Débuter en

Yves Soulet – Hiver 2016

Partie I – Texte – Mathématiques – Graphisme et hypertexte

Préface

Des polys et des docs, il y en a à la pelle, mais la passion d'enseigner fait que l'on a toujours envie d'en peaufiner un de plus.

La première rédaction de ce document a été écrite à l'occasion d'une formation faite au CNES (Centre Natonal d'Etudes Spaciales), centre de Toulouse, avec la collaboration de Marie-Hélène Gellis, en décembre 2000 ; pour différentes raisons, l'édition n'a pu en être préparée à temps. En 2002, la rédaction a été reprise, corrigée, complétée et adaptée à l'enseignement « Techniques de communications » de l'IUP (Institut Universitaire Professionnel) Statistique et Informatique Décisionnelle, Université Paul-Sabatier (Toulouse).

Depuis, elle a été revue et complétée régulièrement (en 2005, 2009 et 2013) en fonction des nouvelles fonctionnalités disponibles ; en particulier, un chapitre concernant la production de fichiers hypertexte au format PDF a été rajouté ainsi qu'un index relativement détaillé facilitant le travail des débutants.

La cinquième édition a été motivée par le désir d'aider Hugues Orounla à développer l'utilisation de LaTeX par les universitaires du Bénin. C'est un modeste document qui ne peut remplacer les ouvrages cités, mais il peut permettre d'aquérir un bon niveau suffisant pour produire des documents de bonne qualité, tant sur le plan présentation que sur le plan pédagogique.

Enfin, je remercie Hugues qui m'a signalé quelques fautes et quelques passages à revoir : pour certains cas, il y aura des compléments dans la partie II : II.A Créer des transparents, II.B Gestion et préparation des fontes, II.C Bibliographie à partir des bases de données.

Yves Soulet
Professeur honoraire
Université Paul Sabatier, Toulouse
Hiver 2016

Table des matières

Partie I.A

Typographier du texte avec LaTeX

Chapitre 1

Présentation de l'ensemble TEX-LATEX

Donald Knuth, Université de Stanford, a développé (avec ses collaborateurs) la première version de TEX en 1978. Précisons tout de suite qu'il faut prononcer TEX comme la première syllabe du mot technique, c'est le souhait de l'auteur. La petite histoire raconte que c'est au retour d'une visite chez un éditeur que l'auteur décida (au vu du coût de l'édition des ouvrages scientifiques) de lancer cette opération.

L'objectif en était de pouvoir éditer des documents scientifiques et techniques de grande qualité éditoriale sans l'utilisation prolongée des photocomposeuses, machines très coûteuses, surtout à cette époque-là.

Par la suite TEX a évolué et, en 1982, Leslie Lamport à écrit LATEX. On va donner quelques précisions préliminaires avant de continuer cette petite introduction historique :

TEX : désigne à la fois le langage de programmation et le programme (l'exécutable) servant à composer les documents à partir de fichiers-sources saisis avec un éditeur quelconque (nous reviendrons là-dessus en détail par la suite) ;

LATEX : est un ensemble de sous-programmes et de macros-commandes écrits avec les primitives du langage TEX pour pouvoir réaliser la composition de documents scientifiques et techniques sans avoir la nécessité de programmer ;

TEX : n'est pas un programme de composition (ou de formatage) de documents du type WYSIWYG (*What You See Is What You Get*) :

- on écrit un fichier-souce TEX contenant le texte, les formules, les commandes d'appel des insertions graphiques et les commandes de formatage,
- on compose (on compile ou encore on traite) ce fichier-source avec l'exécutable `latex` ou avec l'exécutable `pdflatex`,
- enfin, avec d'autres programmes (pilotes d'écran, pilotes d'imprimantes, etc.), on visualise et imprime le résultat de la composition.

A ce niveau là, on est tenté de dire : pourquoi TEX n'est-il pas un système WYSIWYG ? La réponse viendra au cours de cette initiation ; on peut, pour le moment, donner une réponse très partielle mais qui éclairera tout de même : l'uniformité

des blancs (des espaces) dans un paragraphe est une caractéristique de qualité de la composition ; en conséquence, ces blancs, élastiques pour assurer la justification, ne sont déterminés que lorsque TEX a « absorbé » tous les mots du paragraphe : le contenu des dernières lignes du paragraphe influence donc les largeurs des blancs (et les césures des mots) du début du paragraphe !

TEX et LATEX se sont imposés dans le monde universitaire et de la recherche. Dès le début, *leurs auteurs ont mis à la disposition de la communauté scientifique internationale leur travail y compris les sources.* Par la suite, des centaines de chercheurs ont ajouté de multiples extensions : il s'est alors posé le problème de portabilité et de compatibilité de ces multiples extensions de LATEX.

En 1994, l'arrivée de LATEX2e met de l'ordre dans cette profusion d'extensions :

- un seul format, LATEX2e, que l'on peut précompiler ;
- des extensions utilisant ce format, respectant des règles précises et donc compatibles entre elles ;
- la rationalisation de l'utilisation des fontes (NFSS : *New Font Selection Scheme*).

Il faut préciser que le dernier point ci-dessus avait été rendu nécessaire par la mise à disposition, pour pratiquement toutes les langues, de toutes les fontes existantes sous forme numérisée (mise à disposition mais généralement pas dans le domaine public !). Il y a de nombreux ouvrages concernant LATEX disponibles, nous n'en citerons que quelques uns [1, 2, 3, 4].

On insiste encore sur le fait que l'ensemble TEX-LATEX (avec tous les utilitaires nécessaires) est du domaine public. Cet ensemble est maintenant *le fruit du travail d'un important nombre d'informaticiens de haut niveau, travail non motivé par le profit* ; cela explique l'excellente optimisation de tous ces programmes.

On terminera en disant qu'il y a toujours des irréductibles dans le domaine professionnel qui préfèrent écrire des programmes de formatage directement avec les primitives de TEX au lieu d'utiliser LATEX et ses extensions appropriées (les packages) qui ne permettent pas toujours de réaliser exactement ce qu'on veut. Il y a aussi des ouvrages correspondants à ce choix [5, 6]. Nous n'en reparlerons pas.

Exercices

Préparation de la configuration de travail

Pour travailler valablement il faut organiser systématiquement l'ensemble des fichiers ; ci-après, voilà un exemple d'organisation.

On ouvre d'abord une fenêtre de commande. Pour cela, dans le menu Démarrer de Windows, choisir Exécuter et saisir la commande `cmd`, puis `OK` ; la fenêtre apparaît alors. Remonter à la racine `c:` et créer un répertoire pour l'ouvrage que l'on va éditer : on choisi `poly`. Ensuite, il est assez logique de placer les fichiers de saisie dans un sous-répertoire `c:\poly\t` et les fichiers graphiques à intégrer dans un sous-répertoire `c:\poly\d` (les entrées des fichiers de saisie se font avec les commandes

`\input{t/nom-fichier}` (on verra plus loin la commande d'insertion des figures). On peut aussi créer un sous-répertoire `c:\poly\tini` pour placer des éventuels fichiers saisis préalablement sans la syntaxe LATEX. Le répertoire `c:\poly`, qui sera le répertoire courant, sera réservé à :

- tous les fichiers spécifiques de formatage `poly.sty`, `poly.ist`, `macsup.tex`, etc.,
- le « fichier maître » `livre.tex` dont le rôle sera décrit plus loin,
- les fichiers créés au cours de la composition `livre.log`, `livre.aux`, `livre.dvi`, `livre.ps`, `livre.pdf`, `livre.toc`, `livre.idx`, `livre.ind`, etc.

Il est ensuite commode d'y placer quelques fichiers de commandes dont les exemples ci-après sont donnés avec la syntaxe des fichiers de commandes sous DOS-Windows[1] :

`ed.bat = nom-de-l'éditeur t\%1.tex :`
pour éditer les fichiers-sources ;

`tt.bat = latex livre.tex = tex &latex livre.tex :`
pour lancer la composition qui crée le fichier `livre.dvi` à partir du fichier « maître » `livre.tex` (qui contient l'appel des fichiers formatage et des fichiers de saisie) ;

`ps.bat = dvips -P pdf -o livre.ps livre.dvi :`
pour lancer la création du fichier PostScript `livre.ps` à partir du fichier `livre.dvi` ; il faut ensuite transformer ce ficher `livre.ps` en `livre.pdf` l'aide de EPSTOPDF (public) ou encore DISTILLER (non public) ;

`pdf.bat = pdflatex livre.tex = pdftex &latex livre.tex :`
pour générer directement un fichier `livre.pdf` sans passer par les étapes fichier `livre.dvi` et fichier `livre.ps` (solution conseillée).

Le fichier `livre.ps` se visualise et s'imprime avec GSVIEW (versions récentes, gratuit) ; le fichier `livre.pdf` se visualise et s'imprime aussi avec GSVIEW et bien sûr avec ACROREAD (gratuit) et ACROBAT (non public).

Les opérations correspondant aux commandes ci-dessus (à l'exception de la première) peuvent être lancées par l'intermédiaire d'une interface graphique incluse dans certains éditeurs ; citons par exemple WinShell et WinEdit (dépassés), TEXnicCenter (sous Windows) et Emacs (sous UNIX) ; TEXworks et TEXstudio (sous Windows) ainsi que TEXshop (sous Mac-OS) sont des ensembles qui proposent le travail avec deux fenêtres : à gauche la saisie et à droite la compilation ; le gros avantage est que, lorsqu'il y a une erreur de compilation, la position de l'erreur s'affiche sur la saisie.

On comprend beaucoup moins bien ce que l'on fait avec ces dernières méthodes de travail et elles deviennent désavantageuse lorsque l'on travaille simultanément sur plusieurs documents nécessitant des lignes de commandes différentes.

[1] Sous UNIX, ces fichiers sont appelés « scripts » ; ils n'ont pas nécessairement d'extension spécifique, par contre ils doivent être déclarés « exécutables » avec CHMOD (attribut `x`).

Premier pas en LaTeX

Faire un fichier-source `1.tex` contenant les phrases :

```
Je débute en \TeX{} ; j'espère progresser rapidement.
Essai (accents et ligatures) : é è ê ë \'e \`e \^e \"e \oe vre.
```

Placer la commande d'entrée du fichier `1.tex` dans le fichier maître `livre.tex` élémentaire ci-dessous. Compiler avec `pdflatex` (commande `pdf`) et visualiser GSVIEW.

```
% Fichier maitre elementaire pour divers tests
\documentclass[12pt]{book}
% La configuration par défaut est suffisante
\begin{document}
\input t/1 % Par defaut l'extension .tex est ajoutee
\end{document}
```

Chapitre 2

Généralités sur l'édition avec TEX-LATEX

On va commencer à entrer dans le cœur du système pour en découvrir les immenses possibilités. Il est hors de question de mémoriser tout ce qui peut être utilisé pour l'édition d'un document ; il suffit de comprendre le fonctionnement et le rôle des différentes sortes de fichiers pour pouvoir travailler valablement avec le polycop sous la main. Petit à petit on se libérera de l'utilisation continuelle des documents.

2.1 Étapes de l'édition

2.1.1 Saisie des fichiers-sources

Avec un éditeur quelconque (avec la commande `ed` si on a adopté l'environnement de travail proposé en page 5), on saisit le texte, les formules mathématiques s'il y en a et les commandes de formatage. Traditionnellement, pour un document relativement long, l'expérience montre qu'il est avantageux de créer un fichier principal appelé « fichier maître » contenant un préambule de commandes de formatage suivi des commandes d'entrée des fichiers de saisie. L'utilisation du caractère de commentaire % permet de sélectionner très facilement les fichiers que l'on souhaite faire entrer.

2.1.2 Composition

On dit composition ou compilation ou encore traitement : elle se fait à l'aide de la commande `tt`, voir page 5. La première partie du fichier maître `livre.tex` fait l'objet du préambule ; elle contient les commandes de formatage générales et spécifiques au document créé ; la deuxième partie comprend les commandes d'entrée des fichiers saisis et des fichiers créés par la composition elle-même (table des matières, index, liste des figures, etc.).

A partir du fichier `livre.tex`, la composition crée le fichier `livre.dvi` (dvi pour *device independent*). Ce fichier `livre.dvi` contient des informations du type :

> « A telle position (coordonnées) de telle page (numéro) il y a telle lettre (code du caractère) de telle fonte (nom codé permettant la portabilité) ou telle figure (nom du fichier de la figure). »

TEX a donc lu les fichiers métriques des fontes (`.tfm`) utilisées pour y prendre les dimensions des caractères et le début des fichiers de figure pour y prendre les dimensions des figures afin de composer les pages, c'est-àdire positionner tous les caractères et toutes les figures et « garder » les places correspondantes.

2.1.3 Création du fichier PostScript

Le langage PostScript est un puissant langage graphique d'utilisation universelle : imagerie, graphiques, fontes, etc. Le programme DVIPS, appelé par la commande `ps` de la page 5, crée, à partir du fichier `livre.dvi`, un fichier PostScript `livre.ps` contenant le résultat complet de la composition où auront été intégrés les dessins des caractères contenus dans les fichiers PostScript de description de fontes, extension `.pfa` ou `.pfb`, ainsi que les objets graphiques contenus dans les fichiers PostScript encapsulés extension `.eps`.

2.1.4 Visualisation

Les pilotes d'écran GSVIEW (sous Windows) ou GHOSTSCRIPT (sous UNIX) montrent à l'écran le résultat de la composition. Ces pilotes sont en quelque sorte des interpréteurs PostScript avec une sortie dirigée vers l'écran.

2.1.5 Impression

L'impression peut se faire directement en envoyant le fichier `livre.ps` vers l'imprimante où l'interpréteur PostScript commande l'impression à partir du contenu du fichier. Sous DOS-Windows par exemple, dans la fenètre de commande, il suffit de taper `copy livre.ps prn:` (cette simple commande permet de tester si l'imprimante est munie réellement d'un interpréteur PostScript).

Dans la majorité des installations, les pilotes d'écran ci-dessus mentionnés permettent de commander l'envoi du fichier `livre.ps` à l'imprimante ; ils permettent même de n'envoyer que la partie du fichier correspondant aux pages sélectionnées. Sous Windows, GSVIEW peut aussi utiliser son propre interpréteur PostScript et diriger la sortie vers une imprimante non munie d'un interpréteur PostScript (solution de pis-aller car, en général, il y a perte de qualité).

2.1.6 Visualisation et impression sans l'étape PostScript

Cette sous-section est là pour son intérêt pédagogique (et éventuellement pour des cas particulier utilisant des fontes exceptionnelles).

Il existe des pilotes d'écran sous DOS-Windows, WINDVI (un peu ancien) et YAP (récent) par exemple, qui permettent la visualisation mais aussi l'impression à partir du fichier `livre.dvi`. Il y a aussi des utilitaires correspondants sous UNIX qui peuvent varier avec le type d'interface graphique utilisée.

Il existe aussi des pilotes d'imprimante, par exemple sous DOS-Windows, DVIHP pour imprimer directement à partir du fichier `livre.dvi` et sur des imprimantes reconnaissant le langage PCL.

Les pilotes cités ci-dessus travaillent à partir du fichier `livre.dvi` ; ils doivent donc utiliser des fichiers de fontes contenant le dessin des caractères autres que les fichiers PostScript `.pfa` ou `.pfb`. Ils utlisent les fichiers `.pk` contenant une description des caractères du type *bitmap* correspondant à la résolution de l'imprimante. Il est donc difficile d'avoir une grande qualité car, même avec une haute résolution (fichiers `.pk` énormes) on n'arrive à la qualité rendue par le PostScript.

C'est là que l'on réalise l'intérêt du langage PostScript : le fichier `livre.ps` est indépendant de la résolution souhaitée ; il est utilisable avec une petite imprimante personnelle (possédant un interpréteur PostScript bien entendu) mais aussi avec une imprimante professionnelle de haute résolution (4800 points par pouce).

On dit que les fontes dont le dessin des caractères est donné par les fichiers PostScript `.pfa` ou `.pfb` sont des fontes au format *type 1* ; l'appellation *fonte vectorielle* désigne les fontes qui ne sont pas décrites par des fichiers du type *bitmap*, les fontes de *type 1* en particulier. Les fontes *true type* sont aussi des fontes vectorielles mais ne sont pas du *type 1* ; la machinerie TEX comprend des utilitaires pour passer du format *true type* au format *type 1*.

Il peut arriver que l'on doive utiliser une fonte créée avec METAFONT et qui n'est donc disponible qu'en format *bitmap*[1]. La compilation peut se faire normalement avec le fichier métrique `.tfm` ; ensuite, on peut créer un fichier PostScript : DVIPS, ne trouvant pas de fichier `.pfa` ou `.pfb`, cherche le fichier `.mf` ainsi que le programme METAFONT pour lancer la création du fichier `.pk` à la résolution que l'on veut, puis il transforme le dessin des caractère utilisés du format *bitmap* au format PostScript, de façon complètement transparente ; le résultat final est évidemment moins bon. Bien entendu, les fichiers `.pk` sont stockés pour leur réutilisation.

2.1.7 Génération directe d'un fichier au format PDF

A partir du fichier PostScript `livre.ps` il est possible de générer un fichier `livre.pdf` au format PDF (*Portable Document Format*) avec des utilitaires tels que EPSTOPDF (public) ou DISTILLER (non public). Cette méthode est intéressante si l'on doit intégrer beaucoup de fichiers graphiques disponibles au format PostScript car, l'imprimerie professionnelle n'utilise aujourd'hui que le format PDF.

On dispose aussi de l'exécutable `pdftex` en remplacement de l'exécutable `tex` et de l'exécutable `pdflatex` en remplacement de l'exécutable `latex` qui donne directement un fichier `livre.pdf` au format PDF à partir du fichier `livre.tex`. C'est une simplification appréciable, largement adoptée, d'autant plus que les fontes de *type 1* sont utilisées par défaut ; elle est peut-être moins « maîtrisable » que la méthode indirecte décrite ci-dessus. Pour cette compilation, on utilise la commande `pdf` de la page 5. Quelle que soit la méthode utilisée, le fichier `livre.pdf` est visionné et imprimé à l'aide de GSVIEW (versions récentes, gratuit), ACROREAD (gratuit) ou ACROBAT (non gratuit).

La production de documents hypertexte sera introduite au chapitre 13.

[1] Fonte d'usage très restreint qui ne sera jamais publiée au format PostScript

2.2 Récapitulation des types de fichiers

Il est utile de rassembler les noms des différents types de fichiers en précisant leur rôle. Comme l'ensemble TEX-LATEX est du domaine public, les noms ou les extensions des fichiers rappellent leur rôle afin de faciliter la compréhension. Voilà les extensions des des types de fichiers couramment utilisés (il y en a d'autres utilisés pour des compositions spécifiques, voir [4] page 8) ;

Fichiers utilisés pour la composition :

`.tex` : fichiers-source (ou de saisie), fichier maître et fichiers de formatage ;

`.fmt` : fichier de format précompilé chargé à la ligne de commande, voir la commande `tt` page 5, leur nom est précédé du caractère `&` ;

`.tfm` : fichiers métriques des fontes contenant les dimensions des caractères et des dimensions caractéristiques de la fonte : em, ex, espace inter-mot minimun, etc. ainsi que les positionnements des exposants supérieurs et inférieurs s'il s'agit d'une fonte mathématique, etc. ;

`.dvi` : fichier contenant le résultat de la composition comme expliqué à la sous-section 2.1.2, page 7 ;

`.pk` : fichiers de fontes contenant une description de type *bitmap* des caractères ; ils correspondent à une résolution donnée (utilisés si l'on ne passe pas par l'étape PostScript), voir sous-section 2.1.6, page 8 ;

`.pfa .pfb` :
fichiers de fontes contenant une description vectorielle des caractères en langage PostScript, voir sous-section 2.1.3, page 8, ainsi que page 9 ;

`.ps` : fichier PostScript contenant l'ensemble de la composition créé par DVIPS, voir sous-section 2.1.3, page 8 ; c'est aussi l'extension de certains fichiers PostScript de configuration ;

`.eps` : fichiers graphiques (figures, photos, etc.) au format PostScript contenant, dans leur préambule, les dimensions de la figure (ces dimensions sont utilisées par TEX au moment de la composition pour réserver la place nécessaire à l'insersion de cette figure, voir la section 11.1) ;

`.pdf` : fichier au format PDF contenant l'ensemble de la composition obtenue à partir du fichier `.ps` ou directement par la commande `pdf` de la page 5, voir sous-section 2.1.7 ; c'est aussi l'extension de tous le fichiers graphiques au format PDF qui sont insérés dans le document composé directement au format PDF par la commande `pdf`.

Fichiers programmes spécifiques à LATEX :

`.cls` : fichiers de classes ; exemple `book.cls` ;

`.sty` : fichiers de style de la distribution ou personnels ;

`.clo` : fichiers d'options ; exemple `bk10.clo` ;

`.fd` : fichiers de définition des fontes courantes ;

Fichiers annexes créés à la composition : ils ont le nom du fichier maître :

`.log` : fichier contenant des informations sur le déroulement de la composition (utile parfois pour retrouver des erreurs en particulier celles qui n'arrêtent pas la composition) ; d'une façon générale, tout exécutable produit un fichier de trace concernant son exécution avec éventuellement les erreurs rencontrées ;

`.aux` : fichiers auxiliaires contenant des informations stockées entre les passes pour l'index, les références croisées, la bibliographie, etc. ;

`.toc .lof` , `.lot` :
fichiers de table des matières, listes des figures et des tableaux ;

`.idx .ind .ist .bib .bbl .bst` :
fichiers d'index : non classé, classé et formatage, fichiers de bibliographie : données, références et formatage.

2.3 Caractères spéciaux et commandes

Catégories de caractères et `\catcode`

Pour couvrir toutes les possibilités, saisie du texte, des formules et des commandes, il faut, comme pour tout langage de programmation, que des caractères aient une fonction spécifique. La différence est que dans le cas de TEX tous les caractères doivent aussi pouvoir être typographiés. Le tableau suivant montre le classement des caractères en 16 catégories repérées par un nombre, le `\catcode`.

Table 2.1 Catégories des caractères et valeurs du `\catcode`

Caractère et utilisation	Obtention en mode texte	Obtention en mode math	Valeur du `\catcode`
`\` : début de commande		`\backslash`	0
`{` : début de groupe	`\{`	`\{`	1
`}` : fin de groupe	`\}`	`\}`	2
`$` : entrée et sortie en mode math	`\$`		3
`&` : caractère d'alignement	`\&`		4
`^^M` : fin de ligne (entrée)			5
`#` : paramètre de remplacement	`\#`		6
`^` : mise en exposant			7
`_` : mise en indice			8
`^^@` : caractère ignoré (null)			9
␣ : espace et fin de commande			10
`A,...,Z,a,...,z` : lettres	`A` etc.	`A` etc.	11
Autres (que ceux cités avant et après)	`. ; ,` : etc.	`. ; ,` : etc.	12
`~` : caractère actif (commande)			13
`%` : caractère de commentaire	`\%`		14
`^^P` : caractère invalide (del)			15

Il est important de remarquer que ce classement est accessible par l'utilisateur : par exemple, pour typographier des poésies, il suffit de mettre le `\catcode` du caractère `^^M` (touche retour à la ligne) à 13 (caractère actif) et de faire une définition disant que la commande `^^M` fait une fin de paragraphe au lieu d'une fin de ligne :

```
{ \catcode`\^^M=13 \def\^^M{\par}
Poésie avec allers à la ligne à la fin des vers }
```

On se rend déjà compte que la conception de base du langage permet une véritable programmation qui va être utilisée pour faciliter la saisie mais aussi pour obtenir une typographie uniforme et de grande qualité.

Syntaxe des commandes

Il y a les commandes de base du langage TEX, appelées habituellement *primitives*, et les commandes définies par les utilisateurs (celles définies dans les fichiers LATEX et celles définies par l'utilisateur lui-même au fur et à mesure des besoins).

Du point de vue syntaxe, il y a trois types :

`\suite-de-lettres` :
: le caractère suivant la commande doit être un caractère « non lettre », le plus souvent c'est un blanc ; exemples :
`\bfseries mots` : active la mise en gras des mots suivants ;
`\textbf{suite de mots}` : commande avec paramètre mettant en gras la suite de mots contenus dans le groupement ;

`\caractère-quelconque-non-lettre` :
: sans restriction pour le caractère suivant, exemples :
`\\` termine une ligne, `\%` compose %,
`\'` place un accent aigu sur la lettre suivante, `\,` fait un petit espace,

`caractère-quelconque` :
: à condition d'être déclaré en catégorie 13 préalablement et sans restriction pour le caractère suivant ; exemple : les deux-points qui sont, en typographie française, une commande qui enlève l'espace avant s'il y en a un, place un espace spécifique (espace fine) et place le caractère deux-points.
Ceci est complètement transparent pour l'utilisateur et est défini dans le fichier `frenchb.ldf` contenant l'ensemble des définitions relatives au fichier de style appelé par la commande `\usepackage[frenchb]{babel}` figurant dans le fichier maître décrit au prochain chapitre.

Les commandes peuvent prendre de 1 à 9 paramètres ; dans les lignes qui suivent, `\nomcom` désigne le nom de la commande que l'on veut définir et `n` est le nombre de paramètres qu'elle doit prendre. Les syntaxes primitives de définition sont respectivement :
`\def\nomcom`{action-de-la-commande}
`\def\nomcom#1#2...#n`{action-de-la-commande-fonction-des-n-paramètres}
suivant que la commande ne prend pas de paramètre ou en prend $n \leq 9$.

