La programmation en LaTeX

Denis ROEGEL

roegel@loria.fr

1997

Les classes standard donnent une présentation par défaut, qui ne convient pas forcément. Par exemple, taille des caractères, polices employées, marges, interligne, numérotation, etc.

Comment ceci peut-il être personnalisé ?

1. Comment changer une définition ?

1. Elle peut être amendée au début du fichier principal ;
2. Elle peut être modifiée dans la classe ou le *package*, après avoir *renommé* celui-ci.

Pour localiser une classe ou un *package*, on peut regarder le déroulement d'une session qui l'utilise :

```
This is TeX, Version 3.1415 (C version 6.1)
(ex-euro.tex
LaTeX2e <1996/12/01>
(/usr/local/lib/texmf/tex/latex/base/article.cls
...
```

ou même appeler LaTeX sur le fichier cherché (mais cela génère une erreur) :

```
This is TeX, Version 3.1415 (C version 6.1)
(/usr/local/lib/texmf/tex/latex/base/article.cls
LaTeX2e <1996/12/01>
...
```

2. Hiérarchie des fichiers

2.1. Fichiers du format

Lorsque LaTeX est lancé, un certain nombre de commandes sont disponibles. Ce sont celles qui figurent dans les fichiers suivants : `fontmath.ltx`, `fonttext.ltx`, `hyphen.ltx`, `latex.ltx`, `ltpatch.ltx`, `preload.ltx`, le fichier `latex.ltx` comportant la presque totalité des définitions.

Ces fichiers ne sont pas lus au moment d'une compilation mais sont utilisés au moment de la création du format LaTeX (fichier de macros précompilées).

Si ces fichiers existent sur le système, ils peuvent normalement être localisés en leur appliquant `latex`.

2.2. Classes, *packages*

Normalement, le premier fichier lu après le format est la classe (par exemple `book.cls`) et ce fichier peut lui-même en charger d'autres. Tous les fichiers chargés globalement (et donc pas ceux lus ligne à ligne avec les commandes de bas niveau de TeX) apparaissent normalement à l'écran (ils n'apparaissent pas en mode batch), ce qui permet de voir leur hiérarchie.

3. Rappels sur les mécanismes de définition

– `\def` : commande de base de TEX
– `\newcommand` : commande plus simple de LATEX (mais cette commande est définie avec `\def`)

3.1. Définition sans paramètres

```
\newcommand{\commande}{Ceci est l'expansion de la commande}
```

3.2. Définition avec paramètres

```
\newcommand{\commandeII}[1]{Ceci est l'expansion et le parametre est #1.}
```

On peut aussi définir une commande avec un paramètre optionnel.

3.3. Emploi de '%' dans les définitions

En général, on s'en sert pour supprimer la fin de ligne qui peut introduire un espace non désiré. Toutefois, '%' est inutile après un nom de commande car un nom de commande absorbe automatiquement l'espace qui suit.

4. Personnalisations diverses

Beaucoup de personnalisations consistent à modifier un fragment d'une classe ou d'un *package*. Si ce fragment utilise '@' dans une commande, il faut :

– ou bien entourer la redéfinition par `\makeatletter` et `\makeatother`
– ou bien mettre la redéfinition dans un *package*.

4.1. Personnalisations : commandes de sectionnement

Une commande de sectionnement, comme `\section`,

- produit un nombre reflétant la hiérarchie
- stocke le titre pour la table des matières
- stocke le titre pour l'en-tête
- formate le titre

- Compteurs :

  Chaque commande de sectionnement est associée à un compteur. Exemple : à `\section` est associé le compteur '`section`'.

  Par défaut, dans la classe `book` (book.cls) :

```
\newcounter{part}
\newcounter{chapter}
\newcounter{section}[chapter]
...
```

  Ceci signifie que `part` et `chapter` sont indépendants mais que les sections sont remises à 0 lorsque l'on passe à un nouveau chapitre.

  Exercice :

  Faire une copie de `book.cls` et faire en sorte que les chapitres soient remis à 0 à chaque partie. Tester sur un exemple.

Il est aussi possible de faire la même modification au sein du fichier principal, en utilisant `\@addtoreset` :

```
\@addtoreset{chapter}{part}
```

Exercice :

Essayer ce qui précède.

– Formatage :

Le formatage des numéros se fait via des commande préfixées par `\the`. Par défaut, le formatage des chapitres revient à faire :

```
\renewcommand\thechapter{\arabic{chapter}}
```

Exercice :

Cherchez ces définitions dans `book.cls` et commentez les éventuelles différences. Analysez la définition `\thesection`.

Les compteurs sont aussi utilisés pour les références et toute modification au niveau du numéro apparaissant au début de la section sera aussi répercutée là où la section est référencée.

Exercice :

Essayez de faire en sorte que le numéro de section soit encadré (avec `\fbox`*) en modifiant* `\thesection` *et testez le mécanisme des références.*

Le formatage du compteur proprement dit est fait par la commande `\@seccntformat`, définie dans `latex.ltx`.

