquant sur le bouton `setup`. La fenêtre suivante vous propose d'installer le logiciel dans le répertoire `c:\gs` et de mettre le nom `GhostScript` au niveau du menu. Il n'y a pas de raison de refuser, cliquez sur `Install`. L'installation de GhostScript est très rapide.

L'installation du reste du système TEX se déroule dans la foulée. Chez moi, avec un disque dur tournant à 7200 T/mn, un processeur Pentium 700, une RAM de 128 Mo et un lecteur de Cédérom 52×, cela a pris un peu plus de dix minutes. Vous pouvez donc aller prendre un café.

Une fenêtre indique que tous les packages ont été installés dans le répertoire adéquat (enfin, j'espère). Cliquez sur `Suivant`. La configuration du système est alors lancée (une trentaine de seconde). Cliquez sur `Suivant`.

La procédure d'installation s'achève par un message de félicitations et une petite windoserie vous invitant à redémarrer l'ordinateur (case que vous laisserez cochée si vous voulez utiliser TEX immédiatement). Pour les cases `View the documentation` et `View the log file`, je vous propose de ne pas les cocher. Cliquez sur le bouton `Reboot` (en laissant le Cédérom en place). L'ordinateur redémarre.

Éventuellement, quelques fenêtres restent ouvertes (en particulier celle de GhostScript) : vous pouvez les fermer. Dans le menu `Démarrer/Programmes` doit se trouver les items `GhostScript` et `GsView` qui permettent de lancer le visualisateur PostScript et l'item `TeXlive` qui permet d'obtenir de la documentation, de lancer le visualisateur dvi, l'éditeur TeXShell et de désinstaller la distribution TEXlive.

Lancer TEXShell pour les dernières modifications. Dans le menu `Options/Program Calls ...`, le texte correspondant à la troisième ligne (`DVI Previewer` doit être `windvi "%P%F.dvi"` au lieu de l'appel au logiciel yap qui y figure par défaut. Dans le menu `Options`, cliquer sur l'item `WordWrap` ce qui permettra d'avoir des retours à la ligne automatique lors de la frappe

Ouf ! Vous voilà en présence d'une plate-forme encore peu personnalisée mais néanmoins fonctionnelle. TEXShell est relativement facilement configurable mais il faut faire l'effort de lire son menu `Help` pour comprendre comment cela fonctionne. Pour l'instant, ce qui est proposé par défaut suffira.

3 Premiers documents

Un des aspects les plus déroutant de LATEX, lorsqu'on est habitué à des traitements de texte de type Word, est qu'il ne s'agit pas d'un environnement Wysiwyg (What You See Is What You Get : ce qu'on voit est ce qu'on obtient). Comme c'est le premier reproche fait à LATEX, une petite mise au point s'impose. Tout d'abord, le Wysiwyg n'est qu'à peu près Wysiwyg : l'affichage écran ne peut de toute façon pas rendre toutes les nuances possibles au niveau d'une impression, même sur une imprimante actuelle très bas de gamme. D'autre part, il faut se persuader que la prétendue supériorité du Wysiwyg n'est en fait qu'une histoire d'habitude. Passer de Word à LATEX demande donc de perdre certaines habitudes, ce qui n'est jamais agréable au départ. Je pense même (et je suis loin d'être le seul à le penser) que le Wysiwyg est plus un défaut qu'une qualité : elle oblige l'utilisateur à faire plus attention à la forme physique qu'à la forme logique d'un document. C'est également la porte ouverte à tout un tas de mauvaises habitudes typographiques. Pour clore le sujet, je pense que pour une lettre à un ami ou une feuille d'exercices destinée à des élèves, ce n'est pas excessivement important, en revanche, dès que le travail commence à être un peu plus conséquent, la supériorité de LATEX (ou de TEX) est manifeste. Dernier reproche fait à LATEX : les documents demandent plus de temps à être tapés. C'est évident lorsqu'on ne connaît pas encore bien le logiciel ; c'est complètement faux dès qu'on commence à en avoir une certaine habitude.

Produire un document en utilisant LATEX se fait en deux étapes. La première consiste à taper le source du document et la seconde consiste à le compiler. On retrouve donc les deux étapes essentielles de beaucoup de langages de programmation. En théorie, il faut donc un éditeur de texte pour produire le source puis un appel à un programme qui va se charger de traduire le source en un fichier décrivant le document d'une façon facilement compréhensible pour l'ordinateur (donc totalement incompréhensible pour l'humain). Avec TEXShell, les choses sont un peu plus simples car cet éditeur de texte propose, via des menus et boutons, d'automatiser les différentes étapes de compilation. Je ne rentrerai donc pas dans les détails techniques en renvoyant le stagiaire curieux vers les livres cités en bibliographie.

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

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

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

Le résultat est montré page ci-contre.

Un document LaTeX commence obligatoirement par la déclaration

`\documentclass{`*classe_document*`}`

où *classe_document* est un des mots-clés suivants :

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

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

Les deux lignes suivantes commencent par `\usepackage`. Il s'agit d'un appel à une bibliothèque de commandes appelée *extension* ou *package* permettant d'automatiser certaines actions. La syntaxe générale de cette commande est :

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

Dans notre exemple, on trouve :

`\usepackage[T1]{fontenc}`

qui permet de taper directement les lettres accentuées (TeX est un logiciel américain fait à une époque où le codage ASCII ne prenait pas en compte les lettres dites européennes). On trouve également :

`\usepackage[frenchb]{babel}`

qui permet de se plier aux règles de la typographie française[1]. Nous allons immédiatement voir en quoi cela consiste.

La ligne suivante dit :

`\begin{document}`

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

`\end{document}`

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

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

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

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

La partie située entre `\documentclass` et `\begin{document}` s'appelle le *préambule* et la partie située entre le `\begin{document}` et le `\end{document}` s'appelle le ... *document*. Jusqu'au chapitre 7 (page 45), le préambule sera toujours :

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[frenchb]{babel}
```

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

```
\begin{document}
\end{document}
```

Regardons maintenant ce qui a été produit. On voit que les coupures de lignes n'ont pas du tout lieu au même endroit que dans le source et que les lignes produites sont justifiées (alignement au niveau des marges gauche et droite). TEX calcule automatiquement les endroits où doivent se situer les coupures et procède, le cas échéant, aux césures de mots (comme atten-tion dans notre exemple). Dans la plupart des cas, ces coupures sont correctes ; on peut forcer les coupures de façon manuelle mais, sauf cas particulier, c'est généralement une très mauvaise idée. D'autre part, une fin de ligne au niveau du source est comprise comme un espace ce qui fait que les mots « premiers » et « exemples » sont normalement séparés au niveau de la sortie.

On notera également que la première ligne du paragraphe est indentée (retrait vers la droite de la première ligne) alors que le source ne demandait rien. Il s'agit de la présentation par défaut des paragraphes (en typographie française). On verra au chapitre 4 comment spécifier les formes que devront avoir les paragraphes.

Le terme « `\LaTeX{}` », dans le source, a été traduit par « LATEX » dans le document. De façon générale, tout ce qui commence par une contre-oblique est une macro (macro-instruction, une sorte de commande en fait). Ici, le nom de cette commande est `LaTeX` (notez l'emploi des majuscules et minuscules, par exemple, la commande `latex` n'existe pas). Il s'agit d'une suite d'instructions permettant d'obtenir le curieux logo avec ses lettres qui ne sont pas alignées verticalement.

Le gros bloc :

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

a été traduit par une liste d'items. On notera la syntaxe qui permet de déclarer où commence la liste d'items (`\begin{itemize}`) et où elle se termine (`\end{itemize}`). Ces blocs qui commencent par un `\begin{bidule}` et qui se terminent par un `\end{bidule}` s'appellent des *environnements*.

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

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

On peut remarquer également, et ce n'est pas très esthétique, que les marges sont très importantes et que le numéro de page est situé très haut dans la page. Ce comportement malheureux a deux origines. La taille des marges par défaut est de un pouce (environ 2,54 cm) ; c'est KNUTH qui en a décidé ainsi et KNUTH aime bien les marges importantes. D'autre part, TEX a été créé par un américain qui devait écrire des textes sur du papier américain. Or, le format standard outre-atlantique est « letter » alors qu'en France, il s'agit du format A4. Le document a été créé en supposant que le papier était du type letter ce qui explique la position

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

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

bizarre du numéro de page (il suffit de couper la feuille au format letter et on retrouve quelque chose de plus propre !) Bien évidemment, ce comportement peut être réglé (Cf. chapitre 7). Pour le moment, on indiquera qu'on travaille sur du papier au format A4 en spécifiant un paramètre optionnel au niveau de la commande `\documentclass`. En fait, j'avais menti précédemment en disant que tous nos documents commenceraient par cette commande ! En réalité, ils commenceront par la commande :

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

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

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

Ce premier exemple a permis de présenter le processus général pour obtenir un document et de voir les notions de classe de document, d'extension (ou package), de préambule, de macro (ou commande) et d'environnement. Un dernier mot pour spécifier ce qu'est exactement un nom de macro et pourquoi les accolades qui suivent `\LaTeX` n'ont pas été traduites par des accolades au niveau du document final.

LATEX permet d'écrire n'importe quel caractère au niveau du source (les caractères accentués nécessitant quand même la présence d'un `\usepackage[T1]{fontenc}` au niveau du préambule) sauf les dix caractères suivants qui ont chacun un rôle très particulier. Une utilisation erronée de ces caractères produira invariablement des erreurs lors de la compilation et des résultats surprenants au niveau de la sortie.

`\`	début de nom de macro	`%`	commentaire	`~`	espace insécable	`{`	début de groupe	`}`	fin de groupe
`$`	mode mathématique	`_`	indice	`^`	exposant	`#`	paramètre de macro	`&`	colonne de tableau

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

\ sert à indiquer une macro, les caractères qui suivent donnant le nom de cette macro. TEX a des règles qui peuvent sembler bizarres pour déterminer ce qu'est un nom de macro. Une série de lettres (majuscules et/ou minuscules) qui se suivent déterminent un nom de macro, un caractère autre qu'une lettre (y compris un des 10 symboles spéciaux) détermine seul un nom de macro. Voici quelques exemples pour bien comprendre :

`\bonjour` est la macro `bonjour` ;
`\\bonjour` est la macro \ suivie du texte « bonjour » ;
`\\\bonjour` est la macro \ suivie de la macro `bonjour` ;
`\␣` où ␣ désigne un espace est la macro ␣ (le nom est constitué par un espace : bizarre n'est-ce pas ?) ;
`\formatA4` est la macro `formatA` suivie du texte « 4 ».

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

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

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

Les caractères `{` et `}` sont très importants sous TEX. Ils délimitent des *groupes*. Expliquer en détails ce qu'est un groupe sortirait du cadre de ce stage. Pour simplifier à outrance, disons qu'un groupe délimite

[4] Ce n'est pas tout à fait vrai mais c'est une bonne approximation de la vérité. Voir page 42 pour l'influence que peut avoir un `%` pour le document.

une zone à l'intérieur de laquelle on peut obtenir des comportements qui n'existeront plus à l'extérieur de ce groupe. Dans le source on a employé la construction `\LaTeX{}`, c'est-à-dire la macro `LaTeX` suivie d'un groupe vide. Sans ce groupe vide, le source :

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

aurait produit la sortie suivante :

Mes premiers pas avec LaTeXsont un peu émouvants

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

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

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

Source

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

On notera les deux façons supplémentaires d'écrire le mot « LaTeX » en mettant une espace à la suite, le fait que le commentaire « `comme celui-ci` » (ligne 12) n'apparaît pas au niveau de la sortie, et les ribambelles d'espaces au niveau du source qui n'ont donné qu'une seule espace au niveau du document final (lignes 10 à 16).

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

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

Source