Les syntaxes LATEX correspondantes sont :
`\newcommand{\nomcom}{`action-de-la-commande`}`
`\newcommand{\nomcom}[n]{`action-de-la-commande-fonction-des-n-paramètres`}`
Il y a aussi la possibilité d'une définition de commande où le premier paramètre est facultatif : s'il n'est pas donné, il prend par défaut la valeur donnée entre la deuxième paire crochets :
`\newcommand{\nomcom}[n][valeur-par-défaut]{`action-de-la-com...`}`
L'avantage de ces syntaxes est que, si la commande est déjà définie, alors on a un message d'erreur qui le signale : il faut alors choisir un autre nom ou bien remplacer `\newcommand` par `\renewcommand` qui écrase la première définition. Voilà quatre exemples :

- si le texte contient souvent l'adverbe μ-presque partout, on définit :
  `\newcommand{\mupp}{$\mu$-presque partout}`
  et on saisit `\mupp` ;
- si le texte contient des mots en italique gras, on définit :
  `\newcommand[1]{\itgras}{\textbf{\textit{#1}}}`
  et on saisit `\itgras{mots en italique gras}` ;
- si les formules contiennent souvent $L^i(\Omega, \mathcal{A}, \mu_j)$ pour différentes valeurs de i et j, on définit :
  `\newcommand{\loamu}[2]{L^{#1}(\Omega,\mathcal A,\mu_{#2})}`
  et on saisit `$\loamu{1}{\alpha}$` pour avoir $L^1(\Omega, \mathcal{A}, \mu_\alpha)$, etc.
- dans le cas où la valeur du premier argument est la plupart du temps 7, alors on définit :
  `\newcommand{\loamuu}[2][7]{L^{#1}(\Omega,\mathcal A,\mu_{#2})}`
  et on saisit en général `$\loamuu{\alpha}$` pour avoir $L^7(\Omega, \mathcal{A}, \mu_\alpha)$
  et `$\loamuu[\sigma]{\beta}$` pour avoir $L^\sigma(\Omega, \mathcal{A}, \mu_\beta)$ dans les cas particuliers ou le premier paramètre est différent de 7.

Il faut enfin signaler qu'après l'exécution de :
`\let\Nomcom=\nomcom`
la commande `\Nomcom` joue le rôle de la commande initiale `\nomcom` ; on peut alors utiliser le nom `nomcom` pour définir une autre macro ; ceci est souvent utilisé dans le cas où on veut modifier localement l'action d'une commande et réutiliser ensuite sa forme forme initiale : `\nomcom` sera par exemple la variante de la macro et on reviendra à la macro initiale avec le `\let` inversé (`\let\nomcom=\Nomcom`).

2.4 Caractères accentués, ligatures et espaces

Lettres accentuées

Les problèmes proviennent initialement toujours de la même cause, à savoir : l'économie de mémoire informatique au début des années quatre-vingt ; les américains n'utilisant pas (sauf exceptionnellement) de caractères accentués, les fontes initiales de TEX, les CM, sont des fontes codées sur 7 bits (2^7=128 caractères). Depuis longtemps maintenant, des fontes TEX codées sur 8 bits (2^8=256 caractères)

sont disponibles ; elles ont un ensemble de caractères accentués suffisant pour toutes les langues européennes d'alphabet latin.

Suivant le clavier disponible, on saisit par exemple pour les minuscules accentuées dérivées de e et pour le ç (et idem pour les majuscules) :

`é è ê ë` et `ç` ou `\'e \`e \^e \"e` et `\c␣c`

- Cas des fontes sur 7 bits
  - `\'e` utlise la commande définie par `\def\'#1{\accent"13 #1}` où le paramètre `#1` est, dans ce cas, affecté de la valeur particulière `e` : cette commande écrit son paramètre (la lettre e) puis place l'accent aigu (code 13 du codage OT1) sur la lettre e (idem pour les autres caractères accentués et à cédille) ; cette ancienne commande a été très utile car seules les fontes CM avaient une version vectorielle (nécessaire pour avoir une bonne qualité d'impression comme expliqué à la sous-section 2.1.6, page 8).
  - `é` est déclaré caractère actif, c'est-à-dire pouvant représenter une commande, commande définie `\defé{\'e}` ; la lettre é est donc formée comme précédemment ; ceci a été possible parce qu'il a été permis de faire des noms de commande avec des lettres accentuées.
- Cas des fontes virtuelles : en attendant une version vectorielle des fontes CM, on a disposé d'une fausse version vectorielle des CM sur 8 bits (dite « virtuelle »), les AE ; ces fontes, bien que basées sur les fichiers `.pfb` des CM (fontes sur 7 bits), s'utilisent comme des fontes sur 8 bits au codage T1 (la contruction des lettres accentuées se fait par le biais de la « machinerie » des fontes virtuelles et non plus par des commandes LaTeX utilisant la primitive `\accent` ; on entre là dans le domaine des TeXperts.
  - `é` écrit directement la lettre virtuelle é (code 233 du codage T1) comme toute lettre non accentuée.
  - `\'e` est une commande qui écrit la lettre virtuelle é (`\def\'#1{#1}`).
- cas des fontes sur 8 bits :
  - `é` écrit directement le caractère é.
  - `\'e` est la commande qui écrit le caractère é.

Ligatures

Les ligatures sont des caractères spécifiques dont le dessin est formé à partir des dessins des caractères composants :

- il y a des ligatures que l'on obtient grâce à une commandes spécifique : `\ae \oe \AE \OE` donnent respectivement æ œ Æ Œ ; cette commande est nécessaire car il y a des cas où a est suivi de e et où on ne fait pas la ligature ;
- d'autres ligatures au contraire s'obtiennent de façon transparente pour l'utilisateur : ce sont les ligatures ff fi ffi fl ffl qui ne sont pas constituées par un simple rapprochement de lettres composantes comme on le voit sur l'exemple suivant. La forme du f est conservée mais sa hampe est allongée pour se fondre sur le point du i qui a été abaissé. Ces ligatures sont gérées par des lignes des fichiers

métriques des fontes, fichiers `.tfm`, suivant un algorithme élémentaire facile à reconstituer.

f i fi

On arrive à des particularités assimilables aux ligatures. Il y a les trois tirets :

- le trait d'union saisi `-` : ci-joint ;
- le tiret moyen saisi `--` et utilisé dans le sens d'un intervalle : pages 23–35 ;
- le tiret long de ponctuation — (dialogue, etc.) saisi `---` ; là encore un algorithme élémentaire choisi le caractère correspondant.

Il y a ensuite les guillemets :

- les guillemets américains s'obtiennent avec `` ` ` `` (deux accents graves) et `'' ` (deux accents aigus) ; cela donne "entre guillemets" ;
- Les guillemets français s'obtiennent avec les commandes `\og` et `\fg` qui permettent d'obtenir « entre guillemets » et ceci avec les espacements avant et après corrects ; pour une saisie en latin1, on peut saisir directement (`\char19` et `\char20`) les caratères de code 19 et 20 (en codage T1) lieu d'utiliser les commandes ci-dessus.

Espacements

Una espacement est un espace resté blanc en conséquence d'une commande.

- L'espace obtenu avec la barre d'espace du clavier, une suite de tels espaces et le retour à la ligne donnent un espace intermot. Il est élastique ; sa largeur va être déterminée à la composition et aura une valeur entre une valeur minimum et une valeur maximum respectivement plus petite et plus grande que l'espace intermot de la fonte courante (et d'une telle façon que l'on peut y avoir accès par le biais des paramètres `\tolerance` et `\pretolerance`). Il est évident que, pour des longueurs de ligne faible, on aura beaucoup de mots coupés ; on peut éviter cela en augmentant ces paramètres, c'est-à-dire en « tolérant » des espaces intermots plus grands que d'habitude).
- La commande `\␣` permet d'avoir un espace supplémentaire forcé si nécessaire (la notation ␣ représente dans les explications un espace obtenu avec la barre d'espace (caractère blanc, code ASCII 32) ; l'espace ainsi produit à la composition est l'espace intermot de la fonte courante et n'est pas élastique ;
- Une ligne blanche ou la commande `\par` donne la fin de paragraphe.
- Il faut dire que les espaces spécifiques en avant des caractères `:` `;` `!` et `?` sont automatiquement insérés (ces caractères sont déclarés en catégorie 13 et sont définis en tant que commande, voir page 12).
- En ce qui concerne les dimensions, TEX et donc LATEX acceptent les unités suivantes : `pt` `in` `mm` `cm` et `em` `ex`. Les deux dernières sont des unités variables en fonction de la fonte ; elles sont données dans le fichier métrique : l'`em` et l'`ex` sont respectivement la largeur du m et la hauteur du x ; on peut donc faire des

constructions, dont les espaces s'ajustent parfaitement lors du changement de fontes (de famille, de taille, de graisse ou de forme) ; voilà un exemple :
taille courante : et ...
taille courante : et ...
On y remarquera la position des deux-points, gouvernée par une commande d'espacement horizontal exprimée en `em`, s'adapte automatiquement lors du passage de la taille courante à une grande taille ; la même remarque vaut pour la séparation de points de suspension.

- On peut imposer des espacements avec les commandes :
  - `\hspace{longueur-avec-unité}` ; cet espace horizontal n'est pas pris en compte en début et en fin de ligne sauf si l'on utilise la version `\hspace*{...}` ;
  - `\vspace{longueur-avec-unité}` ; cet espace vertical n'est pas pris en compte en début et en fin de page sauf si l'on utilise la version `\vspace*{...}` ;
  - la `longueur-avec-unité` peut être donnée avec la possibilité d'ajustement à la composition sous la forme
    `\vspace{10mm plus 3mm minus 1mm}`
    qui permet des variations proportionnelles aux deux longueurs annexes données au moment de la mise en page ;
  - la commande `\kern 10mm` fait un espace horizontal insécable 10 mm ; elle est particulièrement utile pour construire un glyphe ou un logo composé de plusieurs caractères ; exemple : le point-tiret . — des mathématiciens codé `.\kern0.35em---` (il n'y aura jamais coupure de ligne entre le point et le tiret) ; un autre exemple est le logo TEX ;
  - on dispose des espacements appelés « ressorts » `\hfill` et `\vfill` qui remplissent l'espace libre jusqu'aux limites, par exemple `\hfill\break` fait du blanc jusqu'à la fin de la ligne et fait passer les mots suivants à la ligne suivante, c'est ce que fait la commande `\newline` ; même effet verticalement avec `\vfill\pagebreak`, que l'on peut obtenir avec la commande `\newpage`.
  - enfin, il y a les espacements appelés « ressorts négatifs » `\hss` et `\vss` très utiles pour faire des recouvrements dans des aligements.

Exercices

Faire une ligne contenant & $ % et # (T) traiter et (V) visualiser.

Faire une ligne contenant `\alpha` suivi d'un mot sans accents et du mot été (T). Mettre `$` devant `\alpha` (T). Mettre `$` après `\alpha` (T,V).

Faire une ligner contenant & et # (T). Lire les messages.

Faire un paragraphe commençant par `\catcode‘\é=13 \defé{\par}` et continuer par du texte avec des mots contenant é (T,V).

Faire à la suite un nouveau paragraphe où les caractères é apparaissent de nouveau (il y a deux manières de faire).

Chapitre 3

Fichier « maître » et fichiers-sources

Le fichier « maître » est, on le rappelle, le fichier donné en paramètre aux exécutables `latex` ou `pdflatex` pour faire la composition ; il contient

- un préambule où se trouvent les entrées des fichiers de commandes de la distribution et des fichiers de commandes personnels ;
- un corps incluant les commandes d'entrée des fichiers-sources (ou fichiers de saisie) et des commandes d'exécution de la table des matières, de l'index, etc.

On trouvera ci-dessous un exemple type de fichier maître. Seules les lignes avec un seul % sont à décommenter en fontion des choix.

```
%% Fichier << maître >> pour le present polycop
%%%%%%%%%%% PREAMBULE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\documentclass[12pt]{book}
```

Cette commande charge (ou appelle) l'entrée de la classe `book` (entre `{}`), avec l'option `12pt` (entre `[]`). Il y a aussi les classes `article`, `amsart`, `report`, `slides`, `seminar`, `beamer` (ces trois dernieres sont consacrées à la projection) ; ces fichiers de classe ont l'extension `.cls`. Les fichiers d'option `a4paper`, `landscape`, `10pt`, `11pt`, etc. ont l'extension `.clo`. Il peut y avoir plusieurs options séparées par des virgules[(1)].

Toutes les commandes d'entrées `\usepackage` qui suivent appellent des fichiers dits de style (extension `.sty`) qui complètent les fichiers de classe ou ajoutent d'autres fonctionnalités (l'appellation « style » est à prendre dans un sens étendu[(2)]).

```
\usepackage[xxxxx]{xcolor}%  ;
   %% xxxxx est dvips ou pdftex ; voir le package graphics ;
   %% option usenames : propose plus de 100 couleurs prédéfinies
%\usepackage[cp850]{inputenc}% Pour saisie sous DOS
\usepackage[latin1]{inputenc}% Défaut ; saisie sous Windows et Unix
%\usepackage[applemac]{inputenc}% Inutile avec editeur Mac latin1
```

(1) Pour connaître les classes et leurs options, il faut se reporter aux documentations disponibles : distributions, site CTAN et ouvrages cités. Même remarque pour les styles (ou paquetages) ;

(2) C'est pour cette raison que certains francophones utilisent le mot « paquetage ».

TEX a été conçu pour être universel. Les caractères ont un codage donné en interne : pour pouvoir être utilisé sous tous les systèmes d'exploitation, il faut lui donner le codage des fichiers-sources (codage en entrée) ; c'est le rôle du fichier `inputenc.sty` avec ses options ; (`latin1` est le codage en entrée par défaut ; on n'a proposé que les trois plus utilisés mais l'option `utf8` est disponible).

```
%\usepackage[OT1]{fontenc}% Défaut ; pour utiliser les fontes CM
\usepackage[T1]{fontenc}% Pour utiliser les fontes codees sur 8 bits
```

Pour la même raison, afin de pouvoir composer avec toutes les fontes disponibles, quel que soit son propre codage, il faut donner à TEX ce codage, ce que l'on fait à l'aide d'une option du style `fontenc` (codage en sortie).

- OT1 est le codage sur 7 bits (128 caractères) des fontes CM de TEX (les lettres accentuées sont formées par des macros plaçant l'accent sur les voyelles comme expliqué à la section 2.4, page 13). C'est le codage en sortie par défaut car la majorité des publications scientifiques sont en anglais (lettres accentuées inutiles). Cependant, ces fontes ont continué à être utilisées par les francophones pour des éditions de haute qualité car c'étaient les seules fontes disponibles en version vectorielle de *type 1* dans le domaine public.
- T1 est le codage adopté il y a quelques années par monde TEX ; c'est un codage sur 8 bits (256 caractères). Ce codage est suffisant pour les langues européennes utilisant l'alphabet latin. Il y a plusieurs « versions » des fontes CM en 8 bits ainsi que beaucoup d'autres fontes :
  - les EC existent en version *bitmap* dans la distribution TEX-LATEX usuelle, elles ne sont pas utilisables pour une édition de qualité (en particulier le fichier `.ps` ainsi produit donne par la suite un très mauvais fichier `.pdf`) ;
  - les AE de la distribution usuelle sont de *type 1* car elles utilisent, par le biais de la machinerie des « fontes virtuelles », les fichiers `.pfa` et `.pfb` des CM, et donc sont du domaine public ;
  - les CMsuper en version *type 1* ont un léger défaut (dimensions légèrement différentes par rapport aux CM, d'où des difficultés pour les rééditions ;
  - enfin les LM, également en version *type 1*, n'ont pas le défaut des CMsuper. Leur utilisation est maintenant conseillée par l'assocation GUTenberg ;
  - il y a aussi les fontes Fouier-GUTenberg et les fontes Times dont on parlera à la page suivante.

```
%\usepackage{mltex}% Pour les fontes CM et les lettres avec accent
```

Cet ancien fichier contient le nécessaire pour utiliser les fontes CM (construction des lettres accentuées par des macros comme expliqué à la section 2.4, page 13) ; il permet de consever des césures correctess des mots incluant ces macros de contruction de lettres accentuées [7].

Ce fichier est toujours maintenu. Il peut être utilisé pour une réédition ou par ceux qui utilisent toujours les CM.

```
\usepackage{amsmath,amssymb,amsfonts,theorem}% Packages de base
     %% pour disposer de tous les symboles mathematiques et de
     %% certaines possibilites de composition des formules
     %% et d'environnements : theorem, definition, proof, etc.
```

Les trois premiers fichiers `amsmath.sty`, etc. de l'*Americal Mathematial Society* permettent de nombreuses constructions de formules ainsi que l'utilisation de nouvelles fontes mathématiques. Le style `theorem` permet la création d'environnements particulièrement bien adaptés aux publications de mathématiques.

```
\usepackage{array}% Supplement pour améliorer les tableaux
\usepackage[xxxxx]{graphicx}% Fondamental pour integrer des figures
\usepackage{float}% Pour placer les figures plus facilement
\usepackage{multicol}% Indispensable pour l'index double colonne
```

Pour package `graphicx`, comme le package `xcolor`, l'option `xxxxx` doit être `dvips` si on passe par le fichier PostScript intermédiaire et `pdftex` si l'on produit le PDF directement. Ce package permet l'inclusion des figures et la manipulation d'objets graphiques : encadrement, rotation, redimensionnement, etc.

Les commandes suivantes sont décrites dans le fichier lui-même et ne nécessitent que peu de commentaires supplémentaires.

On rappelle que les fontes CM sont utilisées par défaut.

```
%\usepackage{aeguill}% Pour utiliser les fontes AE, version
  %% virtuelle 8 bits des fontes CM (evite l'anciene
  %% "machinerie mltex" tout en continuant à utiliser la version
  %% vectorielle des CM pour conserver la haute definition)
\usepackage{textcomp,lmodern}% Pour utiliser les fontes LM
  %% textcomp.sty contient les symboles \texteuro \textsection
  %% \textperthousand \textparagraph \textcopyright \textsurd etc.
%\usepackage{fourier}% Pour utiliser les fontes Fourier
%\usepackage{times}% Pour utiliser les fontes Times
```

Les caractères cités sont € § ‰ ¶ © √.

Les fontes Fourier sont basées sur les Utopia ; les symboles mathématiques sont dus à Michel Bovani, membre de GUTenberg (elles peuvent être utilisée librement). Les fontes Times ne sont pas du domaine public ; elles comprennent un supplément MathTimes (non public également). Il est à noter que les fontes mathématiques des CM peuvent s'accorder avec de nombreuses fontes texte, ce qui permet de sortir du graphisme des fontes TEX au moins dans les cas où le volume du texte est dominant.

```
\usepackage[frenchb]{babel}% Pour la langue française :
   %% cesures des mots et specificites typographiques
```

Le style `babel` avec l'option `frenchb`, en plus d'assurer les bonnes césures de mot en français, définit les commandes qui changent *Chapter* en Chapitre, etc., qui donnent les guillemets français, qui changent la valeur de certains espaces, etc. Il

introduit des modifications car le « goût américain » est bien différent des traditions de la typographie française, [4] pages 579 et suivantes.

```
%% APPEL DES PACKAGES PLUS SPECIFIQUES
\usepackage{poly}% Permet d'obtenir le format du présent document avec
   %% une saisie LaTeX normale et quelques commandes supplémentaires
\input{navi.tex} fait la barre de navigation
\usepackage{makeidx}% Package pour faire l'index
\makeindex % Commande la création l'index livre.idx
%% POUR PRODUIRE LES HYPERLIENS : xxxxx est dvips ou pdftex
%% voir le package graphics
%\usepackage[xxxxx,colorlinks,hyperindex,raiselinks=true,
%linktocpage,pagebackref,plainpages=false,pdfpagelabels]{hyperref}
  %% option est dvips ou pdftex (voir graphics et xcolor)
\StandardLayout% Pour prendre en compte les modifications par rapport
   %% à frenchb introduites dans poly.sty ou d'autre styles
```

Le rôle du style `hyperref` sera détaillé, ainsi que ses options les plus courantes, au chapitre 13. Les lignes suivantes ne sont pas nécessaires : elles sont données à titre d'exercice pour tester les coupures de mots en français et en anglais.

```
%% POUR FAIRE DES TESTS DE COUPURE (français et anglais)
\lccode`\'=`\' % Pour couper les mots contenant une apostrophe
%% Tester en enlevant les % devant les commandes \showhyphens
%% et devant \end{document}\endinput, PUIS LES REMETTRE
%% Pour l'anglais enlever PROVISOIREMENT le % devant \language=0
%% Les lignes sont a corriger si Windows, Unix ou Mac
%\language=0 %% =1 par defaut a cause de l'option frenchb de babel
%\showhyphens{signal conteneur diagonale évènement l'intérêt}
%\showhyphens{signal conteneur \'ev\`enement l'int\'er\^et}
%\showhyphens{SIGNAL DIAGONALE \'EV\`ENEMENT L'INT\'ER\^ET}
%\end{document}\endinput

%%%%%%%%%%%% DOCUMENT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begin{document}
%% POUR FAIRE DES ESSAIS DIVERS ENLEVER LE % SUIVANT PUIS LE REMETTRE
%\input {nom-de-fichier-test}\end{document}\endinput
```

Les lignes qui terminent le fichier maître appellent les fichiers-sources. On y trouvera les commandes de table des matières, de bibliographie et d'index (on détaillera tout cela plus loin, au chapitre 6).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TITRE, PREFACE, INTRO, NOTATION, etc. et TABLE DES MAT.
\frontmatter % Pour classe book seulement
%% La pagination est en chiffres romains minuscules
%% Preface, introduction... etc.
%\input {nom-de-fichier}
```

```
%\tableofcontents % Prepare la table des matieres. a la premiere
                  %% passe, puis la compose a la deuxieme

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PARTIE PRINCIPALE
\mainmatter % Pour classe book seulement
%% La pagination est pour toute la suite en chiffres arabes
%% APPEL DES FICHIRS DE CHPITRES...
%\input {nom-de-fichier}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ANNEXES
%\appendix
%% "Chapitre" est automatiquement changé en "Annexe"
%% APPEL DES FICHIRS DES ANNEXES...
%\input {nom-de-fichier}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BIBLIOGRAPHIE ET INDEX
\backmatter
%% Conclusion, liste des figures et des tableaux, etc.
%\input{livre.bbl} %ou commande \bibliography{livre.bbl}
% si bibliographie automatique (cf. partie II.C)
%\printindex % Compose l'index a partir du fichier livre.ind
%% obtenu par classement du fichier livre.idx avec makeindex
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\end{document}
```

Le fichier de bibliographie `livre.bbl` peut être saisi directement ou produit à l'aide de l'utilitaire `bibtex` à partir des bases de données existantes : cette méthode, de plus en plus utilisée, est décrite en détail dans la partie II.C.

Il est conseillé de faire un tel fichier « maître » personnel où l'on ne fait qu'ajouter et retrancher des commandes en fonction du document que l'on prépare (ne pas oublier une description sommaire afin de s'y retrouver par la suite : le caractère de commentaire % est fait pour cela !).

Exercices

Prendre une option du package `inputenc` qui ne correspond pas à votre système et faire un test.

Faire les tests de césure de mot ; y rajouter des commandes de césures d'autres mots.textcomp

Faire un test des symboles proposés page 19 à l'entrée du package `textcomp.sty`.

Chapitre 4

Fontes et polices

Le mot *fonte* est, en principe, réservé à un ensemble de *polices* ayant un aspect graphique commun ; par exemple, on parle des fontes CM qui comprennent une grande variété de polices caractérisées par la *famille*, la *forme*, la *graisse* et la *taille*. La gestion des fontes sera approfondie et complétée dans la partie II.B.

4.1 Famille, forme et graisse

On a résumé sur le tableau ci-dessous les commandes disponibles en LaTeX. Pour des cas non inclus dans ce tableau on a toujours la possibilité de définir une nouvelle fonte (si l'on dispose des fichiers nécessaires) par la commande du type commutateur

`\newfont{\commande-de-la-fonte}{nom-du-fichier}`

où le dernier argument peut être complété par un redimensionnement du type `at 18pt` ou du type `scaled \magstep 3` (avec la convention que `\magstep n` signifie un facteur d'échelle de $1{,}2^n$) ; on trouvera un exemple en fin de chapitre.

Table 4.1 Commandes de famille, forme et graisse

Commande type commutateur	Commande avec argument	Effet obtenu
Famille		
`{\rmfamily ...}`	`\textrm{...}`	romain
`{\ttfamily ...}`	`\texttt{...}`	`machine à écrire`
`{\sffamily ...}`	`\textsf{...}`	sans sérif
Forme		
`{\upshape ...}`	`\textup{...}`	droit
`{\itshape ...}`	`\textit{...}`	*italique*
`{\slshape ...}`	`\textsl{...}`	*oblique*
`{\scshape ...}`	`\textsc{...}`	PETITES CAPITALES
Graisse		
`{\bfseries ...}`	`\textbf{...}`	**gras** (bold face)
`{\mdseries ...}`	`\textmd{...}`	maigre (medium)

On peut théoriquement choisir une caractéristique dans chaque groupe et les appliquer simultanément ; en réalité, dans les distributions habituelles de TEX : le romain petites capitales n'existe qu'en maigre, le machine à écrire n'existe qu'en maigre droit et maigre italique, et le sans sérif n'existe qu'en droit maigre et droit gras. Voilà quelques exemples (ne pas oublier que, par défaut, les famille, forme et graisse sont : romain, droit et maigre) :

`{\slshape\bfseries ...}` ***romain penché gras***

`{\sffamily\bfseries ...}` **sans sérif gras**

`{\scshape ...}` ROMAIN PETITES CAPITALES MAIGRE ≠ CAPITALES

Toutes ces commandes ont, comme on l'a constaté, une forme commutateur et une forme commande avec paramètre ; les saisies ci-dessous sont équivalentes :

