LaTeX 2$_\varepsilon$, un aperçu

Michel GOOSSENS

CERN, Division CN, 1211 Genève 23, Suisse, <Michel.Goossens@cern.ch>

Résumé. Cet article passe en revue les commandes utilisateurs modifiées ou nouvellement introduites dans LaTeX 2$_\varepsilon$, la nouvelle version de LaTeX, en comparaison avec la version précédente LaTeX 2.09. Après une discussion des nouvelles commandes disponibles dans le préambule, nous introduisons les extensions pour définir les commandes ou environnements, pour traiter les longueurs ou construire les boîtes. Puis nous parlons des commandes permettant de gérer les polices en mode mathématique et en mode texte dans le contexte du nouveau schéma de sélection des polices (NSSP) [1], qui fait partie intégrante de LaTeX 2$_\varepsilon$. En particulier nous montrons la facilité avec laquelle on peut utiliser différentes familles de polices. Une liste des classes et extensions disponibles avec LaTeX2e est donnée. Les nouvelles possibilités pour gérer la mise en page et le placement des éléments flottants sont présentées. Tous ces sujets sont traités en profondeur et avec beaucoup plus de détails dans « The LaTeX Companion » [1] et la seconde édition du manuel de référence LaTeX [2].

Abstract. *This article gives an overview of the new or extended user commands available with LaTeX 2$_\varepsilon$, the new LaTeX release, compared to the previous version LaTeX 2.09. After introducing the new preamble commands, the extensions for defining new commands and environments, and handling length and boxes are discussed. The new font selection commands are explained, both for text and math, and it is shown how to easily use different font families. A list of supported class and package files is given and new possibilities for controlling page contents and floats are discussed. Most of this material is described in much greater detail in "The LaTeX Companion" [1] and in the second edition of the LaTeX Reference Manual [2].*

1. Pourquoi LaTeX 2$_\varepsilon$?

Après l'apparition généralisée de LaTeX en 1986, sa popularité n'a cessé d'augmenter et beaucoup d'extensions ont vu le jour. Malheureusement, ces extensions introduisirent des incompatibilités dans les formats, par exemple, il y avait LaTeX « standard », avec ou sans NSSP, SLITeX, $\mathcal{AMS}$-LaTeX, et ainsi de suite. En examinant seulement le document source LaTeX, il était difficile de déterminer pour lequel de ces formats (ou même d'autres) la source était destinée, et, comme différents sites pouvaient avoir des configurations différentes, la portabilité des documents posait souvent problème.

Déjà en 1989 à la conférence annuelle du TUG à Stanford, Frank Mittelbach et Rainer Schöpf s'étaient joints à Leslie Lamport pour discuter de ces problèmes (parmi d'autres)

1. La traduction de l'anglais *New Font Selection Scheme* et son sigle NFSS.

et ils ont alors publié leurs idées sur des voies possibles pour développer LaTeX dans *TUGBoat* [4, 5]. Ceci conduisit quelques années plus tard au lancement du projet à long terme LaTeX3 [6–9].

Cependant, pour essayer d'éliminer toute confusion pour les utilisateurs actuels de LaTeX, lors d'une visite de Leslie Lamport à Frank Mittelbach à Mainz (Mayence, Allemagne) au printemps de 1993, il fut décidé de développer une nouvelle version normalisée de LaTeX, appelée LaTeX 2_ε, préparée et maintenue par l'équipe du projet LaTeX3. Cette décision fut annoncée officiellement à la conférence annuelle du TUG à Aston (Birmingham, Royaume-Uni) en juillet 1993 [3].

Les objectifs majeurs de LaTeX 2_ε sont :

- créer un format unique, remplaçant la multiplicité de formats incompatibles utilisés antérieurement ;
- proposer les fonctions des anciens formats dans des fichiers d'extensions, par exemple `amstex` remplace $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX, et `slides` remplace SliTeX ;
- arrêter la prolifération de dialectes incompatibles de LaTeX 2.09 ;
- introduire NSSP comme le système « standard » de sélection des polices ;
- ajouter un petit nombre de fonctions qui ont souvent été réclamées ;
- garder la compatibilité au niveau document (c.-à-d. que les sources LaTeX 2.09 seront composées de la même façon qu'auparavant) ;
- rester conforme aux conventions LaTeX 2.09, pour rendre l'apprentissage des nouveautés aussi aisé que possible.

La première version « bêta » de LaTeX 2_ε a été rendue publique à la fin 1993, et une deuxième « bêta » est disponible depuis février 1994. La première version de production sera distribuée vers la fin du printemps 1994. À partir de cette date, régulièrement, deux fois par an (au « printemps » et en « automne ») une mise à jour de tous les fichiers de la distribution sera réalisée, même si certains fichiers n'ont pas changé. Ceci garantira la synchronisation de tous les logiciels. Les rapports d'erreur seront gérés centralement en invitant tous les utilisateurs à remplir un formulaire électronique, disponible dans la distribution LaTeX 2_ε, qui devra être envoyé par courier électronique à l'adresse `latex-bugs@rus.uni-stuttgart.de`. Ces rapport seront pris en compte seulement si la version de LaTeX 2_ε qui produit l'erreur ne date pas de plus d'un an. Il existe également une liste de discussion dédiée spécifiquement à LaTeX 2_ε, `LATEX-2E@DHDURZ1.BITNET`, où l'on peut soumettre des questions (ou y poster des réponses).

2. Les déclarations au début du document

Dans cette section nous discuterons de commandes qui peuvent seulement être utilisées avant et dans le préambule d'un document. En particulier, les deux commandes cidessous peuvent apparaître uniquement *devant* la commande `\documentclass`.

2.1. Commandes avant le préambule

`\NeedsTeXFormat{`*nom-version*`}[`*date-version*`]`

Cette commande, qui normalement se trouve à l'intérieur de fichiers de classe ou d'extension, peut également être utile dans un document utilisateur pour garantir que ce dernier soit traité par LaTeX 2_ε.

```
\NeedsTeXFormat{LaTeX2e}[1995/01/01]
```

Par exemple, la référence ci-dessus (à une version fictive) produit le message suivant :

```
LaTeX2e <1994/02/03> PRELIMINARY TEST RELEASE
LaTeX Warning: You have requested version
    `1995/01/01' of LaTeX2e,
    but only version `1994/02/03'
    is available.
```

Si l'on essaie d'utiliser LaTeX 2.09 ou plain, on obtiendra un message d'erreur expliquant relativement clairement le problème :

```
LaTeX Version 2.09 <25 March 1992>
! Undefined control sequence.
l.1 \NeedsTeXFormat
                   {LaTeX2e}
?
```

```
This is TeX, C Version 3.141
(latex2e.tex
! Undefined control sequence.
l.1 \NeedsTeXFormat
                   {LaTeX2e}
?
```

Pour être certain que des documents peuvent être traités sur d'autres sites, il est souhaitable d'inclure toutes les extensions et autres fichiers, dont le document LaTeX 2_ε a besoin, avec le fichier principal. LaTeX 2_ε offre la syntaxe suivante pour ceci :

`\begin{filecontents}{`*nom-du-fichier*`}`
  ⟨*contenu-du-fichier*⟩
`\end{filecontents}`

Quand le document est traité par LaTeX 2_ε le corps de chaque environnement ⟨*contenu-du-fichier*⟩ sera écrit littéralement (verbatim) dans un fichier dont le nom est spécifié comme argument *nom-du-fichier*. Si un fichier avec un tel nom existe déjà dans un des répertoires « visibles » par TeX, un message informatif apparaît, le corps de l'environnement est ignoré, et le fichier original reste intact.

2.2. Commandes dans le préambule

Les commandes de préambule décrites dans cette section sont spécifiquement conçues pour différencier les documents LaTeX 2ε de ceux nécessitant LaTeX 2.09.

Avant d'aller plus loin, arrêtons-nous un moment pour donner quelques notions de base. LaTeX 2ε distingue entre *classes* et *extensions* (la traduction adoptée par GUTenberg pour le terme anglais *package*). Les documents d'une même classe sont caractérisés par une structure générique commune. La classe à laquelle un document appartient est spécifiée par la commande `\documentclass`. Des exemples de classes sont `article`, `report` et `book`. Si l'on veut préciser certains aspects globaux du document (typographiques, comme le corps de la police, le format du papier ou la langue principale) on peut spécifier des « options de classe », par exemple `\documentclass[11pt,a4paper,francais]{article}`. D'un autre côté, les extensions regroupent la définition de commandes et environnements qui augmentent les fonctionnalités de base de LaTeX. Ils sont déclarés par une commande `\usepackage`.