```
1  Le second exemple de source {\LaTeX} va
2  permettre d'illustrer les points que nous
3  venons de voir. Dans les exemples, nous
4  avons vu la macro dont le nom était
5  constitué uniquement d'un espace. Ce
6  source~\LaTeX\ en donne une utilisation
7  possible.
```

Le troisième exemple montre la puissance de TEX. Le premier paragraphe a été repris tel quel sauf qu'on a placé une espace insécable entre deux mots (`source~\LaTeX` en ligne 6). TEX a alors repris tous les calculs de coupure de ligne afin de présenter un résultat correct. Si les mot « source LATEX » avaient été déplacés sur la quatrième ligne du paragraphe, les lignes précédentes auraient été trop serrées (elles auraient présenté des espaces trop étroites entre les mots). TEX a alors complètement revu les coupures de lignes, dès la troisième ligne dans l'exemple, en élargissant un peu plus les lignes précédentes. Pour TEX, il était préférable de rendre les lignes un peu plus lâches que de les serrer. Les calculs entrant en jeu sont très compliqués et assurent d'avoir un résultat proche de celui obtenu de la part d'un typographe professionnel.

Ces deux derniers exemples montrent qu'un source LATEX n'a rien de vraiment compliqué lorsque la présentation voulue n'appelle pas de constructions spéciales. Pour obtenir des résultats non standard, il va falloir faire appel à des macros et des environnements. La suite de ce manuel se charge de cette présentation. Le logiciel TEX connaît environ 300 commandes qu'on appelle des *primitives* car elles sont construites à l'intérieur même du logiciel. Le format Plain TEX (une bibliothèque de macros incontournables) en déclare environ 300 autres : ces 600 commandes sont la base d'un travail sous TEX. LATEX, en tant que surcouche de TEX, en définit très approximativement autant que TEX et Plain TEX réunis ce qui fait qu'au final, un utilisateur dispose de plus de 1 000 commandes pour indiquer ce qu'il veut faire. Ce nombre est assez colossal et apprendre par cœur la syntaxe et le rôle de ce millier de commandes est une tâche aussi ardue qu'inutile (sauf si vous vous destinez à devenir gourou de LATEX, et encore ...).

En premier lieu, un utilisateur lambda n'a pas besoin, loin s'en faut, de toutes ces commandes (ou macros). La plupart du temps, un document n'en nécessite qu'un petit nombre et, normalement, est constitué pour l'essentiel de texte brut. Par exemple, pour écrire ce manuel jusqu'à cette ligne, l'auteur a utilisé exactement 78 macros (dont 7 personnelles) et 11 environnements (dont un personnel). Il est clair que, pour l'essentiel, les macros et environnements utilisés sont très usuels et qu'ils sont connus après peu de temps de pratique. Il n'en reste pas moins que leur apprentissage nécessite un petit effort et qu'il faut avoir de la documentation en permanence lorsqu'on travaille sous LATEX afin de gérer les situations exceptionnelles. Nous allons voir que les macros ont des noms qui permettent souvent de bien s'en souvenir (pour peu que l'anglais ne soit pas complètement inconnu).

4 Commandes de base

Dans ce chapitre, nous allons voir les macros et les environnements qu'on doit absolument connaître. Il s'agit là de gérer la mise en forme des caractères (caractères spéciaux, forme et taille des caractères) et des paragraphes, les espacements importants, les différents types de listes et la composition des formules mathématiques (n'oublions pas que TEX a été créé par un mathématicien-informaticien pour produire des livres à caractère scientifique).

4.1 Caractères particuliers

On a vu précédemment que les 10 caractères `\ % ~ { } $ _ ^ # &` et `&` avaient des rôles précis et ne pouvaient pas être employés pour obtenir les caractères correspondants au niveau de la sortie. Le tableau 1 montre les dix macros permettant de retrouver ces caractères. On voit que pour sept d'entre eux, la macro

caract.	macro	caract.	macro	caract.	macro	caract.	macro	caract.	macro
\	`\(\backslash\)`	%	`\%`	~	`\~{}`	{	`\{`	}	`\}`
$	`\$`	_	`\_`	^	`\^{}`	#	`\#`	&	`\&`

TAB. 1 – Obtention des caractères réservés

a le nom du caractère lui-même. Les exceptions concernent la barre oblique inverse (backslash en anglais), l'accent circonflexe et le niña.

Knuth a expliqué pourquoi il n'a pas voulu utiliser `\\` comme macro permettant d'obtenir la barre oblique ; je n'entrerai pas dans les détails. Remarquons quand même que cette macro doit être entourée des signes `\(` et `\)` qui indiquent que l'intérieur est composé en mode mathématique (Cf. section 4.6).

L'accent circonflexe et le niña sont des exceptions car il ne s'agit pas de macros donnant un caractère mais un accent. Ainsi, `\^` indique que le prochain caractère doit avoir un accent circonflexe. Comme nous voulions l'accent seul, il a suffit de faire suivre cette macro d'un groupe vide. `\^e` donne « ê », `\^a` donne « â » (qui n'existe pas mais TEX s'en moque). N'importe quel caractère peut être accentué. Ainsi, `\^c` donne « ĉ », `\^b` donne « b̂ » (on remarquera que l'accent s'est automatiquement placé plus haut pour cette lettre présentant une hampe), `\^\$` donne « $\hat{\$}$ ». Plus rigolo : avec l'extension `\usepackage[T1]{fontenc}` qui permet d'utiliser directement des caractères accentués, on peut obtenir « û » avec `\^ù`.

En fait, TEX offre la possibilité de placer 14 types d'accents couvrant les besoins de toutes les langues européennes (Cf. tableau 2). Certains utilisateurs refusent d'utiliser le `\usepackage[T1]{fontenc}` et sont

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

TAB. 2 – Accent en mode texte

donc obligés de taper toutes les lettres accentuées de cette façon. C'est une question d'habitude, chacun fait comme il l'entend ! En revanche, un grand nombre de claviers ne proposent pas les capitales accentuées (j'en profite pour rappeler que les capitales doivent TOUJOURS être accentuées lorsque le mot comporte un accent) et ces macros deviennent alors indispensables.

On notera la syntaxe un peu différente pour les macros dont le nom est une lettre (`\u`, `\v`, `\H`, etc.) car une écriture comme `\vo` ne produira certainement pas le caractère « ŏ » mais une erreur de compilation puisque la macro de nom `vo` n'existe pas. On aurait pu écrire cela de différentes façons : `\v{}o`, `\v{o}` produisent le résultat voulu. Si on ne veut pas réfléchir, on peut toujours écrire de façon systématique le caractère à accentuer dans un groupe.

Certains caractères européens ne sont pas souvent présents sur les claviers d'ordinateur, aussi, TEX déclare les macros du tableau 3 pour pouvoir y accéder. Les ponctuations espagnoles ¿ et ¡, appelées par les

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

TAB. 3 – Caractères européens spéciaux

séquences `?‘` et `!‘` ne sont en fait pas des macros mais des ligatures. Cette différence ne présente pas beaucoup d'intérêt pour l'heure. Normalement, un français ne devrait connaître que les quatre premières macros.

Quelques caractères d'usage courant et non forcément disponibles sur les claviers sont également définis (Cf. tableau 4).

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

TAB. 4 – Symboles spéciaux

Nous n'avons présenté que les caractères disponibles par défaut avec LATEX. En fait, grâce aux extensions (Cf. chapitre 7), l'utilisateur a à sa disposition une bibliothèque prodigieuse de symboles. Les caractères particuliers vus ici ne sont que ceux des modes textes. Les modes mathématiques offrent une bien plus grande richesse en raison de la consommation immodérée de ces symboles par les mathématiciens et physiciens.

4.2 Commandes de fonte

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

4.2.1 Taille des caractères

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

Pour changer de taille de caractère, LaTeX propose 10 macros présentées dans le tableau 5 : Attention

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

Tab. 5 – Macros de taille de caractère

à leur utilisation. Si on met une de ces macros dans le source, tout le texte qui va suivre sera de la taille spécifiée. Une façon naturelle de revenir à la taille précédente serait d'écrire `\normalsize` mais ce n'est pas une si bonne idée car la taille précédente n'était pas forcément la taille normale et, en plus, c'est long à écrire ! Une façon beaucoup plus propre et rapide est d'utiliser un groupe. Petit exemple pour bien comprendre :

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

Source

```
1 {\Large Jamais !} Vous entendez ?
2 {\LARGE Jamais} je ne le ferrai.
3
4 {\tiny D'accord.} Ne vous fâchez pas.
```

4.2.2 Types de fonte

Il existe 9 paires de macros permettant de spécifier le type de caractère voulu. Ces macros se répartissent en trois groupes : les macros indiquant la forme (droit (up), italique (italic), penché (slanted) et petite capitale (small cap)), les macros indiquant la graisse (moyen (medium) ou gras (boldface)) et les macros indiquant la famille (romain (roman), sans empattement (sans serif) et non proportionnel dite type machine à écrire (typewriter)). Le tableau 6 présente ces 18 macros.

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

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

Macro I	Macro II	Résultat
Macros pour la forme		
`\textup`	`\upshape`	Un exemple pour voir
`\textit`	`\itshape`	*Un exemple pour voir*
`\textsl`	`\slshape`	*Un exemple pour voir*
`\textsc`	`\scshape`	Un exemple pour voir
Macros pour la graisse		
`\textmd`	`\mdseries`	Un exemple pour voir
`\textbf`	`\bfseries`	**Un exemple pour voir**
Macros pour la famille		
`\textrm`	`\rmfamily`	Un exemple pour voir
`\textsf`	`\sffamily`	Un exemple pour voir
`\texttt`	`\ttfamily`	`Un exemple pour voir`

Tab. 6 – Macros de type de caractère

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

Source

```
{\sffamily {\bfseries Qui}, sur la terre,
  voudrait utiliser du {\bfseries gras}
  sans empattement ?}

\textbf I want \textbf that !
```

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

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

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

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

I want **that**

Source

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

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

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

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

Dans le livre *LaTeX est* meilleur *que les autres.*
Nous verrons que LaTeX est *meilleur* que les autres.

Source

```
Dans le livre \textit{\LaTeX{} est
  \emph{meilleur} que les autres}.

Nous verrons que \LaTeX{} est
\emph{meilleur} que les autres.
```

4.3 Commandes de paragraphe

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

Avant de voir ces macros, il est bon de savoir qu'un paragraphe, pour TeX, est un texte qui se termine soit par une ligne vide (comme on a déjà vu), soit par la macro `\par`[5].

Il existe quatre présentations classiques de paragraphe : justifié (ce qui est la présentation par défaut), au fer à gauche[6] (seul le bord gauche du paragraphe est aligné avec la marge), au fer à droite (le bord droit du paragraphe aligné avec la marge de droite) et centré. Pour obtenir les trois dernières présentations, il existe à chaque fois une macro et un environnement. Voyons les environnements en premier :

Source

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

\begin{flushright}
  Voici un paragraphe au fer à droite où
  l'alignement ne se fait plus qu'au
  niveau du bord droit. La syntaxe
  anglaise peut être traduite par
  \emph{aligné~à~droite}.
\end{flushright}\par % pour changer un peu
\begin{flushleft}
  Un paragraphe au fer à gauche. La syntaxe
  anglaise est évidemment cohérente en
  remplaçant droite~par~gauche.
\end{flushleft}

\begin{center}
  Paragraphe centré. Seules les lignes
  relativement courtes ont un intérêt sinon,
  on ne voit pas très bien le caractère
  centré de la chose !
\end{center}
```

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

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

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

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

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

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

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

[6] La dénomination exacte serait plutôt « en drapeau au fer à gauche » ce qui est plus explicite.

Source

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

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

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

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

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

Source

```
{\centering Exemple qui tombe à l'eau.}\par
{\centering C'est mieux ainsi !\par}
```

Exemple qui tombe à l'eau.

C'est mieux ainsi !

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

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

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

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

TeX étant très scrupuleux, il permet d'utiliser toutes les unités typographiques usuelles anglo-saxonnes et françaises ainsi que les unités courantes du système internationale plus une unité à lui qui est le point d'échelle et trois unités qui dépendent de la fonte en cours. Le tableau 7 montre toutes les unités à disposition.

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

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

En plus de l'indentation, on peut également régler les sauts verticaux entre paragraphes. Là aussi, il existe une dimension qui indique cette distance, en l'occurrence `\parskip`. On peut évidemment procéder au même type de réglage que pour l'indentation mais, pour cette dimension, on a tout intérêt à utiliser une caractéristique fondamentale des dimensions de TeX : il s'agit en fait de ressorts comportant un certain étirement et compression plutôt que de longueurs fixes. Le mot-clé `plus` introduit l'étirement et le mot-clé `minus` introduit la compression.

unité	nom anglais	nom français	correspondances	
1 bp	big point	gros point	$= 1{,}003\,74$ pt $= 0{,}352\,68$ mm $= 0{,}938\,06$ dd	$= 65\,781$ sp $= 1{,}388\,55 \times 10^{-2}$ in
1 cc	cicero	cicéro	$= 12{,}840\,1$ pt $= 4{,}512\,51$ mm $= 12$ dd	$= 841\,789$ sp $= 0{,}177\,658$ in
1 cm	centimeter	centimètre	$= 28{,}452\,74$ pt $= 10$ mm $= 26{,}591$ dd	$= 1\,864\,679$ sp $= 0{,}393\,692$ in
1 dd	didot point	point didot	$= 1{,}07$ pt $= 0{,}376\,066$ mm	$= 70\,124$ sp $= 1{,}480\,865 \times 10^{-2}$ in
1 in	inch	pouce	$= 72{,}27$ pt $= 25{,}4$ mm $= 67{,}541\,5$ dd	$= 4\,736\,286$ sp
1 pc	pica	pica	$= 12$ pt $= 4{,}217\,18$ mm $= 11{,}214\,86$ dd	$= 786\,432$ sp $= 0{,}166\,031$ in
1 pt	point	point	 $= 0{,}351\,14$ mm $= 0{,}934\,57$ dd	$= 65\,536$ sp $= 1{,}382\,446 \times 10^{-2}$ in
1 mm	millimeter	millimètre	$= 2{,}845\,26$ pt $= 2{,}659\,09$ dd	$= 186\,467$ sp $= 3{,}936\,768 \times 10^{-2}$ in
1 sp	scale point	point d'échelle	$= 1{,}528\,78 \times 10^{-5}$ pt $= 5{,}362\,851 \times 10^{-6}$ mm $= 1{,}426 \times 10^{-5}$ dd	$= 2{,}109 \times 10^{-7}$ in
1 em	em-space	cadratin	Il s'agit normalement de la largeur de la lettre « M » de la fonte courante mais il s'agit en fait d'une unité déclarée par le concepteur de la fonte qui a le droit de faire ce qu'il veut.	
1 ex	ex-space	hauteur de 'x'	Il s'agit normalement de la hauteur de la lettre « x » de la fonte courante. La remarque a propos de l'unité **em** s'applique.	
1 mu	math unit	unité mathématique	Correspond à $1/18^{\text{e}}$ de cadratin dans la fonte mathématique courante. Cette unité n'a de sens qu'en mode mathématique.	

TAB. 7 – Unités de mesures

Par exemple, dans cette fin de section, la dimension `\parskip` a été fixée à :
`\setlength{\parskip}{18pt plus4pt minus2pt}`

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

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

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

Le fait de n'avoir spécifié aucun étirement et compression pour la dimension `\parindent`

fait que ces deux composantes sont nulles. Ainsi, l'indentation de paragraphe sera toujours exactement de 3 cm (8 mm dans le reste du manuel). C'est un comportement souhaitable pour l'indentation mais malheureux pour l'espace entre les paragraphes si on souhaite aligner les bas de pages les uns avec les autres.

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

4.4 Espacements

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

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

Source

```
Un\hspace{1.2cm}espace important

\vspace{1.2cm}
Un saut vertical important.
```

Un espace important

Un saut vertical important.