```
début en romain {\itshape suite en italique} fin en romain
début en romain \textit{suite en italique} fin en romain
début en romain \itshape suite en italique \upshape fin en romain
```

Par défaut, LATEX utilise les fontes CM comme cela a été dit page 18. Pour le présent document, on a utilisé initialement les fontes AE qui sont une version 8 bits des CM (commande `\usepackage{aeguill}`, page 19, basée sur les mêmes fichiers `.pfb`. Au cours d'une mise à jour, on a opté pour les fontes LM. On aurait aussi pu opter pour les fontes Times tout en gardant les fontes CM pour les mathématiques (l'expérience a montré que cette juxtaposition est parfaitement acceptable).

4.2 Taille en mode texte et en mode mathématique

Les tailles disponibles en LATEX sont obtenues avec les commandes du tableau suivant qui n'exitent que sous la forme commutateur.

Table 4.2 Commandes de taille ; les tailles données (en points) correspondent à l'option `12pt` de la classe `book`, option du présent document (les tailles pour l'option `10pt` sont entre parenthèses)

Taille	Commande	Effet en texte	Effet en mode math
6(5)	`\tiny`	En mode texte	
8(7)	`\scriptsize`	En mode texte	
10(8)	`\footnotesize`	En mode texte	$A = \int_a^b f(x)\,\mathrm{d}x$
11(9)	`\small`	En mode texte	$A = \int_a^b f(x)\,\mathrm{d}x$
12(10)	`\normalsize`	En mode texte	$A = \int_a^b f(x)\,\mathrm{d}x$
14(12)	`\large`	En mode texte	
17(14)	`\Large`	En mode texte	$A = \int_a^b f(x)\,\mathrm{d}x$
20(17)	`\LARGE`	En mode texte	
25(20)	`\huge`	En mode texte	

4.3 Fontes en mode mathématiques

On verra plus loin qu'il y a deux modes fondamentaux : le mode texte et le mode mathématique ; certaine éléments de saisie sont valables dans les deux modes mais produisent des résultats différents ; d'autres sont spécifiques à un seul mode :

- `a` donne a (droit) en mode texte mais donne a (italique math) en mode math ;
- `\alpha` donne α en mode math et donne une erreur de compilation en mode texte ; `â` donne â en mode texte et une erreur en mode math.

Avec les styles de l'AMS, on dispose d'un choix important de polices pour les mathématiques. Les TEXperts savent en introduire autant qu'ils veulent ; c'est plus complexe que pour le mode texte car il faut préparer la machinerie qui, une fois choisie une nouvelle fonte, va sélectionner automatiquement la taille suivant que le caractère est en modes math `\displaystyle` (mode déployé, formules centrées sur une ligne) ou `\textstyle` (mode texte, formule dans le texte) ou en modes math `\scriptstyle` (indices, supérieurs ou inférieurs) ou `\scriptscriptstyle` (indices d'indice, inférieurs ou supérieurs).

Table 4.3 Polices du mode mathématique

Saisie	Effet	Fonte et remarques
`\mathrm{A}`	A	romain
`\mathbf{A}`	$\mathbf{A}$	romain gras
`A`	A	italique math (par défaut)
`\boldsymbol{A}`	$\boldsymbol{A}$	italique math gras
`\mathbb{A}`	$\mathbb{A}$	blackboad (ensembles $\mathbb{R}$, $\mathbb{C}$, etc.)
`\mathfrak{A}`	$\mathfrak{A}$	gothique (fraktur)
`\mathcal{A}`	$\mathcal{A}$	calligraphique
`\mathsf{A}`	A	sans sérif
`\mathtt{A}`	$\mathtt{A}$	machine à écrire

Signalons que la police math italique n'est pas identique à la police italique pour le texte. Elle possède de nombreuses dimensions spécifiques destinées à positionner les exposants et les indices mais n'inclut pas les crénages. Ces crénages sont contenus dans les fichiers `.tfm` des fontes du mode texte. Après avoir placé un caractère, TEX avance le point courant de la largeur du caractère ; ensuite, s'il y a crénage, TEX recule le point courant en fonction du caractère suivant ; voilà un exemple :

Taxe Taxe

Sans crénage on a bien l'impression que le a est trop éloigné du T. Ce sont tous ces détails qui font la qualité de la composition et qui ont une grande importance

pour la lisibilité (cf. l'uniformité des espaces entre les mots d'un même paragraphe dont l'importance a été soulignée citée à la page 3). Le profane resent tous ces types d'imperfections qui rendent la lecture pénible (ou plutôt moins agréable) mais, dans la majorité des cas, il ne peut en identifier l'origine. Il faut se plonger dans les ouvrages de typographie pour découvrir toute la finesse de cette discipline ... et D. Knuth a conçu TEXpour pouvoir en respecter toutes ses exigences.

Exercices

Tester les différentes commandes du chapitre.

Supposons que l'on ne dispose que du commutateur `\slshape`, construire la macro avec argument `\slant` qui met son argument en penché.

Pour que le crénage n'est pas lieu, il suffit d'introduire un groupe vide `{}` pour « tromper » TEX : il ne « voit » pas alors la lettre suivante et ne recule pas le point courant comme il l'aurait fait nomalement. Traiter l'exemple du mot AVOIR composé en très gros caractères (définir la commande de la fonte nécessaire avec `\newfont{\liga}{ec-lmss10 at 50 pt}` et la mettre en action avec le commutateur `\liga` ... sans oublier le groupement `{...}`!).

Chapitre 5

Mise en page

5.1 Dimensions de mise en page

On trouvera, page 555 de la référence [3] et page 204 de la référence [4], un schéma de la page illustrant ce qui va suivre. Il y a d'abord les dimensions de la feuille de papier :

`\paperheight` et `\paperwidth`

Il y a ensuite les dimensions des éléments constituant la page.

Dimensions dans le sens vertical

Table 5.1 Principales dimensions de la page dans le sens vertical (à partir du haut)

Dimension	Description
`\voffset`	offset vertical (`+1in`)
`\topmargin`	marge haute
`\headheight`	hauteur de la tête de page
`\headsep`	distance entre le bas de tête de page et le haut du miroir
`\textheight`	hauteur du miroir
`\footskip`	distance entre le bas du miroir et le bas du pied de page

Dimensions dans le sens horizontal

Table 5.2 Principales dimensions de la page dans le sens horizontal (à partir de la gauche)

Dimension	Description
`\hoffset`	offset horizontal (`+1in`)
`\evensidemargin`	marge gauche des pages paires
`\oddsidemargin`	marge gauche des pages impaires
`\textwidth`	largeur du miroir
`\marginparsep`	distance entre le miroir et les notes de marge
`\marginparwidth`	largeur des notes de marge
`\headwidth`	largeur des têtes et pieds de page

On fixe les valeurs des dimensions à l'aide des commandes du type :
`\setlength{\textheight}{297mm}`
et on les modifie de la manière suivante :
`\addtolength{\textwidth}{3mm}`
La notation `(+1in)` signifie que si l'on écrit `\voffset=14mm` alors l'offset vertical vaudra 1 in plus 14 mm ; la même remarque vaut pour `\hoffset`. Cela a été fait pour que, en ne donnant aucune valeur, l'impression se fasse avec une mage haute et une marge gauche de 1 in, au début où les imprimantes ne pouvaient pas imprimer près du bord des feuilles.

Les marges de gauche des pages paires et impaires peuvent être différentes pour permettre la reliure. Si l'on utilise des notes de marge (notes ou petites figures), on peut décider de « couvrir » ou non les notes de marge avec le haut de page (par défaut `\headwidth` vaut `\textwidth`).

Autres dimensions accessibles

D'autres commandes sont accessibles au utilisateurs ; elle sont en général déterminées par la classe utilisée ; on n'en cite trois :

`\topskip` :
distance minimum entre le haut du miroir et la ligne de base de la première ligne (défaut : 10 pt) ;

`\maxdepth` :
distance maximun entre le bas du miroir et le bas de la dernière ligne de la page (défaut : 3 pt) ; c'est dans cette « profondeur » que se trouvent les pieds des minuscules g, j, p, etc.

`\baselineskip` :
distance minimum entre les lignes de base successives.

Elle a une valeur par défaut dépendant de l'option de classe choisie, voir page 17 ; elle est adaptée à la fonte et à la taille courante.

`\renewcommand{\baselinestretch}{x}` :
commande qui modifie la valeur de `\baselineskip` en redéfinissant le facteur de dilatation `\baselinestretch` (ce facteur est un nombre positif qui vaut 1 par défaut) ; en général, il vaut mieux ne pas le modifier, surtout si l'on manque d'expérience.

5.2 Espacements

On a déja vu en détail les commandes d'espacements verticaux et horizontaux `\vspace{...}`, `\hspace{...}`, `\kern...`,`\vfill` et `\hfill`.

On a aussi introduit `\baselineskip`, distance entre les lignes de base de deux lignes successives ; cette distance peut être augmentée automatiquement si le bas des caractères d'une ligne est trop proche du haut des caractères de la ligne suivnte.

L'indentation des paragraphes a une longueur `\parindent` ; l'espace vertical supplémentaire ajouté à `\baselineskip` à la fin d'un paragraphe est `\parskip`. On peut empêcher l'indentation en écrivant `\noindent` avant la premier mot de paragraphe.

Avant et après les formules mathématiques centrées, on dit aussi déployées, TEX place des espaces verticaux choisis en fonction de la largeur de la formule : `\beforedisplayskip`, `\beforeshortdisplayskip`, etc. Il est recommandé de ne pas toucher à ces espaces qui ont été optimisés. La même remarque vaut pour les nombreux environnements proposés dans les divers packages, par exemple les environnements du type `theorem`. On peut composer un document sans ajouter des espacements, excepté dans quelqes situations particulières : une débauche d'espacements mal choisis et mal placés nuit à la clarté du document.

Il y a enfin trois espacements verticaux, un peu élatiques, que l'on peut modifier et que l'on apprend à utiliser par l'expérience : `\smallskip`, `\medskip` et `\bigskip`.

Toutes les dimensions d'espacements verticaux (`\baselineskip` excepté) sont élastiques : c'est ce qui permet la mise en page ; à titre d'exemple, pour ce document on a choisi `\parskip=4pt plus 2pt minus 1pt`.

5.3 Coupure de mot, de ligne et de page

Les mots sont coupés automatiquement par TEX à partir d'un fichier de « motifs » de césure spécifique à chaque langue et, en ce qui concerne le français, contenant la quasi-totalité des exceptions. En général, il doit rester au moins deux lettres en fin de ligne et il doit y avoir trois lettres en début de ligne : élève ne peut être coupé ! Le fichier des motifs de césure pour le français (fichier `frhyph.tex`) est dû à un certain nombre de bonnes volontés francophones. Il ne faut pas toucher à ces problèmes qui ont demandé beaucoup de travail pour la mise au point. On peut faire des tests avec la commande `\showhyphens{...}` dont l'argument est une liste de mots séparés par des virgules : les coupures proposées apparaissent alors à la console. Si cela est nécessaire, on peut proposer les césures pour des mots que l'on veut couper d'une façon spécifique ou pour des mots qui auraient échappé à l'analyse minutieuse des spécialistes : dans le préambule, on place la commande `\hyphenation{...}` dont l'argument est la liste de mots séparés par un espace avec un tiret à chaque possibilité de coupure acceptable.

`\newline` ou `\\` :
: coupe la ligne sans l'étirer (sans la justifier) et passe à la suivante (pas d'indentation et pas d'espace en supplément de `\baselineskip` ; ce n'est pas une fin de paragraphe).

`\\[longueur-avec-unité]` :
: idem mais ajoute l'espacement vertical de la hauteur donnée.

`\linebreak` :
coupe la ligne à l'endroit demandé en étirant les espaces intermots pour assurer la justification. Si ces espaces deviennent trop grands, alors le paramètre de « laideur » de la ligne devient plus grand qu'une certaine limite au delà de laquelle il s'affiche avec un message à la console au moment du traitemement ; on peut alors lire :
`Underfull \hbox at paragraph line xxx - yyy`.

`\linebreak[1,2,3 ou 4]`:
« encourage » la coupure de ligne d'autant plus que l'indice choisi est grand ; par défaut, l'indice facultatif est pris égal à 4.

`\nolinebreak[1,2,3 ou 4]` :
commande réalisant l'inverse de la précédente (« décourage » au lieu de « encourage »).

`~` : produit un espace où la coupure ne peut se produire ; c'est très utile dans de nombreuses situations, où une coupure est mal venue :
`Université P.~Sabatier, de dimension~$4$, etc.`

`\newpage` :
fait passer à la page suivante en terminant le paragraphe et sans étirer verticalement la page courante (ce qui fait qu'il peut rester du blanc en bas de page).

`\pagebreak` :
fait passer à la page suivante à la fin de la ligne courante et étire verticlement la page courante en dilatant les espaces élastiques. Si ces espaces deviennent trop grands, alors le paramètre de « laideur » de la page devient plus grand qu'une certaine limite au delà de laquelle il s'affiche dans un message à la console au moment du traitemement ; on peut alors lire :
`Underfull \vbox when output is active ...[...]`.

`\pagebreak[1,2,3 ou 4]` :
« encourage » la coupure de page d'autant plus que l'indice choisi est grand ; par défaut, l'indice facultatif est pris égal à 4.

`\nopagebreak[1,2,3 ou 4]` :
commande réalisant l'inverse de la précédente (« décourage » au lieu de « encourage »). Cette commande est nécessaire pour assurer une composition de qualité : on peut souhaiter par exemple qu'une phrase importante par son contenu ne soit pas coupée par un changement de page.

Il est parfois difficile d'obtenir les coupures de ligne et de page souhaitées lorsque l'on recherche la perfection ; on peut par exemple modifier une coupure de ligne pour éviter que deux petites formules mathématiques avec exposant et indice ne donnent plus l'impression de se toucher lorsqu'elle sont, par malchance, l'une au-dessus de l'autre dans deux lignes successives. On arrive vite à des affaires de TEXperts.

5.4 Style de page

Cette section est intéressante du point de vue programmation mais aussi parce que le style de page par défaut est d'un goût typiquement américain particulièrement orthogonal aux traditions de la typographie européenne...

Les styles de mise en page sont fixés par deux commandes :

`\pagestyle{style-de-page}` :
: choisi le style `style-de-page` pour tout le document ;

`\thispagestyle{style-de-page}` :
: choisi le style `style-de-page` pour la seule page courante.

Les styles prédéfinis par LaTeX sont :

`empty` :
: pas de tête et pas de pied de page ;

`plain` :
: pas de tête de page et un pied de page constitué par le folio centré (utilisé par la macro `\chapter{...}` pour la page contenant le titre de chapitre) ;

`headings` :
: tête de page et pas de pied de page (d'un goût particulièrement horrible).

Traditionnellement, on doit utiliser deux styles : `empty` pour les pages de titre, les pages de début de chapitre et les pages blanches et un autre pour les autres pages. Bien qu'un peu complexe pour les débutants, il est important de pouvoir créer un style de page pour ne pas utiliser le style par défaut. On est particulièrement aidé dans cette tâche par le package `fancyhdr` ; voici ce que l'on peut faire et adapter à son désir sans aucune difficulté (voir le fichier `poly.sty` qui gère les hauts de page du présent document). Les principales commandes sont :

`\RequirePackage{fancyhdr}` :
: appelle le fichier de style nécessaire s'il n'était pas déjà chargé ;

`\pagestyle{fancy}` :
: choisi le style `fancy` modifiable à loisir par quelques macros (dont la plupart suivent ci-après) ; on peut ainsi créer pratiquement tous les styles de page non extravagants ;

`\fancyhf{}` :
: annule complètement les têtes et les pieds de page définis par défaut ou définis dans un fichier déjà chargé ;

`\fancyhead[LE,RO]{\small\thepage}` :
: pour la tête de page ; écrit le folio à gauche `L` sur les pages paires `E` et à droite `R` sur les pages impaires `O` ;

`\fancyhead[CE]{\footnotesize\rightmark}` :
: pour la tête de page ; écrit le contenu du registre `\rightmark` au centre `C` sur les pages paires `E` ;

`\fancyhead[CO]{\footnotesize\leftmark}` :
pour la tête de page ; écrit le contenu du registre `\leftmark` au centre sur les pages impaires `O` (pages de droite) ; les noms des deux registres cités (`\rightmark` et `\leftmark`) ont été choisis en pensant à des styles où le titre de chapitre est sur les pages paires (pages de gauche) et le titre de section sur les pages impaires (pages de droite), ce qui est l'inverse du choix fait ici pour le présent document où le registre `\leftmark` se place sur la page de droite(`CO` : `O`, *odd*, impaire, droite).

`\renewcommand{\chaptermark}[1]{\markboth{#1}{#1}}` :
redéfinit la commande `\chaptermark` qui est utilisée par la macro `\chapter` ; avant `#1`, on pourrait insérer `\thechapter.␣` dont le résultat serait de faire précéder le titre de chapitre du numéro du chapitre suivi d'un point et d'un espace ;

`\renewcommand{\sectionmark}[1]{\markright{#1}}` :
redéfinit la commande `\sectionmark` qui est utilisée par la macro `\section` ; on peut imaginer toute modification comme pour la macro précédente ; pour avoir le numéro de section devant le titre de section, il faut remplacer `#1` par `\thesection. #1` ;

`\renewcommand{\headrulewidth}{0pt}` :
fait disparaître le filet qui, par défaut, se trouve en bas de la tête de page et qui ne convient pas pour ce type de document ;

Remarques et explications

Cette fin de chapitre est intéressante du point de vue programmation mais n'est pas utile à l'utilisateur moyen. *Rappel* : `\rightmark` se place sur la page gauche et `\leftmark` sur la page de droite.

Pour illustrer la nécessité et les raisons de la complexité des macros `\rightmark`, `\leftmark`, `\chaptermark` et `\sectionmark`, imaginons que l'on est dans la situation où la macro `\section` placerait directement le titre de section dans le registre `\rightmark`. Imaginons maintenant que TEX fasse une coupure telle que le page de gauche se termine par le dernier paragraphe de la section « Abeilles » et la page de droite commence par le titre de la section « Guêpes ».

Suivons l'algorithme de coupure de page : TEX ajoute successivement des lignes à la page de gauche, calcule la hauteur obtenue, ceci tant que cette hauteur est inférieure à la hauteur du mirroir `\textheight`. Dans la situation envisagée, la coupure va se faire quand TEX aura lu le titre « Guêpes » et constaté qu'il ne rentre pas. Le fait qu'il ait lu ce titre fait qu'il a mis « Guêpes » dans le registre `\rightmark`. Le processus de coupure de page se déclanche : TEX met de côté le titre de la section « Guêpes » (pour le placer au début de la page de droite) et éjecte la page de gauche se terminant par le dernier paragraphe de la section « Abeilles » avec un haut de page constitué du contenu du registre `\rightmarck` qui est « Guêpes » !

Ce résultat est aberrant ; il a tout de même permis d'exposer simplement la raison pour laquelle il a fallu imaginer tout une machinerie (dépassant le niveau du présent

document) qui assure que les registres `\rightmark` et `\leftmark` ne sont modifiés qu'après que le processus d'éjection de la page courante ait eu lieu, ce qui n'est pas le cas de la situation simpliste imaginée.

Exercices

Espacements prédéfinis

Faire des essais avec `\smallskip`, `\medskip` `\bigskip`.

Coupure de ligne et de page

Faire des tests pour bien voir (et différencier) l'effet des commandes `\par`, `\newline` et `\linebreak` puis `\newpage` et `\pagebreak`.

Styles de pages à la demande

Ecrire les macros pour avoir le titre de l'ouvrage en haut à gauche, le titre de chapitre en haut à droite et le titre de section au centre en bas sur toutes les pages.

Réflexion

On peut échanger `\rightmark` et `\leftmark` en opérant comme suit ;

```
\let\Rightmark=\rightmark
\let\Leftmark=\leftmark
\renewcommand{\rightmark}{\Leftmark}
\renewcommand{\leftmark}{\Rightmark}
```

Expliquer le fonctionnement du groupe de commandes ci-dessus.

Chapitre 6

Structuration d'un document

Ce chapitre est consacré aux commandes de structuration d'un document du type « livre niveau grandes classes, université, recherche » (parties, chapitres, sections, etc.) mais aussi aux commandes relatives aux structures annexes (table des matières, index, bibliographie, etc.). L'exposé correspond à la classe de document `book` ; cependant, on donnera pour certaines commandes quelques indications relatives à leur utilisation éventuelle dans le cas de quelques autres classes de document.

6.1 Pages de titre et de début

Cette partie du livre correspond à la partie du fichier « maître » débutant par la commande `\frontmatter`. Traditionnellement elle a une numérotation en chiffres romains minuscules et est constituée comme suit :

- pages de garde (pages 1 et 2),
- page de petit titre et page blanche (pages 3 et 4),
- page de titre et page réservée à des indications générales (autres ouvrages de l'auteur, série de laquelle l'ouvrage fait partie, numéro de l'édition, adresse de l'éditeur, copyright, etc. (pages 5 et 6),
- page de dédicace et page blanche (pages 7 et 8),
- ensuite préface, introduction, etc,
- enfin table des matières, voir la section 6.4.

6.2 Corps du document

Cette partie du livre correspond à la partie du fichier « maître » débutant par la macro `\mainmatter` (numérotation en chiffre arabes recommençant à 1 et continuant jusqu'à la fin de l'ouvrage).

Dans cette section, mais aussi dans la suite, on va introduire les commandes LaTeX mais aussi des commandes du style `poly.sty` qui seront repérées par le sigle POLY. Il est entendu que le style `poly.sty` modifie le résultat de certaines commandes LaTeX mais sans en modifier leur syntaxe (ce qui fait que cela n'entraîne aucun changement pour l'utilisateur).

Les commandes suivantes sont valables pour les classes `book`, `report` et `article` (sauf `\part` et `\chapter` bien entendu).

`\part{Titre de partie}` :
fait un titre de partie numérotée (sur une page impaire entière), voir page 1, et apparaissant dans la table des matières ;

`\partsn{Titre de partie}` :
POLY, fait un titre de partie non numérotée (sur une page impaire entière) apparaissant dans la table des matières ;

`\chapter{Titre de chapitre}` :
fait un titre de chapitre numéroté (sur une nouvelle page impaire), voir page 35, et apparaissant dans la table des matières ; cette commande n'existe pas pour la classe `article` ;

`\section{Titre de section}` :
fait un titre de section numérotée apparaissant dans la table des matières, voir ci-dessus le titre de la présente section ;

`\sectionsn{Titre de section}` :
POLY, fait un titre de section non numérotée mais apparaissant dans la table des matières ; bien entendu, l'absence de numérotation implique une répercussion sur certaines références croisées (l'utilisation de cette commande, intéressante dans plusieurs cas, demande une attention particulière) ; elle peut être utilisée pour une biblographie par chapitre, une liste d'exercices, etc. ;

`\subsection{Titre de sous-section}` :
fait un titre de sous-section numérotée apparaissant dans la table des matières ;

On trouve ensuite :

`\subsubsection{...}`, `\paragraph{...}` et `\subparagraph{...}`.

d'effet évident et d'utilisation à éviter pour ne pas arriver à une trop grande complexité de structure. Ces niveaux ne sont pas numérotés car, par défaut, le compteur `secnumdepth` qui limite le niveau de numérotation vaut 2 (pour la classe `book`, les niveaux sont : -1 pour partie, 0 pour chapitre, 1 pour section, 2 pour sous-section, etc.). Le style POLY ne prévoit que l'utilisation de la commande `\paragraph` utile pour clarifier l'exposé en séparant des points différents sans pour cela introduire une nouvelle numérotation avec la commande `\subsubsection`.

La syntaxe complète de ces commandes est la suivante (explication sur le cas de la commande `\section`) :

`\section*[titre]{Titre}`

`Titre` est le titre qui sera composé, `titre` est le titre qui apparaîtra dans la table des matières (titre abrégé par exemple) et l'$*$ empêche la numérotation mais aussi la mise dans la table des matières (d'où les macros spécifiques POLY telles que `sectionsn`).

Le style POLY permet d'introduire la commande `\\` dans l'argument des commandes `\part`, `\chapter`, `\section` et `\subsection` (sans qu'elle se retrouve dans les hauts de page ou dans les entrées de la table de matières).

Les compteurs impliqués dans la numérotation automatique sont `part` (non utilisé par défaut excepté pour la numérotation sur la page de titre de partie), `chapter`, `section` et `subsection`, les autres ne jouent pas de rôle si `secnumdepth` vaut 2. Par défaut, `chapter` continue à progresser aux changements de parties (on peut facilement imposer la mise à zéro à chaque partie mais il faut faire un peu de programmation pour garder le bon fonctionnement des références croisés) ; `section` est remis à zéro en début de chapitre et `subsection` est remis à zéro en début de section.

Les numérotations se font par des commandes telles que :

```
\newcommand{\thechapter}{\arabic{chapter}}
\newcommand{\thesection}{\thechapter.\arabic{section}}
\newcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

que l'on peut varier avec les commandes `\arabic`, `\roman`, `\Roman`, `\alph` et `\Alph` de significations évidentes : chiffres arabes, romains minuscules, romains majuscules, lettres minuscules et lettres majuscules.

Le compteur `equation` est remis à zéro en début de chapitre et la numérotation des formules se fait par une commande de type :

```
\newcommand{\theequation}{\thechapter.\arabic{equation}}
```

et peut être modifiée également.