`\documentclass[`*liste-des-options*`]{`*nom-de-classe*`}[`*date-de-version*`]`

Cette commande ou «déclaration» remplace le `\documentstyle` de LaTeX 2.09.

Il faut qu'il y ait exactement *une* déclaration `\documentclass` dans chaque document, et qu'elle soit la première commande exécutable (excepté pour les commandes «initiales» décrites ci-dessus).

liste-des-options **:** une liste (optionnelle) d'options dont chacune peut modifier la présentation de certains éléments du document comme définis dans le fichier de classe *nom-de-classe* ou dans l'un des fichiers d'extension chargé par une commande `\usepackage`, comme décrit ci-dessous.

nom-de-classe **:** nom du fichier de classe, caractérisé par le suffixe `.cls`.

date-de-version **:** paramètre (optionnel), qui spécifie la date de distribution du fichier de classe. Le format utilisé est `AAAA/MM/JJ`, par exemple `1994/02/03`, pour le 3 février 1994. Si TeX trouve un fichier de classe avec une date antérieure à celle spécifiée sur la commande `\documentclass`, il affiche un avertissement.

`\documentstyle[`*liste-des-options*`]{`*nom-de-style*`}[`*date-de-version*`]`

Cette commande, qui est disponible pour des raisons de compatibilité, est similaire à `\documentclas`, mais elle commute LaTeX 2ε en «mode de compatibilité», où toutes les commandes sont redéfinies pour qu'elles aient le même effet qu'avec LaTeX 2.09, ce qui permet de traiter les anciens fichiers sans changements avec LaTeX 2ε. Notons quand-même qu'avec ce mode de fonctionnement, on n'a pas accès aux extensions LaTeX 2ε décrites dans le présent article.

On charge un fichier d'extension avec la commande `\usepackage`, qui a la syntaxe :

`\usepackage[`*liste-des-options*`]{`*nom-extension*`}[`*date-de-version*`]`

liste-des-options **:** une liste (optionnelle) d'options dont chacune peut modifier la présentation de certains éléments du document définis dans le fichier d'extension.

nom-extension **:** le nom du fichier extension, caractérisé par le suffixe `.sty`. Un fichier extension peut :

- définir de nouvelles commandes ;
- modifier des commandes définies dans le fichier de classe ou dans un fichier d'extension chargé antérieurement ;
- élargir la gamme de documents qui peuvent être traités.

date-de-version **:** paramètre (optionnel), qui spécifie la date de distribution du fichier d'extension (voir la commande `\documentclass` ci-dessus).

On peut utiliser un nombre arbitraire de commandes `\usepackage`. Une commande `\usepackage` peut inclure dans son argument *nom-extension* les noms de plusieurs extensions, par exemple `\usepackage{calc,ifthen}`. LaTeX 2ε vérifie que chaque fichier d'extension est seulement chargé une seule fois. La commande `\usepackage` traite non seulement les options spécifiées dans l'argument *liste-des-options* mais également celles présentes dans *liste-des-options* de la commande `\documentclass`.

Plus généralement, les options spécifiées dans les arguments optionnels des commandes `\documentclass` et `\usepackage` peuvent être subdivisées en options *globales* (les options de `\documentclass` vues par un fichier d'extension importé avec `\usepackage`) ou *locales* (toutes les autres). L'ordre dans lequel les options sont traitées est celui dans lequel elles sont déclarées à l'intérieur des fichiers. Les options non utilisées dans une classe sont mémorisées et exportées comme options globales vers les extensions. Si dans une extension une option locale n'est pas référencée, un message d'erreur est affiché, et l'utilisateur est invité à réintroduire le nom de l'option.

Finalement, en atteignant la commande `\begin{document}` LaTeX 2ε affichera un message d'avertissement pour toute option globale référencée ni par la classe, ni par les extensions.

`\listfiles`

En introduisant cette commande dans le préambule du document, on obtient en fin d'exécution une liste avec les noms des différents fichiers lus par le document. Voici une partie de la liste obtenue en traitant le présent article :

```
*File List*
article.cls    1994/02/03 Standard LaTeX2e document class
 size10.clo    1994/02/03 LaTeX2e document class size dependent file
```

```
  french.sty
 cahiers.sty
rotating.sty
graphics.sty    1994/04/13 v0.4a Standard LaTeX Graphics (DPC,SPQR)
    trig.sty    1994/03/15 v1.07 sin cos tan (DPC)
   dvips.def    1994/04/24 v1.3 Driver-dependant file (DPC,SPQR)
   t1enc.sty
  epsfig.sty    94/04/121.1(SPQR)
egraphics.sty   1994/04/14 v0.4a Enhanced LaTeX Graphics (DPC,SPQR)
  keyval.sty    1994/02/01 v1.06 (DPC)
  ifthen.sty    1994/01/24 v0.1e LaTeX2e package
    calc.sty
vspaceex.eps    Graphic file (type eps)
```

2.3. Exemples de préambules

L'exemple suivant fait référence à la classe de document article avec les options (globales) twocolumn et a4paper. Puis il charge les fichiers d'extensions multicol et babel, en spécifiant pour ce dernier les options german et dutch. D'autres paramètres globaux associés au document (comme `\textheight`) peuvent également être définis dans le préambule.

```
\documentclass[twocolumn,a4paper]{article}
\usepackage{multicol}
\usepackage[german,dutch]{babel}
\addtolength{\textheight}{2cm}
\begin{document}
    ...
\end{document}
```

Ci-dessous on trouvera trois façons équivalentes pour charger des fichiers d'extensions :

```
\documentclass[german]{book}
\usepackage[german]{babel}
\usepackage[german]{varioref}
\usepackage{multicol}
\usepackage{epic}
```

Un peu moins long :

```
\documentclass[german]{book}
\usepackage[german]{babel,varioref}
\usepackage{multicol,epic}
```

En spécifiant l'option german comme globale, on peut raccourcir davantage :

```
\documentclass[a4paper,german]{book}
\usepackage{babel,varioref,multicol,epic}
```

Un document un peu plus complexe peut avoir une structure comme :

```
\NeedsTeXFormat{LaTeX2e}[1994/05/01]
\begin{filecontents}{varioref.sty}
   ....                     % Source LaTeX de l'extension varioref
\end{filecontents}
\listfiles                  % Imprimer la liste des fichiers référencés
\documentclass[a4paper,german]{book} % La classe de document << class >>
\usepackage{varioref}
\begin{document}
%-----------%-------------------- matériel préliminaire
\maketitle
\section*{...}              % par exemple une section nommée << Préface >>
\tableofcontents            % chapitre << Table des matières >>
\listoffigures              % chapitre << Liste des figures >>
\listoftables               % chapitre << Liste des tableaux >>
%-------------------------------- corps du document
\part{...}
\chapter{...}
   \section{...}
\chapter{...}
\part{...}
%-----------%-------------------- matériel postliminaire
\appendix
\chapter{...}               % chapitres étiquetés << Appendice >>
\chapter{...}
\begin{thebibliography}
  ...                       % entrées bibliographiques
\end{thebibliography}
\begin{theindex}
  ...                       % entrées dans l'index
\end{theindex}
\end{document}
```

Pour être certain que le destinataire du fichier est capable de traiter le document sans problèmes la source LaTeX du fichier d'extension varioref est incluse dans le fichier, dans le corps d'un environnement `filecontents` *précédant* la commande `\documentclass`.

3. Définir des commandes et des environnements

Cette section explique comment l'utilisateur peut définir des commandes et environnements nouveaux ou redéfinir d'anciennes déclarations.

3.1. La (re)définition d'une commande

LaTeX 2_ε propose plusieurs façons pour définir ou redéfinir une commande :

`\newcommand{\`*macom*`}[`*narg*`][`*défaut*`]{`*définition de la commande*`}`
`\renewcommand{\`*macom*`}[`*narg*`][`*défaut*`]{`*redéfinition de la commande*`}`
`\providecommand{\`*macom*`}[`*narg*`][`*défaut*`]{`*définition de la commande*`}`

Les deux premières formes `\newcommand` et `\renewcommand` proposent des extensions par rapport à LaTeX 2.09 en offrant la possibilité de spécifier un argument *optionnel* lors de la (re)définition de la commande `\macom`. La possibilité d'un tel paramètre optionnel dans la commande utilisateur est signalée par la présence du second paramètre optionnel *défaut*, qui donne la valeur par défaut prise par l'argument en question lorsqu'il n'est pas spécifié explicitement quand la commande est utilisée. La troisième forme `\providecommand` est surtout intéressante dans des fichiers d'utilité générale, qui sont inclus dans d'autres documents, et qui ne sont pas toujours sous le contrôle de l'utilisateur (par exemple des bases de données bibliographiques BIBTeX). Si la commande `\macom` n'est pas encore définie, `\providecommand` agit comme `\newcommand` et la définit ; dans le cas contraire la définition existante reste inchangée.