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

Source

```
Un\hspace{1cm}texte\hspace{1cm}difficile%
\hspace{1cm}à\hspace{1cm}composer à cause
des contraintes d'espacement.

Un\hspace{1cm minus4mm}texte%
\hspace{1cm minus4mm}difficile%
\hspace{1cm minus4mm}à%
\hspace{1cm minus4mm}composer à
cause des contraintes d'espacement.
```

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

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

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

Les commandes `\hspace` et `\vspace` sont encore plus puissantes que ce qui vient d'être dit : lorsqu'une commande `\hspace` survient en début de ligne ou en fin de ligne, son rôle est inhibé. De même, une commande `\vspace` en début ou en fin de page n'a aucune action. C'est une bonne chose, par exemple, c'est ce qui permet

de donner une présentation correcte au en-têtes de section. De façon terriblement schématique, la commande `\section` dans ce document commence par les commandes suivantes :

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

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

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

qui indique que l'espacement entre deux paragraphes sera normalement de 4 points mais pourra en fait varier entre 3 points et 6 points. (Voir section 6.3 page 44 pour la macro `\setlength`.)

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

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

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

Source

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

Un texte.
Un autre texte.
Un autre texte.

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

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

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

Source

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

Super
Super Super Super Super

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

4.5 Listes

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

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

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

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

Voyons trois exemples simples pour commencer :

Source

```
\begin{itemize}
\item en français, les premiers éléments
  d'une liste se terminent par un point
  virgule ;
\item chaque élément commence par une
  minuscule ;
\item le dernier élément a un point.
\end{itemize}

\begin{enumerate}
\item C'est le premier point.
\item Voici le deuxième.
\item Et enfin le dernier.
\end{enumerate}

\begin{description}
\item[Un] Article indéfini singulier.
\item[Deux] Ce n'est plus un article.
  Parfois article introduisant le dual mais
  pas en français.
\item[Trois et etc.] Définitivement plus des
  articles et toujours pluriels.
\end{description}
```

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

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

Un Article indéfini singulier.

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

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

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

Source

```
\begin{itemize}
\item[\P] une liste avec de jolis symboles ;
\item[\&] très esthétique ;
\item[?] et illisible !
\end{itemize}
```

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

Les listes peuvent être imbriquées les unes dans les autres. Pour la liste `enumerate`, chaque niveau dispose de son propre symbole de numérotation. LaTeX autorise quatre niveaux d'imbrication ce qui est largement suffisant et déjà pas facile à suivre !

Les matières enseignées sont :

1. les mathématiques en raison :
   (a) de leur rigueur ;
   (b) de leur clarté ;
   (c) de leur beauté.
2. la physique parce que :
   (a) il faut que tout le monde vive :
   – les ingénieurs ;
   – les profs ;
   – les autres.
   (b) il y en a quand même un peu besoin.

Source

```
Les matières enseignées sont :
\begin{enumerate}
\item les mathématiques en raison :
  \begin{enumerate}
  \item de leur rigueur ;
  \item de leur clarté ;
  \item de leur beauté.
  \end{enumerate}
\item la physique parce que :
  \begin{enumerate}
  \item il faut que tout le monde vive :
    \begin{itemize}
    \item les ingénieurs ;
    \item les profs ;
    \item les autres.
    \end{itemize}
  \item il y en a quand même un peu besoin.
  \end{enumerate}
\end{enumerate}
```

4.6 Modes mathématiques

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

Il existe deux modes mathématiques : le mode en texte et le mode hors texte. Le mode en texte permet de construire une formule destinée à être incorporée dans le texte d'où une étendue verticale réduite au minimum comme dans $\frac{1}{2}x = \frac{x}{2}$. Le mode hors texte qui permet de composer des formules centrées (par défaut) et qui utilisent toute la ligne rien que pour elles ce qui permet des espacements verticaux plus généreux. Voici la même formule en hors texte :

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

LaTeX permet des constructions telles que $\frac{1}{2}x = \frac{x}{2}$, c'est-à-dire composer une formule en texte comme elle apparaît en hors texte mais, contrairement à ce que pensent beaucoup de personnes, c'est une erreur de vouloir le faire. L'interlignage est complètement détruit ce qui est une faute typographique de débutant !

Comment tape-t-on la formule qui a servi d'exemple ? Tout d'abord, il faut indiquer à LaTeX quand commence et se termine la formule. Le tableau 8 suivant résume les possibilités :

délimiteurs	mode	exemple
`$ ... $`	en texte	$\frac{1}{2}x = \frac{x}{2}$
`\( ... \)`	en texte	$\frac{1}{2}x = \frac{x}{2}$
`\begin{math} ... \end{math}`	en texte	$\frac{1}{2}x = \frac{x}{2}$
`$$ ... $$`	hors texte	$$\frac{1}{2}x = \frac{x}{2}$$
`\[ ... \]`	hors texte	$$\frac{1}{2}x = \frac{x}{2}$$
`\begin{displaymath} ... \end{displaymath}`	hors texte	$$\frac{1}{2}x = \frac{x}{2}$$

TAB. 8 – Délimiteurs des modes mathématiques

Les délimiteurs utilisant le signe dollar sont un codage TEX. Il est préférable d'utiliser un des deux codages purement LATEX pour éviter certains effets bizarres qui peuvent survenir dans des situations complexes. J'avoue que faisant la navette entre TEX et LATEX, j'utilise essentiellement les dollar mais ce n'est pas bien !

Maintenant que nous savons nous mettre en mode mathématique, il reste à voir ce qu'on peut mettre dedans ! Durant cette première approche, nous ne verrons que les bases, les compositions plus complexes seront traitées plus tard à la section 5.5 page 34.

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

4.6.1 Opérations élémentaires

TEX gère évidemment les lettres et les quatre opérations de base. Examinons la formule $x + 3 = y - 5$ obtenue avec le code `\(x+3=y-5\)`. On peut remarquer que des espaces de tailles variées ont été automatiquement ajoutées autour des signes plus, moins et égal. En fait, des espaces mis dans le source n'ont strictement aucune action en mode mathématique (hormis, bien sur, celui de stopper un nom de macro). Par exemple `\(x + 3   = y - 5\)` aurait donné strictement le même résultat. On remarquera également que les lettres sont en italique ; c'est la règle pour les formules de math sauf qu'en France, la règle complète indique que les majuscules ne doivent pas être en italique ce que ne fait pas TEX par défaut (dans ce manuel, on transgressera cette règle typographique et même les majuscules seront en italique).

Les signes plus, moins et égal sont présents sur le clavier mais pas ceux de la multiplication et de la division (celle de l'école primaire). On peut bien évidemment taper un « / » pour la division mais la macro `\div` donne le symbole ÷. Le signe multiplié est obtenu avec la macro `\times` qui donne le symbole ×.

TEX est intelligent et la formule `x\times-3` est comprise comme il se doit, c'est-à-dire comme le produit de x et de -3 ce qui fait que les espaces sont correctement gérées : $x \times -3$. En définitive, si vous tapez votre formule correctement, TEX la composera correctement d'un point de vue typographique.

4.6.2 Structures indispensables

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

Exposants et indices Les exposants et les indices utilisent deux caractères réservés de TEX, à savoir respectivement `^` et `_`. Leur syntaxe est on ne peut plus simple : ce qui suit immédiatement `^` est placé en exposant (même chose avec `_` pour les indices). Il est également possible de mélanger exposant et indice sans ce soucier de l'ordre de spécification. Dans les exemples qui suivent, regardez avec attention la hauteur de ces éléments.

x^2, l^2, x_2, y_2 x^2, x_i, x_i^2 et x_i^2

x^2

x_i

x_i^2

Source

```
1 \(x^2\), \(l^2\),  \(x_2\),     \(y_2\)
2 \(x^2\), \(x_i\),  \(x^2_i\) et \(x_i^2\)
3 \[x^2\]  \[x_i\]   \[x^2_i\]
4 \(x^2\times y^2\), \(x^2y^2\) et \(10^-3\)
```

$x^2 \times y^2$, x^2y^2 et 10^-3

Le dernier exemple est certainement une faute ; on pensait sans doute à 10^{-3} en tapant cette formule. TEX a fait ce qu'on lui a demandé, en l'occurrence écrire 10, suivi de $-$ en exposant puis un 3 (qui n'est plus en exposant car ne suivant pas immédiatement le symbole `^`). Pour obtenir la forme désirée, il faut faire suivre l'accent circonflexe d'un groupe qui contiendra tout ce qui sera en exposant : la formule correcte est obtenue avec la syntaxe `10^{-3}`.

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

Source

```
\(f'(x)=2x\)

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

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

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

Fractions On peut bien sûr employer le symbole / et c'est d'ailleurs mieux dans les formules en texte car les fractions à barre horizontale dans une formule en texte sont quelque peu ridicules (comme dans l'exemple $\frac{1}{2}$). En revanche, dans les formules hors texte, il est souvent préférable de construire une fraction à barre horizontale. Celles-ci se construisent grâce à la macro `\frac` suivi de deux groupes : le premier indiquant le numérateur et le second le dénominateur. Voici un exemple pour clarifier les idées :

Source

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

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

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

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

Source

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

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

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

Source

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

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

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

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

Source

```
\[\sqrt{\sqrt{25}+4} = 3\]
en texte cela donne :
\(\sqrt{\sqrt{25}+4} = 3\)
```

$$\sqrt{\sqrt{25}+4} = 3$$

en texte cela donne : $\sqrt{\sqrt{25}+4} = 3$

La macro `\sqrt` permet des racines $n$ième en précisant le n entre crochets juste après la macro.

Source

```
  \[f_n(x)=\sqrt[n]{x+1}\]

```

$$f_n(x) = \sqrt[n]{x+1}$$

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

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

La macros `\vdots` place trois points verticalement ($\vdots$) et la macro `\ddots` les place en diagonal ($\ddots$). On les utilise plutôt dans des matrices (Cf. section 5.5).

$$f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$$

Source

```
\[f(x_1,\ldots,x_n) = x_1+\cdots+x_n\]
```

4.6.3 Symboles en vrac

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

Lettres grecques Elles sont obtenues en faisant précéder leur nom d'une contre-oblique. Par exemple `\(\beta\)` donnera β. Les majuscules sont obtenues en mettant la première lettre de leur nom en majuscule : `\(\Omega\)` donnera ainsi Ω.

Seules les lettres n'existant pas dans l'alphabet romain sont spécifiées. Pour les minuscules, cela n'exclut que le omicron (`\omicron` est donc une macro inexistante) obtenu avec un « o », en revanche, c'est beaucoup plus fréquent pour les majuscules puisqu'il n'y a que 11 majuscules grecques n'ayant pas d'équivalent graphique dans l'alphabet romain (`\Gamma` Γ, `\Delta` Δ, `\Theta` Θ, `\Lambda` Λ, `\Xi` Ξ, `\Pi` Π, `\Sigma` Σ, `\Upsilon` Υ, `\Phi` Φ, `\Psi` Ψ et `\Omega` Ω).

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

`\epsilon`	ϵ	et	`\varepsilon`	ε		`\theta`	θ	et	`\vartheta`	ϑ
`\pi`	π	et	`\varpi`	ϖ		`\rho`	ρ	et	`\varrho`	ϱ
`\sigma`	σ	et	`\varsigma`	ς		`\phi`	ϕ	et	`\varphi`	φ

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

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

Source