On peut aussi modifier les valeurs de ces compteurs par les commandes :

```
\setcounter{nom-du-compteur}{nombtre-voulu}
\addtocounter{nom-du-compteur}{nombre-a-ajouter}
```

Il y a aussi la possibilité d'avoir des annexes : il suffit de placer la commande

```
\appendix
```

et de continuer avec la macro `\chapter`. Automatiquement le mot chapitre dans les pages de titre est remplacé par le mot annexe et la numérotation se fait en lettres majuscules A, B, etc. :

« Chapitre 1 » est remplacé par « Annexe A ».

6.3 Pages de fin

Les structures annexes que nous allons examiner sont situées dans la partie de l'ouvrage que l'on appelle pages de fin par opposition à la partie pages de début. Dans le fichier maître, elles sont précédées de la commande `\backmatter` ; on peut y trouver les listes suivantes (en fonction du type d'ouvrage) :

- liste des figures,
- liste des tableaux,
- liste des notations,
- liste des abréviations,
- liste des exemples,

- liste des théorèmes et liste des définitions (livres de mathématiques).

  Par contre, il y a toujours les deux structures suivantes
- bibliographie, voir la section 6.6.
- index (et parfois index des auteurs citée), voir la section 12.

  Les structures les plus utilisées font l'objet des sections suivantes.

6.4 Table des matières

La commande

`\tableofcontents`

crée le fichier de table des matières `livre.toc` à la première passe et compose la table des matières à la deuxième passe, voir le fichier maître page 20.

Le compteur `tocdepth` donne le niveau de structure le plus profond figurant dans la table des matières ; par défaut ce compteur vaut 2 : il n'y a dans la table des matières que les parties, les chapitres, les sections et les sous-sections. Ce compteur est indépendant du compteur `secnumdepth` défini plus haut, page 36 (ils peuvent avoir des valeurs différentes).

On peut ajouter une commande ou une entrée « à la main » dans la table des matières en ajoutant des commandes du type

```
\addtocontents{toc}{\vspace[2mm]} espace supplémentaire
\addcontentsline{toc}{\subsection}{Entrée supplémentaire}
```

dans le corps du document, à l'endroit voulu. « Entrée supplémentaire... » figurera (avec la pagination correspondante) dans la table des matières avec les caractéristiques des sous-sections. Ceci est particulièrement utile, par exemple pour signaler dans la table des matières quelques illustrations importantes dans le cas où on ne crée pas de liste des figures (cas très fréquent).

Il ne faut pas oublier que toute addition ou amélioration directe faite dans le fichier `livre.toc` est perdue puisque, à la passe suivante, ce fichier sera créé à nouveau ! C'est toujours quand tout est presque fini que l'on constate qu'il faut améliorer la table des matières (coupures intelligentes pour les titres longs par exemple) et faire des corrections dans le corps du document ; le style POLY propose la solution qui consiste à « couper » la macro `\tableofcontents` en deux parties : `\maketab` qui produit le fichier `livre.toc` et `\printtab` qui compose le fichier `livre.tab` s'il existe. Après la compilation que l'on pense être définitive, on copie `livre.toc` en `livre.tab` où l'on peut faire des modifications qui ne seront pas écrasées lors des ultimes traitements (en contre partie, il faut s'assurer que les corrections ne changent pas la pagination).

6.5 Liste des figures et liste des tableaux

Ces listes sont créées et composées exactement comme la table des matières (deux passes nécessaires) avec les commandes :

`\listoffigures` et `\listoftables`

Ces listes ne comportent que les figures ou les tableaux intégrés par l'intermédiaire des environnements `figure` ou `table` qui seront abordés plus loin. On peut aussi y ajouter une commande ou du texte avec les commandes `\addtocontents` ou `\addtocontentsline` où l'argument `toc` est remplacé par `lof` ou `lot`.

6.6 Bibliographie

On expose ici la bibliographie « à la main » utilisée en débutant. La bibliographie à partir des bases de données sera exposée dans la partie II.C. Les citations sont faites avec la commande

```
\cite{xx,yy,...}
```

où `xx`, `yy`... sont des clés, séparées par ses virgules, associées aux références. Ces références sont rassemblées, par ordre de citation, avec l'environnement spécifique `thebibliography` dans un fichier `livre.bbl` du type :

```
\begin{thebibliography}{nn}

\bibitem{xx}
Donald E. \textsc{knuth},
\textit{The \TeX book},
Addison-Wesley, Reading, 1986.

\bibitem{yy}
Franck \textsc{Mittelbach} and Rainer \textsc{Sch\"opf},
\textit{A new font selection scheme for \TeX},
TUGboat, \textbf{10} (2), 222--238, 1989.

...

\end{thebibliography}
```

Le paramètre `nn` de l'environnement est un chiffre s'il y a au plus 9 références, deux chiffres s'il y a au plus 99 références, etc. (il s'agit de permettre la détermination d'une largeur de retrait). La bibliographie est composée à l'endroit où l'on fait entrer ce fichier, généralement en fin d'ouvrage (cf. le fichier maître page 21) avant l'index. L'environnement `thebibliography` fait une page de titre automatiquement en utilisant la commande `\chapter`. Ceci concerne la bibliographie générale en fin d'ouvrage mais le style POLY définit l'environnement `thebibliographypc` qui, avec des fichiers de données bibliographiques similaires, permet de composer des bibliographies en fin de chapitre ; il suffit pour cela de faire entrer ces fichiers à la fin des chapitres concernés. Le titre de ces bibliographies de chapitre est un titre de niveau section qui se fait automatiquement par appel de la commande `\sectionsn`.

6.7 Index

La génération de l'index est lancée par la commande `\makeindex` de l'extention `makeidex` placée en début du fichier maître, voir page 20. Le fichier `livre.idx` ainsi créé à la première passe n'est pas classé. On obtient le fichier classé `livre.ind` en lançant un utilitaire fourni avec la distribution TEX par la commande

```
makeindex -o livre.ind -s ppur.ist livre.idx
```

où `ppur.ist` est un fichier de formatage de l'index qui donne un index sous une forme sobre, claire et traditionnelle.

A la passe suivante, l'index sera imprimé à l'endroit du document où l'on a placé la commande `\printindex`, voir le fichier maître, page 21.

Différentes possibilités d'indexations

Parmi toutes les possibilités, on ne va exposer que les plus utiles :

`\index{mot}` :
: place *mot* dans l'index ;

`\index{mot|textbf}` :
: place *mot* dans l'index et met en gras le numéro de page ;

`\index{mott@mot}` :
: place *mot* dans l'index mais le classement se fera avec *mott* ; ceci est particulièrement utile dans certains cas, par exemple :
`\index{alpha@$\alpha$}` fait que α sera classé à la lettre A ;

`\index{mot@\textbf{mot}}` :
: place *mot* et le met en gras ;

`\index{mot|(}` et `\index{mot|)}` :
: place *mot* dans l'index avec pour pagination l'intervalle allant de la pagination de la première commande à la pagination de la deuxième commande (toutes les indexations intermédiaires de *mot* ne seront pas prises en compte).

Il est recommandé de saisir la commande `\index{...}` immédiatement après la saisie du mot pour éviter d'avoir des paginations erronnées :

```
Voilà comment on indexe un mot\index{mot} avec \LaTeX
```

Entrées hiérarchisées

Pour expliquer simplement cette possibilité considérons le cas où, dans un document, on a la suite d'indexations pour les mots et/ou groupes de mots : groupe de Lie, groupe, groupe d'automorphismes d'évolution, groupe d'automorphismes, groupe d'automorphismes de translation. On va saisir

```
\index{groupe!de Lie}
\index{groupe}
\index{groupe!d'automorphismes!d'évolution}
```

```
\index{groupe!d'automorphismes}
\index{groupe!d'automorphismes!de translation}
```

et on obtiendra la disposition suivante dans l'index

groupe, 56
  – de Lie, 40
  – d'automorphismes, 72
    – d'évolution, 61
    – de translation, 87

Il faut aussi mentionner que l'on peut ajouter des entrées dans le fichier `livre.idx` (sans retraiter l'ensemble) avec la commande `\indexentry{mot}{numero-de-page}` placée n'importe où.

On peut faire une semblable insertion dans le fichier `livre.ind`, toujours en donnant le numéro de page ; par contre, l'insertion doit se faire en respectant l'ordre alphabéthique. On découvre toujours un gros travail d'amélioration à faire : coupures de ligne intelligentes mais surtout corrections d'erreurs dues à des erreurs de syntaxe d'indexation : singulier et pluriel (le pluriel est exceptionnement nécessaire), entrées hiérarchisées illogiques (les problèmes de ce type n'apparaissent qu'après composition)

Remarque sur le classement de l'index

Le classement de l'index fait par l'utilitaire `makeindex` pose des problèmes dans le cas des langues utilisant des lettres accentuées. La tradition typographique pour le français consiste à faire un classement où les accents sont ignorés. Si, dans le fichier source, on a saisi `\index{\'evolution}`, ce mot sera classé en début de l'index car le code latin1 de `\` est 92 et celui de `a` est 97. Si l'on a saisi `\index{évolution}`, ce mot se retrouvera en fin d'index car le code latin1 de `é` est 233 et celui de `z` est 122. La manière sûre d'avoir un résultat parfait, pour le moment, consiste à coder `\index{evol...@\‘evol...}` ou `\index{evol...@évol...}`.

Il y a un utilitaire très puissant, XINDY, qui permet de résoudre tous ces problèmes par le biais de tables d'équivalences : on peut par exemple déclarer dans ce type de table que les chaînes de caractères `\'e`, `\'E`, `é` et `É` sont classées à la position de la chaîne `e`. Hélas, ce programme est d'utilisation assez complexe ; il sera exposé dans la partie II.D.

6.8 Références croisées

Comme LaTeX numérote lui-même les éléments de sectionnement, les équations, les environnements du type `theorem`, les niveaux des énumérations et les notes de bas de page, il faut un système de références n'utilisant pas cette numérotation encore inconnue à la saisie. Pour cela LaTeX utilise un système de labels qui fonctionne de la manière suivante : imaginons une section de titre « Abeilles » qui va devenir la section 4.3 et débuter en page 65. A la saisie on écrit

```
\section{Abeilles}\label{abe}
```

et plus loin, lorsqu'on voudra faire référence à cette section, on écrira :

```
\dots{} comme exposé à la section \ref{abe} en page\pageref{abe} \dots
```

qui donnera, à la deuxième passe, le texte :

... comme exposé à la section 4.3 en page 65 ...

Une fois un label attribué par la commande `\label`,

- `\ref{...}` donne la numérotation de sectionnement (chapitre, section, etc.) ou bien la numérotation des environnements `equation`, `figure`, `table` ou du type `theorem`, ou bien encore les niveaux d'énumération, ou bien enfin le numéro de note de bas de page où se trouve placé le `label{...}`. Les numéros de référence des équations devant être entre parenthèses, `\eqref{...}` rajoute les deux parenthèses.
- `\pageref{...}` donne le numéro de page où se trouve le `label{...}`.

Dans les environnements `figure` et `table`, le label doit être positionné après la commande `\caption`.

6.9 Titre de document

Les commandes de titre qui suivent ne sont pratiquement pas utilisées pour composer le titre d'un document du type livre (classe `book`) sauf éventuellemnet s'il s'agit d'une rédaction provisoire (c'est pour cette raison qu'elles sont ramenées en fin de chapitre). Par contre elles sont fondamentales pour les classes `article` et `report` car le titre n'occupe dans ces cas qu'une partie du début de la première page.

Ces commandes de titre sont les suivantes :

`\title{Titre}` :
: pour donner le texte du titre,

`\author{Auteur, Auteur et Auteur}` :
: pour donner les noms des auteurs,

`\thanks{texte}` :
: commande qui, placée dans l'argument des deux commandes précédentes, permet de mettre des notes de bas de page (utile pour préciser l'organisme des auteurs, remercier pour une aide financière, etc.),

`\date{date}` :
: pour donner la date (l'omission de cette commande provoque automatiquement la capture de la date système),

`\maketitle` :
: produit l'impression du titre.

Pour la classe `book`, les mêmes commandes font un page de titre.

Pour la classe `slides`, on a la même chose que pour la classe `book` sauf que la commande `\thanks` n'est pas définie.

En outre pour la classe `book`, il y a un environnement spécial qui permet de faire une page de titre sans tête ni pied de page et avec mise à zéro du compteur de page :

```
\begin{titlepage}
toutes commandes jugées utiles, y compris l'insertion
d'un logo, pour faire une page de titre personnalisée
\end{titlepage}
```

Exercices

Sectionnement

Faire un fichier contenant une partie, un premier chapitre court et un deuxième chapitre avec sections, sous-sections et paragraphes (il suffit de disposer d'un paragraphe que l'on répète indéfiniment pour constituer des pages terminées par `\newpage` et dans lesquelles on ajoute des sectionnements).

Ecrire les commandes pour avoir le numéro de section sous la forme [A], [B], etc.

Structures annexes

Faire la table des matières ; introduire préalablement une coupure avec la commande `\\` dans un titre de section apparaissant dans un haut de page ; vérifier le résultat dans la table des matières et dans le haut de page.

Faire un fichier de bibliographie (voir l'exemple en section 6.6 page 39). Avec cet unique fichier faire une bibliographie générale de fin d'ouvrage et une bibliographie à la fin du deuxième chapitre ; faire la table des matières et la contrôler.

Introduire des mots dans les pages (les mettre en gras pour les repérer facilement) et les indexer ; faire l'index, le classer et le composer. Tester l'indexation avec la commande `\index{mott@mot}` et faire un exemple d'entrée hiérarchisée.

Références croisées

Tester les macros de références croisées `\label{.}`, `\ref{.}` et `\pageref{.}`.

Chapitre 7

Dispositions du texte

Les dispositions spécifiques (centrage, liste, énumération, note de bas de page, tableau, etc.) se font, pour la quasi-totalité, à l'aide d'un environnement :

```
\begin{nom-de-l'environnement}
... texte ...
\end{nom-de-l'environnement}
```

7.1 Positionnement

Les environnements de position par rapport au miroir sont :

`flushleft`, `flushright` et `center`.,

Le passage à la ligne se fait avec `\\`. Ces trois positionnements peuvent être aussi obtenus avec les commandes

`\raggedleft`, `\raggedright` et `\centering`

si l'on se trouve déjà dans un autre environnement ou dans un `\parbox` (cette dernière commande étant définie à la section 7.7 page 50). On donne deux exemples :

Exemple de l'environnement `center`
pour faire des titres

Environnement `flushright`
pour un nom d'auteur

On dispose encore de deux primitives TEX qui permettent de composer des paragraphes avec des retraits variées ; on écrit par exemple

```
{\leftskip=15mm\rightskip=20mm plus 10mm
... texte ... \par}
```

et l'on obtient :

> Un paragraphe avec un retrait à gauche de 15 mm et un retrait à droite variant entre 20 mm et 30 mm. Il faut qu'il y ait la commande de fin de paragraphe entre le dernier mot du dernier paragraphe et l'accolade fermante du groupement.

On remarque qu'il n'y a pas de césure : elle sont très improbables du fait de la justification en drapeau à droite. Pour éviter avec certitudes les césures il faut écrire

```
\rightskip=20mm plus 1fill
```

où `fill` joue le rôle d'un unité de longueur complètement extensible et compressible. On donne un autre exemple d'utilisation de `fill` : si l'on fait une boîte d'une certaine longueur dans laquelle on veut placer A au premier tiers, on écrit

```
\hbox to 80mm{(\hskip 1fill A\hskip 0pt plus 2fill)}
```

et l'on obtient

(A).

La macro `\hbox to ...` sera vue à la section 7.7, page 50 ; notons que le même résultat peut être obtenu, par définition même de la macro `\hfill`, avec la syntaxe :

```
\hbox to 80mm{(\hfill A\hfill\hfill)}
```

7.2 Mode verbatim

Cet environnement permet de composer du texte comprenant tous les caractères réservés de TEX tout en conservant les espaces et les retours à la ligne ; il utilise la fonte machine à écrire. Il est utilisée pour reproduire des parties de fichiers de commandes TEX dans ce document ; il s'utilise pour la composition des livres d'informatique car l'utilisation de la fonte machine à écrire (rappelant les fontes des consoles) permet de bien différencier les commandes du texte explicatif (le symbole ␣ rend visibles les espaces pour éviter toute confusion ; il s'obtient automatiquement en ajoutant `*` après `verbatim` ou après la commande `\verb` donnée ci-après) ; comme les espaces sont conservés et que la fonte utilisée est à espacement fixe, on peut présenter les algorithmes de façon claire en jouant avec les alignements (c'est comme en mode WYSIWYG !).

Pour reproduire une seule commande ou quelques commandes, on peut utiliser la commande `\verb|\commande{argument}|` qui donne `\commande{argument}`. Malheureusement cette commande ne peut pas être placée dans l'argument d'une autre macro ; pour la macro `\item[...]` utilisée dans ce document pour les listes de commandes de la page 42, il a fallu désactiver l'effet des caractères réservés de TEX en écrivant

```
\item[\string\chapter\string{Titre de chapitre\string}]
```

où la commande `\string` « écrit » explicitement le caractère qui suit ; il faut encore noter que cette méthode ne marcherait pas si les arguments de la macro `\chapter` devait être entre crochets (c'est un problème de TEXperts).

Si cela est nécessaire, cet attirail peut être simplifié par l'introduction de macros bien « pensées ». Il faut signaler que des fichiers de style ont été écrits pour typographier des programmes : pour quelques lignes, on peut se contenter de ce qui est dit ci-dessus ; pour un ouvrage complet, il vaut mieux chercher dans la documentation des packages appropriés ... ou écrire son propre package si l'on en est capable !

7.3 Listes

Environnement `itemize`

Il y a quatre niveaux possibles ; on peut trouver un exemple en page 14. La saisie se fait comme suit :

```
\begin{itemize}

\item texte du premier élément de la liste

\item texte du deuxi\‘{e}me élément de la liste

...

\end{itemize}
```

Les lignes blanches n'ont ici aucun effet ; elles sont utiles pour clarifier le fichier lors des corrections. Les symboles servant à faire ressortir les éléments de la liste sont en général définis par défaut dans les styles de langage utilisés : `english`, `frenchb`, etc. Il peuvent être changés pour tout élément car la commande `\item` admet un paramètre facultatif saisi entre crochets ; par exemple `\item[$\diamondsuit$]` produit le symbole $\diamondsuit$ à la place du symbole $\bullet$.

Le style POLY modifie les différents espacements verticaux et horizontaux et recommande de n'utiliser que deux niveaux, exceptionnellement trois. Les symboles par défaut et dans l'ordre : $\bullet$ - et $\cdot$ (obtenus avec `\Bille` `\Tiret` et `\Point`) et la commande `\labelsitemization{.}{.}{.}` permet d'en changer l'ordre à n'importe quel endroit du document.

Environnement `enumerate`

Cet environnement a aussi quatre niveaux et la saisie se fait exactement comme la saisie pour l'environnement `itemize`. Par défaut et successivement par niveau la numérotation est 1. 2. ..., (a) (b) ..., i. ii. ... et A. B. ... Cette numérotation peut être changée avec quatre commandes dont la première est

```
\renewcommand{\theenumi}{\xxx{enumi}}
```

où `\xxx` est mis pour `\arabic`, `\alph`, `\roman`, `\Alph` ou `\Roman` ; les trois autres commandes se déduisent de celle-là par le simple changement de `i` en `ii`, `iii` et `iv`.

Le style POLY modifie également les espacements et recommande de ne pas utiliser plus de deux niveaux, trois exceptionnellement. La numérotation par défaut est 1) 2) ..., a) b) ... et i) ii) ... Cette numérotation peut ête changée à tout endroit du document par la commande `\labelsenumeration{.}{.}{.}` dont les arguments sont à choisir entre `\arabic` `\alph` `\greek` et `\roman`. Le style POLY propose en plus trois environnements d'énumération avec un seul niveau `enuma`, `enumg` et `enumr` dont les numérotations respectives sont a) b) ..., α) β) ... et i) ii) ... sont également disponibles.

Environnement ***description***

L'environnement `description` est utilisé ici sous une variante `com` pour donner des commandes suivies par leur description ; voilà la saisie de la description de la page 36 (où la macro `\s` est l'bréviation de `\string`) :

```
\begin{com}

\item[\s\part\s{Titre de partie\s}] fait un titre de partie ...

\item[\s\partsn\s{Titre de partie\s}]  \poly, fait un titre de ...

\item[\s\chapter\s{Titre de chapitre\s}] fait un titre de ...

...

\end{com}
```

La seule différence avec l'environnement `description` de LaTeX réside dans le fait que les différentes dimensions ont été adaptées aux styles des environnements précédemment décrits et dans l'utilisation de la fonte machine à écrire pour le titre de l'élément au lieu de la fonte romain gras usuelle.

7.4 Notes de bas de page

Les notes de bas de pages s'obtiennent par la saisie :

```
L'association GUTenberg\footnote{Crée en 1988.} édite une revue ...
```

Il y a des problèmes lorsque l'appel de la note se trouve à l'intéreur de certains environnements : environnement `tabular` par exemple ; alors il faut

- appeler la note à l'endroit voulu avec la commande `\footnotemark`,
- donner le texte de la note avec la commande `\footnotetext{...}` placée juste après cet environnement.

On peut changer la largeur et l'épaisseur du filet séparant le bas du texte des notes de bas de page, l'espace avant ce filet, l'espace après ce filet ainsi que l'espace séparant les notes ; les commandes sont (avec des dimensions types)

```
\renewcommand{\footnoterule}{\vspace*{5pt}
  \rule{50mm}{0.4pt}\vspace*{7pt}}
\setlength{\footnotesep}{3pt}
```

On peut aussi avoir accès au compteur de notes `footnote` ; par défaut, il est remis à zero en début de chapitre. Les notes appelées dans l'environnement `minipage` (cf. section 7.6, page 49) apparaissent au bas de la minipage et avec une numérotation spécifique en lettres minuscules italiques. Le style POLY change légèrement certaines dimensions et place le numéro de note entre parenthèses[(1)].

[(1)] Le numéro de note apparaît entre parenthèses.

7.5 Notes de marge

Les notes de marge se placent avec la commande

```
\marginpar{Texte}
```

Cette commande peut être utilisée pour insérer des petites figures (à condition que ces figures soient de grande qualité et peu lettrées sinon la figure se retrouve recouverte de lettrages et donc pratiquement illisible).

On peut souhaiter que le texte de la note apparaisse différemment suivant qu'il s'agit d'une page paire (à gauche) ou impaire (à droite) ; cela se fait en utilisant le paramètre facultatif de la commande

```
\marginpar[Texte pour côté gauche]{Texte pour côté droit}
```

Le texte peut être le même des deux côtés mais différencié par des commandes de positionnement ; par exemple en drapeau à gauche pour la page de gauche et réciproquement.

7.6 Minipage

Cet environnement produit un bloc de texte ayant les propriétés d'une page (notes de bas de page avec numérotation spécifique, possibilité d'inclure des tableaux, etc.). La saisie se fait comme suit :

```
\begin{minipage}[position]{largeur}

... texte ...

\end{minipage}
```

où `largeur` est une dimension déjà définie ou une dimension exprimée avec une unité et où `position` prend les valeurs suivantes

- `b` : la ligne de base de la dernière ligne de la minipage est alignée sur la ligne courante (si TEX est en mode horizontal, c'est-à-dire en train de composer une ligne),
- `t` : la ligne de base de la première ligne de la minipage est alignée sur la ligne courante,

enfin, si le paramètre facultatif n'est pas donné, alors la minipage est centrée verticalement sur la ligne de base. Voilà une illustration des trois positions possibles

Positionnement obtenu avec le paramètre `b` et `\raggedright`

Positionnement obtenu avec le paramètre `t` et `raggedleft`

Positionnement obtenu sans paramètre et `centering`

montrant en outre l'effet des commandes `\raggedright`, `\raggedleft` et `\centering` correspondant respectivement aux définitions :

`\setlength{\leftskip}{0pt plus 1fil}`, la même avec `\rightskip` et les deux à la fois (cf. section 7.1, page 45).

7.7 Boîtes

On peut faire des blocs de texte de petite longueur avec les commandes suivantes.

`\mbox{quelques mots}` :
pour une longueur de texte inférieure à la longueur de la ligne.

`\makebox[largueur][position]{quelques mots}` :
encore pour une longueur de texte inférieure à la longueur de la ligne ; `largueur` est une dimension déjà définie ou une dimension donnée avec une unité ; `position` peut être `l` : texte positionné à gauche, `r` : texte positionné à droite et `s` : texte « étiré » sur la largeur de la boîte ; si ce paramètre est omis, alors le texte est centré dans la boîte.

`\fbox{...}` :
comme `\mbox{...}` avec en plus un entourement par un filet d'épaisseur `\fboxrule` à la distance `\fboxsep` du texte (ces dimensions sont bien entendu modifiables avec la commande `\setlength{.}{.}`).

`\framebox[...][...]{...}` :
comme `\makebox[.][.]{.}` mais avec un filet comme pour `\fbox{.}`.

`\parbox[position]{largueur}{texte}` :
s'utilise pour des quantités de texte moyennement longues et sans complications telles que tableaux, notes de bas de page, etc. ; les paramètres `position` et `largeur` ont les mêmes rôles que pour l'environnement `minipage`, voir page 49.

`\raisebox{depl}[haut][prof]{\boite}` :
déplace verticalement la boite `\boite` (qui peut être réduite à quelques mots) d'une distance (algébrique) `depl` ; si les paramètres optionnels sont donnés, alors la boite déplacée se comporte comme une boite de hauteur `haut` et de profondeur `prof`.