Le nombre d'arguments, dans lequel l'argument optionnel doit être compté, est compris entre 0 et 9. Si la commande n'a pas d'arguments, le `[0]` peut être omis. Dans la partie *définition de la commande* les arguments sont référencés de `#1` à `#`*narg* ; si la commande a été déclarée avec un argument optionnel, alors celui-ci porte le numéro un (`#1`).

Pour illustrer ce qui précède, comparons les déclarations suivantes, où l'on définit successivement une commande sans arguments, avec un argument obligatoire, une argument optionnel, et finalement avec deux arguments, le premier optionnel, et le deuxième obligatoire. Notons en particulier le cas des commandes `\seqo` et `\seqom`, qui peuvent avoir un argument optionnel ; si elles sont spécifiées sans leur argument optionnel, la valeur par défaut est substituée.

```
\newcommand{\seq}{x_{0},\ldots\,x_{n}}
\newcommand{\seqm}[1]{#1_{0},\ldots\,#1_{n}}
\newcommand{\seqo}[1][k]{x_{0},\ldots\,x_{#1}}
\newcommand{\seqom}[2][k]{#2_{0},\ldots\,#2_{#1}}
\[\seq      \quad \seqm{z}     \qquad
  \seqo     \quad \seqo[l]     \qquad
  \seqom{y}\quad \seqom[i]{q}       \]
```

Ce qui donne :

$$x_0,\ldots\,x_n \quad z_0,\ldots\,z_n \qquad x_0,\ldots\,x_k \quad x_0,\ldots\,x_l \qquad y_0,\ldots\,y_k \quad q_0,\ldots\,q_i$$

Pour garantir qu'une commande fonctionnera correctement tant en mode mathématique, qu'en mode texte, on doit attacher une importance particulière à sa définition. Pour rendre cette tâche plus facile, LaTeX 2ε offre la commande suivante :

`\ensuremath{`*source LaTeX à composer en mode mathématique*`}`

Comme son nom l'indique la commande `\ensuremath` garantit que son argument est toujours composé en mode mathématique en l'entourant, si nécessaire, par des signes `$`. En particulier, les deux premières lignes de l'exemple ci-dessus peuvent être réécrites de la façon suivante :

```
\renewcommand{\seq}{\ensuremath{x_{0},\ldots\,x_{n}}}
\renewcommand{\seqm}[1]{\ensuremath{#1_{0},\ldots\,#1_{n}}}
\seq,\quad\seqm{z} ou \(\seq,\quad\seqm{z}\)
```

$x_0, \ldots x_n, \quad z_0, \ldots z_n$ ou $x_0, \ldots x_n, \quad z_0, \ldots z_n$

3.2. La (re)définition des environnements

Dans LaTeX 2.09 les environnements sont définis ou redéfinis en utilisant les commandes suivantes :

`\newenvironment{`*monenv*`}[`*narg*`]{`*déf_entrée*`}{`*déf_sortie*`}`
`\renewenvironment{`*monenv*`}[`*narg*`]{`*déf_entrée*`}{`*déf_sortie*`}`

Comme pour les commandes LaTeX le nombre d'arguments d'un environnement doit être compris entre zéro et neuf, et, si l'environnement n'a pas d'arguments, le `[0]` peut être omis. Dans la partie définition, *déf_entrée*, ces arguments sont référencés de `#1` à `#`*narg*. Notons que les arguments peuvent uniquement être spécifiés lorsque l'on *entre* l'environnement avec la commande `\begin{monenv}` comme indiqué ci-dessous.

`\begin{monenv}{`arg_1`}...{`arg_k`}`

Lorsqu'on *quitte* l'environnement avec la commande `\end{monenv}` aucun paramètre ne peut être spécifié. En plus, les valeurs des arguments spécifiés avec la commande `\begin{monenv}` à l'entrée de l'environnement (voir ci-dessus) ne sont plus disponibles dans la partie « sortie » *déf_sortie* de la définition où sont groupées toutes les actions que LaTeX doit exécuter en quittant l'environnement *monenv*.

En analogie avec les commandes, LaTeX 2ε offre également pour les environnements la possibilité d'avoir un argument optionnel (en première position seulement).

`\newenvironment{`*monenv*`}[`*narg*`][`*valeur_par_défaut*`]{`*déf_entrée*`}{`*déf_sortie*`}` [*valeur_par_défaut*] spécifie la valeur par défaut à utiliser pour l'argument optionnel au cas où il ne serait pas donné explicitement lors de l'utilisation de l'environnement avec la commande `\begin{monenv}`. Dans la partie « entrée » *déf_entrée* de la définition de l'environnement les arguments sont référencés de `#1` à `#`*narg* ; si l'environnement a un argument optionnel, alors celui-ci porte le numéro un (`#1`).

Comme exemple d'un environnement utilisant un argument optionnel, construisons une variante, appelée `Defliste`, de l'environnement LaTeX standard `description`. Cet environnement `Defliste` se comporte comme en environnement `description` normal lorsqu'il est utilisé sans argument optionnel. Toutefois s'il a un argument optionnel, sa valeur est utilisée pour définir la largeur du terme de la liste de description. Ainsi, en spécifiant le texte du terme le plus large comme argument optionnel à la liste `Defliste`, on est certain que toutes les parties de définition de la liste sont alignées.

L'utilisation de la liste `Defliste` est montrée plus en détails ci-dessous, où dans le premier cas (sans argument optionnel), elle se comporte comme `description` (à part que nous utilisons la commande `\emph` pour mettre en évidence le terme ou l'étiquette de la liste), alors que, dans le deuxième cas, en utilisant la largeur du texte le plus large, la partie descriptive est justifiée à gauche.

```
\newenvironment{Defliste}[1][\quad]%
 {\begin{list}{}{\renewcommand{\makelabel}[1]{\emph{##1}\hfil}%
                \settowidth{\labelwidth}{\emph{#1}}%
                \setlength{\leftmargin}{\labelwidth+\labelsep}}}
 {\end{list}}
\begin{Defliste}
  \item[Court]            Ceci est un terme court.
  \item[Plus long]        Ceci montre un terme plus long.
  \item[Encore plus long] Et voici le terme le plus long.
\end{Defliste}
\begin{Defliste}[Encore plus long]
    .....
\end{Defliste}
```

Court Ceci est un terme court.

Plus long Ceci montre un terme plus long.

Encore plus long Et voici le terme le plus long.

Court	Ceci est un terme court.
Plus long	Ceci montre un terme plus long.
Encore plus long	Et voici le terme le plus long.

`sp`	le point réduit (*scaled point*), 65536 sp = 1 pt, la plus petite unité TeX	
`pt`	le point, 1 pt = 1/72,27 in = 0,351 mm	
`bp`	le gros point (*big point*), ou point PostScript, 72 bp = 1 in	
`dd`	le point Didot, égal à 1/72 d'un pouce français, 1 dd = 0,376 mm	
`mm`	le millimètre, 1 mm = 2,845 pt	
`pc`	le pica, 1 pc = 12 pt = 4,218 mm = 0,166 in	
`cc`	le cicéro, 1 cc = 12 dd = 4,531 mm	
`cm`	le centimètre, 1 cm = 10 mm = 2,371 pc	
`in`	le pouce anglo-saxon (*inch*), 1 in = 25,4 mm = 72,27 pt	
`ex`	hauteur de la lettre minuscule «x» dans la police courante	
`em`	largeur de la lettre majuscule «M» dans la police courante	
`mu`	unité mathématique (*math unit*), 18 mu = 1 em	

TAB. 1 - *Les unités disponibles avec LaTeX*

4. La manipulation des longueurs

Les longueurs (dimensions) LaTeX *fixes* (anglais «rigid») ou *élastiques* (anglais «rubber») sont déclarées, définies et modifiées à l'aide des commandes suivantes :

`\newlength{`*cmd*`}`	`\setlength{`*cmd*`}{`*longueur*`}`
`\addtolength{`*cmd*`}{`*longueur*`}`	
`\settowidth{`*cmd*`}{`*texte*`}`	`\width`
`\settoheight{`*cmd*`}{`*texte*`}`	`\height`
`\settodepth{`*cmd*`}{`*texte*`}`	`\depth`
	`\totalheight`

LaTeX 2ε introduit les nouvelles commandes `\settoheight` et `\settodepth`, en analogie avec la commande `\settowidth`, qui existait déjà en LaTeX 2.09. Elles permettent, respectivement, de «mesurer» la hauteur et la profondeur de matériel TeX. Parallèlement, les longueurs `\width`, `\height`, `\depth` et `\totalheight`, également nouvellement introduites dans LaTeX 2ε, peuvent être utilisées avec les commandes pour manipuler les boîtes LaTeX discutées plus bas à la section 5. Le tableau 1 montre les différentes unités dans lesquelles on peut exprimer les longueurs LaTeX.