```
Soit \(f\) une fonction appartenant à
\(\mathcal{F}\).
```

Seules les majuscules sont accessibles.

Opérateurs binaires En plus des signes + et − on trouve un grand nombre d'opérateurs binaires. Ces signes seront séparés de ce qui les entoure de la même façon que pour une addition ou une soustraction. La liste complète est donnée au tableau 9.

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

TAB. 9 – Opérateurs binaires

Relations En plus des signes =, > et <, on trouve également un grand nombre de relations. Ces signes seront séparés de ce qui les entoure par les mêmes espacements que ceux entourant le signe =. La liste complète est donnée dans le tableau 10

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

TAB. 10 – Relations

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

Grands opérateurs Ils sont indiqués par soucis d'exhaustivité mais nous les étudierons à la section 5.5. Indiquons toutefois que leur taille varie selon qu'on se trouve en mode en texte ou en mode hors texte. Le tableau 11 en dresse la liste complète :

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

TAB. 11 – Grands opérateurs

Délimiteurs Les délimiteurs sont des symboles qui encadrent une sous-formule. Ils sont présentés au tableau 12 et, comme pour les grands opérateurs, ils ne seront étudiés qu'à la section 5.5.

(	$($	)	$)$	\uparrow	$\uparrow$
[	$[$	]	$]$	downarrow	$\downarrow$
\{	$\{$	\}	$\}$	\updownarrow	$\updownarrow$
\lfloor	$\lfloor$	\rfloor	$\rfloor$	\Uparrow	$\Uparrow$
\lceil	$\lceil$	\rceil	$\rceil$	\Downarrow	$\Downarrow$
\langle	$\langle$	\rangle	$\rangle$	\Updownarrow	$\Updownarrow$
/	$/$	\backslash	$\backslash$		
|	$\|$	\\|	$\|\|$		

TAB. 12 – Délimiteurs

Flèches LaTeX propose une vaste panoplie de flèches. Le tableau 13 en dresse la liste complète.

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

TAB. 13 – Flèches

Symboles inclassables Tous ces symboles sont des symboles ordinaires, c'est-à-dire qu'ils se comporteront comme des lettres au niveau des espacements. Ils sont présentés au niveau du tableau 14.

\aleph	$\aleph$	\prime	$\prime$	\forall	$\forall$	\infty	∞
\hbar	$\hbar$	\emptyset	$\emptyset$	\exists	$\exists$	\triangle	$\triangle$
\imath	$\imath$	\nabla	∇	\neg	$\neg$	\clubsuit	$\clubsuit$
\jmath	$\jmath$	\surd	$\surd$	\flat	$\flat$	\diamondsuit	$\diamondsuit$
\ell	ℓ	\top	$\top$	\natural	$\natural$	\heartsuit	$\heartsuit$
\wp	$\wp$	\bot	$\bot$	\sharp	$\sharp$	\spadesuit	$\spadesuit$
\Re	$\Re$	\\|	$\|$	\backslash	$\backslash$	\partial	∂
\Im	$\Im$	\angle	$\angle$				

TAB. 14 – Symboles mathématiques divers

Noms de fonctions Sous ce terme générique, on trouve tous les mots qui exigent d'être écrits en caractères romains (alors que par défaut, les lettres sont écrites en italique). Le tableau 15 indique toutes les possibilités mises à part les macros \bmod et \pmod. \bmod produit le texte « mod » mais le considère comme un opérateur binaire (donc avec des espaces de chaque côté). \pmod est une macro qui écrit « mod » avec ce qui suit entre parenthèses. Un petit exemple pour mieux comprendre :

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

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

Source

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

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

TAB. 15 – Noms de fonction

accents Les macros servant à accentuer en mode texte ne sont quelquefois plus valides en mode mathématique. Ce dernier offre 10 sortes d'accents plus 2 de taille variable. Le tableau 16 montre ces macros.

Les deux accents de taille variable n'ont en fait que trois tailles possibles (les deux plus grandes tailles étant montrées dans le tableau.

`\acute`	$\acute{x}$	`\check`	$\check{x}$	`\grave`	$\grave{x}$	`\vec`	$\vec{x}$	
`\bar`	$\bar{x}$	`\ddot`	$\ddot{x}$	`\hat`	$\hat{x}$	`\widehat`	$\widehat{xy}$	$\widehat{xyz}$
`\breve`	$\breve{x}$	`\dot`	$\dot{x}$	`\tilde`	$\tilde{x}$	`\widetilde`	$\widetilde{xy}$	$\widetilde{xyz}$

TAB. 16 – Accents mathématiques

Synonymes Certaines macros sont particulièrement utilisées dans les textes mathématiques et elles possèdent des synonymes. Ces synonymes ont pour but de leur donner un nom plus court ou plus parlant (pour un anglophone). Le tableau 17 donne les équivalents possibles.

`\not=` ($\neq$)	ou `\ne` ou `\neq`		
`\geq` ($\geq$)	ou `\ge`	`\leq` ($\leq$)	ou `\le`
`\}` ($\}$)	ou `\rbrace`	`\{` ($\{$)	ou `\lbrace`
`\leftarrows` ($\leftarrow$)	ou `\gets`	`\rigtarrow` ($\rightarrow$)	ou `\to`
`\wedge` ($\wedge$)	ou `\land` (logical and)	`\vee` ($\vee$)	ou `\lor` (logical or)
`\neg` ($\neg$)	ou `\lnot` (logical not)	`\ni` ($\ni$)	ou `\owns`
`\\|` ($\|\|$)	ou `\Vert`	`|` ($\|$)	ou `\vert`

TAB. 17 – Synonymes de macros mathématiques

5 Quelques présentations plus évoluées

5.1 Structuration des documents

5.1.1 Commandes pour le plan

LaTeX connaît 7 niveaux de plan. Dans l'ordre d'importance on trouve `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` et `\subparagraph`. La syntaxe est on ne peut plus simple : il suffit d'indiquer la macro suivie par un groupe qui donnera le titre. Ainsi les trois titres précédents ont été obtenus avec les lignes :

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

Pourquoi les titres de chapitre de ce manuel sont composés avec la macro `\section` au lieu de la macro `\chapter` ? C'est que pour limiter la colère à la photocopie, le style de document est `article` et non pas `report` ou `book`. Dans la classe `article` la macro `\chapter` n'existe pas. D'autre part, la mise en page n'est pas la même en fonction de la classe de document. Essentiellement, les espaces verticaux seront plus ou moins importants. Par exemple, un `\chapter` prendra une page complète et sera composé sur une page impaire dans le style `book` alors qu'avec la classe `report` ce ne sera pas le cas. Il serait trop long de décrire exactement ce que font ces 7 macros selon les 3 classes possibles de document. Il suffit d'être logique et tout se passera bien.

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

Si le document a été correctement structuré avec ces 7 macros, il devient simplissime de construire la table des matières. En effet, un appel à la macro `\tableofcontents` suffit à faire tout le travail. Certaines présentations nécessitent deux compilations successives pour être correctes. La construction d'une table des matières fait partie de celles-ci. La première compilation construit un fichier auxiliaire dans lequel sont rangés les renseignements concernant les titres (énoncés et numéros de page) et la seconde compilation lit ce fichier pour construire effectivement la table des matières.

5.1.2 Références

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

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

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

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

Ainsi, la phrase :

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

sera composée comme suit :

Les commandes relatives au plan peuvent être vues à la section 5.1.1 page 27. (Vous pouvez vérifier, je suis certain du résultat puisque ce n'est pas moi qui ai écrit les numéros.)

5.2 Table des matières, index

Comme on vient de le voir, la table des matières est on ne peut plus simple à produire : il suffit de taper la macro `\tableofcontents` au niveau où on veut que celle-ci soit écrite et d'avoir pris soin de construire le plan avec les macros vues à la section 5.1.1.

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

```
\usepackage{makeidx}
```

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

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

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

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

5.3 En-têtes, pieds de page, notes

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

Par défaut, LaTeX propose quatre types d'en-tête et pied de page :

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

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

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

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

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

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

```
\pagestyle{headings}
```

a été écrite dans le préambule de ce manuel.

On peut vouloir modifier localement cette présentation. Pour cela, il suffit d'appeler `\thispagestyle` suivie, là aussi, d'un groupe indiquant le type choisi pour la page courante. Seule la page où se situe cette macro prendra le style précisé, les pages suivantes reprenant le style indiqué avec la macro générale `\pagestyle`. Il existe une petite exception (importante en pratique) pour la première page du document. Si on veut obtenir un style particulier pour cette première page, il faut taper la macro `\thispagestyle` au niveau du préambule. Par exemple, l'auteur de ce manuel a tapé la commande :

```
\thispagestyle{empty}
```

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

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

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

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

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

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

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

De plus, un appel à la macro `\pagenumbering` remet le compteur de page à 1. Ainsi, toujours dans ce manuel, une fois que la page de titre a été composée, les trois lignes suivantes étaient :

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

Traduisons ces trois lignes en français :
- commencer une nouvelle page (`\newpage`) ;
- passer alors en numérotation romaine en commençant à « i » (`\pagenumbering{roman}`) ;
- construire la table des matières (`\tableofcontents`) ;
- commencer une nouvelle page (`\newpage`) ;

– passer alors en numérotation arabe en commençant à « 1 ».

Vous pouvez vérifier que c'est effectivement ce qui s'est passé !

Comme celle-ci.

Il nous reste à voir les notes de bas de page[7] et les notes marginales. Les notes de bas de page sont appelées grâce à la macro `\footnote` suivie d'un groupe donnant le contenu de la note. En particulier le numéro de l'appel est géré de façon automatique (par défaut, il est remis à 1 au début de chaque section avec la classe `article` ou au début de chaque chapitre avec les classes `book` et `report`. Quant aux notes marginales, elles sont appelées avec la macro `\marginpar` et sont suivies, là aussi, d'un groupe indiquant leur contenu.

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

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

5.4 Tableaux

Ah ! Les tableaux ! C'est en partie à l'aisance avec laquelle un utilisateur construit un tableau compliqué qu'on reconnaît un expert de LaTeX. Autant dire qu'il s'agit là d'une notion un peu délicate à manipuler. C'est logique : construire un tableau est quelque chose de très subtil qui nécessite souvent une bonne dose de réflexion. Dans cette section, nous ne verrons que les macros définies directement par LaTeX ; quelques présentations un peu plus compliquées seront présentées lors de l'étude des extensions (Cf. section 7.4).

Un tableau est défini grâce à l'environnement `tabular` où le groupe qui suit le `\begin{tabular}` indique le motif du tableau (nombre et types des colonnes). Il y a quatre type de base pour les colonnes :

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

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

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

p{*d***}** indique une colonne qui peut accueillir des **p**aragraphes dont la largeur est donnée par la dimension *d* (Cf. tableau 7 page 17 pour les unités acceptées).

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

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

Le *contenu du tableau* va indiquer le texte qui va apparaître dans chaque cellule du tableau. Pour passer d'une cellule à une autre, on emploie le caractère spécial `&` et pour passer à la ligne suivante la séquence de caractères `\\`. Voici deux exemples :

[7] Comme celle-là.

gauche	centré	droite	paragraphe de deux centimètres
xxx	yyy	zzz	ppp

gauche	centré	droite	paragraphe de deux centimètres
xxxxxxxx	yyyyyyyy	zzzzzzzz	pppppppp

Source

```
\begin{tabular}{lcrp{2cm}}
  gauche & centré & droite &
  paragraphe de deux centimètres \\
  xxx & yyy & zzz & ppp
\end{tabular}

\vspace{1cm}
\begin{tabular}{lcrp{2cm}}
  gauche & centré & droite &
  {\raggedright paragraphe de deux
    centimètres\par} \\
  xxxxxxxx &
  yyyyyyyy &
  zzzzzzzz &
  pppppppp
\end{tabular}
```

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

Sans précaution particulière, le motif p produit des résultats peu esthétiques car sa largeur est généralement très faible et le paragraphe est construit par défaut pour être justifié ce qui est incompatible (voyez le piètre résultat du premier tableau). La solution est, souvent, de forcer la composition au fer à gauche grâce à la macro `\raggedright`. N'oublions pas alors de faire agir cette macro sur un vrai paragraphe (obtenu ici grâce à la macro `\par`). La présence des accolades peut sembler mystérieuse ; sans celles-là, il y aurait eu une erreur à la compilation car la macro \\ indique de passer à la ligne suivante mais il s'agit d'une commande interdite lorsqu'on se trouve entre deux paragraphes. Les accolades permettent de créer un paragraphe à l'intérieur sans qu'il soit visible à l'extérieur : tordu n'est-ce pas ? C'est le genre de petite feinte qu'il vaut mieux connaître pour ne pas perdre trop de temps lors des premiers pas sous LaTeX.

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

Pour obtenir des tableaux qui en soient vraiment, il reste à gérer les réglures. Les réglures verticales, celles séparant les colonnes, sont déclarées au niveau du motif avec le caractère |. Les réglures horizontales sont déclarées dans le contenu du tableau avec la macro `\hline` qui fera suite à la fin de ligne. Reprenons l'exemple précédent en ajoutant des réglures :

gauche	centré	droite	paragraphe de deux centimètres
xxx	yyy	zzz	ppp
xxx xxx	yyy yyy	zzz zzz	ppp ppp

Source

```
\begin{tabular}{|l|c|r|||p{2cm}|}
  \hline
  gauche & centré & droite &
  {\raggedright paragraphe de deux
    centimètres\par}\\
  \hline
  xxx & yyy & zzz & ppp\\
  \hline\hline
  xxx xxx & yyy yyy & zzz zzz & ppp ppp\\
  \hline
\end{tabular}
```

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

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

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

où *nb* indique le nombre de colonnes devant être fusionnées (pour un changement de motif sans fusion, on prendra $nb = 1$), *motif* est le motif de remplacement pour cette cellule et *texte* est le texte qui sera placé dans cette cellule. Voici un premier exemple moins artificiel que les précédents où on ne fait que modifier le motif pour obtenir une ligne de titre.

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

Source

```
\begin{tabular}{|l|r|l|}
  \hline
  \multicolumn{1}{|c|}{Corps} &
  \multicolumn{1}{c|}{\O\ (km)} &
  \multicolumn{1}{c|}{densité} \\ \hline
  Soleil   & 1\,392\,000 & 1,409 \\
  Mercure  &      4\,840 & 5,50  \\
  Vénus    &     12\,390 & 5,25  \\
  Terre    &     12\,760 & 5,517 \\
  Mars     &      6\,800 & 3,94  \\ \hline
\end{tabular}
```

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

La macro `\,` n'a rien à voir avec les tableaux : il s'agit d'une espace fine (on verra les différents types d'espaces à la section 5.5).

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

Source

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

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

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

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

sera un peu embêtante à taper, sans compter les risques d'erreur (taper 9 ou 11 cellules au lieu de 10). LaTeX permet un raccourci d'écriture sous la forme :

`|*{10}{c|}`

qui indique que le motif sera constitué d'un « | », puis d'une répétition de 10 fois le texte « c| » ce qui donnera bien le résultat souhaité. L'astérisque est suivi de deux groupes : le premier indique le nombre de répétitions et le second groupe le sous-motif qui devra être répété. Dans notre exemple, cela vaut à peine le

coup puisque il n'y a que deux répétitions. Enfin ! Pour des raisons pédagogiques, voici la traduction du motif précédent :

```
|l|*{2}{|c|c|c|}
```

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

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

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

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

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

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

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

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

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

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

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

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

Cela n'a rien de particulièrement excitant ! De plus, avec ce qui a été présenté jusqu'à maintenant, il est tout à fait possible de le composer même si cela va entraîner un côté un peu pénible. En fait, le côté un peu magique de l'affaire et qui ne transparaît pas au niveau de la sortie est que le corps de ce tableau a été tapé sous la forme :

```
2  &     & 3  &     \\
4  & non & 5  &     \\
```

```
6  & non & 7  &     \\
8  & non & 9  & non \\
10 & non & 11 &     \\
```

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

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

5.5 Mathématiques complexes

Voici le deuxième chapitre consacré aux mathématiques. Nous verrons encore quelques points supplémentaires lors de la présentation de l'extension *amsmath* à la section 7.3. Pour l'instant, nous en resterons à ce que LaTeX offre de manière standard.

5.5.1 Fontes mathématiques

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

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

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

Les fontes mathématiques présentent les mêmes possibilités de modification que celles du texte courant mais en modifiant le nom des macros. Ainsi, certaines macros `\text...` vues à la section 4.2.2 ont un équivalent dans les modes mathématiques en changeant le « `text` » en « `math` ». Le tableau 18 montre l'effet de toutes ces macros existant en mode mathématique : on voit que seules les lettres romaines, les chiffres et les majuscules

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

TAB. 18 – Fontes mathématiques

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

TeX gère automatiquement la taille des caractères dans les formules en fonction de leurs emplacements (exposant, exposant d'exposant, composantes de fraction, etc.) et du mode hors texte ou en texte. En tout, il y a quatre tailles prédéfinies : la taille hors texte (display), texte (text), scripte (script) et sous-scripte (scriptscript). La taille peut alors être forcée grâce aux macros `\displaystyle`, `\textstyle`, `\scriptstyle` et `\scriptscriptstyle` (la macro `\displaystyle` avait été présentée brièvement à la section 4.6.2). Tout ce

qui se trouve après ces macros va prendre la taille spécifiée, à moins que des automatismes de LaTeX entrent en jeu pour réduire cette taille (fraction, exposant, ...). Voici un exemple de leur utilisation :

Source