`\rule[depl]{larg}{haut}` :
trace un rectangle noir de largeur `larg` et de hauteur `haut` : le paramètre optionnel est un déplacement du type de celui de la commande précedente.

Voilà une illustration des deux commandes ci-dessus ;

```
\begin{center}\rule{5mm}{0.3pt}\raisebox{5mm}{Haut}
\rule{5mm}{0.3pt}\raisebox{-3mm}{Bas}\rule{5mm}{0.3pt}
\rule{2mm}{5mm}\rule{5mm}{0.3pt}\rule[-3mm]{2mm}{5mm}
\rule{5mm}{0.3pt}\end{center}
```

Haut

Bas

Boîtes de largeur nulle

Imaginons que nous voulons écrire en début de ligne le prénom suivi du nom de telle manière que le nom débute à 24 mm du début de la ligne ; on code

```
\newcommand{\appel}[2]{\noindent\hspace*{24mm}%
   \makebox[0pt][r]{#1 }#2\par}
```

et cela donne (en fait le prénom et l'espace sortent à gauche de la boîte) :

Pierre Durand

Jean Dumoulin

Boîtes de largeur donnée

La primitive `\hbox to xxx`, où `xxx` est une dimension explicite avec unité ou un registre de dimension deja définie, est souvent utilisée par les TEXpert pour des constructions délicates ; un exemple à été donné en page 46.

Jouer avec les fantômes ou entrer dans le monde des TEXperts

Voici trois boîtes

balle pomme graphe

elles sont différentes en ce qui concerne la hauteur (au-dessus de la ligne de base) et la profondeur (en dessous de la ligne de base) ; on « résoud » cette imperfection avec le fantôme vertical, commande `\vphantom{lg}` abrégée en `\vfan` dans le fichier source, qui est un objet graphique de largeur nulle (donc invisible et ne perturbant pas les positions dans le sens horizontal des autres caractères) et dont la hauteur est celle du l et la profondeur celle du g : les boîtes sont parfaites !

balle pomme graphe

Il y a aussi les commandes `\hphantom` et `\phantom` dont les effets se devinent facilement à partir de leur nom.

7.8 Tabulations

Cet environnement concerne un seul paragraphe ; les taquets posés sont annulés à la fin du paragraphe. Voilà un petit algorithme suivi de son codage :

```
if next-char=i   print lig fi endfi
if next-char=l   print lig fl endfi
if next-char=f   if next-char=i        print lig ffi endfi
                 if-next-char=l        print lig ffl endfi
                 else                  print lig ff+next-char endfi
else             print f+next-char endfi
```

expliquant ce que fait TEX après avoir trouvé la lettre f. La saisie en est la suivante :

```
\begin{tabbing}
if next-char=m \=print f+next-char endfi \=\kill
if next-char=i \>print lig fi endfi\\
if next-char=l \>print lig fl endfi\\
if next-char=f \>if next-char=i \>print lig ffi endfi\\
\>if-next-char=l \>print lig ffl endfi\\
\>else \>print lig ff+next-char endfi\\
else \>print f+next-char endfi
\end{tabbing}
```

La première ligne sert seulement à poser les taquets de tabulation avec la commande `\=` ; on place entre les taquets le texte le plus long qui doit figurer dans la colonne correspondante ; on termine la ligne avec la commade `\kill` (cette ligne ne ne sera pas imprimée). Les lignes suivantes comportent les taquets représentés par la commande `\>` et la fin de ligne est signalée par `\\` comme on l'a déjà vu (tous les taquets après lesquels il n'y a plus de texte peuvent être omis). La seule difficulté est la nécessité de choisir le texte le plus long : ce n'est pas simple avec les fontes à caractères à largeur variable. L'environnement `tabular` qui suit calculera lui-même les largeurs des colonnes.

7.9 Tableaux

Les environnements `tabular`, `tabular*` et `array` ont la même syntaxe de saisie : les deux premiers pour le texte et le dernier pour les mathématiques, ce qui n'empêche pas de mettre des formules entre `$...$` dans le premier cas et du texte dans une `\mbox{...}` dans le second cas. L'environnement `tabular*` permet d'imposer la largeur du tableau en la donnant en paramètre facultatif, c'est-à-dire en première position et entre `[...]`.

La syntaxe de saisie qui suit nécessite l'entrée du package supplémentaire `array` :

```
\begin{tabular}[position]{preambule}

... lignes et filets horizontaux ...

\end{tabular}
```

où `position` est `t` ou `b` ou est absente ; la position est alors gérée comme pour les minipages, voir section 7.6 page 49 ;

et où le préambule est une suite d'éléments suivants

`|` : fait un filet vertical ; `||` fait un filet double ;

`l` : fait une colonne justifiée à gauche ;

`r` : fait une colonne justifiée à droite ;

`c` : fait une colonne centrée ;

`p{largeur}` :
fait une colonne de `\parbox[t]{largeur}{texte}` ; le positionnement vertical est donc du type `t` ;

`b{largeur}` :
fait une colonne de `\parbox[b]{largeur}{texte}` le positionnement vertical est donc du type `b` ;

`m{largeur}` :
fait une colonne de `\parbox{largeur}{texte}` ; le positionnement vertical est donc du type centré ;

`@{xxx}` : supprime l'espace intercolonne et insère à la place la chaîne de caractères `xxx` ; cette commande permet d'aligner des colonnes de nombres décimaux (en prenant `,` pour chaîne `xxx` et en plaçant les parties entières en colonne du type `r` à gauche et les parties décimales en colonne du type `l` à droite) mais aussi des colonnes de nombres en notation exponentielle tels que $1{,}45 \times 10^{15}$ avec alignement sur le signe $\times$;

`>{xxx}` : insère la chaîne `xxx` (qui peut être du texte ou une commande) devant les arguments de la colonne qui suit ; elle doit être placée dans le préambule devant `l`, `r`, `c`, `p{.}`, `b{.}` et `m{.}` ;

`<{xxx}` : insère la chaîne `xxx` après les arguments de la colonne qui précède (ne concerne que les six types de colonnes cités ci-dessus) ; si une colonne centrée d'un tableau fait avec `tabular` ne contient que des éléments en mode mathématique, on écrira dans le préambule `>{$}c<{$}` qui évitera de placer tous les `$` nécessaires (mais il y a bien d'autres applications possibles).

Les « commandes de ligne » sont les suivantes :

`texte & texte & texte & ... & texte \\` :
est la saisie d'une ligne où `&` est le séparateur de colonnes et où `texte` peut être des mots, des boîtes, des `\parbox`, des formules mathématiques entre `$... $`, etc. ; `\\` indique la fin de ligne comme d'habitude ;

`\hline` : fait un filet horizontal ; pour un double filet, il suffit de répéter la commande ;

`\cline{m-n}` :
fait un filet horizontal de la colonne `m` à la colonne `n` incluses ;

`\multicolumn{m}{positon}{texte}` :
écrit `texte` (qui peut être des mots, des `\parbox`, etc. comme déjà précisé) sur `m` colonnes à partir de la colonne où l'on place la commande ; `position` peut être `l` (positionnement à gauche sur la largeur des `m` colonnes), `r` (positionnement à droite), `c` (position centrée) ;

`\ctext{m-n}{positon}{texte}` :
est une commande POLY qui écrit du texte à la place d'un filet ; elle est décrite dans le dernier exercice du présent chapitre.

Remarques

L'écriture avec la commande `\multicolumn{.}{.}{.}` sur `m` colonnes supprime `m-1` caractères de séparation `&` comme on le voit sur les deux lignes de saisie suivantes :

```
\begin{tabular}{|l|c|c|c|c|c|r|}
texte & texte & texte & texte & texte & texte & texte \\
\multicolumn{4}{|c|}{texte} & texte & texte & texte
\end{tabular}
```

Cette même commande avec `m=1` sert à changer exceptionnellement le positionnement d'un argument de colonne comme ci-dessous où `texte` sera centré bien que placé dans une colonne justifiée à gauche dans le préambule :

```
\begin{tabular}{|r|l|c|}
texte & texte & texte \\
texte & \multicolumn{1}{c|}{TEXTE} & texte
\end{tabular}
```

On remarque encore sur les exemples ci-dessus que, s'il y a des filets en bordure de la multicolonne, il faut les rappeler en les plaçant dans le paramètre `position` de la commande en respectant la règle suivante : si la multicolonne touche le côté gauche du tableau, il faut reproduire le filet de gauche et de droite ; sinon on ne reproduit que le filet de droite (ceci avec la convention que « reproduire le filet » signifie composer le filet s'il est demandé dans le préambule).

Réglage des tableaux

`\renewcommand{\arraystretch}{x}` :
: permet de modifier l'écartement des lignes : `\arraystretch` joue dans les environnements du type `tabular` le rôle de `\baselinestretch` pour le texte, voir page 28 ; le coefficient `x`, qui vaut 1 par défaut, peut être pris légèrement inférieur à 1 ;

`\addtolength{\extrarowheight}{dimension}` :
: permet d'augmenter la hauteur des lignes (la profondeur n'étant pas modifiée) ; cette commande est utilisée pour « fignoler » des tableaux contenant beaucoup de lettres capitales (qui, sans cette correction, semblent toucher le filet du dessus de la ligne) ;

`\addtolength{\tabcolsep}{dimension}` :
: ajoute `dimension` (dimension donnée avec unité et pouvant être négative) à la demi-largeur de l'espace horizontal séparant deux colonnes voisines dans l'environnement `tabular` ;

`\addtolength{\arraycolsep}{dimension}` :
: commande identique à la précédente pour l'environnement `array` ;

`\small` :
: permet de passer à un corps plus petit, voir section 4.2 page 24.

Il faut prendre garde de placer ces commandes avant d'ouvrir les environnements `tabular`, etc. et d'enfermer le tout dans un groupement `{...}`.

Exercices

Environnements, listes et boites

Tester les environnements de position et `\verbatim`,puis la commande `\verb{...}`.

Composer un exemple de chaque environnement `itemize` et `enumerate` en se limitant à deux niveaux ; regarder la macro `\greek` dans le fichier `poly.sty` et en déduire une macro qui numérote les éléments d'une liste avec +, ++, et +++ (se limiter à trois niveaux).

Introduire quelques notes de bas de page.

Composer des exemples de minipages, boîtes et `\parbox`.

Tableaux : exercice important

Faire le tableau suivant ; la commande POLY `\ctext{m-n}{position}{texte}` écrit, à la place d'un filet partiel horizontal que l'on ferait avec la commande `\cline{m-n}`, `texte` de la colonne `m` à la colonne `n`, à gauche, à droite ou au milieu suivant que `position` vaut `l`, `r` ou `c`.

Nom et prénom		Âge	Taille (en m)	Lieu des études	
				secondaires	supérieures
Durand	Martin	25	1,78	Revel	Toulouse
Dumoulin	Sophie	102	1,8	Limoux	Carcassonne

Les tableaux composées avec LaTeX méritent une correction. Pour découvir le défaut, faire une `mbox{...}` centrée avec l'environnement `center` et contenant 3 fois le même tableau (tableau de gauche ci-dessous). Dans ce tableau, il faut mettre `\displaystyle` dès l'entrée en mode mathématique (après le premier `$`) sinon les intégrales seraient en mode texte. Rajouter la commande

```
\setlength{\extrarowheight}{1.5pt}
```

avant la deuxième copie et remarquer comment le haut des capitales s'en trouve décollé du filet supérieur. Dans la troisième copie, placer le code de l'intégrale en paramètre de la macro POLY `\gabarit` : cette macro place la formule dans un boite test puis en mesure la hauteur et la profondeur ; ces deux quantités sont augmentées de 3pt et un fantôme vertical est formé avec ces deux dimensions ; ensuite la formule est posée précédée du fantôme ; remarquer alors que les hauts et bas de l'intégrale sont alors bien décolés des filets.

Objet	Formule
Intégrale	$\int_p^k f(s)\ \mathrm{d}x$

Objet	Formule
Intégrale	$\int_p^k f(s)\ \mathrm{d}x$

Objet	Formule
Intégrale	$\int_p^k f(s)\ \mathrm{d}x$

Faire un tableau utilisant les commandes du type `\m{...}`.

Partie I.B

Composer des mathématiques avec LaTeX

Chapitre 8

Mode mathématique

Pour composer des formules de mathématiques, TEX doit être en mode dit mathématique, mode dans lequel il gère automatiquement les espaces et les positionnements des symboles les uns par rapport aux autres.

8.1 Différents modes mathématiques

8.1.1 Mode « en ligne »

Ce mode est celui utilisé pour écrire une petite formule de mathématiques dans le texte, par exemple :

soit f la fonction $C^{\inf}$ décrivant

dont la saisie est :

```
soit $f$ la fonction $C^{\inf}$ décrivant
```

On entre et sort du mode en ligne avec

`$` et `$`, `\(` et `\)` ou `\begin{math}` et `\end{math}`

L'espace avant l'entrée et l'espace après la sortie du mode mathématique en ligne est le même que l'espace intermot (un blanc, plusieurs blancs ou un retour à la ligne). Les espaces et les positions des caractères à l'intérieur des modes mathématiques (mode en ligne et mode déployé présenté dans la section suivante) sont gérées par le programme : les blancs saisis ne sont pas pris en compte excepté dans le cas où ils jouent le rôle de fin de commande (en conséquence, on les utilise très souvent pour « aérer » la saisie afin de retrouver facilement les erreurs). Voici un exemple suivi des explications correspondantes : $\alpha b\gamma\delta\mathbb{C}\mathcal{D}\boldsymbol{\varepsilon}$ est saisi :

```
$\alpha b\gamma\delta\mathbb{C}\mathcal D\boldsymbol\varepsilon$
```

Dans cet exemple, il faut un blanc (un au moins) entre `\alpha` et `b` et entre la commande `\mathcal` et son argument `D` ; dans tous les autres cas, ils ne sont pas nécessaires du point de vue syntaxique (il est commode d'en utiliser comme expliqué ci-dessus) ; si l'on ne met pas ces blancs obligatoires, on aura, à la compilation, des remontrances telles que :

`Undefined control sequence \alphab` ou `Und... cont... seq... \mathbbC`

Dans cet exmple, on a utilié la possibilité suivante : l'argument d'une commande doit être donné entre accolades mais, si l'argument ne contient qu'un seul caractère,

alors les accolades ne sont plus nécessaires (cas de `\mathcal D`). On remarque ainsi que, dans tous les cas de l'exemple, la commande est bien délimitée à droite par un caractère de catégorie autre que lettre, c'est-à-dire de `\catcode` différent de 11 (cf. le tableau 2.1, page 11).

Les précisions ci-dessus concernant les espaces sont valables pour tous les modes mathématiques (elles ne seront pas répétées). Par contre, on verra à la prochaine section comment on peut ajouter en mode mathématique de très petits espaces supplémentaires pour séparer des caractères ou des grands espaces pour bien différencier les éléments d'une même formule.

8.1.2 Mode déployé

C'est le mode utilisé pour les formules importantes par leur rôle dans l'exposé et/ou par leur complexité et leur « encombrement » ; par exemple

$$f(a,b) = \int_a^b f(x)\,\mathrm{d}x$$

dont la saisie est :

```
\og encombrement \fg{} ; par exemple
$$
f(a,b) = \int_{a}^{b} f(x) \dr x
$$
dont la saisie est :
```

avec la définition `\def\dr{\,\mathrm{d}}`.

On entre et sort du mode déployé de plusieurs manières :

Mode déployé sans numérotation (commande TEX)

L'entrée et la sortie de ce mode se fait avec

`$$` et `$$`, `\[` et `\]` ou `\begin{displaymath}` et `\end{displaymath}`

c'est le cas de l'exemple donné ci-dessus.

Mode déployé avec numérotation automatique (commande LATEX)

L'entrée et la sortie de ce mode se fait avec

`\begin{equation}` et `\end{equation}`

L'exemple ci-dessus devient ainsi

$$f(a,b) = \int_a^b f(x)\,\mathrm{d}x \tag{8.1}$$

dont la saisie est :

```
\begin{equation}
f(a,b) = \int_{a}^{b} f(x) \dr x
\end{equation}
```

La numérotation automatique peut être supprimée par le simple ajout de `*` dans les commandes d'entrée et de sortie : `\begin{equation*}` et `\end{equation*}`. Elle se fait par défaut par chapitre sous la forme (1.2), (1.3) ... (2.1), (2.2) ... etc. mais on peut mettre en œuvre une numérotation par section avec les commandes :

```
\numberwithin{equation}{section}
\renewcommand{\theequation}{\thesection.\arabic{equation}}
```

et la numérotation devient : (1.1.1), (1.1.2) ... (1.2.1), (1.2.2) ... etc. La première commande remet à zéro le compteur `equation` au changement de section et recouvre la numérotation par chapitre habituellement utilisée par défaut. La deuxième fixe la manière dont la numérotation sera affichée.

Mode déployé avec alignement (commande TEX, à éviter)

L'entrée et la sortie de ce mode se font avec :

`\begin{eqnarray}` et `\end{eqnarray}`

Il s'utilise pour une suite d'équations avec alignements verticaux ou pour une très longue équation que l'on doit couper. Cet environnement utilise la syntaxe de l'environnement tabular ; la numérotation est automatique pour chaque ligne ; si on ne veut pas une numérotation, on met `\nonumber` en fin de ligne ; si on ne veut aucune numérotation, on ajoute `*` après `eqnarray` comme dans le mode précédent.

Voici un exemple d'équations alignées où l'on remarque le grave défaut de cette macro : les espacements des deux côtés des signes = sont différents dans l'égalité $M = N$ et dans $N =$ (alignement) :

$$\begin{aligned} M = N &= +A_1 + A_2 + A_3 + A_4 + A_5 && (8.2)\\ &= B_1 + B_2 + B_3 + \ldots + B_7 + B_8 + B_9 && (8.3) \end{aligned}$$

Voici une équation avec alignements et coupures ... et avec le défaut signalé :

$$\begin{aligned} A_1 + A_2 + A_3 + \ldots + A_7 + A_8 + A_9 &= \\ &= K(x^2 + y^2) + B_1 + B_2 + B_3 + B_4 + B_5 \\ &\quad + (C_1 + C_2 + C_3 + \ldots + C_7 + C_8 + C_9) \times \\ &\quad\quad \times (C_1 + C_2 + C_3 + \ldots + C_7 + C_8 + C_9) \\ &\quad + D_1 + D_2 + D_3 + \ldots + D_7 + D_8 + D_9 \\ &= P + Q + 2S && (8.4) \end{aligned}$$

où l'on remarque les divers décalages des sommes et des produits aux coupures qui seront expliqués par le suite. La saisie de ces deux blocs d'équtions est :

```
\def\SUM#1{#1_{1}+#1_{2}+#1_{3}+#1_{4}+#1_{5}+#1_{5}}
\def\SUMM#1{#1_{1}+#1_{2}+#1_{3}+...+#1_{7}+#1_{8}+#1_{9}}
\begin{eqnarray} % premier bloc
  M = N & =& \SUM B \\
        & = & \SUM C + D
\end{eqnarray}
```

```
\begin{eqnarray} % deuxieme bloc
   \lefteqn{\SUMM A = } \quad\quad\quad\quad\nonumber \\
        & =& K(x^2+y^2) + \SUM B \nonumber \\
        &  & \quad {} + ( SUMM C) \times  \nonumber\\
        &  & \quad \quad {}\times( \SUMM D) \nonumber \\
        &  & \quad {} + \SUMM D \nonumber \\
        & =& P + Q + 2 S
\end{eqnarray}
```

Dans le dernier exemple, on a utilisé, pour améliorer la lisibilité de la formule, la commande `\quad` qui fait un espace, appelé cadratin, de la largeur de la lettre m de la fonte courante : décalage d'un cadratin pour les termes de la somme et de deux cadratins pour les facteurs internes d'un même terme de la somme. Ces conseils pour améliorer la lisibilité des formules seront aussi utilisés avec les autres environnements d'alignement que l'on va exposer au chapitre 10. La commande `\lefteqn{...}` placée en première colonne permet à l'argument qu'elle contient de dépasser sur les deuxième et troisième colonnes, afin de diminuer la largeur totale.

Il semble que l'environnement `eqnarray` créant le mode déployé avec alignement a la même syntaxe que l'environnement `tabular` : c'est en partie vrai mais une remarque s'impose : cet environnement entraîne automatiquement l'entrée en mode mathématique, c'est-à-dire que les arguments des alignements sont écrits sans être mis entre `$...$`, ce qui serait nécessaire pour placer une formule dans une case d'un tableau construit avec l'environnement `tabular`.

On dispose encore de l'environnement `array` qui permet de faire des alignements complexes d'éléments de mathématiques. Cet environnement ne fait pas entrer dans le mode mathématique, il doit être utiliseé a l'intérieur du mode mathématique : environnements `$...$`, `\$\$...\$\$`, `equation` ou `eqnarray`. Dans l'exemple suivant, une mactrice est construite avec l'environnement `array` et, parce qu'elle fait partie d'une formule elle-même faisant partie d'un alignemement, elle se trouve à l'intérieur de l'environnement `eqnarray` (voir la saisie à la suite) :

$$\begin{aligned} A &= \left|\begin{array}{cc} a & b \\ c & d \end{array}\right| \\ &= B + 2\alpha C \end{aligned}$$

```
\begin{eqnarray*}
A & = & \left|\begin{array}{cc}
              a & b \\  c & d
        \end{array}\right| \\
& = & B + 2 \alpha C
\end{eqnarray*}
```

Les environnements ci-dessus permettent de réaliser la plupart des formules ; cependant les packages de l'AMS proposent d'autres environnements plus « productifs » et plus rigoureux (cf. le défaut signalé). Cet environnement sera délaissé au profit de l'environnement `align` défini à la page 75 ... et qui sera suivi de quelques autres particulièrement appréciés.

8.2 Espaces en mode mathématique

Les espaces et les positions relatives des caractères les uns par rapport aux autres sont gérés automatiquement par TEX. Les blancs et les retours à la ligne introduits à la saisie sont ignorés (à l'exception des espaces suivant une commande comme expliqué à la section 8.1.1 en page 59).

Dans certaines situations, on a besoin, pour améliorer la lisibilité et l'interprétation des formules, d'introduire des espaces supplémentaires (en plus des espaces prévus par TEX) ;

les commandes sont `\,` pour ajouter un petit espace (moins d'un point)
`\:` pour ajouter un espace moyen
`\;` pour ajouter un grand espace (moins de trois points)
`\!` pour ajouter un petit espace négatif (moins d'un point)

Voici un exemple : sans correction d'abord, avec correction ensuite et sans correction de nouveau

$$\left(\frac{A}{B}\right)^2 \quad \left(\frac{A}{B}\right)^2 \quad \left[\frac{A}{B}\right]^2$$

C'est la « rondeur » de la parenthèse très agrandie qui nécessite un rapprochement de l'exposant avec deux petits espaces négatifs (saisie : `...^{\!\!2}`) ; par contre les crochets carrés ne nécessitent pas cette correction ; il ne faut pas se laisser impressionner par ce type de détail, c'est déjà du domaine des TEXperts chez qui ces corrections (peu souvent nécessaires) sont devenues des réflexes acquis par l'expérience. On dispose ensuite de deux espaces importants le cadratin (largeur de la lettre m de la fonte courante) et le double cadratin, `\quad` et `\qquad` ; ils s'utilisent pour séparer des formules sur la même ligne, voir ci-dessus, et pour séparer les deux parties d'une même formule, par exemple

$$f(x) = g(x) \quad \forall x \in [0, 1]$$

Exercices

Composer de petites formules simples en mode en ligne puis des formules connues plus complexes en mode déployé (revenir sur cet excice quand on aura vu les macros de somme, de fraction, de racine carrée, etc.).

Faire le tableau des coefficients du binôme :

$$\begin{array}{ccccc} 1 & & & & \\ 1 & 1 & & & \\ 1 & 2 & 1 & & \\ 1 & 3 & 3 & 1 & \\ 1 & 4 & 6 & 4 & 1 \end{array}$$

Mettre en œuvre la numérotation des formules à gauche (option `leqno` de l'extension `amsmath`).

Tester les petits espaces en rapprochant un indice à droite d'une lettre et un exposant à gauche d'une lettre, (tester en même temps les grands espaces pour séparer des parties de formules les unes des autres) :

$$B_u \quad F_u \quad F_u \quad {}^uB \quad {}^uB \quad {}^uA \quad {}^uA$$

Remarque

Que l'on se rassure ! Les déplacements des indices et des exposants sont donnés ici à titre d'exercice. Les TEXperts eux-mêmes ne les utilisent que très rarement ; d'une part, les exposants à gauche, qui appellent le plus ce type de corrections sont très rares ; d'autre part, la nécessité de déplacement d'un indice à droite (supérieur ou inférieur) ne se fait sentir que pour certaines lettres indicées, et encore dans le cas où il y aurait dans le voisinage d'autres lettres indicées appelant une comparaison !

Chapitre 9

Caractéristiques du mode mathématique

Ce chapitre est destiné à entrer en profondeur dans le mode mathématique pour que l'essentiel en soit examiné, ceci afin que le lecteur soit en mesure de réaliser les compositons les plus courantes ; le souhait de l'auteur de ces lignes est que le lecteur sache, lorsqu'il est devant une difficulté, ce qu'il doit aller chercher dans les ouvrages de référence (qu'il est inutile et irraisonnable de vouloir réécrire !).