Les lignes suivantes montrent comment déclarer, définir, changer ou utiliser les longueur LaTeX. Notons que dans la plupart des cas on peut utiliser tant les longueurs fixes que les longueurs élastiques.

```
% Déclaration de la longueur \MaLgr
\newlength{\MaLgr}                          MaLgr = \the\MaLgr
```

MaLgr = 0.0pt

```
% Définition et utilisation d'une longueur fixe
\setlength{\MaLgr}{10mm}                    MaLgr = \the\MaLgr
```

MaLgr = 28.45274pt

```
% Définition et utilisation d'une longueur fixe
\setlength{\MaLgr}{5mm plus 1mm minus .5mm}  MaLgr = \the\MaLgr
```

MaLgr = 14.22636pt plus 2.84526pt minus 1.42262pt

```
% Exemple d'une opération arithmétique
\setlength{\MaLgr}{1em plus 1ex}
Un em avec une partie élastique d'un ex est égal à \the\MaLgr.
\addtolength{\MaLgr}{1pc}
En y ajoutant un pica on obtient \the\MaLgr.
```

Un em avec une partie élastique de 1 ex est égal à 10.0pt plus 4.47998pt. En y ajoutant un pica on obtient 22.0pt plus 4.47998pt.

```
% Utilisation des commandes pour mesurer les dimensions d'une boîte.

\settoheight{\MaLgr}{Le mois de juin} La hauteur du texte est \the\MaLgr
\settoheight{\MaLgr}{\Large Le mois de juin} et maintenant  \the\MaLgr.
```

La hauteur du texte est 6.74994pt et maintenant 8.09993pt.

```
\settodepth{\MaLgr}{Le mois de juin} La profondeur du texte est \the\MaLgr
\settodepth{\MaLgr}{\Large Le mois de juin} et maintenant \the\MaLgr.
```

La profondeur du texte est 2.08496pt et maintenant 2.50195pt.

```
\settowidth{\MaLgr}{Le mois de juin} La largeur du texte est \the\MaLgr
\settowidth{\MaLgr}{\Large Le mois de juin} et maintenant \the\MaLgr.
```

La largeur du texte est 62.49933pt et maintenant 74.99922pt.

Les longueurs élastiques (variables) sont particulièrement intéressantes comme outil de micro-typographie, parce qu'elles permettent un contrôle fin du placement de l'information sur la page. LaTeX offre dans ce domaine quelques commandes prédéfinies.

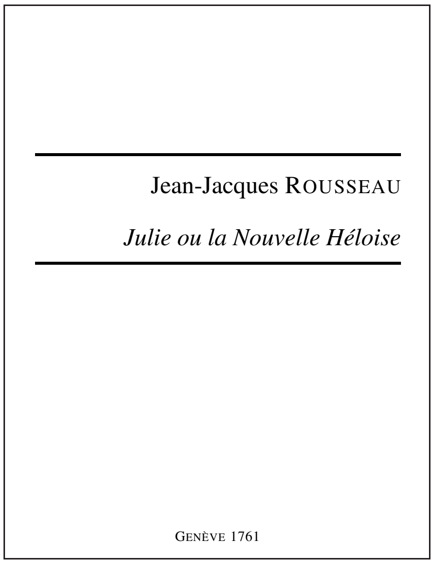
La commande `\fill` représente une longueur élastique avec une longueur naturelle égale à zéro. La longueur `\fill` peut s'étirer jusqu'à n'importe quelle valeur positive. Sa définition ne peut pas être modifiée.

La commande `\stretch{`*nombre_décimal*`}` offre plus de flexibilité par la présence de son argument, qui spécifie l'élasticité par rapport à celle de `\fill`, c.-à-d., `\fill` est équivalent à `\stretch{1}`. Plus généralement, `\stretch{`*nom_déc*`}` a une élasticité *nom_déc* fois celle de `\fill`. Cette caractéristique est très commode pour l'ajustement précis de textes, horizontalement ou verticalement, comme le montrent les exemples ci-dessous :

```
\newcommand{\EH}[1][1.]{\hspace{\stretch{#1}}}
\begin{center}
  gauche \hfill                            droite\\
  gauche \EH[.5]\fbox{$\frac{1}{3}$}\hfill droite\\
  gauche \EH         milieu \hfill         droite\\
  gauche \hrulefill\ milieu \hrulefill\    droite\\
  gauche \dotfill\                         droite\\
  gauche \dotfill\ \EH[.5] \dotfill\       droite\\
  gauche \dotfill\ \EH     \dotfill\       droite\\
  gauche \dotfill\ \EH[2.] \dotfill\       droite
\end{center}
```

gauche droite
gauche $\boxed{\frac{1}{3}}$ droite
gauche milieu droite
gauche ________________ milieu ________________ droite
gauche .... droite
gauche .... droite
gauche .... droite
gauche .... droite

```
\documentclass{article}
\usepackage{t1enc,times}
\thispagestyle{empty}
\newcommand{\HRule}%
           {\rule{\linewidth}{1mm}}
\setlength{\parindent}{0mm}
\begin{document}
  \vspace*{\stretch{1}}
  \HRule
  \begin{flushright}
    \Huge
    Jean-Jacques \textsc{Rousseau}\\[8mm]
    \emph{Julie ou la Nouvelle Héloise}
  \end{flushright}
  \HRule
  \vspace*{\stretch{2}}
  \begin{center}
    \Large\textsc{Genève 1761}
  \end{center}
\end{document}
```



5. Plusieurs types de boîtes

La notion de «boîtes» est au cœur même du modèle de composition du moteur typographique de TeX, tel qu'il fut conçu par Knuth, et LaTeX offre donc plusieurs commandes pour faciliter leur utilisation.

5.1. Les boîtes simples

`\mbox{`*texte*`}`	`\makebox[`*largeur*`][`*positionnement*`]{`*texte*`}`
`\fbox{`*texte*`}`	`\framebox[`*largeur*`][`*positionnement*`]{`*texte*`}`

Le paramètre (optionnel) de positionnement a une valeur de défaut égale à `[c]`, ce qui centre les texte à l'intérieur de la boîte. D'autres possibilités sont `[l]`, pour un ajustement à gauche, ou `[r]`, pour un ajustement à droite. En plus, LaTeX 2ε offre la spécification `[s]` qui étale le texte (en jouant sur l'espacement entre les mots) de façon qu'il occupe toute la largeur de la boîte en question. Comme noté précédemment à la section 4, LaTeX 2ε admet également l'utilisation dans la partie *largeur* des commandes de manipulation des boîtes les paramètres de longueur spéciaux `\width`, `\height`, `\depth` et `\totalheight`. Ces paramètres contiennent les dimensions naturelles (largeur, hauteur et profondeur) de l'argument *texte* ; `\totalheight` (la hauteur totale) correspond à la somme de `\height` et `\depth`.

Les exemples suivants montrent comment ces différents paramètres peuvent être utilisés pour contrôler la composition à l'intérieur d'une boîte [2].

```
\par\framebox                 {Ceci est une petite boîte \LaTeX.}
\par\framebox[\width + 15mm][s]{Ceci est une petite boîte \LaTeX.}
\par\framebox[1.7\width]       {Ceci est une petite boîte \LaTeX.}
```

Ceci est une petite boîte LaTeX.

Ceci est une petite boîte LaTeX.

Ceci est une petite boîte LaTeX.

5.2. Les filets

Les filets sont des rectangles remplis d'encre qui sont surtout utilisés pour tirer des traits.