```
Un nombre énorme :
\(2^{2^{2^{2^{2^{2^{2^{2}}}}}}}\approx
3{,}4\times10^{38}\)

Un nombre énorme :
\(2^{2^
  {\scriptstyle2^
    {\scriptstyle2^
      {\scriptstyle2^
        {\scriptstyle2^
          {\scriptstyle2^
            {\scriptstyle2}}}}}}}\approx
3{,}4\times10^{38}\)
```

Un nombre énorme : $2^{2^{2^{2^{2^{2^{2^{2}}}}}}} \approx 3{,}4 \times 10^{38}$

Un nombre énorme : $2^{2^{2^{2^{2^{2^{2^{2}}}}}}} \approx 3{,}4 \times 10^{38}$

ou encore, plus parlant :

Source

```
Le début du développement du nombre d'or en
fraction continue donne :
\[1+\frac{1}{1+
  \frac{1}{1+
    \frac{1}{1+
      \frac{1}{1}}}}=1{,}6\]

Le début du développement du nombre d'or en
fraction continue donne :
\[1+\displaystyle\frac{1}{1+
  \displaystyle\frac{1}{1+
    \displaystyle\frac{1}{1+
      \displaystyle\frac{1}{1}}}}=1{,}6\]
```

Le début du développement du nombre d'or en fraction continue donne :

$$1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1}}}}=1{,}6$$

Le début du développement du nombre d'or en fraction continue donne :

$$1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1}}}}=1{,}6$$

5.5.2 Grands opérateurs

Les grands opérateurs sont des opérateurs pouvant changer de taille en fonction du mode mathématique et s'écrivant souvent avec des indices et/ou des exposants. Le tableau 11 page 25 en dresse la liste complète ; nous allons maintenant voir comment les utiliser.

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

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

Source

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

On sait que
\[\sum_{n=0}^\infty \frac{1}{n!}\]
peut être la définition de \(e\).
```

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

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

peut être la définition de e.

Comme il a été précisé à la section 4.6, on peut être tenté d'écrire quelque chose du type : « On sait que $\displaystyle\sum_{n=0}^{\infty} \frac{1}{n!}$ peut être la définition de e ». Toujours pour ne pas massacrer l'interligne, cette façon de faire est fortement déconseillée ! Incidemment, le code pour la formule qui a permis cette horreur a été :

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

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

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

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

Source

```
Comme \(\int_0^2 t\,dt=2\),
un triangle \ldots

Comme
\[\int_0^2 t\,dt=2\]
un triangle \ldots
```

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

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

un triangle ...

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

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

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

Source

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

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

5.5.3 Délimiteurs

Les délimiteurs sont des symboles destinés à encadrer des sous-formules. La liste complète des délimiteurs a été présentée au tableau 12 page 25. Si ces symboles n'étaient que cela, ils ne seraient guère utiles et n'auraient de sens que pour l'humain. En réalité, les délimiteurs peuvent voir leur taille modifiée de façon fine, soit manuellement, soit automatiquement. Nous ne verrons ici que la façon automatique (de toute façon plus utile) et nous renverrons le lecteur avide au stage de perfectionnement ou à des ouvrages spécialisés !

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

Source

```
\[f(\frac{1+x}{x})=x^2+(\frac{1}{x})^2\]
```

$$f(\frac{1+x}{x}) = x^2 + (\frac{1}{x})^2$$

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

Source

```
\[f\left( \frac{1+x}{x} \right)
  =x^2+\left( \frac{1}{x} \right)^2\]
```

$$f\left(\frac{1+x}{x}\right) = x^2 + \left(\frac{1}{x}\right)^2$$

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

5.5.4 Espacements

Comme nous l'avons déjà dit, un espace tapé au niveau d'une formule ne sert à rien sauf à stopper un nom de macro ou bien à présenter les choses de façon lisible pour un humain. Or, il peut arriver que l'on veuille

insérer des espaces au niveau d'une formule. Les macros d'espacement horizontal vues pour le texte — \␣ (macro espace), \hspace — ainsi que l'espace insécable ~ fonctionnent également en mode mathématique. Le tableau 19 montre sept types d'espaces que LaTeX permet en mode mathématique (dont quatre disponibles en mode texte) :

Macro	Signification	Exemple	Mode texte
\!	espace négative	‖	non
\,	espace fine	||	oui
\:	espace moyenne	||	non
\;	grande espace	||	non
\␣	espace intermot	| |	oui
\quad	espace cadratin	| |	oui
\qquad	double espace cadratin	| |	oui

TAB. 19 – Macros d'espacements

Le tableau 20 montre quelques utilisations classiques de ces différentes espaces qu'il est bon de connaître pour obtenir une lecture facile d'un texte. On arrive vraiment à de la typographie très fine et il n'est pas obligatoire de se souvenir de toutes ces présentations. D'autre part, certaines constructions sont sujettes à discussions (parfois enflammées) donc vous prenez et vous en faites ce que vous voulez ! Tous ces exemples ont été repris tels quels dans le TeXbook. Pensez-y surtout pour les intégrales multiples et méfiez-vous de la racine carrée : avec ces deux conseils, vous pourrez éviter les présentations malheureuses.

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

TAB. 20 – Exemples classiques de correction d'espacement

5.5.5 Empiler verticalement

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

Nous avons déjà vu les accents mathématique et le tableau 16 page 27 en dressait la liste complète. Il existait un accent permettant de surligner une lettre (`\bar`) mais son emploi pour placer une barre au-dessus d'une formule plus longue n'est pas correct :

$\bar{x}+\bar{y}$

Source
```
\(\bar x \bar + \bar y\)
```

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

$\overline{x+y}$

Source
```
\(\overline{x+y}\)
```

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

$\underline{x+y}$

Source
```
\(\underline{x+y}\)
```

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

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

Pour n pair, $2^n = \overbrace{2\times\cdots\times 2}^{\frac{n}{2}} \times \underbrace{2\times\cdots\times 2}_{\frac{n}{2}}$

Source
```
Pour \(n\) pair, \(2^n=
  \overbrace{2\times\cdots\times2}
    ^{\textstyle\frac{n}{2}}
\times
  \underbrace{2\times\cdots\times2}
    _{\textstyle\frac{n}{2}}\)
```

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

$2^n = \overbrace{2\times\underbrace{2\times\cdots\times 2}_{n-2}\times 2}^{n}$

Source
```
\(2^n=\overbrace{2\times\underbrace{
    2\times\cdots\times2}_{n-2}
  \times2}^n\)
```

et un exemple délirant pour voir que tout est permis !

$\overbrace{\overbrace{\overbrace{2\times2}^{2}\times\overbrace{2\times2}^{2}}^{4}\times\overbrace{\overbrace{2\times2}^{2}\times\overbrace{2\times2}^{2}}^{4}}^{8}$

Source
```
\(\overbrace{\overbrace{\overbrace
    {2\times2}^2
    \times\overbrace
    {2\times2}^2
    }^{\scriptstyle 4}
  \times\overbrace{\overbrace
    {2\times2}^2
    \times\overbrace
    {2\times2}^2
    }^{\scriptstyle 4}
  }^{\scriptstyle 8}\)
```

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

$\vec{x} \overset{\text{déf}}{=} (x_1, x_2, \ldots, x_n)$
$x \overset{f}{\longmapsto} f(x)$

Source
```
\(\vec x\stackrel{\mathrm{d\acute ef}}{=}
  (x_1,x_2,\ldots,x_n)\)

\(x\stackrel{f}{\longmapsto} f(x)\)
```

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

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

5.5.6 Matrices

De façon générale, les matrices sont considérées comme des tableaux, la seule différence étant que les tableaux étaient construits en utilisant l'environnement `tabular` alors que les matrices seront construites avec l'environnement `array`. Un premier exemple pour voir que ce n'est vraiment pas différent :

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

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

Pour les « vraies » matrices, il faudra ajouter des parenthèses sous la forme de délimiteurs extensibles (donc avec les préfixes `\left` et `\right`). La forme pour les déterminants suivra le même principe en prenant des barres verticales comme délimiteurs. L'exemple précédent devient alors :

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

Source
```
\[ \varphi(\lambda) = \left|
  \begin{array}{*{4}{c}}
  1-\lambda & 2 & 3 & 4 \\
  2 & 3-\lambda & 4 & 1 \\
  3 & 4 & 1-\lambda & 2 \\
  4 & 1 & 2 & 3-\lambda \\
\end{array} \right| \]
```

LaTeX définit le délimiteur « vide » qui ne produit aucun symbole. Il est souvent utile dans certains cas faisant appel aux tableaux. Par exemple :

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

Source
```
\[\left.\begin{array}{l}
    x\in E \\
    \quad\mbox{ou} \\
    x\in F \\
  \end{array}\right\}
\iff x+x^2=0
\]
```

Un autre exemple fréquent est donné par la construction :

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

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

Cet exemple sera repris lors de l'étude de l'extension *amsmath* à la section 7.3 page 47. Notons ici l'utilisation d'un mode mathématique en texte (le `\(i-j\)`) à l'intérieur d'un mode texte (les boîtes `\mbox`) à l'intérieur d'un mode mathématique hors texte (la formule complète) !

6 Macros

6.1 Principe des macros

Nous avons étudié un certain nombre de macros jusqu'à maintenant et nous pouvons nous faire une idée générale de ce qu'est une macro : une commande permettant des actions complexes. En réalité, la signification du terme « macro » est un peu plus précise : une macro est un mini-programme construit à partir de commandes directement compréhensibles par le compilateur (en l'occurrence TEX) et/ou d'autres macros définies préalablement. Ainsi, un utilisateur pourra vouloir se construire ses propres macros pour automatiser certaines tâches répétitives.

Reprenons un des exemples précédents où nous avions écrit la formule :

$$\vec{x} \stackrel{\text{déf}}{=} (x_1, x_2, \ldots, x_n)$$

et supposons que le symbole « $\stackrel{\text{déf}}{=}$ » soit souvent utilisé dans le document. Il y aura tout intérêt à définir une macro qui se charge de recopier la définition de ce symbole au lieu de retaper à chaque fois tous les ordres nécessaires à son obtention.

La première étape va être de choisir un nom (les `\tata`, `\titi` et autre `\toto` sont à proscrire sans hésitation). Appelons notre macro `\defeq` pour des raisons de cohérence avec les autres macros de LATEX. Ensuite le texte que la macro doit remplacer doit être bien compris et on doit être sûr de son résultat. Ici, pas de problème, la macro doit remplacer la suite :

```
\stackrel{\mathrm{d\acute ef}}{=}
```

Il ne reste alors plus qu'à appeler la macro permettant de construire des macros, en l'occurrence `\newcommand`, suivie du nom de la macro que l'on veut définir puis d'un groupe donnant son contenu. Avec notre exemple, cela va donner :

```
\newcommand{\defeq}{\stackrel{\mathrm{d\acute ef}}{=}}
```

Et voilà, c'est tout ! Il ne reste plus qu'à l'utiliser :

$$\vec{x} \stackrel{\text{déf}}{=} (x_1, x_2, \ldots, x_n)$$

Source

```
\[\vec x\defeq(x_1,x_2,\ldots,x_n)\]
```

Il y a d'autres façons de définir des macros mais celle exposée ci-dessus est fortement conseillée par les défenseurs de LATEX. Elle offre, entre autres avantages, la sécurité de ne pas redéfinir des macros déjà existantes. En reprenant l'exemple précédent, on pourrait légitimement se dire que plus un nom de macro est court, plus rapide sera sa frappe et décider d'appeler `\def` notre jolie macro. Si cela avait été possible, il y aurait eu une série de catastrophes et plus rien n'aurait fonctionné correctement car `\def` est une primitive du langage TEX abondamment utilisée (par exemple, la macro `\newcommand` utilise cette primitive) et tout ce qui ferait appel directement ou indirectement à `\def` serait corrompu. D'autre part, les messages d'erreurs affiché par le terminal auraient été réellement abscons et sans rapport évident avec la redéfinition de la primitive donc une erreur très délicate à comprendre.

6.2 Macros à paramètres

Les macros comme celle définie à la section précédente seraient assez utiles mais la puissance de ce concept vient surtout de la possibilité de mettre des paramètres (de 1 à 9). Illustrons cela par un exemple un peu plus évolué. Supposons qu'à plusieurs endroits d'un ouvrage on veuille écrire des citations avec le texte composé au fer à droite et en italique suivi, à la ligne, du nom de l'auteur composé en petite capitale et enfin du titre de l'ouvrage séparé du nom de l'auteur par une virgule, le tout placé dans un bloc de 6 cm de large. Voici un petit exemple pour voir sa réalisation sans définir de nouvelle macro :

Source

```
\begin{flushright}
  \begin{tabular}{@{}p{6cm}@{}}
    {\raggedleft \itshape C'est parce que quelque chose des objets
      extérieurs pénètre en nous que nous voyons les formes
```

```
      et que nous pensons.\par}\\
    {\raggedleft \textsc{\'Epicure}, lettre à Hérodote.\par}
  \end{tabular}
\end{flushright}
```

C'est parce que quelque chose des objets extérieurs pénètre en nous que nous voyons les formes et que nous pensons.