9.1 Polices du mode mathématique

Par défaut les lettres latines majuscules et minuscules et les lettres grecques minuscules sont en italique math ; les chiffres ainsi que les parenthèses, la ponctuation et les capitales grecques sont en romain droit. Les polices disponibles ont déjà été données et nous les rappelons ci-dessous

Table 9.1 Polices du mode mathématique

Saisie	Effet	Fonte et remarques
`\mathrm{A}`	A	romain
`\mathbf{A}`	$\mathbf{A}$	romain gras
`A`	A	italique math (par défaut)
`\boldsymbol{A}`	$\boldsymbol{A}$	italique math gras
`\mathbb{A}`	$\mathbb{A}$	blackboad (ensembles $\mathbb{R}$, $\mathbb{C}$, etc.)
`\mathfrak{A}`	$\mathfrak{A}$	gothique (fraktur)
`\mathcal{A}`	$\mathcal{A}$	calligraphique
`\mathsf{A}`	A	sans sérif
`\mathtt{A}`	$\mathtt{A}$	machine à écrire

Pour les mathématiques, on n'a pas malheureusement de code typographique universellement accepté ; cependant il y a un certain nombre de règles assez généralement respectées ; voici quelques exemples : on désigne par $\boldsymbol{a}$ un vecteur et par

a sa longueur, les différents ensembles de nombres sont représentés par $\mathbb{N}$, $\mathbb{R}$, $\mathbb{C}$ et $\mathbb{Q}$, l'élément différentiel figurant dans une intégrale s'écrit en droit et doit être légèrement séparé de l'intégrant, voir l'intégrale à la section 8.1.2, page 60, etc.

9.2 Styles

Pour une taille donnée choisie (option `12pt` pour le présent document), il y a quatre styles dont les deux premiers ont de nombreuses caractéristiques identiques ; en ce qui concerne les tailles des caractères, on se reportera à la section 4.2 page 24. TEX choisit automatiquement un de ces styles mais ils peuvent être imposés par une commande qui est habituellement utilisée pour les nommer ;

`\textstyle` :
la taille de tous les caractères correspond à la commande `\normalsize` ; il est utilisé par défaut dans le mode en ligne.

`\displaystyle` :
preque tous les caractères ont encore la taille donnée par la commande `\normalsize` ; la différence avec le précéent style est détaillée ci-après ; ce style est utilisé par défaut dans le mode déployé.

Ces deux premiers style diffèrent par des positionnements relatifs de certains caractères les uns par rapport aux autres et par la taille de certains signes mathématiques ; voilà un exemple d'une formule dont la saisie identique donne deux résultats différents ; on obtint succèssivement : en mode ligne $A=\sum_{n=1}^{n=\infty}a_n$ et en mode déployé avec, à la suite, une répétition avec style imposé par la commande `\textstyle` placée devant la deuxième formule :

$$A=\sum_{n=1}^{n=\infty}a_n \qquad \textstyle A=\sum_{n=1}^{n=\infty}a_n \tag{9.1}$$

dont la saisie est :

```
\begin{equation}
A=\sum_{n=1}^{n=\infty}a_{n} \quad \quad
\textstyle A=\sum_{n=1}^{n=\infty}a_{n}
\end{equation}
```

Voilà les deux autres styles pour les exposants et les indices :

`\scriptstyle` :
les caractères sont à la taille `\scriptsize` ; ce style est utilisé par défaut pour composer les exposants et les indices, les numérateurs et dénominateurs de premier niveau des fractions en mode ligne, par exemple $\frac{5}{8}$, et les numérateurs et dénominateurs de deuxième niveau des fractions en mode déployé comme on peut le voir sur la première fraction de la ligne suivante

$$A=\frac{2x+1}{y+\frac{3x}{2y-5}} \qquad A=\frac{2x+1}{y+\dfrac{3x}{2y-5}} \tag{9.2}$$

La saisie de la ligne ci-dessus est :

```
\begin{equation}
A=\frac{2x+1}{y+\frac{3x}{2y-5}} \quad
A=\frac{2x+1}{y+\ds\frac{3x}{2y-5}}
\end{equation}
```

Beaucoup de TEXpert n'aiment pas la composition par défaut et préfèrent, pour une fraction du mode en ligne, l'écriture 5/8 et, pour le mode déployé, la deuxième composition ci-dessus obtenue avec la commande `\ds`, abréviation de `\displaystyle`, placée devant la deuxième fraction du dénominateur ; cette composition est plus agréable visuellement et améliore la lisibilité dans le cas des grandes formules complexes.

`\scriptscriptstyle` :
les caractères sont à la taille `\scriptscriptsize` ; ce style est utilisé pour composer les exposants et les indices d'exposants et d'indices ainsi que pour le fractions à plusieurs niveaux (comme expliqué ci-dessus mais en considérant le niveau de fraction suivant). On a encore la possibilité d'imposer le style `\scriptsize`, commande abrégée en `\sc`, dans une situation où le style par défaut est le style `\scriptscriptsize`.

9.3 Classes d'objets mathématiques

Les objets mathématiques sont répartis en classes correspondant à leur sens mathématique et nécessitant par conséquent des spécificités d'espacement et de positionnement. Bien que du domaine des Grands TEXperts, on peut dire comment cela est géré au plus profond du langage : chaque caractère a un paramètre, le `\mathcode`, qui, en hexadécimal contient, dans l'ordre, un chiffre caractérisant la classe (dont on est en train de parler), un chiffre caractérisant la police (lorsque vous écrivez par exemple la commande `\mathbb` vous forcez le changement de ce chiffre) et deux chiffres donnant la position du caractère dans la fonte ; cette explication nous permettra de comprendre comment un unique caractère peut avoir deux fonctions distinctes. Ces classes sont les suivantes.

Symboles ordinaires - classe 0 :
Ce sont les lettres latines et grecques ainsi que les chiffres mais aussi de très nombreux symboles :

$$a\ b\ \ldots\ A\ B\ \ldots\ \alpha\ \beta\ \ldots\ \Gamma\Delta\ \ldots\ \infty\ \forall\ \exists\ \partial\ \nabla \text{ etc.} \tag{9.3}$$

dont la saisie (sans les commandes d'environnement déployé) est :

```
a\;b\; \ldots\; A\;B\; \ldots\;
\alpha\;\beta\; \ldots \; \Gamma \Delta\; \ldots\;
\infty\; \forall\; \exists\; \partial\; \nabla
```

Grands opérateurs - classe 1 :
Ce sont les opérateurs du type somme et intégrale avec des limites qui

ont, automatiquement comme cela a déjà été montré sur l'équation 9.1, des compositions différentes en mode déployé et en mode en ligne :

$$\sum_{n=1}^{n=\infty} \quad \int_{n=1}^{n=\infty} \quad \text{mode déployé} \tag{9.4}$$

$$\textstyle\sum_{n=1}^{n=\infty} \quad \int_{n=1}^{n=\infty} \quad \text{mode en ligne} \tag{9.5}$$

$$\oint \quad \prod \quad \bigcup \quad \bigcap \quad \bigoplus \quad \bigotimes \quad \text{etc.} \tag{9.6}$$

où la saisie des symboles (avec `\ts` abréviation de `\textstyle`) est :

```
\sum_{n=1}^{n=\infty}\quad \int_{n=1}^{n=\infty}
\ts \sum_{n=1}^{n=\infty}\quad \int_{n=1}^{n=\infty}
\oint\;\prod\;\bigcup\;\bigcap\;\bigoplus\;\bigotimes
```

Il faut aussi noter que l'on peut, en mode déployé et tout en conservant la grande variante de l'opérateur, avoir les limites positionnées comme en mode en ligne avec la commande `\nolimits` placée juste après la commande correspondante de l'opérateur :

$$\sum_{a}^{b} \quad \textstyle\sum_{a}^{b} \tag{9.7}$$

On remarquera que le symbole intégrale est ainsi défini par défaut (très probablement pour un question de place). La commande `\limits` permet de faire l'inverse : $\sum_a^b$ devient $\sum\limits_a^b$ ce qui écarte les lignes !

Opérations binaires - classe 2 :

Ce sont l'addition, la soustraction, les différentes sommes et les différents produits, etc.

$$+ \; - \; \pm \; \oplus \; \cdot \; \times \; * \; \wedge \; \cup \; \otimes \; \text{etc.} \tag{9.8}$$

de saisie :

```
+\; -\; \pm\; \oplus\; \cdot\; \times\;
\ast\; \wedge\; \cup\; \otimes
```

Relations - classe 3 :

Ce sont l'égalité et toutes les relations d'équivalence et d'ordre :

$$= \neq > < \geq \leq \gg \ll \in \notin \subset \not\subset \equiv \not\equiv \sim \propto \perp \; \text{etc.} \tag{9.9}$$

de saisie :

```
=\;\not=\;>\;<\;\geq\;\leq\;\gg\;\ll\;\in\;\notin\;\subset
\not\subset\;\equiv\;\not\equiv\;\sim\;\propto\;\perp
```

ainsi que les nombreuses flèches correspondant aux applications, limites, etc.

$$\rightarrow \longrightarrow \Rightarrow \Longrightarrow \Leftrightarrow \mapsto \uparrow \downarrow \nearrow \searrow \; \text{etc.} \tag{9.10}$$

dont la saisie est :

```
\rightarrow\; \longrightarrow\; \Rightarrow\;
\Longrightarrow\; \Leftrightarrow \;\mapsto \;
\uparrow\; \downarrow\; \nearrow\; \searrow
```

Délimiteurs ouvrants et fermants - classes 4 et 5 :

Ce sont les parenthèses, les accolades, les barres, les doubles barres, les crochets, etc.

$$(.)\quad \{.\}\quad [.]\quad |.|\quad \|.\|\quad \langle.\rangle \tag{9.11}$$

dont la saisie est :

`(.)\;\;\{.\}\;\;[.]\;\;|.|\;\;\|.\|\;\;\langle.\rangle`

caractérisés par l'adaptation, grâce aux commandes `\left` et `\right`, de la grandeur à l'ensemble des objets entourés. Voici trois exemples :

$$\left(\frac{a}{s}\right)\quad \left(\begin{array}{cc}a&b\\c&d\end{array}\right)\quad \left(\begin{array}{ccc}A&B&C\\D&E&F\\G&H&K\end{array}\right) \tag{9.12}$$

qui s'obtient avec la saisie :

```
\begin{equation}
\left(\frac{a}{s}\right)\quad
\left(\begin{array}{cc}a&b\\c&d
\end{array}\right)\quad
\left(\begin{array}{ccc}A&B&C\\D&E&F\\G&H&K
\end{array}\right)
\end{equation}
```

Si l'on souhaite n'avoir qu'un des deux délimiteurs, il faut alors remplacer celui que l'on ne veut pas par un point (qui n'est pas bien sûr imprimé) tout en conservant les commandes `\left` ou `\right` correspondantes.

Signes de ponctuation - classe 6 :

Ce sont les signes de ponctuations habituels avec la virgule et le point-virgule automatiquement suivis d'un petit espace

$$a, b; c \quad 2,2 \quad 2.2 \tag{9.13}$$

Ce petit espace pose le problème de l'utilisation de la virgule pour les nombre décimaux où il ne faut pas d'espace après la virgule en typographie française. On peut changer la virgule de classe, c'est-à-dire remplacer `\mathcode‘\,="613B` par `\mathcode‘\,="013B` (premier chiffre 0 pour la classe ordinaire au lieu de 6 pour la classe ponctuation) ; mais alors on ne pourra plus écrire $\boldsymbol{x} = (x_a, x_b, x_c, x_d)$ où le petit espace après la virgule est nécessaire pour faciliter la lecture des formules. Si on a un document avec beaucoup de nombres décimaux, la solution consiste à configurer la virgule en classe ordinaire avec `\mathcode‘\,="013B` et définir la virgule de ponctuation par `\matchardef\vg="613B`. Dans le cas contraire, on laisse la configuration de la virgule en ponctuation (défaut) et on définit une virgule pour les nombres décimaux `\matchardef\vg="013B`. On peut aussi utiliser les deux définitions en les nommant `\vgp` (p : ponctuation) et `\vgd` (d : décimale). Ces manipulations ne concernent pas la virgule dans le texte.

9.4 Caractères disponibles

Comme déjà annoncé, ce document n'a pas la prétention d'être un livre de référence ni de tenter de les recopier. L'essentiel concernant les principes de base du mode mathématique a été expliqué ; par contre pour avoir une vue complète sur l'ensemble des caractères disponibles, il faut se reporter à la référence [4], tableaux 8.8 à 8.27, aux documents disponibles dans les distributions complètes de TEX ou encore à des documents disponibles sur le Web [8, 9].

Exercices

Le lecteur est invité à composer soigneusement toutes les formules du chapitre.

Prendre les tables de caractères disponibles et tester quelques caractères de chaque classe.

Dans le cas où les virgules décimales sont nombreuses et groupées, vérifier que l'on peut résoudre facilement la dificulté en encadrant cette région comme suit :
`{\virdef ... $0,1$ ... }`
avec la définition :
`\def\virdef{\mathcode'\,="013B"}`.

Chapitre 10

Constructions d'objet mathématiques

Ce chapitre rassemble les constructions de base essentielles pour composer un document mathématique sans complexité extrême. Ces constructions sont directement obtenues avec des commandes primitives du langage TEX ou résultent de macros LATEX de base ou de macros des packages de l'AMS. Et ensuite, pour des travaux spécifiques, il y a des nombreuses extensions spécialisées correspondant soit à des domaines scientifiques précis, soit à des constructions spécifiques complexes ; par exemple, il existe une extension pour faire les formules de chimie et il existe une extension (`longtable`) pour faire des tableaux sur plusieurs pages, etc.

10.1 Indices et exposants

Ils sont obtenus avec les caractères `^` et `_` ; cela a été vu à la section 2.3 concernant les caractères spéciaux, page 12. Si l'exposant a plusieurs caractères et/ou commandes, il faut les inclure dans un groupement ; on donne quelques exemples

$$A^2 \quad A_2 \quad A^\alpha \quad A^{a+c} \quad A_{B^2}^{a+c} \quad {}_3A^4 \quad \left(\left(1+x^2\right)^2+A^2\right)^{\!2} \tag{10.1}$$

dont la saisie est[1] :

```
\begin{equation}
A^2\quad A_2\quad A^\alpha\quad A^{a+c}\quad A_{B^2}^{a+c}
\quad {}_3A^4\quad \left(\left(1+x^2\right)^2+A^2\right)^{\!2}
\end{equation}
```

On remarque sur le cinquième exemple les tailles spécifiques pour les exposants et les indices en `\scriptstyle` et en `\scriptscriptstyle`.

[1] Pour que les saisies figurent le plus près possible des exemples correspondants, et aussi pour ne pas trop allonger le présent document, les saisies présentées le sont d'une manière compacte ce qui est fortement déconseillé ; il y a intérêt à mettre chaque élément de formule sur une nouvelle ligne pour s'assurer facilement de la justesse de la syntaxe et pour y apporter facilement corrections et/ou modifications ; cette remarque vaut pour l'ensemble du document

10.2 Radicaux

On les obtient simplement avec la commande `\sqrt[...]{...}`

$$\sqrt{3} \quad \sqrt{1+x^2} \quad \sqrt{\frac{5x}{1+3x^2}} \quad \sqrt[3/5]{\frac{1+A}{1+2B^2}} \quad \left(\frac{1+A}{1+2B^2}\right)^{5/3} \tag{10.2}$$

obtenue avec la saisie :

```
\begin{equation}
\sqrt 3 \quad \sqrt{1+x^2} \quad \sqrt{\frac{5x}{1+3x^2}}
\quad \sqrt[\ss3/5]{\frac{1+A}{1+2B^2}}\quad
\left({\frac{1+A}{1+2B^2}}\right)^{\!\!5/3}
\end{equation}
```

En général, à partir d'une certaine complexité de l'argument la notation exponentielle est préférable car plus lisible comme on peut le constater sur le dernier exemple composé des deux façons possibles.

10.3 Fractions

On a déjà mentionné à la section 9.2 en page 66 qu'en mode en ligne la saisie `$\frac{8x+5}{c(a+4)}$` donne pour résultat $\frac{8x+5}{c(a+4)}$ où le numérateur et le dénominateur sont en `\scriptstyle` ce qui écarte bien souvent les lignes et produit ainsi un effet très désagréable (nettement visible pour la fraction de la deuxième ligne du présent paragraphe).

Il est souhaitable d'opter pour l'écriture linéaire mais alors il faut être très attentif à l'utilisation des parenthèses (et en particulier ne pas mettre de parenthèse inutile comme on le voit souvent) :

$$(8x+5)/2+a \ \text{ est } \ \frac{8x+5}{2}+a \ \text{ alors que } \ (8x+5)/(2+a) \ \text{ est } \ \frac{8x+5}{2+a}$$

En fait, il suffit de respecter les règles des parenthèses des langages de programmation tels que le Fortran.

A la section 9.2 on a aussi vu que, en mode déployé, ce sont les fractions du deuxième niveau qui ont le numérateur et le dénominateur en `\scriptstyle` ; on a vu alors comment éviter cet effet, ce qui a permis d'expliquer la notion de style ; en fait la macro `\dfrac{.}{.}` donne le résultat souhaité :

$$\frac{\frac{1+x}{c+x}}{A+3B+4x} \text{ devient } \frac{\dfrac{1+x}{c+x}}{A+3B+4x} \text{ mais } \frac{1+x}{c+x}(A+3B+4x)^{-1} \text{ est mieux !}$$

10.4 Fonctions prédéfinies et grands opérateurs

Les fonctions dites prédéfinies sont des objets traditionnellement composées en romain et placées dans la classe des grands opérateurs, ce qui introduit automatiquement un petit espace entre la fonction et son argument, par exemple $\cos\alpha$, $\sin x$, etc. Un grand nombre de telles fonctions sont déjà disponibles avec LaTeX et les extensions de l'AMS :

$$\sin\;\cos\;\tan\;\sinh\ldots\exp\;\log\;\sup\;\inf\;\det\;\dim\;\text{etc.} \tag{10.3}$$

dont la saisie est :

```
\sin\;\cos\;\tan\;\sinh\ldots\exp\;\log\;\sup\;\inf\;\det\;\dim
```

En plus des fonctions prédéfinies et des grands opérateurs disponibles, on peut en créer d'autres facilement ; voici quelques exemples :

$$\sin\alpha \quad \sup_{x\in A} f(x) \quad \operatorname{aire} B \quad \liminf_{x\rightarrow\infty} f(x) \quad \mathop{\mathsf{S}}_{p=1}^{p=\infty} A_p \quad \mathsf{S}_{p=1}^{p=\infty} A_p \quad \mathop{\diamond}_{p=1}^{p=\infty} \mathfrak{A}_p$$

dont la saisie est :

```
\begin{equation*}
\newcommand{\aire}{\operatorname{aire}}
\newcommand{\limdow}{\operatornamewithlimits{lim\,inf}}
\newcommand{\somme}{\operatornamewithlimits{\large\textsf{S}}}
\newcommand{\sommebis}{\operatorname{\large\textsf{S}}}
\newcommand{\produit}{\operatornamewithlimits{\mbox{\LARGE$\diamond$}}}
\sin\alpha \quad\sup_{x\in A}f(x)\quad \aire B\quad
\limdow_{x\rightarrow\infty}f(x)\quad\somme_{p=1}^{p=\infty}A_p
\quad\sommebis_{p=1}^{p=\infty}A_p
\quad\produit_{p=1}^{p=\infty}\mathfrak{A}_p
\end{equation*}
```

Les deux exemples de somme avec le caractère S montre la différence entre les deux primitives `\operatorname` et `operatornamewithlimits`.

10.5 Accents, surlignements et soulignements

Les accents en mode mathématique ont une syntaxe spéciale ; on doit saisir `$\hat{e}$` pour avoir $\hat{e}$. Pour les indices textuels (en romain droit), il faut écrire `$R_\mathrm{\acute{e}lec.}$` pour avoir $R_{\mathrm{\acute{e}lec.}}$ (ne pas saisir `$R_{\mbox{élec.}}$` qui donne $R_{\mbox{élec.}}$ ou alors saisir `$R_{\mbox{\scriptsize élec.}}$` qui donne $R_{\mbox{\scriptsize élec.}}$). La première méthode est la seule correcte car si $R_{\mathrm{\acute{e}lec.}}$ est mis en indice, alors les tailles baissent respectivement « d'un cran » pour donner $t_{R_{\mathrm{\acute{e}lec.}}}$ au lieu de $t_{R_{\mathrm{\acute{e}lec.}}}$. On dispose aussi de la macro `\text{...}` qui résoud facilement cette difficulté des tailles en écrivant `$R_\text{élec}$` et `$t_{R_\text{élec}}$`.

L'autre utilité des accents en mode mathématique réside dans le fait que l'on retrouve sous cette forme les traditionnels signes de signification généralement admise

$$\dot{a}\;\ddot{a}\;\bar{a}\;\vec{a}\;\hat{a}\;\tilde{a}\;\ldots \tag{10.4}$$

dont la saisie est :

```
\begin{equation}
\dot{a}\;\ddot{a}\;\bar{a}\;\vec{a}\;\hat{a}\;\tilde{a}\;\ldots
\end{equation}
```

Les principaux surlignements et soulignements sont :

$$\widehat{abc}\ \widetilde{abc}\ \overrightarrow{abcd}\ \overline{abd}\ \underline{abcd} \tag{10.5}$$

obtenus avec :

```
\begin{equation}
\widehat{abc}\;\widetilde{abc}\;\overrightarrow{abcd}\;
\overline{abd}\;\underline{abcd}
\end{equation}
```

On trouve enfin les accolades horizontales dont l'utilisation est illustrée par l'exemple suivant :

$$E_{\mathrm{tot.}}=\underbrace{E_{\mathrm{cin.}}+E_{\mathrm{pot.\,int}}}_{\text{Énergie interne}}+E_{\mathrm{pot.\,ext.}} \tag{10.6}$$

qui s'obtient avec la saisie :

```
\begin{equation}
E_\mathrm{tot.}=
\underbrace{E_\mathrm{cin.}
 +E_\mathrm{pot.\,int}}_{\mbox{\scriptsize\‘Energie interne}}
+E_\mathrm{pot.\,ext.}
\end{equation}
```

10.6 Alignements de toutes sortes !

Environnement ***array*** *et matrices*

L'environnement `array` sert de base à la composition des matrices mais il est plus avantageux d'utiliser les macros disponibles car elles ne demandent pas de donner de préambule (en conséquence de quoi les colonnes sont nécessairement centrées) ; il nécessite d'être employé en mode mathématique :

$$\left(\begin{array}{cc}A & B \\ C & D\end{array}\right)\quad\begin{matrix}A & B \\ C & D\end{matrix}\quad\begin{pmatrix}A & B \\ C & D\end{pmatrix}\quad\begin{bmatrix}A & B \\ C & D\end{bmatrix}\quad\begin{vmatrix}A & B \\ C & D\end{vmatrix} \tag{10.7}$$

dont la saisie est :

```
\begin{equation}
\left(\begin{array}{cc}A & B \\ C & D\end{array}\right)
\quad\begin{matrix}A & B \\ C & D\end{matrix}
\quad\begin{pmatrix}A & B \\ C & D\end{pmatrix}
\quad\begin{bmatrix}A & B \\ C & D\end{bmatrix}
\quad\begin{vmatrix}A & B \\ C & D\end{vmatrix}
\end{equation}
```

On se rend compte (encore une fois) en comparant le premier et le troisième exemple qu'un espace négatif doit être ajouté lors de l'utilisation de certains délimiteurs.

Pour les grandes matrices que l'on ne peut représenter que partiellement, on dispose des macros `\ldots`, `\vdots` et `\ddots` que l'on utilise de la manière suivante :

$$\begin{pmatrix} A_1^1 & A_1^2 & \dots & A_1^n \\ A_2^1 & A_2^2 & \dots & A_2^n \\ \vdots & \vdots & \ddots & \vdots \\ A_n^1 & A_n^2 & \dots & A_n^n \end{pmatrix} \tag{10.8}$$

Environnement `cases`

Les environnements que l'on va présenter ont une version avec `*` pour supprimer la numérotation mais on peut n'en suprimer que certaines avec la commande `\nonumber` en fin de la ligne correspondante. L'environnement `cases`, qui peut avoir plus de deux lignes, permet d'obtenir :

$$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases} \tag{10.9}$$

avec la saisie :

```
\begin{equation}
f(x)=\begin{cases}0&\text{si }x<0\\1&\text{si }x\geq0\end{cases}
\end{equation}
```

Environnement `align`

On arrive à l'environnement `align` qui ne présente pas l'excès d'espacement des deux côtés du signe `=` de l'environnement `eqnarray` : comparer l'alignement suivant avec les équations 8.2 et 8.3, page 61 :

$$\begin{aligned} A &= B + C \\ X + Y &= Z + T \end{aligned} \tag{10.10}$$

obtenu en écrivant (un seul `&` par ligne au lieu de deux pour l'environnement `eqnarray` qu'il vaut mieux oublier) :

```
\begin{align}
A&=B+C\\X+Y&=Z+T\nonumber
\end{align}
```

Environnement `split`

On a ensuite l'environnement `split` constituant l'amélioration de l'environnement `eqnarray` pour les alignements d'équations avec coupures telles que :

$$\begin{aligned} A_1 + A_2 + A_3 + A_4 + A_5 = (B_1 + B_2 + B_3 + B_4 + B_5) \times \\ \times (C_1 + C_2 + C_3 + C_4 + C_5) \\ + D_1 + D_2 + D_3 + D_4 + D_5 \\ A = Y_1 + Y_2 + Y_3 + Y_4 + Y_5 \\ + Z_1 + Z_2 + Z_3 + Z_4 + Z_5 \end{aligned} \tag{10.11}$$

où la saisie est (noter que l'environnement `split` doit être enveloppé dans un environnement `equation` ou `align`, comme l'environnement `array`) :

```
\begin{align}
\begin{split}
\SUM{A}&=(\SUM{B})\times{}\\
       &\quad\quad\times(\SUM{C})\\
       &\quad{}+\SUM{D}
\end{split}\nonumber\\
\begin{split}A&=\SUM{Y}\\&\quad {}+\SUM{Z}\end{split}
\end{align}
```

On aurait pu obtenir une disposition équivalente avec l'environnement `align` mais on n'aurait pas pu numéroter la deuxième équation.