`\rule[`*positionnement_vertical*`]{`*largeur*`}{`*hauteur_totale*`}`

La commande `\rule` dessine un filet d'une hauteur *hauteur_totale* et d'une largeur de *largeur*. Le positionnement est contrôlé par l'argument *positionnement_vertical*, qui

2. Dans cet exemple les opérations arithmétiques à l'intérieur du premier argument optionnel *largeur* nécessite le chargement au préalable de l'extension calc.

spécifie à quelle position la base du filet sera placée verticalement, une valeur positive indiquant plus haut, et une valeur négative plus bas que la ligne courante. Les filets sont également commodes pour manipuler la hauteur d'une boîte. En combinaison avec les paramètres LaTeX qui définissent la hauteur et la profondeur d'une boîte, les filets permettent des ajustements micro-typographiques fins de la représentation visuelle des divers éléments d'un document.

```
\newsavebox{\BAutomne}
\savebox{\BAutomne}{\Large L'automne commence en septembre}
\newlength{\Mpr}\settodepth{\Mpr}{\usebox{\BAutomne}}
\newlength{\Mht}\settoheight{\Mht}{\usebox{\BAutomne}}
\addtolength{\Mht}{\Mpr}

\framebox[1.6\width+1em][s]{\usebox{\BAutomne}}
\par
\framebox[1.6\width+1em][s]{\usebox{\BAutomne}%
          \rule[-2\Mpr]{0mm}{2\Mht}}
```

L'automne commence en septembre

L'automne commence en septembre

Les boîtes de largeur zéro sont également très utiles en d'autres circonstances.

```
\begin{center}
Une phrase centrée.\makebox[0cm][l]{$^{123}$}\\
Un peu de texte au milieu.                 \\
\makebox[0cm][r]{$^{321}$}Une phrase centrée.\\
\end{center}
```

Une phrase centrée.^{123}
Un peu de texte au milieu.
^{321}Une phrase centrée.

```
\makebox[0cm][r]{\(\Leftrightarrow\)}%
Au début de la ligne courante on a placé une boîte avec
une largeur zéro dont le contenu peut dépasser dans la marge.
```

$\Leftrightarrow$Au début de la ligne courante on a placé une boîte avec une largeur zéro dont le contenu peut dépasser dans la marge.

5.3. Le placement des boîtes

Les boîtes peuvent être positionnées verticalement avec la commande :

`\raisebox{`*déplacement_vertical*`}[`*profondeur*`][`*hauteur*`]{`*texte*`}`

Les deux arguments optionnels *profondeur* et *hauteur* demandent à TeX d'utiliser les valeurs indiquées pour la profondeur, respectivement la hauteur, de la boîte dans ses calculs de positionnement. S'ils ne sont pas spécifiés, TeX utilisera les valeurs actuelles correspondant à l'argument *texte*. Le exemple simple ci-dessous devrait clarifier les principes d'utilisation de cette commande.

```
\begin{flushleft}
x111x \raisebox{-1ex}{vers le bas} x222x          \\     % Déplacement
x333x \raisebox{1ex}{vers le haut} x444x          \\[1em]% visible
x111x \raisebox{-1ex}[0cm][0cm]{vers le bas} x222x\\     % Déplacement
x333x \raisebox{1ex}[0cm]{vers le haut} x444x            % invisible
\end{flushleft}
```

x111x vers le bas x222x
x333x vers le haut x444x

x111x vers le bas x222x
x333x vers le haut x444x

Un exemple probablement plus utile est le suivant, qui montre comment on peut insérer du texte « entre » deux lignes d'un tableau (en « dissimulant » le contenu de la boîte, pour que TeX l'ignore dans ses calculs).

```
\begin{center}
\begin{tabular}{|c|c|c|}                                \hline
         & \multicolumn{2}{c|}{Le titre}              \\\cline{2-3}
\raisebox{1.5ex}[0cm][0cm]{Entre les lignes un et deux}
         & A        & B                               \\\hline
20000000 & 10       & 10                              \\\hline
\end{tabular}
\end{center}
```

Entre les lignes un et deux	Le titre	
	A	B
20000000	10	10

Finalement, quand votre pilote d'imprimante le permet, vous pouvez générer des rotations pour les boîtes. Dans cet exemple on voit aussi clairement l'utilité des différents paramètres LaTeX 2_ε décrivant les dimensions de la boîte.

```
\newcommand{\DoT}[1]{\begin{turn}{45}#1\end{turn}}
x1 \DoT{\fbox{Pas très bien.}} x2
\DoT{\raisebox{\depth}{\fbox{Ceci est parfait.}}} x3
\DoT{\raisebox{-\height}{\fbox{Ce choix n'est pas bon.}}} x4
```

x1 Pas très bien. x2 Ceci est parfait. x3 Ce choix n'est pas bon. x4

5.4. La commande `\parbox` et l'environnement `minipage`

Avec LaTeX 2.09 des boîtes qui peuvent contenir plus d'un paragraphe sont construits avec `parbox` et `minipage`, comme suit :

`\parbox[`*positionnement*`]{`*largeur*`}{`*texte*`}`
`\begin{minipage}[`*positionnement*`]{`*largeur*`}`
    texte
`\end{minipage}`

Les paramètres *positionnement* et *largeur* sont identiques à ceux décrits pour les boîtes simples à la section 5.1. Notons toutefois que l'ordre dans lequel ces arguments sont spécifiés est inversé. Voici un simple exemple d'utilisation :

```
\parbox{.3\linewidth}{Ceci est le contenu de la parbox de gauche.}
\hfill \emph{\large Ligne centrale} \hfill
\parbox{.3\linewidth}{Et ceci est la parbox de droite.\\
                     L'ajustement du texte n'est pas parfait comme
                     \LaTeX{} a pas mal de peine à optimiser le
                     texte dans les paragraphes à l'intérieur
                     de colonnes étroites.}
```

Ceci est le contenu de la parbox de gauche.	*Ligne centrale*	Et ceci est la parbox de droite. L'ajustement du texte n'est pas parfait comme LaTeX a pas mal de peine à optimiser le texte dans les paragraphes à l'intérieur de colonnes étroites.

Parfois il est utile de pouvoir définir la dimension verticale d'une boîte de type paragraphe. C'est pourquoi LaTeX 2ε a introduit des arguments optionnels supplémentaires pour `minipage` et `\parbox`.

`\parbox[`*pos_externe*`][`*hauteur*`][`*pos_interne*`]{`*largeur*`}{`*texte*`}`

`\begin{minipage}[`*pos_externe*`][`*hauteur*`][`*pos_interne*`]{`*largeur*`}`
texte
`\end{minipage}`

Alors que l'argument *pos_externe* contrôle la position verticale de la boîte vis-à-vis de la ligne courante, l'argument nouvellement introduit *pos_interne* spécifie la position du matériel textuel *texte* à l'intérieur de la boîte de hauteur *hauteur*. Les valeurs de *pos_interne* peuvent être `[t]`, `[c]`, `[b]` ou `[s]`. Si cet argument est absent la valeur de l'argument *pos_externe* sera utilisée. D'une certaine façon on peut considérer les argument « verticaux » *hauteur* et *pos_interne* comme les équivalents des arguments « horizontaux » *largeur* et *positionnement* de `\makebox`. En utilisant la spécification `[s]` pour *pos_externe* le matériel textuel *texte* s'étalera verticalement pour occuper tout l'espace défini par *hauteur*. Dans ce cas c'est l'utilisateur qui est responsable de l'introduction de commandes d'espacement élastique, en utilisant, par exemple, des commandes `\vspace` et `\stretch` ou `\vfill`.

Tout comme avec les autres commandes qui manipulent ses boîtes, on peut utiliser les paramètres `\height`, `\depth` et `\totalheight` pour faire référence aux dimensions naturelles de la boîte dans l'argument optionnel *hauteur*.

```
xx \fbox{\parbox[b][1.6\height][s]{36mm}
        {Ce texte vient en haut.   \par\vspace{\stretch{3}}
         Celui-ci au milieu.       \par\vspace{\stretch{1}}
         Et ces quelques lignes se trouvent en bas de la boîte.}}
   \fbox{\parbox[b][\height+\baselineskip][s]{40mm}
        {Cette fois on a plusieurs lignes dans la partie
         supérieure de la boîte.
         Mais seulement une seule ligne \par\vfill
         ici en bas.}} xx
```

xx	Ce texte vient en haut. Celui-ci au milieu. Et ces quelques lignes se trouvent en bas de la boîte.	Cette fois on a plusieurs lignes dans la partie supérieure de la boîte. Mais seulement une seule ligne ici en bas.	xx

5.5. La mise en boîte de matériel LaTeX

Du matériel LaTeX peut être composé une seule fois et ensuite sauvegardé dans une boîte étiquetée, d'où il peut être extrait plus tard. LaTeX a plusieurs commandes pour réaliser ces fonctions.