ÉPICURE, lettre à Hérodote.

Petite cerise sur le gâteau : la largeur du texte écrit est exactement de 6 cm car on a pris la peine d'inhiber les espaces intercolonnes en mettant les deux `@{}` au début et à la fin du motif du tableau.

Le tout est un peu embêtant à taper et nous voudrions avoir une macro qui se charge du travail, étant entendu que le texte de la citation, le nom de l'auteur et le titre de l'ouvrage sont variables. Pour cela, il va falloir définir une macro à trois paramètres. Appelons-la `\epigraphe`, le début de sa déclaration se fait sous la forme :

```
\newcommand{\epigraphe}[3]
```

c'est-à-dire comme précédemment mais en rajoutant le nombre de paramètres entre crochets. Ensuite, on tape le texte qui remplacera la macro en indiquant la présence du premier paramètre par la séquence `#1`, celle du deuxième paramètre par `#2` et celle du troisième paramètre par `#3`. Comme nous avons déjà le modèle, cela ne va pas poser de problème insurmontable, il suffit de remplacer les passages correspondants par les séquences `#`n adéquates :

```
\newcommand{\epigraphe}[3]{
  \begin{flushright}
    \begin{tabular}{@{}p{6cm}@{}}
      {\raggedleft \itshape #1\par}\\
      {\raggedleft \textsc{#2}, #3\par}
    \end{tabular}
  \end{flushright}}
```

L'emploi est on ne peut plus simple, il suffit d'écrire la macro puis les trois éléments (dans l'ordre) et le tour est joué. Par exemple :

Source

```
\epigraphe{Voici une macro qui parle d'elle-même (en logique, on appelle cela un quine).
Une macro compliquée qui fonctionne du premier coup est un bonheur rare.}
{Charpentier}{Stage \LaTeX}
```

Voici une macro qui parle d'elle-même (en logique, on appelle cela un quine). Une macro compliquée qui fonctionne du premier coup est un bonheur rare.

CHARPENTIER, Stage LaTeX

L'emploi de macros personnelles ne sert pas qu'à remplacer des séquences longues et répétitives de commandes, il permet également de réaliser des documents faciles à maintenir car mieux structurés. Supposons qu'on réalise un document produisant des recettes de cuisines. Chaque recette va commencer avec le nom du plat qu'on écrira dans un corps plus grand que le reste de la fiche. Il n'y a pas de problème particulier, un source tel que :

```
{\Large Le canard laqué}
```

conviendra parfaitement. Après des semaines de travail, l'ouvrage comporte maintenant plus de 400 fiches lorsque, tout compte fait, les noms des recettes seraient plus jolis s'ils étaient aussi en gras et centrés en haut de la fiche, c'est-à-dire que chaque fiche devrait commencer sur une nouvelle page. Bien évidemment, les éditeurs de texte sont capables de faire des remplacements automatiques mais il s'agit presque toujours d'opérations dangereuses et qui doivent être effectuée sous contrôle humain ; après tout, d'autres textes ont pu être composés dans ce même corps sans être le nom de la recette. Une solution beaucoup plus propre et compréhensible serait de définir dès le début du travail une macro `\titrerecette` dont la déclaration serait :

```
\newcommand{\titrerecette}[1]{{\Large #1}}
```

La présence des doubles accolades est nécessaire car sinon, l'appel `\titrerecette{Le canard laqué}` est remplacé par le texte :

```
\Large Le canard laqué
```

et l'action de la macro `\Large` se poursuivra au-delà de la zone souhaitée alors qu'avec les doubles accolades, la macro est remplacée par le texte :

```
{\Large Le canard laqué}
```

et tout va bien.

Au moment où on veut modifier la présentation des titres des recettes, il suffira de modifier la définition de la macro au niveau du préambule et tout sera recomposé selon les nouvelles exigences. Ici, la macro deviendrait :

```
\newcommand{\titrerecette}[1]{%
  \newpage
  \begin{center}
    \Large #1
  \end{center}
  }
```

Deux petits mots sur cette macro avant de clore le sujet. Nous avons vu précédemment qu'il fallait mettre la macro `\Large` entre accolades pour que son action reste confinée au paramètre de la macro. Ici, ce n'est pas nécessaire car tout environnement se comporte comme des accolades sur ce point. En l'occurrence, l'environnement `center` va limiter la portée de la macro `\Large`. D'autre part, il y a un caractère `%` à la fin de la première ligne. Ce commentaire (vide) sert à ce que TEX ne voit pas la fin de la ligne et ne produisent donc pas d'espace parasite. En fait, cette précaution est ici superflue car une espace précédant un saut de page ne fait de mal à personne mais il existe beaucoup de situations où il faut penser à ce « truc ».

6.3 Compteurs et dimensions

Dans ce manuel, nous avons déjà rencontré les deux dimensions `\parindent` et `\parskip` qui permettaient respectivement d'indiquer l'importance de l'indentation de première ligne et la distance entre deux paragraphes. Nous avons également utilisé, sans le nommer, le registre qui garde le numéro de page en cours. En fait, LATEX gère une multitude de choses en se servant de registres de dimension et de compteurs ; de plus, l'utilisateur peut se créer ses propres registres pour des besoins particuliers.

Étudions d'abord les compteurs, la syntaxe pour les dimensions ne sera pas trop différente. Si on désire utiliser un compteur qui n'est pas défini par LATEX, il va falloir le déclarer. Pour cela, on dispose de la macro `\newcounter` qui sera suivie du nom du compteur (compteur qui, sous LATEX, ne s'écrit pas avec la contre-oblique). Nous allons déclarer un compteur qui servira à une macro destinée à numéroter automatiquement les exercices d'un devoir ; appelons-le `numexo`. Sa déclaration sera donc :

```
\newcounter{numexo}
```

Cette déclaration faite, le compteur `numexo` existe et est initialisé à zéro. Il reste à pouvoir modifier sa valeur, à le transmettre à des macros et à placer sa valeur dans un texte. Voyons ces différents points un à un.

Avec LATEX, il existe deux façons de changer le contenu d'un compteur. On peut placer une valeur particulière grâce à la syntaxe :

`\setcounter{`*cpt*`}{`*val*`}`

qui permet de mettre la valeur *val* dans le compteur *cpt*.

On peut aussi ajouter une valeur en utilisant la construction :

`\addtocounter{`*cpt*`}{`*val*`}`

qui ajoutera la valeur *val* au contenu actuel du compteur *cpt*. Pour une soustraction, il suffit d'ajouter une valeur négative. Pour les multiplications et divisions, vous êtes cordialement invité au stage de perfectionnement ou à lire des documents de référence !

Si le contenu d'un compteur doit être transmis à une macro en tant que paramètre, on pourra utiliser la construction :

`\value{`*cpt*`}`

Un point plus utile que le précédent consiste à afficher le contenu d'un compteur au niveau du document produit. C'est ce que nous voulons avec le compteur d'exercice. Pour cela, LaTeX n'offre pas moins de 7 macros. Nous ne verrons ici que les cinq les plus fréquemment employées :

Commande	Type	Exemple
`\arabic{`*cpt*`}`	nombre arabe	1, 2, 3, ...
`\roman{`*cpt*`}`	nombre romain minuscule	i, ii, iii, ...
`\Roman{`*cpt*`}`	nombre romain majuscule	I, II, III, ...
`\alph{`*cpt*`}`	lettre minuscule	a, b, c, ...
`\Alph{`*cpt*`}`	lettre majuscule	A, B, C, ...

Nous sommes maintenant en possession de tout ce qu'il faut pour construire notre macro gérant automatiquement les numéros d'exercices. Cette macro devra définir un nouveau paragraphe, inhiber l'indentation de première ligne et afficher le texte « **Exercice** *n.* » où *n* sera le numéro de l'exercice en cours suivie d'une espace horizontale assez généreuse. Le code réalisant toutes ces actions peut être défini comme suit :

```
\newcommand{\exo}{\par
  \addtocounter{numexo}{1}%
  \noindent
  \textbf{Exercice \arabic{numexo}}\quad}
```

À la suite de quoi on pourra obtenir quelque chose comme :

Source

```
\exo Repérer toutes les fautes de ce manuel (exercice difficile et long).
\exo Assez tôt dans ce manuel, il avait été dit que le nombre de macros utilisées jusqu'à
ce point était de~78.

Maintenant que ce manuel arrive plutôt vers sa fin, quel est le nombre de macros qui ont
été étudiées ?
\exo Très souvent, l'augmentation d'un compteur avec la macro
\(\mathtt{\backslash}\)\texttt{addtocounter} est de~1 (comme pour l'exemple en cours des
exercices).

Pensez-vous vraiment que \LaTeX{} soit si mal fait qu'il n'ait pas pensé à définir un
raccourci pour une incrémentation d'une unité ?
```

Exercice 1 Repérer toutes les fautes de ce manuel (exercice difficile et long).

Exercice 2 Assez tôt dans ce manuel, il avait été dit que le nombre de macros utilisées jusqu'à ce point était de 78.

Maintenant que ce manuel arrive plutôt vers sa fin, quel est le nombre de macros qui ont été étudiées ?

Exercice 3 Très souvent, l'augmentation d'un compteur avec la macro `\addtocounter` est de 1 (comme pour l'exemple en cours des exercices).

Pensez-vous vraiment que LaTeX soit si mal fait qu'il n'ait pas pensé à définir un raccourci pour une incrémentation d'une unité ?

Facile n'est-ce pas ? Et surtout, cela évite les erreurs de numérotations qui arrivent fréquemment surtout si on remanie l'ordre et le nombre d'exercices en étant un peu pressé (expérience personnelle).

À propos de l'exercice 3 : la réponse est évidemment « non » ! Il arrive effectivement très fréquemment que l'on veuille ajouter 1 à un compteur et la séquence :

```
\addtocounter{cpt}{1}
```

peut être raccourcie en :

```
\stepcounter{cpt}
```

Compliquons un peu les choses en exigeant une macro qui numérote automatiquement les questions à l'intérieur d'un exercice mais qui ait le bon goût de redémarrer à 1 à chaque nouvel exercice. Pour que cela soit joli, le numéro de la question sera écrit en gras et suivi d'un triangle, d'un point et d'une espace normale. Nous appellerons `\question` cette macro et `numq` le compteur associé au numéro de question.

La définition de la macro `\question` ne pose pas de problème : elle ressemble beaucoup à celle de la macro `\exo` :

```
\newcommand{\question}{\par
  \stepcounter{numq}%
  \noindent
  \textbf{\arabic{numq}} \(\triangleright\).\ }
```

Avec les deux macros `\exo` et `\question` telles qu'elles ont été définies, les choses seront correctes pour le premier exercice mais deviendront incorrectes dès le deuxième exercice car le compteur `numq` continuera sur sa lancée. Il existe deux solutions à ce problème : soit on modifie la définition de la macro `\exo` en ajoutant la commande :

```
\setcounter{numq}{0}%
```

au début de la macro, soit on utilise une autre façon de définir le compteur `numq` :

```
\newcounter{numq}[numexo]
```

Cette syntaxe indique que `numq` est un compteur qui sera automatiquement remis à zéro à chaque appel de `\stepcounter{numexo}` (il faudra donc obligatoirement modifier la macro `\exo` pour y remplacer la commande `\addtocounter{numexo}{1}` par la commande `\stepcounter{numexo}`).

Certaines macros gérant les dimensions ressemblent à celles dédiées aux compteurs. Ainsi, on trouve :

```
\newlength{\long}
```

qui permet de définir la dimension *long* (on notera la présence de la contre-oblique pour les dimensions alors qu'il n'y en avait pas pour les compteurs). Lorsqu'une nouvelle dimension est créée, elle est initialisée à `1in` (1 pouce). On a également :

```
\setlength{\long}{val}
```

qui permet de spécifier la valeur *val* à placer dans le registre de dimension *long*. Une longueur doit obligatoirement comporter une des unités listées au tableau 7 page 17 et peut comporter une composante `plus` et/ou une composante `minus` permettant de donner respectivement un étirement et une compression à cette longueur (TEX parle plutôt de ressort), Cf. section 4.4 page 18. Enfin, la dernière macro ressemblant à celles gérant les compteurs est :

```
\addtolength{\long}{val}
```

qui ajoute la valeur *val* au registre *long*. Les trois composantes (longueur, extension et compression) sont ajoutées séparément.

Trois autres macros permettent de spécifier une longueur calculée à partir d'un matériel donné :

```
\settowidth{\long}{objet}
\settoheight{\long}{objet}
\settodepth{\long}{objet}
```

initialisent le registre *long* respectivement avec la largeur, la hauteur et la profondeur de *objet*. Un petit exemple pour bien comprendre qui utilise le fait que la fonte `\texttt{}` a tous ses caractères de la même largeur :

Voici un mot supprimé

Source

```
\newlength{\retour}
\settowidth{\retour}{\texttt{supprimé}}
\texttt{Voici un mot
  supprimé\hspace{-\retour}////////}
```

7 Extensions utiles

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

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

7.1 geometry

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

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

```
\usepackage{geometry}
```

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

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

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

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

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

7.2 times et compagnie

Si vous en avez assez des sempiternelles fontes Computer Modern, il est possible de composer des documents avec d'autres familles de fontes. Le problème est que dans le monde TEX on n'aime pas l'à peu près donc les fontes disponibles sont d'excellentes qualités et par conséquent soit rares, soit chères. Oubliez les fontes True Type qui ne correspondent de toute façon pas aux critères de qualité requis bien qu'il soit possible

de transformer ce genre de fontes sous une forme utilisable par TEX. En résumé et en très gros, deux grands types de fontes sont utilisées dans le monde TEX : les fontes, telles que les Computer Modern, fabriquées à partir du programme METAFONT et les fontes PostScript. Ces fontes sont radicalement différentes. Les fontes produites à partir de METAFONT sont des dessins bitmaps (chaque caractère est décrit sous la forme d'une grille de points) alors que les fontes PostScript sont des fontes vectorielles. La plupart du temps, les fontes METAFONT appartiennent au domaine public (elles sont donc accessibles gratuitement, librement distribuables et/ou modifiables) alors que la plupart des fontes PostScript sont soumises à un copyright classique et sont payantes. Les fontes PostScript offrent un vaste choix de possibilités car elles sont produites par des structures commerciales dont c'est, au moins en partie, le rôle et qui disposent d'un personnel hautement qualifié alors que les fontes METAFONT sont fabriquées par des passionnés de façon autonome sur leur temps libre. Or, produire une fonte de qualité requiert une quantité de travail énorme (j'ai essayé) et beaucoup de connaissances ce qui explique la relative rareté des fontes METAFONT, du moins celles de qualité.