Environnement ***`aligned`***

L'environnement `aligned` fontionne comme l'environnement `align` mais s'utilise à l'intérieur de l'environnement `equation` : il permet les constructions du type suivant :

$$\left\{\begin{aligned}x&=5+k\\y&=4+3k\end{aligned}\right.\quad\quad\left\{\begin{aligned}u&=5+8k\\v&=+2k\\w&=12k\end{aligned}\right. \tag{10.12}$$

résultant de la saisie :

```
\begin{equation}
\left\{\begin{aligned}x&=5+k\\y&=4+3k
\end{aligned}\right.\quad\quad
\left\{\begin{aligned}u&=5+8k\\v&=+2k\\w&=12k
\end{aligned}\right.
\end{equation}
```

Environnement ***`gather`***

Enfin on va citer l'environnement `gather` qui permet de composer une suite d'équations sans alignement et avec un bon espacement entre les équations (plus petit et visuellement plus agréable que si l'on avait placé une succession d'environnements `equation`) ; par exemple :

$$A_1+A_2+A_3+A_4+A_5=B_1+B_2+B_3+B_4+B_5 \tag{10.13}$$

$$y=c_1+c_2+c_3+c_4+c_5$$

$$\gamma=x+y+z_1+z_2+z_3+z_4+z_5 \tag{10.14}$$

qui se saisit tout simplement :

```
\begin{gather}
\SUM{A}=\SUM{B}\\
y=\SUM{c}\nonumber\\
\gamma=x+y+\SUM{z}
\end{gather}
```

10.7 Texte au milieu des mathématiques

On a déja utilisé la commande `\text{...}` pour composer les indices textuels. Plus généralement elle permet du texte dans une formule ; par exemple :

$$A = B \text{ si et seulemement si } A \geq 0 \tag{10.15}$$

dont la saisie est :

```
\begin{equation}
A=B \text{ si et seulemement si } A\geq 0
\end{equation}
```

On aurait pu ne pas mettre des blancs avant et après le texte dans l'argument de la commande `\text` et placer un `\quad` avant et un `\quad` après la commande.

Pour peaufiner la présentation d'une preuve, on peut avoir besoin d'insérer entre des équations, alignées avec l'environnement `align`, quelques mots ou parfois même une ou deux lignes de texte. Si on termine l'environnement pour placer le texte puis si on ouvre un nouvel environnement `align` on a perdu l'alignement du premier ! La commande `\intertext{...}`, placée après la fin de ligne (c'est-à-dire après `\\`), permet d'insérer une ou deux lignes de texte entre deux équations tout en restant dans l'environnement `align`, et donc, en conservant l'alignement des équations situées avant le texte inséré.

10.8 Constructions diverses

Cette section comprend des constructions particulières, importantes par leur utilité, mais difficiles à ranger dans des classes bien définies.

Flèches extensibles pour limites

Lorsque l'on écrit que $f \to 0$, il est souvent nécessaire de préciser les conditions dans lesquelles cette limite a lieu ; si $f(a, x)$ est une fonction dépendant d'un paramètre a, on peut vouloir écrire :

$$f(a, x) \xrightarrow[a\notin\mathbb N]{x\rightarrow\infty} 0$$

dont la saisie est :

```
$$ f \xrightarrow[a\notin\mathbb N]{x\rightarrow\infty} 0 $$
```

Superpositionnenement d'objets divers

On peut être ammené à superposer des objets ; on obtient :

$$\overset{*}{B} = 1 \quad \underset{\text{ext}}{B} + a \quad \underset{\text{ext}}{\overset{*}{B}} = 1 + a$$

avec la saisie :

```
$$
\overset{\ast}{B}=1\quad \underset{\mathrm{ext}}{B}=a \quad
\overset{\ast}{\underset{\mathrm{ext}}{B}}=1+a
$$
```

La commande qui cache un objet sans le cacher !

Le cas le plus fréquent où, tout naturellement, on a envie d'utiliser la commande `\smash` avant même que l'on sache qu'elle existe, est le suivant : dans un paragraphe, on a une ligne avec une formule mathématique contenant un exposant compliqué qu'on ne peut écrire que d'une seule manière ; il écarte la ligne le contenant de la ligne précédente, ce qui n'est pas très beau, alors que, juste au dessus de cet exposant, il n'y a que des lettres sans pied ! TEX écarte les lignes quand la somme (profondeur d'une ligne plus hauteur de ligne suivante plus distance `\lineskip`) dépasse `\baselineskip`. On voit bien que $A^{(4+B)^2}$ soulève la ligne précédente et que l'exposant 2 pourrait monter plus haut tout en restant assez éloigné de la ligne de base précédente. On va faire croîre à TEX que la hauteur de la formule est nulle en l'enfermant dans une boîte virtuelle de hauteur et de profondeur nulle ; voilà le résultat : $A^{(4+B)^2}$. C'est le rôle de la commande `\smash{argument}`.

10.9 Environnements du type « theorem »

Cette section explique les nombreuses possibilités pour la construction d'environnements numérotés automatiquement sous la forme C.1, C.2, etc. où C est le numéro de chapitre courant ; tout cela se fait grâce au style `theorem` ; le compteur correspondant est remis à zéro à chaque nouveau chapitre. Il y a aussi la possibilité de remise à zéro à chaque nouvelle section, solution lourde et à éviter excepté dans des cas bien spécifiques ; pour cela, il suffit de changer `chapter` par `section` dans la deuxième commande ci-dessous (dans ce cas la numérotation est de la forme C.S.1, C.S.2, etc., C et S étant respectivement les numéros de chapitre et de section courants).

Les commandes principales sont les suivantes :

`{\theoremstyle{xxxx} ... } :`
: choisit le style `xxxx` des environnements définis dans la suite jusqu'à la fin du groupement ; les styles `xxxx` disponibles sont :
`plain` qui donne en suivant le titre, le numéro, le titre facultatif et le texte à la suite,
`break` qui donne en suivant le titre, le numéro, le titre facultatif et le texte à la ligne suivante.

`\newtheorem{nom}{Titre}[chapter]:`
: définit l'environnement nommé `nom` affecté du titre `Titre` ainsi que le compteur correspondant, nommé `nom`, servant à la numérotation automatique ; ce compteur est remis à zéro lors du changement de valeur du compteur `chapter`.

`\newtheorem{nombis}[nom]{Titre} :`
: donne le même résultat que la commande précédente sauf que ce nouvel environnement `nombis` utilise le compteur de l'environnement `nom` défini préalablement. Cette possibilité est très utilisée car, en mathé-

matiques par exemple, on aime bien avoir une numérotation commune pour les théorèmes, les lemmes et les corollaires.

`\theoremheaderfont{fonte}` :
choisit la fonte du titre, de la numérotation et du titre annexe facultatif : un seul choix est possible (cette limitation peut être levée mais cela exige un peu de programation).

`\theorembodyfont{fonte}` :
choisit la fonte du texte de l'environnement ; ce choix agit aussi sur la fonte du titre d'où la nécessité d'adjoindre parfois `\upshape` à la fonte du titre lorsque la fonte du texte contient `\itshape`.

`\theorempreskipamount` et `\theorempostskipamount` :
espace avant et espace après élastiques et modifiables par la commande `\global\setlength{\theor...mount}{xxxx}` : les valeurs par défaut de `xxxx` sont 12pt plus5pt minus3pt et 8pt plus3pt minus1.5pt ; elles conviennent bien au choix de l'option `12pt` de la classe `book`.

10.9.1 Construction des environnements

Avec les commandes précédentes, l'utilisateur construit les environnements qu'il souhaite. On donne en suivant les lignes de construction contenues dans le fichier de style POLY (`poly.sty`) ; le lecteur peut compléter à souhait :

```
{\theoremstyle{plain}
\theoremheaderfont{\scshape}
\theorembodyfont{\itshape}
\newtheorem{theo}{Th\'eor\`eme}[chapter]
\newtheorem{lemma}[theo]{Lemme}
\theorembodyfont{\slshape}
\newtheorem{defi}{D\'efinition}[chapter]}

{\theoremstyle{break}
\theorembodyfont{\itshape}
\newtheorem{theopar}[theo]{Th\'eo\`eme}
\theorembodyfont{\slshape}
\newtheorem{defipar}[defi]{D\'efinition}}
```

10.9.2 Test des environnements

Théorème 10.1 (Théorème de Lebesgue) *Ici du texte pour tester l'environnement* `theo` *: disposition, fonte pour les titre, numérotation et titre annexe facultatif et fonte pour le texte.*

Théoème 10.2 (Théorème de complétion)
Ici du texte pour tester l'environnement `theopar` *: disposition, etc.*

Définition 10.1 (Voisinage d'un point) *Ici du texte pour tester l'environnement* `defi` *: disposition, etc.*

Lemme 10.3 *Ici du texte pour tester l'environnement* `lemma` *: disposition, etc. On remarquera que le lemme utilise bien le même compteur que les théorèmes.*

Définition 10.2 (Fermeture d'un ensemble)
Ici du texte pour tester l'environnement `defipar` *: disposition, etc.*

Les saisies respectives sont :

```
\begin{theo}[Théorème de Lebesgue]....\end{theo}
\begin{theopar}[Théorème de complétion]....\end{theopar}
\begin{defi}[Voisinage d'un point]....\end{defi}
\begin{lemma}....\end{lemma}
\begin{defipar}[Fermeture d'un ensemble]....\end{defipar}
```

Exercices

Composer toutes les formules données en exemple (sans utiliser les « sommes » dont le seul but est de provoquer la nécessité de coupures.

Partie I.C

Graphisme et hypertexte avec LaTeX

Chapitre 11

Inclusion de documents graphiques

On commence d'abord, dans un but pédagogique, par l'inclusion[1] d'objets graphiques par la méthode passant par la production d'un fichier PostScript intermédiaire pour arriver au fichier PDF final.

11.1 Inclusion d'un objet graphique au format PostScipt

Principe de base

Cette inclusion est rendue possible par la « commande qui ne fait rien » : l'auteur de TEX a imaginé dès le début la commande `\special{...}` qui n'a aucun effet sur le résultat de la composition mais qui écrit dans le fichier `.dvi` son argument. Cet argument peut être une suite de commandes PostScript et, lorsque l'on va créer le fichier `.ps`, DVIPS va ajouter cette suite de commandes au code PostScript qu'il génère à partir du fichier `.dvi`. Si on saisit :

```
Voilà un exemple simple
{\catcode`\:=12\special{ps:gsave currentpoint translate newpath
0 0 moveto 15 15 neg rlineto 15 15 neg rmoveto  300 300 neg
lineto neg 0 300 neg moveto 300 0 lineto 4 setlinewidth 0.2 setgray
stroke grestore}} donnant une croix...
```

on obtient (après compilation et création du fichier PostScript) :

Voilà un exemple simple donnant une croix de Saint-André de 20 mm environ superposée au texte. Cet objet graphique a son origine (extrémité du bras bas gauche de la croix) au point courant, entre les mots « simple » et « donnant » ; la composition du texte est totalement indépendante de cet objet qui est pour TEX un point de dimension nulle sur la ligne de base (il y a un espace avant résultant du retour à la ligne après « simple » et un espace après résultant de l'espace après « `grestore}}` ».

[1] Cette section ne peut être reproduite telle qu'elle est présentée que par la méthode de composition avec production du fichier PostScript intermédiaire.

Format PostScript « encapsulé »

On a déja parlé du langage PostScript sous lequel sont disponibles la plupart des fontes existantes ; ce sont les fontes dites de *Type 1*. Mais ce langage est utilisé aussi pour d'autres documents graphiques (images numérisées, dessins techniques de CAO, etc.) ; la plupart des logiciels de dessin et des logiciels de conversion de formats graphiques ont la possibilité d'enregistrer les fichiers au format EPS.

Ce format est basé sur le langage PostScript mais il inclut en plus des commentaires (lignes dont les deux premiers caractères sont `%%`) utilisés à de multiples fins, en particulier pour séparer les différentes parties et faciliter la lecture des fichiers.

Pour l'inclusion d'un objet graphique dans un document LaTeX, ce sont les premières lignes de commentaire qui sont intéressantes : il y a la ligne

```
%%BoundingBox:x y u v
```

où les couples `x,y` et `u,v` sont respectivement les coordonnées des sommets bas gauche et haut droit d'un rectangle fictif entourant l'objet graphique, d'où la dénomination : format « EPS » pour *Encapsulated PostScript*. Le rôle de cette ligne est fondamental : TeX va lire les quelques premières lignes du fichier graphique EPS jusqu'à ce qu'il trouve la chaîne de caractères `%%BoundingBox` ; les quatres nombres suivants sont lus et `u-x` et `v-y` vont donner respectivement la largeur et la hauteur de la boîte (vide) qui va être réservée à la composition. Ensuite, à la création du fichier `.ps`, DVIPS va inclure le code PostScript de l'insertion de telle manière qu'elle va se trouver imprimée dans la boîte réservée.

Soit le petit fichier graphique `croix.eps` contenant le code PostScript de la croix de Saint-André tracée plus haut. Ici, les unités sont différentes et les déplacements verticaux se font dans le sens opposé ; la commande `showpage` permet de visualiser ce fichier directement avec GSVIEW et n'est d'aucune utilité pour l'inclusion.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 50 50
newpath 0 0 moveto 50 50 lineto 0 50 moveto
50 0 lineto 1 setlinewidth 0.3 setgray stroke
showpage
```

La commande LaTeX `\includegraphics` résoud le problème de placement du graphique sans recouvrir le texte comme on peut le voir avec la saisie :

```
Voilà le résultat que l'on obtient
\includegraphics{d/croix.eps} sur lequel on remarque bien que la
croix ne déborde plus sur le texte...
```

Voilà le résultat que l'on obtient  sur lequel on remarque bien que la croix ne déborde plus sur le texte car DVIPS l'a placée dans la boîte (jusque là vide) réservée entre les lignes de texte par TeX au moment de la composition. On termine ensuite, comme d'habitude, par la production du fichier au format PDF final.

11.2 Commande d'inclusion et options correspondantes

Quand on génère directement le fichier PDF final, comme cela se fait maintenant (sauf dans des situations particulières), tout se passe exactement de la même manière, si ce n'est que tout se fait de manière complètement transparente ; mais en interne, il y a toujours la composition qui réserve la place de l'objet graphique dont le code est ajouté à l'étape finale de création du fichier PDF.

Pour faire cette inclusion, il faut d'abord charger l'extention `graphicx` avec l'option `pdftex` (ou avec l'option `dvips` si l'on travaille avec la méthode du fichier PostScript intermédiare). Cette extension contient la commande `includegraphics` déja utilisée dans la section précédente mais aussi de nombreuses commandes graphiques (dilatation, rotation et coloration du texte) qui seront vue au chapitre 12.

On donne maintenant la syntaxe détaillée de la commande :

`\includegraphics[liste-de-prescriptions]{nom-de-fichier}`

où `nom-de-fichier` est le nom du fichier graphique (avec éventuellement le chemin d'accès) et où les prescriptions, séparées par des virgules, peuvent être :

`bb=xx yy uu vv` :
: qui remplace la ligne
`%%BoundingBox:x y u v`
du fichier graphique si elle n'existe pas ou si elle est mauvaise[(2)] (coordonnées inexactes) ; les coordonnées doivent être données en points sans unité ;

`trim=xx yy uu vv` :
: qui enlève une bande d'espace à gauche, en bas, à droite et en haut respectivement de largeur `xx`, `yy`, `uu` et `vv` ; les coordonnées doivent être encore données en points sans unité ;

`clip=true` :
: qui n'imprime que l'intérieur de la `BoundingBox` ; cette prescription, jointe à l'une des précédentes, est très importante lorsque l'on se trouve dans le cas d'une figure numérisée avec trop de blanc tout au tour ; dans ce cas on va avoir un grand espace en dessus de la figure, mais cet espace est en réalité couvert par du blanc (de la couleur blanche) de la figure. Si l'on tente de le supprimer avec une commande d'espacement négatif, on va voir ce blanc couvrir le texte précédent la figure ; on se rend compte qu'il faut que la `BoundingBox` soit en fait le plus petit rectangle contenant la figure. Avec GSVIEW, on détermine alors ses dimensions (ou la largeur des bandes à supprimer) ; avec l'une ou l'autre des deux premières prescriptions et la prescription `clip=true` le blanc inutile est supprimé. Ainsi, on élimine toute une partie de la figure sans toucher à son fichier, manipulation délicate et peu recommandable.

[(2)] Une autre possibilité consiste à corriger les dimensions de la BoundingBox directement dans le fichier graphique après avoir relevé les vraies dimensions en visualisant la figure avec GSVIEW.

`width=xx` :
qui impose la largeur `xx` à la figure (en réduisant proportionnellement la dimension verticale) ; cette dimension, ainsi que toutes les autres ci-après, doivent être données avec une unité) ;

`height=yy` :
qui impose la hauteur et réduit proportionnellement la largeur ;

`totalheight=zz` :
qui impose la hauteur totale hauteur + profondeur (à utiliser lorsque l'on a des figures de profondeur non nulle) ;

`scale=f` : qui impose le facteur d'échelle `f` ;

`angle=aa`: qui fait une rotation de `aa` degrés ;

`origin=xx` :
qui définit l'origine de la rotation ; `xx` est formé de deux des lettres suivantes, la première pour la position horizontale et la deuxième pour la position verticale : `l` pour gauche, `r` pour droite, `t` pour haut, `B` pour ligne de base, `b` pour bas, et `c` pour centre dans les deux directions (exception pour le centre de la figure, `cc` est seulement noté `c`) ;

`draft=true` :
qui trace uniquement un rectangle contenant la figure (peut faire gagner du temps pour des mises en page complexes).

On peut aussi, en plus des fichiers graphiques `.eps` (si l'on travaille avec la méthode du fichier PostScript intermédiaire) ou en plus des fichiers `.pdf` (si l'on travaille avec la méthode directe), inclure directement des fichiers au format `.jpg` (format des photographies numériques). Mais il n'y a pas dans ce format l'équivalence de la `BoundingBox` ce qui oblige à donner par exemple la largeur voulue (`width`) ou un facteur de redimensionnement (`scale`). Si nécessaire, on consultera les références [3] et/ou [4], page 635 et suivantes ; cet aspect est très complexe car il y a bien d'autres formats graphiques reconnus `.tif`, `.pcx`, `.bmp`, `pntg` ... suivant le système d'exploitation utilisé. On signale simplement que sous UNIX et LINUX, on peut aussi utiliser des fichiers graphiques compressés qui sont décompressés au fur et à mesure de leur utilisation par un appel système au décompresseur.

Si l'on a des figures créées avec l'extension `pstricks`, il faut travailler avec la méthode du fichier PostScript intermédiaire (que l'on transforme à la fin pour obtenir le fichier PDF du document). On peut aussi transformer tous les fichiers graphiques `.eps` en fichiers graphiques `.pdf` pour créer directement le fichier `.pdf` du document. Pour ces transformations de formats, on dispose dans les distributions TEX de l'utilitaire :

```
epstopdf --outfile=fichier.pdf fichier.eps
```

Il y a d'autres utilitaires pour faire ces transformations, notamment DISTILLER, (utilitaire commercial de Adobe) et CONVERT (utilitaire public) qui permet presque toutes les conversions,

11.3 Positionnement des inclusions graphiques et des tableaux

Il est évident que l'on ne peut pas placer une figure où l'on veut : il peut ne pas y avoir la place sur le reste de la page courante. Le même problème se pose également pour les tableaux. On peut toujours faire des essais et, compte tenu de la hauteur totale de la figure, trouver la solution en un ou deux essais supplémentaires lorsque l'on commence à avoir une certaine expérience. Cela dans le cas où il n'y a pas trop de figures bien entendu !

Pour résoudre ce problème de positionnement des figures (et également des tableaux) les auteurs de LaTeX ont introduit la notion d'objets « flottants » ; pour cela ils ont défini l'environnement `figure` et l'environnement `table` pour le positionnement des tableaux ; ces environnements numérotent automatiquement les figures et les tableaux. La saisie type pour inclure une figure est :

```
\begin{figure}[x]
\begin{center}
\includegraphics[...]{...}
\caption{texte-de-la-légende}\label{...}
\end{center}
\end{figure}
```

où `x` prend les valeurs `h` pour placer la figure « ici si possible », `t` pour la placer en haut de la page courante, `b` pour la placer en bas de la page courante et `p` pour la placer sur une page, avec d'autres figures, après la page courante. Si l'on a fait entrer l'extension `float`, alors la valeur `H` réussit (en général) ce que `h` ne réussit pas à faire. Voici un exemple obtenu avec la valeur `H` pour `x` :

Figure 11.1 Exemple d'intégration de figure avec une longue légende qui se centre automatiquement sous la figure avec retrait à gauche et à droite par rapport au texte.

On remarque que le style POLY compose les légendes des figures, mais aussi des tableaux, avec une taille un peu plus petite que la taille du texte (en `\small`) ; le texte est en retrait à droite et à gauche, centré mais non justifié (tout cela est facilement modifiable).

Il y a de nombreux paramètres pour modifier le comportement du positionnement automatique de LaTeX ; il est hors de question de les recopier avec la description de leur rôle, voir [4], pages 290 et suivantes. Disons par exemple que `\textfraction`, fraction de texte sur une page, vaut 0,2 ce qui peut parfois gêner ; `totalnumber`, nombre maximal de figures, vaut 3 alors que l'on pourrait parfois placer quatre

petites figures. Les valeurs de ces paramètres peuvent être changées (avec l'utilisation des commandes `\renewcommand{\textf...}{.}` et `\setcounter{totaln...}{.}`.

L'inclusion d'un tableau se fait exactement de la même manière avec l'environnement `table` et tout ce qui a été dit pour les figures est valable pour les tableaux excepté que la légende se place traditionnellement avant le tableau en typographie française. Le style POLY apporte une modification nécessaire à ce changement et, compte tenu du fait qu'il est souvent préférable de faire un fichier spécial pour chaque tableau, la saisie type est donnée ci-après (remplacer la commande `\input{fichier-de-tableau}` par le codage du tableau lui-même pour des petits tableaux) :

```
\begin{table}{x}
\begin{center}
\caption{texte-de-la-légende}\label{...}
\input{fichier-de-tableau}
\end{center}
\end{table}
```

Exercices

Insérer une photo numérique dans une page de texte. Pour remplacer l'absence de `BoundingBox` donner par exemple la hauteur souhaitée.

Tester les prescriptions de la commande `\includegraphics` en utilisant le fichier graphique `croix1.pdf` de la figure (11.1). Noter que certaines dimensions doivent être données en points (environ 1/3 de mm).

Chapitre 12

Autres possibilités graphiques de LaTeX

Ce chapitre est consacré à quelques autres possibilités graphiques simples et particulièrement utiles. En ce qui concerne les extensions spécialisées, il faudra se rapporter aux ouvrages de références ; on se bornera à une présentation de PSTRICKS en fin de chapitre.

12.1 Manipulation d'objets LaTeX

On va donner ici quelques commandes transparentes pour l'utilisateur mais qui sont en fait basées sur l'utilisation de la commande `\special` et donc utilisent la puissance du langage PostScript de façon cachée.

`\scalebox{f}[g]{texte ou boîte}` :
est une commande qui redimensionne quelques mots de texte ou une boîte avec `f` pour facteur d'échelle horizontal et `g` pour facteur d'échelle vertical ; si ce dernier est omis, alors il prend par défaut la valeur du premier ; voici un exemple

Exemple de redimensionnement

obtenu avec `\scalebox{2}[3]{Exemple de redimensionnement}`.

`\resizebox{dimh}{dimv}{texte ou boîte}` :
est une commande semblable mais où l'on donne des dimensions au lieu des facteurs d'échelle ; `dimh` est la largeur voulue et `dimv` est la hauteur totale voulue. Si l'on écrit `\resizebox*`, alors le rapport largeur/hauteur-totale est conservé : dans ce cas bien entendu, on ne donne qu'une des deux dimensions, la seconde devant être remplacée par `!` (elle s'ajustera automatiquement).

`\rotatebox[xx]{angle}{texte ou boîte}` :
fait tourner le texte ou la boîte d'un angle de `angle` degrès ; l'origine de la rotation est donnée par le paramètre facultatif `xx` dont les valeurs sont celles utilisées par la prescription de rotation pour l'inclusion de graphiques, section 11.2, page 86 ; par défaut la rotation se fait autour du point de référence (`lB` avec les notations définies précédemment).

Voilà un exemple d'utilisation

Prénom	Nom	Age	Age du mariage	Age de l'obtention de l'ingéniorat
Jean	Dupré	23	21	22
Pierre	Duchemin	25	24	23
Raymond	Delaville	24	23	23

pour les tableaux avec des colonnes étroites dont les titres assez longs sont tounés autour de leur centre (paramètre `c`) ; il y a bien sûr beaucoup d'autres cas d'utilisation pour les tableaux mais aussi en dehors des tableaux.

12.2 Un peu de couleur, mais pas trop

En édition électronique, l'utilisation de la couleur est basée sur des modèles ; on ne va parler que des plus utilisés. Ces modèles sont relativement intuitifs si l'on se réfère aux connaissances élémentaires concernant la décomposition des couleurs en couleurs de base. Ces modèles sont :

- modèle RGB (red, green, blue) rouge, vert, bleu ;
- modèle CMYK (cyan, magenta, yellow, black) cyam, magenta, jaune, noir ; c'est la difficulté matérielle de reconstituer le noir et certaines couleurs foncées (fabrication des encres) qui fait préférer ce deuxième modèle au premier.

L'utilisation de la couleur implique que l'on charge le package `xcolor.sty` qui contient des tableaux de couleurs : 16 couleurs par défaut, 68 couleurs avec l'option `dvipsnames`, 151 couleurs avec l'option `svgnames` et 241 couleurs avec l'option `x11names`. Les couleurs par défaut sont : `black`, `blue`, `brown`, `cyan`, `darkgray`, `gray`, `green`, `lightgray`, `magenta`, `olive`, `orange`, `pink`, `purple`, `red`, `violet`, `white` et `yellow`.

Tous les tableaux de couleurs sont affichées dans la documentation de l'extension.

Les commandes de définition des couleurs sont les suivantes :

`\definecolor{couleur}{rgb}{x,y,z}` :
définit la couleur nommée `couleur` dans le modèle RBG avec trois nombres `x`, `y` et `z` inférieurs ou égaux à 1 ;

`\definecolor{couleur}{cmyk}{x,y,z,t}` :
définit la couleur nommée `couleur` dans le modèle CMYK avec quatre nombres `x`, `y`, `z` et `t` inférieurs ou égaux à 1 ;

`\definecolor{gris}{gray}{x}` :
définit le gris nommé `gris` avec le nombre `x` inférieur ou égal à 1 (attention : 1 définit le banc et 0 définit le noir !).

Les couleurs prédéfinies s'utilisent comme suit :