`\newsavebox{`*cmd*`}`	déclarer une nouvelle boîte
`\sbox{`*cmd*`}{`*texte*`}`	remplir une boîte
`\savebox{`*cmd*`}[`*largeur*`][`*positionnement*`]{`*texte*`}`	remplir une boîte
`\begin{lrbox}{`*cmd*`}` *texte* `\end{lrbox}`	remplir une boîte
`\usebox{`*cmd*`}`	utiliser le contenu d'une boîte

L'environnement `lrbox` est une addition de $\LaTeX 2_\varepsilon$. Les arguments des commandes `\sbox` et `\savebox` sont équivalents à ceux à la section 5.1 pour `\mbox` et `\makebox`. Avant remplissage ou utilisation, la boîte *cmd* doit toujours être créée auparavant à l'aide d'une commande `\newsavebox`. Les commandes `\sbox`, `\savebox` et l'environnement `lrbox` sauvegardent le matériel textuel spécifié à l'aide de l'argument *texte* dans cette boîte et son contenu peut être utilisé ultérieurement avec une commande `\usebox`. `lrbox` est en quelque sorte une variante généralisée sous forme d'environnement de `\sbox`. L'environnement `lrbox` est très utile si l'on veut sauvegarder le contenu d'un autre environnement dans une boîte pour traitement ultérieur.

Pour illustrer cela, l'exemple suivant définit un environnement `fminipage` qui agit comme un environnement `minipage`, mais qui ajoute un cadre autour de son contenu (une variante généralisée de `\fbox`). Dans la définition de l'environnement on remarquera l'utilisation de l'argument optionnel pour contrôler la largeur de la mini page encadrée. En plus on peut y introduire du matériel en mode verbatim. Pour la partie arithmétique de l'exemple, il faut d'abord charger l'extension `calc`.

```
\newsavebox{\fminiboite}
\newlength{\fminilongueur}
\newenvironment{fminipage}[1][\linewidth]% largeur par défaut
{\setlength{\fminilongueur}{#1-2\fboxsep-2\fboxrule}%
   \begin{lrbox}{\fminiboite}
     \begin{minipage}{\fminilongueur}
}{%
     \end{minipage}
   \end{lrbox}
   \noindent\fbox{\usebox{\fminiboite}}%
}

\begin{fminipage}
  Cet environnement peut contenir du texte en mode
```

```
  verbatim \verb=\fminiboite=.

  Voici un deuxième paragraphe qui contient des maths \(e^{\pi}\).
\end{fminipage}
```

Cet environnement peut contenir du texte en mode verbatim `\fminiboite`.
Voici un deuxième paragraphe qui contient des maths e^{π}.

```
\begin{fminipage}[.5\linewidth]
  ....
\end{fminipage}
```

Cet environnement peut contenir du texte en mode verbatim `\fminiboite`.
Voici un deuxième paragraphe qui contient des maths e^{π}.

6. Les commandes de sélection de police

Cette section présente les commandes utilisateurs disponibles en LaTeX 2_ε pour spécifier des polices, tant en mode mathématique, qu'en mode texte. Quelques-unes des extensions les plus connues pour mettre en place une nouvelle famille de police seront présentées et quelques précisions sur la compatibilité avec les commandes LaTeX 2.09 seront données.

6.1. Présentation des nouvelles commandes

La première question qu'on peut se poser est pourquoi les développeurs de LaTeX 2_ε ont décidé d'introduire toute une panoplie de nouvelles commandes pour gérer les choix des polices. Pour tenter de répondre à cette question mentionnons quelques particularités des commandes de sélection des polices dans LaTeX 2.09 :

- leur syntaxe, c.-à-d. `{\it texte}` au lieu de `\it{texte}`, ce qui est différent de la syntaxe utilisée pour toutes les autres commandes LaTeX (sauf celles pour changer la taille des caractères) qui sont spécifiées avec des arguments ;
- les commandes spécifiant les changements de polices n'agissent pas d'une façon *orthogonale*, ce qui veut dire que, par exemple, `\bf\sf` produira du sans serif normal, en d'autres termes seule la dernière commande est honorée ;
- certaines substitutions de polices avaient lieu « à notre insu », par exemple, `\tiny\tt` produisait une petite police mais en caractères romains, parce qu'on supposait qu'à de si petites tailles, la différence entre les styles romain ou machine à écrire est pratiquement invisible, d'où la décision d'utiliser la police romaine à la même taille, qui est déjà chargée ;

– les corrections italiques doivent être introduites à la main, c.-à-d. on doit écrire `{\em mon texte\/}`, et même cette approche n'est pas correcte dans tous les cas.

LaTeX 2_ε résoud ces problèmes en introduisant les nouvelles commandes suivantes pour choisir les polices dans la partie texte d'un document :

`\textmd{`Un texte en caractères normaux`}`
`\textbf{`**Un texte en gras**`}`
`\textup{`Un texte en caractères droits`}`
`\textit{`*Un texte en italique*`}`
`\textsl{`*Un texte en caractères inclinés*`}`
`\textsc{`UN TEXTE EN PETITES MAJUSCULES`}`
`\textrm{`Un texte en caractères romains`}`
`\textsf{`Un texte en caractères linéales (sans serifs)`}`
`\texttt{Un texte en style machine à écrire (à chasse fixe)}`

`\emph{`*Un texte en mode emphatique*`}`.

Les commandes permettant de modifier la taille des caractères restent inchangées, c.-à-d. `\large`, `\scriptsize`, etc. fonctionnent comme précédemment et ne prennent *pas* d'argument ; elles doivent donc impérativement être utilisées à l'intérieur d'un groupe (implicite ou explicite), comme un environnement ou une paire d'accolades.

Ces commandes ne souffrent pas des inconvénients des commandes LaTeX 2.09 énoncés ci-dessus, pour les raisons suivantes :

– la syntaxe est similaire à celle des autres commandes LaTeX ;
– `\textbf{\textsf{`**texte**`}}` produit du gras sans serif ;
– `{\tiny\texttt{texte}}` produit des petits caractères en style machine à écrire ;
– `\emph{`*texte*`}` n'a *pas* besoin d'un `\/`.

Cependant il reste encore quelques restrictions, `\textbf{\texttt{texte}}`, par exemple, produira le texte en style machine à écrire normal, pas gras, parce qu'il n'y a pas de variante grasse de la police « Computer Modern typewriter », associée en LaTeX standard à la commande `\texttt`, mais au moins LaTeX 2_ε affiche un message d'avertissement signalant la substitution.

Dans le domaine des polices mathématiques LaTeX 2_ε a les nouvelles commandes suivantes, qui fonctionnent uniquement en mode mathématique :

`\mathnormal{`$Texte\ normal\ en\ math\ italique$`}`
`\mathcal{`$\mathcal{MATH\ CALLIGRAPHIQUE}$`}`
`\mathrm{`$\mathrm{Texte\ romain\ en\ mode\ mathématique}$`}`
`\mathbf{`$\mathbf{Texte\ gras\ en\ mode\ mathématique}$`}`
`\mathsf{`$\mathsf{Texte\ sans\ serif\ en\ mode\ mathématique}$`}`
`\mathit{`$\mathit{Texte\ italique\ en\ mode\ mathématique}$`}`
`\mathtt{`$\mathtt{Texte\ style\ machine\ à\ écrire\ en\ mode\ mathématique}$`}`

Extension	Police sans serif	Police romaine	Police machine à écrire
times	Helvetica	Times	Courier
palatino	Helvetica	Palatino	Courier
newcent	AvantGarde	NewCenturySchoolbook	Courier
bookman	AvantGarde	Bookman	Courier
avant	AvantGarde		
helvet	Helvetica		
nimbus	URW Nimbus sans	URW Nimbus Roman	
charter		Bitstream Charter	
utopia		Adobe Utopia	
garamond	Optima	Garamond	Courier
basker	Univers	Baskerville	Courier
mtimes	Univers	Monotype Times	cmtt
bembo	Optima	Bembo	Courier
lucid	LucidaSans	Lucida	Courier
lucidbrb	LucidaSans	LucidaBright	LucidaSansTypewriter
lucidbry	LucidaSans	LucidaBright	LucidaSansTypewriter

TAB. 2 - *Les polices chargés par les diffèrentes extensions du système PSNFSS*

6.2. Choisir ses polices

Il est maintenant relativement aisé de remplacer les polices Computer Modern par d'autres familles, pour autant qu'elles soient installées dans un des répertoires où TeX peut les trouver. Plusieurs extensions pour utiliser des polices populaires ont vu le jour, parmi lesquelles on trouve :

– le système PSNFSS, qui donne accès à plusieurs familles PostScript ;
– `\usepackage{amssymb}`, qui donne accès aux polices AMS ;
– `\usepackage{pandora}`, qui donne accès aux polices Pandora ;
– `\usepackage{euler}`, qui donne accès à la famille Euler de Hermann Zapf.