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

```
\usepackage{times}
```

au niveau du préambule et c'est tout.

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

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

Le package `pifont` est un peu spécial car la fonte appelée renferme des symboles et parce que, contrairement aux autres extensions, il définit quelques macros permettant quelques présentations particulières. Le tableau 21 indique tous les symboles existant dans la fonte Dingbat accessibles grâce à la syntaxe :

`\ding{`*numéro*`}`

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

`{\Pifont{psy}` *texte* `}`
ou
`\Pisymbol{psy}{`*numéro*`}`

Le tableau 22 indique l'ensemble de ces symboles

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

Source

```
\begin{dinglist}{93}
  \item petite liste charmante ;
  \item peut-être un peu niaise.
\end{dinglist}

{\centering COUPON \`A RENVOYER\par}
\dingline{34}

\emph{La section du Canon}
({\Pifont{psy}katatomh kan\'onoV})
```

✻ petite liste charmante ;
✻ peut-être un peu niaise.

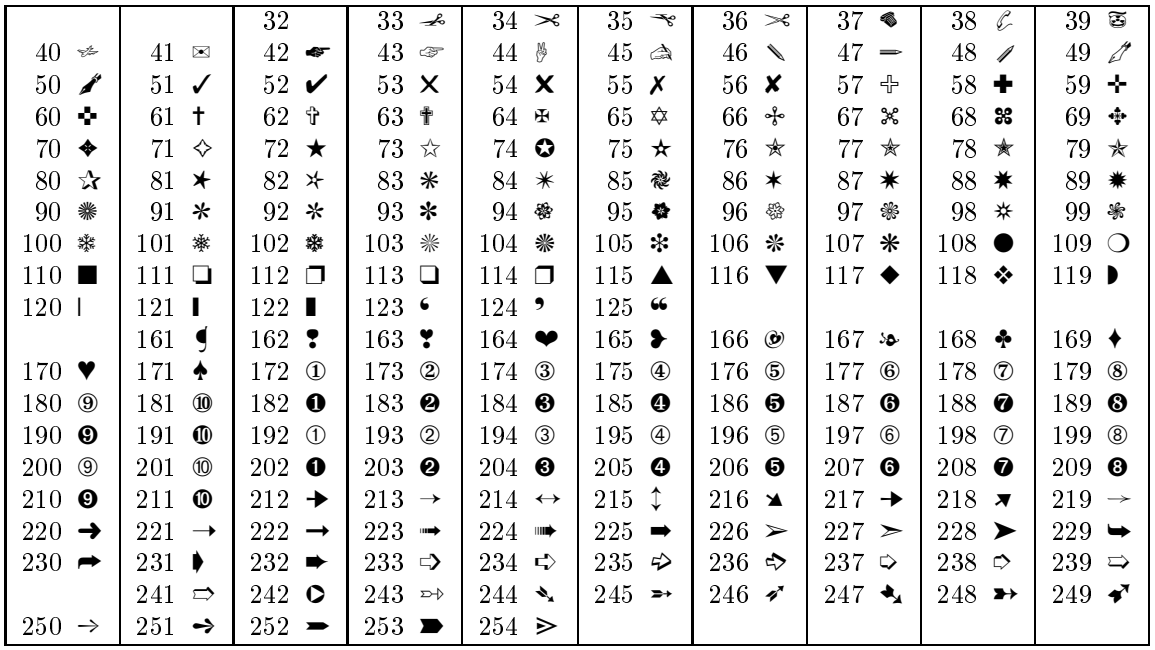
COUPON À RENVOYER

✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂

La section du Canon (κατατομη κανόνος)

On pourra enfin citer l'extension *marvosym* qui procure un grand nombre de symboles dont plusieurs versions du caractère euro. Malheureusement, ce pakage recèle une petite erreur qui empêche l'obtention correcte des flèches à double barre dans les modes mathématiques. Pour corriger cette erreur, on peut procéder de la sorte (je n'expliquerai pas ces incantations) :

```
\let\oldRightarrow\Rightarrow
\usepackage{marvosym}
\let\MarvoRightarrow\Rightarrow
\let\Rightarrow\OldRightarrow
```



TAB. 21 – Caractères de la fonte ZapfDingbat

				32		33	!	34	∀	35	#	36	∃	37	%	38	&	39	∍
40	(| 41 |)	42	*	43	+	44	,	45	−	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	;
60	<	61	=	62	>	63	?	64	≅	65	A	66	B	67	X	68	Δ	69	E
70	Φ	71	Γ	72	H	73	I	74	ϑ	75	K	76	Λ	77	M	78	N	79	O
80	Π	81	Θ	82	P	83	Σ	84	T	85	Y	86	ς	87	Ω	88	Ξ	89	Ψ
90	Z	91	[	92	∴	93	]	94	⊥	95	_	96	‾	97	α	98	β	99	χ
100	δ	101	ε	102	ϕ	103	γ	104	η	105	ι	106	φ	107	κ	108	λ	109	μ
110	ν	111	ο	112	π	113	θ	114	ρ	115	σ	116	τ	117	υ	118	ϖ	119	ω
120	ξ	121	ψ	122	ζ	123	{	124	|	125	}								
		161	ϒ	162	′	163	≤	164	⁄	165	∞	166	*f*	167	♣	168	♦	169	♥
170	♠	171	↔	172	←	173	↑	174	→	175	↓	176	°	177	±	178	″	179	≥
180	×	181	∝	182	∂	183	•	184	÷	185	≠	186	≡	187	≈	188	…	189	|
190	—	191	↵	192	ℵ	193	ℑ	194	ℜ	195	℘	196	⊗	197	⊕	198	∅	199	∩
200	∪	201	⊃	202	⊇	203	⊄	204	⊂	205	⊆	206	∈	207	∉	208	∠	209	∇
210	®	211	©	212	™	213	∏	214	√	215	·	216	¬	217	∧	218	∨	219	⇔
220	⇐	221	⇑	222	⇒	223	⇓	224	◊	225	⟨	226	®	227	©	228	™	229	∑
230	⎛	231	⎜	232	⎝	233	⎡	234	⎜	235	⎣	236	⎡	237	⎧	238	⎣	239	⎜
		241	⟩	242	⌡	243	⎞	244	⎜	245	⌡	246	⎤	247	⎜	248	⌡	249	⎤
250	⎜	251	⌡	252	⎤	253	⎫	254	⌡										

TAB. 22 – Caractères de la fonte Symbol

en lieu et place du bête :
`\usepackage{marvosym}`

7.3 amsmath

L'American Mathematical Society est la dépositaire du logiciel TEX. Il s'agit d'une importante association américaine de mathématiciens. Lorsque ceux-ci ont eu le programme de Knuth en leur possession, ils se sont

empressé de construire toute une série de macros permettant des constructions mathématiques complexes dont ils avaient besoin. Toutes ces macros ont été réunies dans plusieurs extensions dont la plus importante est *amsmath*. Il est hors de question de présenter toutes les possibilités de ce package en raison de la profusion de possibilités mais nous verrons les possibilités offertes les plus utiles.

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

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

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

$$\overrightarrow{A(x)B(x)} = \overrightarrow{g_x}$$

$$\underline{\lim_{n\to\infty}}\ u_n = \overline{\lim_{n\to\infty}}\ u_n = \boldsymbol{a^\lambda} \iff \lim_{n\to\infty} u_n = \boldsymbol{a^\lambda}$$

Source

```
\[\int_C\vec V\cdot d\vec M
  = \iint_D\left(\frac{\partial Q}{dx}
    -\frac{d\partial P}{dy}\right)\,dx\,dy
\] % intégrales multiples automatiques

\[\overrightarrow{A(x)B(x)}
  = \overrightarrow{g_x}\] % grandes flèches

\[\varliminf_{n\to\infty} u_n
  = \varlimsup_{n\to\infty} u_n
  = \boldsymbol{a^\lambda}
  \iff \lim_{n\to\infty} u_n
  = \boldsymbol{a^\lambda}\]
```

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

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

Source

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

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

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

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

Source

```
\[\begin{pmatrix}
  1 & \cos x \\
  -\cos x & 1 \\
\end{pmatrix}\]

\[\begin{vmatrix}
  \cos(x)-X & -\sin x \\
  \sin x & \cos(x)-X \\
\end{vmatrix}\]
```

7.4 Tableaux évolués

Vous vous êtes peut-être aperçu que la construction de tableaux est quelquefois un peu pénible et des problèmes surgiront presqu'obligatoirement si vous vous mettez à composer des textes de façon soutenue avec LaTeX. Ces problèmes ont été évoqués sur des forums de discussion et certaines personnes, très à l'aise dans la programmation de LaTeX ont créé des packages résolvant automatiquement les problèmes rencontrés. Là aussi, il existe beaucoup d'extensions dont le but est de résoudre certains problèmes liés aux tableaux ; nous ne ferons

pas vœux d'exhaustivité et nous ne présenterons que les plus classiques. Seul le package *array* sera présenté avec quelques détails, les autres packages ne seront qu'évoqués, éventuellement avec un exemple représentatif.

Le package *array* offre la possibilité de dire plus de choses au niveau du motif d'un tableau. Le tableau 23 donne le récapitulatif des commandes qu'on peut indiquer au niveau du motif d'un tableau.

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

TAB. 23 – Commandes des motifs de tableaux

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

Source

```
\begin{tabular}{|>{\scshape}llm{5.65cm}%
    r<{ \scshape j-c.}@{ -- }l<{ \scshape j-c.}|}
  \hline
  Abel & Niels Henrick &
  Il montre l'impossibilité de résoudre les équations de degré 5 par radicaux. &
  1802 ap. & 1829 ap. \\ \hline
  Alembert & Jean \textsc{le Rond d'} &
  Collaborateur à l'encyclopédie de Diderot. Il a effectué d'importants travaux sur les
  dérivées partielles, les nombres complexes et la mécanique &
  1717 ap. & 1783 ap. \\ \hline
  Apollonius & de Perge &
  Il a étudié les sections planes du cône (les coniques). En fait, on sait très peu de
  choses sur ce mathématicien. &
  262 av. & \(\sim\) 180 av. \\ \hline
\end{tabular}
```

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

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

Ainsi, le tableau 23 commençait de cette façon :

```
\noindent
\begin{tabularx}{\linewidth}{|c|X|}
\hline
\multicolumn{2}{|c|}{\textbf{Anciennes commandes}} \\\hline
\texttt{l} & Colonne alignée à gauche \\
\texttt{c} & Colonne centrée \\
\texttt{r} & Colonne alignée à droite \\
...
```

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

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

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

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

`\hhline{`*motif*`}`

On commencera avec un petit exemple de ses possibilités et le tableau 24 récapitulera les commandes de motif de cette macro.

a	b	A	+
1	2	α	−
3	4	β	×
5	6	Γ	÷

Source

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

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

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

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

où *nbligne* désigne le nombre de lignes qui devront être fusionnées, *largeur* la largeur de la grande cellule créée et *contenu* le contenu de la cellule qui sera considérée comme un paragraphe (donc pourra être composée sur plusieurs ligne de texte. Voici un exemple de tableau composé grâce à cette macro :

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

TAB. 24 – Syntaxe de la macro `\hhline`

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

Source

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

7.5 multicol

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

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

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

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

8 Petit aperçu des possibilités

Ce chapitre n'est là que pour montrer quelques possibilités offertes par LaTeX : il ne s'agit pas d'étudier ces présentations et il n'y aura strictement aucune explication.

8.1 Ce manuel

Le manuel lui-même comprend des constructions qui n'ont pas été évoquées jusqu'ici. Par exemple, les tableaux ayant une légende et un numéro sont ce que LaTeX appelle des « flottants ». Un flottant est une structure qui sera placée dans le document de façon automatique par LaTeX en fonction de la place disponible, donc pas nécessairement au niveau où cette structure a été tapée dans le source. LaTeX prévoit des flottants pour les tableaux et pour les figures. On notera la présence d'une *Liste des tableaux* à la fin du manuel, liste qui a été créée automatiquement.

La première page présentait des inclusions de dessins dans des paragraphes avec un découpage du texte autour du dessin. Là aussi, tout est fait de façon automatique, le source indique seulement qu'il faut placer un certain dessin et le texte qui l'entourera mais sans se soucier de savoir où se feront les coupures de lignes et quand les lignes auront de nouveau toute la largeur de la gage à leur disposition.

Dans le chapitre présentant l'installation de la distribution TeXlive, il y avait de temps en temps des symboles « ☐ » et « ⊞ », j'avais également prévu le symbole « ☒ ». Ces symboles ont été construits en tant que macros et leur inclusion à n'importe quel endroit était alors très simple.

Les exemples présentant un source et son résultat (soit côte à côte, soit l'un en-dessous de l'autre) ont été faits de façon automatique grâce à un environnement d'un package particulier. Le texte n'a été tapé qu'une seule fois de telle sorte que le résultat montré est forcément celui obtenu par le source en vis-à-vis.

Les tableaux 21 et 22 page 47 auraient été très pénibles à taper si l'auteur n'avait pas créé quelques macros pour se débarrasser d'une bonne partie du travail répétitif. Cela devrait d'ailleurs devenir un réflexe lorsqu'on travaille sous LaTeX : tout travail répétitif devrait faire l'objet d'une ou de plusieurs macros (ou environnements).