```
\noindent texte en noir,
 {\color{red}texte en rouge} ou
\textcolor{red}{texte en rouge} texte en noir.
```

et cela donne : texte en noir, texte en rouge ou texte en vert texte en noir.

On dispose, pour colorer les boites, des commandes suivantes :

`\colorbox{couleur}{texte}` :
fait une `\mbox` contenant `texte` sur un fond de couleur `couleur` ;

`\fcolorbox{couleur-fi}{couleur-fo}{texte}` :
fait une `\fbox` contenant `texte` avec un filet de couleur `couleur-fi` et un fond de couleur `couleur-fo`. On peut aussi fixer la couleur du texte avec la commande `\color` pour obtenir essai de boîte colorée ;

`\pagecolor{couleur}` :
colore la page en entier avec la couleur `couleur` (utilisable pour colorer les transparents).

12.3 D'autres encadrements

On a déjà vu l'encadrement de quelques mots de texte avec

`\fbox{.}` et `\framebox[.]{.}{.}`,

voir page 50 ; on peut modifier la distance de séparation `\fboxsep` (3 pt par défaut) entre le texte et le filet ainsi que l'épaisseur du filet `\fboxrule` (0.4 pt par défaut). Le package `fancybox.sty` apporte de nouvelles possibilités.

D'abord, il propose les nouveaux environnements `Bflushleft`, `Bcenter` et `Bflushright` qui ont les mêmes effets à l'intérieur de la boîte que les environnements `flushleft`, `center` et `flushright` sur la page :

... Texte au fer à gauche ... ... Texte au fer à droite ... ... Ici, texte centré ...

Ce type d'environnement peut aussi servir à faire des éléments de tableaux.

Il y a ensuite de nouvelles possibilités d'encadrement :

`\shadowbox{texte ou environnement B...}` :
fait un encadrement avec une ombre portée ;

`\doublebox{texte ou ...}` :
fait un encadrement avec double filet ;

`\ovalbox{texte ou ...}` :
fait un encadrement avec un filet fin et des coins arrondis ;

`\Ovalbox{texte ou ...}` :
fait le même type d'encadrement que la commande précédente mais avec un filet plus épais.

Voilà un exemple de ces encadrements montrant leur position par rapport à la logne de base (le premier exemple est fait avec `\fbox{...}` :

Titre ... ... Titre ... ... Titre ... ... Titre ... ... Titre

12.4 Pour finir, un aperçu sur PSTRICKS

Il s'agit d'une extension constituée d'un ensemble de fichiers permettant de programmer en PostScript sans le savoir ; ceci car les commandes correspondantes LaTeX sont transformées, grâce à la commande TeX `\special`, en commandes PostScript par DVIPS. Les possibilités sont immenses mais le nombre de commandes est aussi très grand. On va se borner encore à donner un exemple[1] : traversée d'une lentille par un rayon lumineux ; voici la figure suivie de son code PSTRICKS :

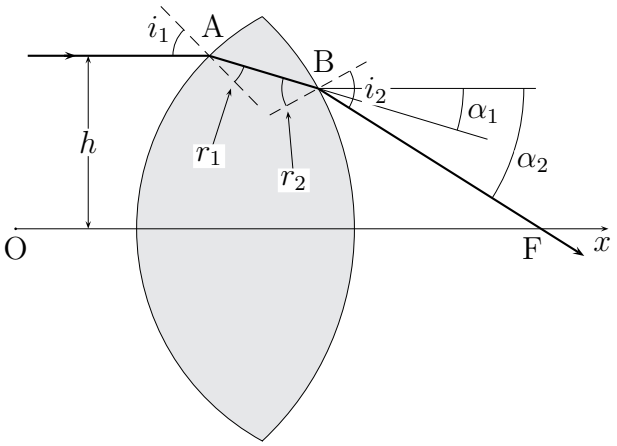

Figure 12.1 Traversée d'une lentille doublement convexe par un rayon lumineux

```
\begin{pspicture}(0,0.32)(7.36,6.24)
\newcmykcolor{grisleger}{0 0 0 0.1}
\def\montre(#1)(#2)[#3]#4{\psline[linewidth=0.3pt]%
                        {#3}(#1)(#2)\rput*(#1){#4}}
\psset{xunit=0.16cm,yunit=0.16cm,unit=0.16cm,labelsep=3pt,framesep=1pt}
\psset{linewidth=0.3pt}
\pscustom[fillstyle=solid,fillcolor=grisleger]{%
                \psarc(30,20){20}{118.7}{-118.7}
\psarc(4,20){24}{-46.9}{46.9}} \psline{->}(0,20)(49,20)
```

[1] Cette section ne peut être reproduite, telle qu'elle est présentée, que par la méthode de composition avec production du fichier PostScript intermédiaire.

```
\psplot[linestyle=dashed]{12}{20}{x 16 sub 1.02 mul neg 34.3 add}
\psplot[linestyle=dashed]{21}{29}{x 25 sub 0.55 mul 31.6 add}
\psarc(16,34.3){3}{134.4}{180} \psarc(16,34.3){3}{-45.6}{-16.7}
\psarc(25,31.6){3}{163.3}{-151.2} \psarc(25,31.6){3}{-32.2}{28.8}
\psarc(25,31.6){12}{-16.7}{0} \psarc(25,31.6){17}{-32.2}{0}
\psline(25,31.6)(39,27.4)%(43,26.2) \psline(25,31.6)(43,31.6)
\psline{<->}(6,34.3)(6,20)
\rput*(6,27.15){$ h $} \pscircle*(0,20){0.5pt}
\uput[d](0,20){$ \mathrm O $} \uput[d](48.5,20){$ x $}
\uput[d](42.7,20){$ \mathrm F $} \uput[u](16.3,35.2){$ \mathrm A $}
\uput[u](25.5,32.3){$ \mathrm B $} \psset{linewidth=0.8pt}
\psline{->}(1,34.3)(5,34.3) \psline(4,34.3)(16,34.3)
\psline(16,34.3)(25,31.6) \psline{->}(25,31.6)(47,17.75)
\uput[l](13.5,36.5){$ i_{1} $} \uput*[r](28.2,31.5){$ i_{2} $}
\uput[r](36.7,29.5){$ \alpha _{1} $}
\uput*[r](40.8,25.5){$ \alpha _{2} $}
\montre(16,26)(17.8,31.6)[->]{$ r_{1} $}
\montre(23,24)(22.5,29.7)[->]{$ r_{2} $}
\end{pspicture}
```

On remarque que l'on trouve exceptionnellement un peu de code PostScript pur dans le dernier argument des macros `\psplot`. La composition du présent document est faite avec la méthode directe ; cette figure étant écrite en PSTRICKS, on a procédé comme suit : on a traité un fichier contenant uniquement le code PSTRICKS de la figure et, avec DVIPS, on a obtenu un fichier PostScript d'une page contenant uniquement la figure entourée de beaucoup de blanc ; ensuite, on a fait le fichier PDF correspondant avec EPSTOPDF et on l'a intégré avec la macro `includegraphics` (le blanc a été éliminé en utlisant la prescription `bb=...`). Cette méthode, à éviter, est parfois bien utile (cas présent où la figure avait été faite pour un livre).

Exercices

Tester les commandes des trois premières sections du chapitre.

Pour la troisième cellule de la première ligne du tableau, on procédera comme suit (la méthode est à retenir pour clarifier le code : dans un premier temps, on construit la boîte, dans un second temps on la fait tourner) :
`\newsavebox{\baa}` crée la boîte nommée `\baa`
`\savebox{\baa}{\begin{tabular}{c}Age du\\mariage\end{tabular}}` met le petit tableau dans la boîte `\baa`
`\usebox{\baa}` extrait le contenu de la boîte `\baa`
Donc, on mettra dans la cellule en question :
`\rotatebox[origin=c]{90}{\usebox{\baa}}`
Faire le tableau complet.

Chapitre 13

De LaTeX à l'hypertexte

Les pages WEB que l'on trouve actuellement sont, en grande majorité, codées en HTML (*HyperText Markup Language*). Cela entraîne deux inconvénients majeurs pour les documents scientifiques :

- impossibilité d'afficher correctement des formules mathématiques (la seule solution parfaite valable consiste à faire une image pour chaque formule !) ;
- la mise en page de l'auteur est en partie perdue : c'est le navigateur « client » qui fait sa propre mise en page.

Le format PDF (*Portable Document Format*) ne présente pas les inconvénients ci-dessus et a toutes les possibilités hypertexte. En outre :

- il donne des documents de très bonne qualité ;
- alors que la lecture du format PostScript nécessite un visualisateur peu courant en dehors du monde TeX (bien que du domaine public), le format PDF est lu et imprimé avec Acrobat Reader distribué gratuitement par Adobe (commande : `acroread nom-de-fichier`).

13.1 Possibilités hypertexte de LaTeX

On ne va pas reprendre tout ce qui a été dit au chapitre 2, sous-sections 2.1.2, 2.1.3 et 2.1.7, concernant la production de documents au format PDF par la méthode directe ou par la méthode utilisant un fichier intermédiaire au format PostScript. On va plonger tout de suite dans la merveilleuse machinerie de l'hypertexte. Les possibilités sont de deux types :

I) toute la machinerie de « référencement au sens large » de LaTeX va être trasformée en un ensemble de liens et de cibles ; par exemple, dans la table des matières un clic sur le numéro de page d'une section du livre fera apparaître la page du début de cette section ;

II) on va disposer de tous les types de liens habituels de navigation rencontrés dans les pages au format HTML : par exemple on va pouvoir afficher un renseignement se trouvant dans une page située sur un serveur distant.

13.2 Préparation du fichier

Il faut charger l'extension `hyperref` à la fin du préambule dans le fichier maître `livre.tex`. Cette unique extension transforme toute la machinerie de références de LaTeX en un ensemble de liens hypertextes :

- table des matières : le clic sur un numéro de page fait apparaître la page de texte correspondante ;
- bibliographie : le clic sur une citation dans le texte fait apparaître la référence correspondante dans la bibliographie ; le(les) nombre(s) qui suit(suivent) le texte de la référence est(sont) le(s) numéro(s) de page où se trouve cette citation ; le clic sur ce(s) numéro(s) de page fait(font) apparaître la(les) pages correspondantes ;
- notes de bas de page ; le clic sur le numéro de note montre la partie de la page où se trouve l'appel de la note (utile si l'écran ne montre qu'une partie de la page).
- références croisées : le clic sur le numéro de sectionnement ou sur le numéro de de page associé fait apparaître le début de ce sectionnement.

Comme les possibilités sont nombreuses, le nombre d'options est grand et on va se limiter aux plus importates. On charge l'extension de la façon suivante :

```
\usepackage[xxxx,bookmarks, ... ]{hyperref}
```

où les options sont :

`xxxx` : est obligatoire : `pdftex` si le PDF final est obtenu directement et `dvips` s'il est obtenu à partir d'un fichier PostScript intermédiaire ;

`bookmarks, bookmarksopen` :
la première option assure la composition des *signets*, la seconde option fait de même et commande en plus leur affichage à l'ouverture du fichier avec ACROREAD ; les *signets* apparaissent dans une colonne à gauche de l'écran, ils constituent la table des matières typique des documents au format PDF ; ces *signets* posent des problèmes comme expliqué à la sous-section 13.3.

`colorlinks` :
commande la coloration des liens (par défaut les couleurs sont rouge pour les liens internes, noir pour les cibles, vert pour les citations, magenta pour les fichiers, cyan pour les URL ; ces couleurs peuvent être changées par des options spécifiques ;

`anchorcolor=xxxx` :
change la couleur des cibles (le noir ne permet par de repérer facilement la cible, il vaut mieux prendre une couleur foncée, exemple `olive` (couleur définie par défaut dans l'extension `xcolor`) ;

`hyperindex` :
est une option qui rend actifs les numéros de page de l'index, le clic sur le numéro de page affiche cette page ;

`linktocpage` :
fait que ce sont les numéros de page qui sont actifs dans la table des matières et non les titres de chapitre et de section (beaucoup plus agréable) ; de la même manière que ci-dessus, le clic sur le numéro de page affiche cette page ;

`pagebackref` :
ajoute à la bibliographie, après chaque référence, les numéros des pages où elle a été citée et rend actifs ces numéros, ce qui permet d'accéder à des pages très facilement ; il faut pour cela que chaque entrée `\bibitem{...}` de la bibliographie soit suivi d'une ligne blanche lors de sa création (saisie ou création automatique) ;

`raiselinks=true` :
remonte l'affichage des liens qui par défaut ne montre que la ligne de base (si la cible est en haut de page, on ne pourrait pas voir le mot sans cette option) ; uniquement valable pour les cibles LaTeX ;

`plainpages=false,pdfpagelabels` :
l'ensemble de ces deux options résolvent le problème de la double numérotation de la classe `book` présentée à la page 35.

En fait les hyperliens sont créés par défaut ; les options ci-dessus ne sont que des options de présentation ou des améliorations.

Il faut savoir qu'il y a des dizaines d'options ; en voilà un aperçu :

- des options concernant le fichier : `pdfauthor=nom`, `pdfsubjet=sujet`, etc.
- des options pour commander l'affichage : `pdfmodepage=FullScreen`, `pdfpagelayout=Fit`, numéro de page à montrer à l'ouverture du fichier, afficher la barre des menus, etc.
- des options pour le passage d'une page à l'autre (par dissolution, balayage, mouvement de rideaux...), etc.
- des options pour les signets : numérotation, profondeur de la numérotation , etc.

Toutes ces options sont données et sommairement commentées dans la doc de l'extension `hyperref` (mise à jour en 2012).

13.3 Attention : difficultés possibles

Problème avec les signets

Les signets sont créés avec les entrées de la table des matières : donc les commandes TeX figurant dans ces entrées ne sont pas reconnues : logo `\TeX`, formules mathématiques, `\og` et `\fg`, les ligatures `\oe` et `\ae`, etc. Cela est signalé par un message du type :

```
Package hyperref Warning
Token not allowed in a PDFDocEncoded string: ...
```

Pour résoudre ces difficultés, le package `hyperref` fournit la macro

`\texorpdfstring{saisie en syntaxe TeX}{saisie en syntaxe PDF}`

utilisée comme suit :

`\section{\texorpdf...{Fichier \og maître \fg}{Fichier ‘‘maître’’}}`

En ce qui concerne les ligatures, il faut les supprimer dans l'argument en syntaxe PDF. Il peut y avoir des cas plus difficiles à traiter : on se trouve alors obligé de modifier le titre en restant le plus près possible du titre en syntaxe TEX. Cela sera facilité avec le codage en entrée `utf8` (cf. page 18).

Problème avec les numérotations de page

Un autre problème survient lorsqu'il y a deux pages ayant le même numéro, ce qui est le cas avec la classe `book` où l'introduction et la table des matières ont une numérotation spécifique en chiffres romains minuscules ; ce cas est résolu par l'utilisation des options `plainpages=false` et `pdfpagelabels`. Mais tout autre numérotation multiple demande une très délicate programmation.

13.4 Ajout de liens hypertexte supplémentaires

On peut ajouter au fichier de saisie tous les liens hypertexte habituellement utilisés dans le langage HTML.

Hyperliens internes au fichier

A l'intérieur d'un même document, on peut faire un lien avec les macros :

`\hyperlink{nom-du-lien}{mots-liens}`
`\hypertarget{nom-du-lien}{mots-ciblés}`.

Ces macros colorent les *mots-liens* et les *mots-cibles* (par défaut : rouge pour les premiers et noir pour les seconds, autres couleurs possibles avec des options).

Voilà un exemple : `mots-liens` = « parlé », `mots-cibles` = « fichier maître »et `nom-du-lien` = `testint`. Si on écrit :
... le fichier maître, dont on a déja parlé, à la fin du premier chapitre ...
le clic sur « parlé » montre la ligne contenant les mots « fichier maître ».

Le retour au point de départ se fait avec la barre de navigation, bouton « Reviens » (cf. section suivante).

Hyperliens internes à l'ordinateur

On dispose d'un couple de macros qui permet d'ouvrir un fichier à une page donnée (il faut bien entendu préparer ce fichier en y plaçant une cible, c'est la deuxième macro qui joue ce rôle) :

`\hyperref{nom-fichier}{categorie}{nom-du-lien}{mots-cibles}`
`\hyperdef{categorie}{nom-du-lien}{mots-cibles}`.

Ici, le rôle du nom du lien est le couple `catégorie` + `nom-du-lien` (cela est imposé par des raisons de compatibilié avec d'autres langages). Il faut penser que,

lorsque l'on aura consulté les pages voulues, il faudra revenir au premier fichier, à l'endroit d'où l'on est parti ; cela signifie que l'on devra utiliser deux paires des macros ci-dessus : une paire pour l'aller et une paire pour le retour.

Texte du premier fichier `polycop.pdf` et macros correspondantes :
... les tailles disponibles sont listées au troisième chapitre ...
`\hyperref{testlink.pdf}{text}{ficmaster}{listées}` : lien aller,
`\hyperdef{text}{ficmasterbis}{troisième chapitre}` : cible retour.

Texte du second fichier à atteindre `testlink.pdf` et macros correspondantes :
... (première ligne) Le mot *fonte* est en principe réservé à un ...
`\hyperref{polycop.pdf}{text}{ficmasterbis}{principe}` : lien retour,
... Les tailles disponibles en LaTeX sont ...
`\hyperdef{text}{ficmaster}{tailles disponibles}` : cible aller.
On a mis le lien retour dans la première ligne du fichier visité. Si l'on a été amené à se déplacer loin de la cible aller, on trouvera ainsi directement le lien retour.

Mais, si le fichier secondaire a une bare de navigation, on peut faire le retour avec le bouton « Reviens » et les deux macros du lien retour sont alors inutile. Cette possibilité est testée dans la partie II.A « Préparer des transparents » où son avantage est détaillé (possibilité d'utiliser des fichiers annexes avec tout fichier principal, sans devoir ajouter des liens retour spécifiques aux fichiers pricipaux).

Pour ouvrir un fichier à la première page, on a la commande simplifiée :

```
\href{nom-du-fichier}{mots-liens}
```

Il n'y a pas de cible à désigner dans le second fichier, mais il faut y faire le lien de retour comme pour le cas précédent : on laisse au lecteur le soin de construire un exemple. On note que cette commande peut ouvrir un ficher HTLM : voici l'ouverture du fichier `histoiregs.html` : le problème de retour ne se pose pas dans ce cas car l'ouverture et la fermeture de l'explorateur n'agissent pas sur l'affichage du premier fichier qui est un fichier PDF.

Hyperliens externes

La commande`\href` a encore trois applications : en voilà d'abord deux pour l'accès à une URL et à un fichier dont on connait le chemin dans l'URL et le nom :

```
\href{http://adresse-url}{mots-liens}
\href{http://adresse-url/chemin/fichier}{mots-liens}
```

Enfin, la troisième application permet l'envoi d'un message par l'intermédiaire de l'utilitaire de messagerie utilisé par défaut : EUDORA, THUNDERBIRD, etc. :

```
\href{mailto:adresse-internet}{mots-liens}
```

13.5 Barre de navigation

Le package `hyperref` permet de créer facilement une barre de navigation spécifique (voir en détail le fichier `navi.tex`). La macro fondamentale est :

```
\Acrobatmenu{menuoption}{nom-du-bouton}
```

où `menuoption` est une commande de navigation PDF, par exemple : FullScren (plein écran), FitPage (une page sur l'écran), LastPage, NextPage, Goback, GoForward, etc. et `nom-du-bouton` peut être un mot ou plusieurs mots ou un mots dans une `\fbox` qui fait penser à un bouton (et avec un peu de couleur, c'est encore mieux).

Ces boutons sont placés en ligne pour former une sorte de texte linéaire nommé `\footmark`. Le package `fancyhdr`, section 5.4, utilisé pour faire les hauts de pages (cf. fichier de style `poly.sty`) peut aussi faire les pieds de pages : il suffit de rajouter :

`\fancyfoot[CO]{\footmark}` : centre du pied des pages impaire
`\fancyfoot[CE]{\footmark}` : centre du pied des pages impaire

et de redéfinir le style de page `plain`, qui est le style des pages blanches ainsi que des pages de titre de partie et de titre de chapitre. Toutes cela se trouve dans le petit ficher `navi.tex` qui est chargé après le fichier `poly.sty`.

Le fichier `polycop.pdf` s'ouvre sur la page de titre ; dès l'ouverture, on conseille de cliquer sur les boutons Page et Ecran (on sort du mode plein écran avec Escape).

On réalisera rapidement l'apport particulièrement précieux sur le plan pédagogique des boutons « Reviens » et « Repars » et de tous les autres pour naviguer entre deux points quelconques du document.

Toute la machinerie hypertexte repose sur le chargement du style `hyperref` et le fichier `navi.tex` chargé juste à la suite. Pour une impression, il faut donc annuler l'action du style `hyperref` en ajoutant l'option `draft`[1] et annuler le chargement du fichier `navi.tex`. Le présent package contient les fichiers `polycop.pdf` pour la lecture à l'écran avec navigation et le fichier `polyimp.pdf` pour l'impression (*la mise en page est faite pour une impression recto-verso*).

Avec cet ensemble de liens hypertexte on peut réaliser des documnents pédagogiques permettant, au fur at à mesure de la progression, des retours en arrière pour revoir certains points et des consultations d'autres documents pour avoir des explications complémentaires. Cela est particulièrement vrai pour la production de documents à projeter où l'auteur des transparents peut préparer à l'avance la possibilité d'afficher des documents complémentaires suivant les réactions et/ou les questions de l'auditoire. Cette possibilité sera développée dans la partie II.A de ce document sous la forme d'une manuel-présentation du package `pdfscreen` accompagné du package `texpower` pour afficher progressivement les lignes des pages projetées.

Comme exercice, l'auteur doit se lancer sur un travail qu'il a en projet : avec ce qu'il a trouvé dans ce présent document, il doit pouvoir réussir ... et cela d'autant plus aisément qu'il dispose de *tous les fichiers spécifiques* associés (joints à ce document par l'auteur). Il ne faut pas oublier de consulter les documentations disponibles, sur le site `ctan.org` [9], le DVD annuel distribué par le groupe d'utilisateurs GUTenberg [8], le contenu des ditributions dans les sous-répertoires `doc` et ... les collègues ; ne pas oublier que l'esprit TEX-LaTeX est comme l'esprit LINUX : ceux qui savent aident ceux qui débutent.

---

[1] Cette option inhibe toutes les macros hypertexte que l'on a introduites dans la saisie ... sinon il faudrait les effacer !

Bibliographie

[1] Denis Bitouzé et Jean-Côme Charpentier
LaTeX
Pearson Education, Paris, 2006. 4

[2] Christian Rolland,
LaTeX par la pratique,
O'Reilly, Paris 1999. 4

[3] Helmut Kopka & Patrick W. Daly,
A guide to LaTeX,
Addison-Wesley, London 1999. 4, 27, 86

[4] Frank Mittelbach, Michel Goosens,
LaTeX companion, 2e édition
Pearson Education France, Paris 2005. 4, 10, 20, 27, 70, 86, 87

[5] Donald E. Knuth,
The TeXbook,
Addison-Wesley, Reading 1986. 4

[6] Raymond Séroul,
Le petit livre de TeX,
InterÉditions 1989. 4

[7] Daniel Flipo,
Francisation d'un format LaTeX : nouveautés,
Cahiers GUTenberg, **32** (1999) p. 63–70. 18

[8] GUTenberg *(Association des utilisateurs francophones de TeX)*,
`http://www.gutenberg.eu.org`
Forum de discussion : `gut@ens.fr` 70, 100

[9] CTAN *(Comprehensive TeX Archive Network)*,
`http://ctan.org` 70, 100

Index