Discutons un peu plus en détail le système PSNFSS, dont les principales composantes sont regroupés dans le tableau 2. Comme le montre ce tableau, ces extensions remplacent en général les trois familles de base pour la partie texte d'un document, c.-à-d., sans serif, romain et machine à écrire avec la famille PostScript indiquée dans chaque cas. Les extensions dans la partie supérieure du tableau font seulement référence aux polices disponibles dans la mémoire morte (ou sur le disque) des imprimantes PostScript, alors que les extensions dans la seconde partie du tableau utilisent des polices que l'on doit acheter séparément (à part Courier et cmtt).

Les polices mathématiques ne seront pas affectées par les extensions mentionnées ci-dessus, sauf dans le cas où l'extension en question contient également les polices adéquates. Ainsi est-il possible de remplacer une famille de polices mathématiques (qui définit les groupes mathématique, comme math italique, symboles et caractères d'extension) par une autre famille en chargeant les extensions suivantes (et en achetant les polices nécessaires, le cas échéant) :

– lucmath : remplace les polices Computer Modern par Lucida Math ;
– lucibrb (lucibry) : les remplace par Lucida Bright maths de Y&Y ;
– mathtime : de Michael Spivak les remplace par ses propres polices *mathtime* ;
– mathptm : d'Alan Jeffrey les remplace par la police Symbol (mais nécessite encore quelques caractères des polices Computer Modern).

6.3. La compatibilité

Les anciennes commandes LaTeX 2.09 pour la sélection des polices (`\rm`, `\bf`, etc.) sont toujours disponibles en LaTeX 2ε, mais elles ne font plus partie du «noyau» de LaTeX. Elles doivent maintenant êtres définies dans les fichiers de classe, où les définitions des commandes pour changer la taille des caractères (comme `\huge` et `\tiny`) ont toujours résidées. C'est donc le concepteur d'une nouvelle classe de document qui doit décider de la définition exacte de ces anciennes commandes. En ce qui concerne les «classes standards», comme article et book, ces commandes donnent le même résultat qu'auparavant.

Une autre remarque importante est qu'en mode compatibilité, c.-à-d. en traitant un document commençant avec `\documentstyle`, LaTeX 2ε émule LaTeX 2.09 *sans* NSSP. Pour émuler LaTeX 2.09 *avec* NSSP, il faut utiliser :

```
\documentstyle[newlfont]{...}
```

7. Les classes standard LaTeX 2ε

Cette section présente les fichiers les plus importants qui font partie de la distribution LaTeX 2ε et discute quelques unes des extensions qui sont déjà mises à jour pour être utilisées avec LaTeX 2ε.

Les fichiers associés à LaTeX 2ε sont caractérisés par les suffixes :

– *nom*`.cls` : les fichiers de classe ;
– *nom*`.clo` : les fichiers d'options ;
– *nom*`.sty` : les fichiers d'extensions ;
– *nom*`.cfg` : les fichiers (optionnels) de configuration, qui sont lus à l'exécution et contiennent des commandes de personnalisation.

Les classes de document « standard » distribuées avec LaTeX 2ε sont article, report, book, letter, slide, proc, et ltxdoc. Disons quelques mots sur chacune d'entre elles :

article, report, book

– se comportent comme les anciens styles LaTeX 2.09 ;

- twocolumn et openbib ont été recodées comme options internes ;
- une série de nouvelles options internes a été ajoutée pour gérer le format et l'orientation de la page : a4paper, a5paper, b5paper, letterpaper, legalpaper, executivepaper et landscape.

letter

- se comporte comme l'ancien style LaTeX 2.09 ;
- une série de nouvelles options est mainenant disponible : a4paper, a5paper, b5paper, letterpaper, legalpaper et executivepaper.

slides

- offre les même fonctions que le format SLITeXavec LaTeX 2.09 ;
- on peut personnaliser la sélection des polices en spécifiant son choix dans le fichier de configuration `sfonts.cfg` ;
- une série d'options est disponible : a4paper, a5paper, b5paper, letterpaper, legalpaper, executivepaper et landscape ;
- l'option twocolumn n'est pas disponible.

proc

- pour faciliter la composition d'actes (anglais *proceedings*) de congrès ou conférences ;
- est basée sur la classe de document article ;
- les options a5paper, b5paper, onecolumn et titlepage sont désactivées.

ltxdoc

- est utilisée pour composer une version documentée de code source LaTeX ;
- est basée sur la classe de document article et nécessite l'extension doc ;
- utilise le fichier de configuration `ltxdoc.cfg` ;
- définit les commandes `\DocInclude` et `\GetFileInfo` ;
- désactive l'option a5paper.

Actuellement les extensions suivantes sont disponibles :

- graphics, pour inclure des fichiers graphiques externes contenant des images PostScript, gif, ou autres et pour exécuter des rotations ou déformations (agrandissement, retrécissement) des éléments du texte. Ceci suppose que le pilote d'imprimante supporte ces fonctions [3] ;
- ifthen, pour construire des structures de contrôle. En plus des commandes `\ifthenelse` et `\whiledo` précédemment disponibles avec LaTeX 2.09, on a introduit les commandes `\newboolean`, `\setboolean`, et `\boolean` ;
- makeidx et showidx pour la génération d'un index ;

3. Cette extension, ainsi que egraphics, sa généralisation, et la façon dont la couleur est supportée en LaTeXe, seront décrites dans un prochain numéro des Cahiers.

– doc et shortvrb pour faciliter la production de documentation des fichiers de classes et d'extensions ;
– oldlfont et newlfont comme fichiers de compatibilité avec la version 1 de NSSP de LaTeX 2.09 ;
– latexsym charge la police symbolique LaTeX `lasy`. Comme cette police n'est plus chargée par défaut par LaTeX 2_ε elle doit être chargée séparément lorsqu'on en a besoin. Ceci n'est toutefois pas nécessaire si l'on utilise les extensions amsfonts ou amssymb ;
– exscale définit des polices mathématiques supplémentaires ;
– eufrak et euscript donnent accès aux alphabets Euler Fraktur et Euler Script ;
– oldgerm définit les polices de caractères allemands anciens de Yannis Haralambous ;
– pandora définit la famille de polices Pandora de Neenie Billawala ;
– syntonly limite l'action de LaTeX à la seule vérification de la syntaxe du document ;
– tracefnt permet de suivre les actions de NSSP lorsque LaTeX traite le document ; le niveau de détail des informations est géré par les différentes options errorshow, warningshow, infoshow, et debugshow ;
– varioref généralise la commande `\ref` en ajoutant le numéro de page à la référence, si nécessaire en utilisant des textes modulables (comme «le tableau 1 à la page suivante» ou «la figure 2 à la page précédente»).

Plusieurs autres extensions disponibles sur les serveurs CTAN sont déjà compatibles avec LaTeX 2_ε ou le seront bientôt. Dans la première catégorie nous trouvons entre autres a4 (qui présente quelques extensions pour composer des pages A4, comme la commande `\widemargins`), epsfig, exams, labels, layout, la famille de classe de document NTG[4] artikel1, rapport3, etc., subeqnarray, psnfss, textfit. La deuxième catégorie contient la collection babel, changebar, supertabular, la collection TUGboat (ltugboat, etc.), la collection «Mainz», avec array, ftnright, multicol, theorem et verbatim.

8. Quelques trucs bons à connaître

Cette section décrit quelques caractéristiques de LaTeX 2_ε dont on n'aura probablement pas besoin tous les jours, mais qui pourront s'avérer utiles pour résoudre certains problèmes pratiques dans la composition des documents.