8.2 Graphiques

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

- construire une image grâce à un programme externe et l'inclure sous forme d'un fichier PostScript (PDF TeX et PDFLaTeX offrent plus de possibilités sur le type des fichiers graphiques pouvant être inclus) ;
- utiliser des extensions permettant de mettre des commandes graphiques dans le source du document.

Ces deux méthodes présentent chacune des avantages et des inconvénients. Voici quelques exemples des possibilités. Ils ont tous été composés avec la seconde méthode (et même à partir du seul groupe d'extensions lié à *PsTricks*) pour que tout le document tienne dans un minimum de fichiers. Le source de ces différents exemples n'est pas indiqué mais le stagiaire curieux pourra toujours se procurer le source de ce manuel pour connaître le fin mot de l'histoire.

La figure 1 reproduit fidèlement ma jolie calculatrice.

La figure 2 illustre un classique de géométrie.

La figure 3 est tirée du LaTeX Graphics Companion et est due à Denis Girou. Il s'agit bien sûr d'un calligramme de Guillaume Apollinaire, Denis Girou n'a fait que le coder de façon informatique[8] !

[8] Qu'il soit remercié sur sept générations pour avoir autorisé cette reproduction.

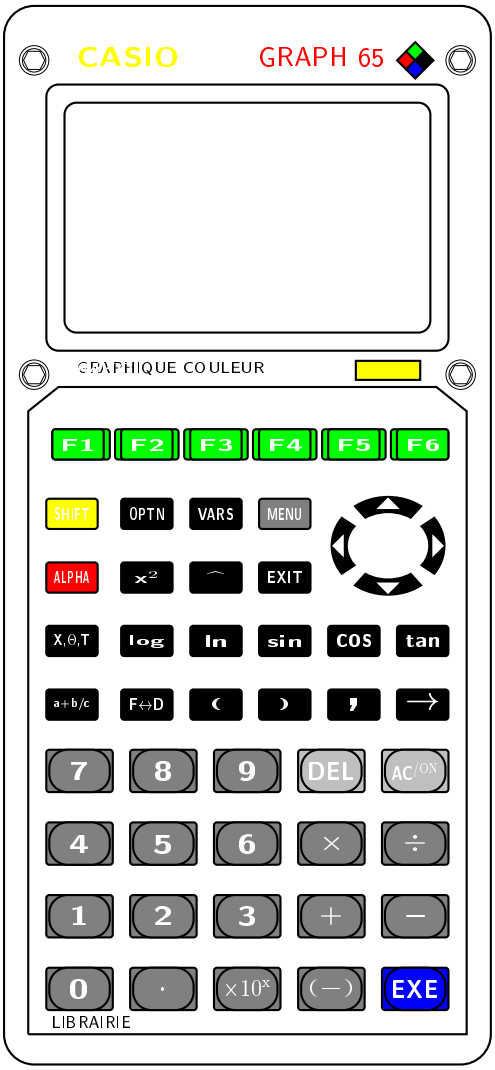
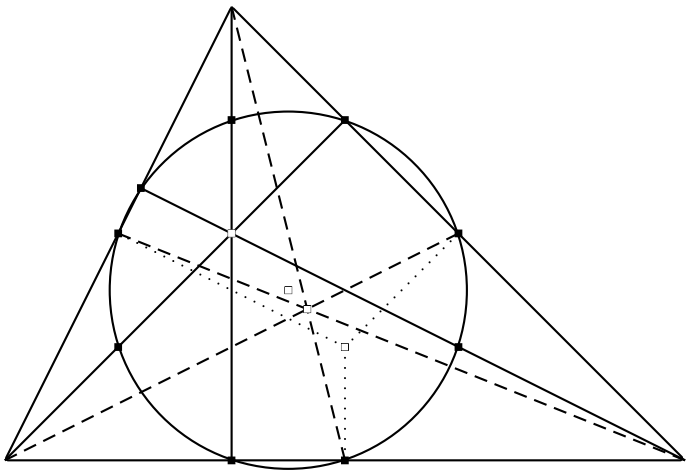
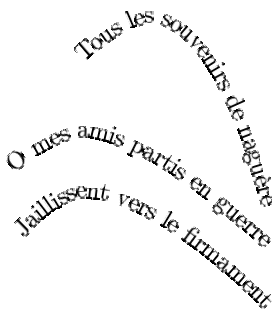


FIG. 1 – Dessin utilisant l'extension *Pstricks*



Cercle des 9 points

FIG. 2 – Figure géométrique utilisant l'extension *Pstricks*


Tous les souvenirs de naguère
O mes amis partis en guerre
Jaillissent vers le firmament

t

A Bibliographie commentée

La liste des références présentée ici n'a pas pour vocation d'être complète. Il s'agit d'ouvrages que je possède, ou que j'ai eu un certain temps en ma possession, et de sites internet sur lesquels j'ai l'habitude d'aller, ce qui me permet de donner mon avis et non un avis pris dans un livre quelconque. Bien entendu, tous les commentaires sont hautement subjectifs ! Les prix indiqués sont ceux pratiqués lorsque j'ai acheté l'ouvrage et sont donc susceptibles d'une certaine variation.

A.1 Ouvrages sur la typographie

[LEX] *Lexique des règles typographiques en usage à l'Imprimerie Nationale*, Imprimerie nationale, 1990. (89 F)
Ouvrage de référence sur les questions de typographie. Il est à noter qu'il existe d'autre corpus de règles que celles-ci mais, dans l'ensemble, ces règles sont plus ou moins admises par tout le monde (en France).

[MTF] *Manuel de typographie française*, Yves Perrousseaux, Atelier Perrousseaux, 1996. (120 F)
Cet ouvrage présente les règles de typographie avec des références historiques et indique comment les mettre en œuvre sur un système informatique.

[MPI] *Mise en page & imprimerie*, Yves Perrousseaux, Atelier Perrousseaux, 1999. (200 F)
Il s'agit du complément logique de l'ouvrage précédent.

[PLT] *Petites leçons de typographie*, Jacques André, document libre, 1990. (0 F)
Petit document (25 pages de corps) mais bien présenté et clair.

A.2 Ouvrages pour débuter

[LGP] *LaTeX, guide pratique*, Christian Rolland, Campus Press, 199?. (??? F)
C'est très souvent le premier ouvrage qu'ont eu les LaTeXiens confirmés. La présentation se veut résolument pratique et la nouvelle édition a été fortement augmentée d'où une certaine exhaustivité au niveau des packages importants.

[JML] *Joli manuel pour LaTeX 2ε*, Benjamin Bayart, document libre, 1995. (0 F)
Il s'agit d'un document destiné initialement aux élèves de l'ESIEE (ingénieurs en électronique) mais qui a été mis à disposition sur internet. Dès que les rudiments de LaTeX sont connus, c'est-à-dire, par exemple, après un stage comme celui-ci, ce guide est très utile et très complet. Je le recommande chaudement.

[MPM] *A User Manual for METAPOST*, John D Hobby, document libre, 199?. (0 F)
Le manuel de référence de METAPOST écrit par son auteur. METAPOST est un logiciel permettant de construire des figures au format PostScript en utilisant une bonne partie de la syntaxe du langage METAFONT. Ce livre est nettement plus accessible que le METAFONTbook mais moins complet. Il existe une excellente traduction faite par Jean-Côme Charpentier et Pierre Fournier (un peu de publicité) disponible sur le site Syracuse (Cf. infra).

[LAG] *LaTeX. Apprentissage, guide et référence*, Bernard Desgraupes, Vuibert, 2000. (∼ 250 F)
Comme son nom l'indique, il s'agit vraiment d'un livre permettant d'apprendre, d'être guidé et d'avoir uen référence (très complète) sur LaTeX. À mon sens, le meilleur ouvrage car il a réussit le pari d'être aussi utile au débutant qu'au LaTeXien confirmé.

A.3 Ouvrages plus difficiles

[TEX] *The TeXbook*, Donald Knuth, Addison-Wesley, 1996. (41,95 $ US)
LA référence en ce qui concerne le langage TeX. Cet ouvrage se veut lisible par un débutant mais, soyons sérieux, la majeure partie de son contenu est d'une très haute technicité. À posséder absolument si on attrape le virus TeXien.

[MTL] *La maîtrise de TeX et LaTeX*, Thomas Lachand-Robert, Masson, 1995. (380 F)
Contrairement à ce que son titre pourrait laisser croire, il n'est quasiment pas question de LaTeX. Il s'agit de

l'ouvrage français le plus complet sur le langage TEX, un peu l'équivalent du TEXbook. Indispensable si on ne lit pas l'anglais et qu'on veuille découvrir les entrailles de TEX.

[MFB] *The METAFONTbook*, Donald KNUTH, Addison-Wesley, 19??. (??? $ US)
LA référence en ce qui concerne le langage METAFONT. Les mêmes remarques que celles faites pour le TEXbook s'appliquent à ce livre. Il semble malheureusement épuisé.

[DPS] *A document Preparation System*, Leslie LAMPORT, Addison-Wesley, 1999. (36,95 $ US)
Un ouvrage de référence sur LATEX écrit par son auteur. L'ouvrage ne traite que de LATEX et ne présente pas les extensions. Personnellement, je lui préfère la référence suivante mais mon opinion n'est pas forcément toujours partagée !

[COM] *The LATEX Companion*, Michel GOOSSENS, Frank MITTELBACH, Alexander SAMARIN, Addison-Wesley, 1994. (36,95 $ US)
C'est mon livre de chevet ! Il est complet au-delà de toute espérance : un travail titanesque. Une traduction française existe (publiée chez Campus Press) et qui porte le même nom (249 F). La traduction reprend certains points rendus obsolètes (elle date de 2000) et est moins chère que l'ouvrage originale mais la beauté (physique) du livre est moins bonne : les ouvrages de chez Addison-Wesley sont souvent de qualité.

[LGC] *The LATEX Graphics Companion*, Michel GOOSSENS, Sebastien RAHTZ, Frank MITTELBACH, Addison-Wesley, 1997. (36,95 $ US)
C'est l'équivalent du précédent pour tout ce qui touche au graphisme dans le monde LATEX. Il est aussi beau et aussi complet que *The LATEX Companion*.

[FAQ] *FAQ LATEX*, Marie-Paule KLUTH, ???, ???. (??? F)
Pourquoi tous ces points d'interrogations ? Parce que je n'ai pas cet ouvrage que pourtant tout le monde devrait posséder. Pourquoi je n'ai pas cet ouvrage alors que je n'en pense que du bien ? Réponse à la section suivante. À propos, FAQ signifie « Frequently Asked Questions » très joliment traduit en « Foire Aux Questions ».

A.4 Références sur internet

`http://www.ctan.org/ctan`
CTAN est l'acronyme de « Comprehensive TeX Archive Network ». En clair, tous les fichiers informatiques qui existent (officiellement) sur TeX se trouvent sur ce site. En conséquence, lorsqu'on cherche la perle rare, c'est un des premiers endroits où aller. Le site indiqué est le site principal basé au États-Unis, il existe des sites mirroirs un peu partout dans le monde : la liste de ces sites mirroirs se trouve sur tous les sites CTAN.

`http://tex.loria.fr/index.html`
Il s'agit du Loria : un site français de référence pour tout ce qui concerne la documentation de TEX et des programmes satellites. J'ai réellement pillé ce site !

`http://gutenberg.eu.org/pub/GUTenberg`
GUTenberg est le pendant français du TUG (TEX User's Group). Il s'agit d'une association loi 1901 qui fédère les passionnés de TEX et de LATEX. Cette association permet à ses adhérents de recevoir un bulletin d'information, de s'abonner pour un prix modique aux cahiers de GUTenberg et de participer à des réunions aux thèmes divers et variés touchant au monde TEX.

`http://melusine.eu.org/syracuse`
Site Poitevin regroupant les utilisateurs des logiciels libres et des utilisateurs des programmes du monde TEX. Site très sympathique géré par des personnes non moins sympathiques quoique peu nombreuses ce qui ajoute à leur mérite. C'est par exemple sur ce site qu'a été placé la traduction du manuel de METAPOST réalisée avec Pierre FOURNIER.

`fr.comp.text.tex`
Attention, il ne s'agit pas d'un site mais d'un forum de discussion (newsgroup comme disent ceux qui ont du mal avec la langue française). On y trouve des personnes de tout niveau, on peut se contenter de poser des questions, de lire les questions et réponses des autres ou bien de répondre soi-même à certaines questions. Il s'agit d'un forum très vivant : soyez un minimum poli, posez votre question clairement et vous obtiendrez presqu'à coup sûr une réponse. Enfin, comme sur tout forum de discussion ou presque, une FAQ est postée tous les mois. Cette FAQ a été établie à partir de celle de Marie-Paule KLUTH qui s'en occupait il y a quelques années. Elle s'enrichit de mois en mois ce qui explique ma réticence à acheter celle du commerce.

Liste des tableaux

1 Obtention des caractères réservés . . . 11
2 Accent en mode texte . . . 12
3 Caractères européens spéciaux . . . 12
4 Symboles spéciaux . . . 12
5 Macros de taille de caractère . . . 13
6 Macros de type de caractère . . . 14
7 Unités de mesures . . . 17
8 Délimiteurs des modes mathématiques . . . 21
9 Opérateurs binaires . . . 24
10 Relations . . . 25
11 Grands opérateurs . . . 25
12 Délimiteurs . . . 25
13 Flèches . . . 26
14 Symboles mathématiques divers . . . 26
15 Noms de fonction . . . 26
16 Accents mathématiques . . . 27
17 Synonymes de macros mathématiques . . . 27
18 Fontes mathématiques . . . 34
19 Macros d'espacements . . . 37
20 Exemples classiques de correction d'espacement . . . 37
21 Caractères de la fonte ZapfDingbat . . . 47
22 Caractères de la fonte Symbol . . . 47
23 Commandes des motifs de tableaux . . . 49
24 Syntaxe de la macro \hhline . . . 51