8.1. Contrôler la mise en page

Parfois, dans le stade *final* de la composition d'un document on peut vouloir donner un coup de main à LaTeX pour garantir que les changements de pages se font à des endroits adéquats pour optimiser la lisibilité du document. LaTeX 2.09 avait les commandes comme `\clearpage` et `\samepage`, qui n'étaient pas toujours d'une efficacité sans failles. De son

4. Nederlandse TeX Gebruikersgroep, ou le groupe des utilisateurs néerlandophones de TeX.

côté LaTeX 2_ε introduit en outre des commandes qui augmentent ou diminuent la hauteur de la page en cours d'une quantité *size* par rapport à sa taille «naturelle» (`\textheight`).

`\enlargethispage{`*size*`}` `\enlargethispage*{`*size*`}`

Par exemple, `\enlargethispage{-\baselineskip}` diminue la hauteur de la page en cours d'une ligne, alors que `\enlargethispage*{2\baselineskip}` l'allonge de deux lignes par rapport à sa hauteur naturelle.

La forme étoilée (avec astérisque) tente de rétrécir au maximum l'espace vertical élastique présent sur la page pour essayer d'inclure le plus d'information possible sur la page en cours.

8.2. Les éléments flottants

Une nouvelle commande et un nouvel identificateur permettent un contrôle plus fin de l'emplacement des éléments flottants dans un document.

`\suppressfloats[`*placement*`]`

Cette commande interdit tout placement supplémentaire d'élément flottant sur la page en cours. Le paramètre optionnel *placement* peut être `[t]` *ou* `[b]` (pas les deux en même temps), pour indiquer que la restriction s'applique seulement au placement d'éléments flottants en haut ou en bas de la page.

Nouvel identificateur de placement d'un élément flottant : `!`

Ce nouvel identificateur peut être utilisé en combinaison avec au moins un des anciens identificateurs de LaTeX 2.09 (`h`, `t`, `b` et `p`) dans l'argument optionnel déterminant le placement des éléments flottants.

Si l'identificateur `!` est spécifié, alors, seulement pour l'élément flottant en question, LaTeX ignore :

– toutes les restrictions sur le nombre d'éléments flottants sur la page en cours ;
– toutes les restrictions explicites sur la fraction de la page qui doit être occupée par des éléments flottants ou du texte.

Ce mécanisme essaie toutefois de générer des pages qui ne sont pas trop pleines, et d'imprimer les éléments flottants d'un type donné dans le bon ordre.

La présence de l'identificateur `!` n'a aucun effet pour les pages qui contiennent uniquement des éléments flottants. En plus, il supprime l'effet d'une commande `\suppressfloats` éventuelle pour l'élément flottant en question.

Annexe : pour inconditionnels seulement : comment fabriquer une commande avec deux arguments optionnels ?

Dans cet appendice on veut montrer comment on peut, en LaTeX 2ε standard, définir une commande ou un environnement avec plus d'un argument optionnel. Supposons, par exemple, que nous voulions construire une liste pour laquelle on ait la possibilité de spécifier à l'aide d'argument optionnels, non seulement la largeur de l'étiquette (comme avec l'environnement `Defliste` introduit à la section 3.2), mais aussi si la liste doit être composée d'une façon compacte ou non. Nous choisissons la syntaxe suivante :

```
\begin{Deflisteplus}[<largeur>][<style>]
```

Pour définir un environnement avec deux arguments optionnels, on utilisera une astuce en introduisant une approche en deux étapes [5]. L'exemple montre également comment on peut paramétrer différentes présentations typographiques de la liste en utilisant des commandes pour contrôler la largeur, la police et l'ajustement à l'intérieur de l'étiquette.

```
\newcommand{\Deflistepluslabel}[1]{%    police et ajustement
  \mbox{\Deflisteplusfont #1}\hfil}%    de l'étiquette

\newcommand\Deflisteplusfont{\itshape}% police de l'étiquette
\newcommand\Deflisteplusmargin{}%       largeur de la marge

%%%%%%%%%%%%%%%%%%%%%%%%% première étape %%%%%%%%%%%%%%%%%%%%%%%%%%
\newenvironment{Deflisteplus}[1]%       traite premier argument optionnel
 [\kern\leftmargin]%                    valeur par défaut de cet argument
 {%                                     Partie << entrée >> de
  %                                     l'environnement Deflisteplus
   \renewcommand{\Deflisteplusmargin}{#1}% redéfinition de la largeur de
                                        % la marge en utilisant le premier
                                        % argument optionnel de la liste
   \xDeflisteplus                       % exécuter \xDeflisteplus qui
                                        % s'occupe de la 2ième étape
 }%
 {\endlist}%                            Partie << sortie >> de
  %                                     l'environnement Deflisteplus.
  %                                     Elle termine la liste.

%%%%%%%%%%%%%%%%%%%%%%%%% deuxième étape %%%%%%%%%%%%%%%%%%%%%%%%%%
\newcommand{\xDeflisteplus}[1]%         traite deuxième argument optionnel
 [normal]%                              valeur par défaut de cet argument
{\list{}{%                              début de la liste
  \settowidth{\labelwidth}%             calcule la largeur de l'étiquette
  {\mbox{\Deflisteplusfont\Deflisteplusmargin}}% en utilisant le premier
                                        % argument optionnel sauvegardé
```

5. On aura besoin aussi des extensions calc et ifthen.

```
                                        % dans \Deflisteplusmargin
  \setlength{\itemindent}{0pt}%         le renfoncement initial
  \setlength{\leftmargin}{\labelwidth+\labelsep}% la marge de gauche
  \let\makelabel\Deflisteplusl abel%     comment composer l'étiquette
  \ifthenelse{\equal{#1}{compact}}%     utiliser 2ième argument optionnel
   {\setlength{\itemsep}{0pt}%          de l'environnement Deflisteplus.
    \setlength{\topsep}{.5\topsep}}{}%  Si <<compact>>, réduire l'espace
 }%                                     Fin définition de la liste
}%                                      Fin définition de \xDeflisteplus

Ceci est un peu de texte juste précédant la liste.
\begin{Deflisteplus}
  \item[Petite] Ceci est la première entrée dans la liste
  \item[Longue étiquette] Ceci est la deuxième entrée dans la liste.
  \item[] L'étiquette de cette entrée est absente.
\end{Deflisteplus}

\begin{Deflisteplus}[Longue étiquette]
  ...
\end{Deflisteplus}

\begin{Deflisteplus}[Longue étiquette][compact]
  ...
\end{Deflisteplus}
```

Ceci est un peu de texte juste précédant la liste.

Petite Ceci est la première entrée dans la liste

Longue étiquette Ceci est la deuxième entrée dans la liste.

L'étiquette de cette entrée est absente.

Ceci est un peu de texte juste précédant la liste.

Petite Ceci est la première entrée dans la liste

Longue étiquette Ceci est la deuxième entrée dans la liste.

L'étiquette de cette entrée est absente.

Ceci est un peu de texte juste précédant la liste.

Petite Ceci est la première entrée dans la liste

Longue étiquette Ceci est la deuxième entrée dans la liste.

L'étiquette de cette entrée est absente.

Références bibliographiques

[1] Michel Goossens, Frank Mittelbach et Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, USA, 1994.

[2] Leslie Lamport. *LaTeX—A Document Preparation System,* Seconde édition. Addison-Wesley, Reading, USA, 1994.

[3] Frank Mittelbach et Chris Rowley. LaTeX 2_ε, une nouvelle version de LaTeX. *La LETTRE GUTenberg*, 2:3–4, février 1994.

[4] Frank Mittelbach et Rainer Schöpf. Towards LaTeX 2.10. *TUGBoat*, 10(3):400–401, novembre 1990.

[5] Frank Mittelbach. E-TeX: Guidelines for future TeX extensions. *TUGBoat*, 11(3):337–345, septembre 1990.

[6] Frank Mittelbach et Chris Rowley". LaTeX 2.09 $\hookrightarrow$ LaTeX3. *TUGBoat*, 13(1):96-101, avril 1992.

[7] Chris Rowley. LaTeX3 update. *TUGBoat*, 13(3):390-391, octobre 1992.

[8] Frank Mittelbach, Chris Rowley et Michael Downes. Volunteer work for the LaTeX3 project. *TUGBoat*, 13(4):510-515, décembre 1992.

[9] Frank Mittelbach et Chris Rowley. Volunteer work for the LaTeX3 project. *TeX and TUG NEWS*, 3(1):7-11, janvier 1994.