Exercice :

Modifiez cette définition de manière appropriée et comparez à la précédente solution.

– Formatage général d'un titre de section : commande `\@startsection`

Les commandes `\section`, `\subsection`, etc., sont définies avec `\@startsection`. La syntaxe est :

`\@startsection{`⟨*name*⟩`}{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}`

Exemple :

```
\newcommand\section{%
      \@startsection
         {section}% nom du compteur
         {1}% niveau
         {\z@}% indentation par rapport a la marge
         {-3.5ex \@plus -1ex \@minus -.2ex}% espace avant
         {2.3ex \@plus.2ex}% % espace apres
         {\normalfont\Large\bfseries}}% style
```

Suivant le signe du *saut avant*, le premier alinéa sera est indenté (positif) ou non (négatif).

Les parties `\@plus` et `\@minus` correspondent à une flexibilité de l'espacement.

Le signe de l'*espace après* indique s'il y a ou non un saut entre le titre de section et le premier alinéa.

Exercice :

Redéfinissez `\paragraph` *pour qu'il y ait une ligne blanche entre le titre du paragraphe et le texte qui suit. Faites en sorte que le premier alinéa soit indenté.*

Les premiers alinéas peuvent être tous indentés avec le *package* `indentfirst`.

– Titres de chapitres et de parties :

Ces titres ne sont pas faits avec `\@startsection`.

`\chapter` par exemple, est défini dans `book.cls` à partir de `\secdef\@chapter\@schapter`, signifiant que `\chapter` fera appel à `\@chapter` et `\chapter*` à `\@schapter` ('s' pour 'star').

La commande `\@chapter` va appeler `\@makechapterhead` pour formater le titre.

```
\def\@makechapterhead#1{%  #1 est le titre du chapitre
  \vspace*{50\p@}%
  {\parindent \z@ \raggedright \normalfont
    \ifnum \c@secnumdepth >\m@ne
      \if@mainmatter
        \huge\bfseries \@chapapp\space \thechapter
        \par\nobreak
        \vskip 20\p@
      \fi
    \fi
    \interlinepenalty\@M
    \Huge \bfseries #1\par\nobreak
    \vskip 40\p@
  }}
```

Exercice :

Mettez le numéro du chapitre et le titre sur la même ligne, après avoir supprimé le mot 'chapitre',

et alignez le tout à droite.

– Composantes fixes des titres : `\chaptername`, `\partname`, etc. contiennent les noms ‘Chapter’, ‘Part’, etc. Ces commandes sont redéfinies avec `\renewcommand` : `\renewcommand{\abstractname}{Summary}`

4.2. Personnalisations : table des matières

La table des matières est générée dans un fichier `.toc`. Elle comporte des entrées de la forme

`\contentsline{`⟨*type*⟩`}{`⟨*text*⟩`}{`⟨*page*⟩`}`

où

– `type` vaut ‘section’, ‘subsection’, etc. Cela sert à déterminer l’indentation, le style, etc.
– `text` est ce qui apparaît.
– `page` est le numéro de la page.

Cette liste est générée automatiquement par les commandes de sectionnement qui exécutent des commandes `\addtocontents` ou `\addcontentsline`. Par exemple, la définition de `\@chapter` dans `book.cls` est :

```
\def\@chapter[#1]#2{%
   \ifnum \c@secnumdepth >\m@ne
        \if@mainmatter
           \refstepcounter{chapter}%
           \typeout{\@chapapp\space\thechapter.}%
           \addcontentsline{toc}{chapter}%
               {\protect\numberline{\thechapter}#1}%
        \else
           \addcontentsline{toc}{chapter}{#1}%
        \fi
   \else
        \addcontentsline{toc}{chapter}{#1}%
   \fi
   \chaptermark{#1}%
   \addtocontents{lof}{\protect\addvspace{10\p@}}%
   \addtocontents{lot}{\protect\addvspace{10\p@}}%
   \if@twocolumn
        \@topnewpage[\@makechapterhead{#2}]%
   \else
        \@makechapterhead{#2}%
   \@afterheading
   \fi}
```

Chaque entrée de la table des matières entraine en fait un appel de `\@dottedtocline` dont la

syntaxe est :

`\@dottedtocline{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}{`⟨*text*⟩`}{`⟨*page*⟩`}`

Les arguments ‘text’ et ‘page’ sont ceux de `\contentsline`. Les trois premiers arguments sont définis pour chaque commande de sectionnement, via une commande préfixée par `\l@` :

```
\newcommand*\l@subsection{\@dottedtocline{2}{3.8em}{3.2em}}
```

– Le niveau permet éventuellement de controler le nombre d’entrées affichées via le compteur ‘tocdepth’.
– L’indentation est celle depuis la marge gauche.
– ‘numwidth’ est l’espace disponible pour le numéro de la section.

La largeur disponible pour le numéro de page est contrôlée par la valeur de `\@pnumwidth` défini ainsi dans `book.cls` :

```
\newcommand\@pnumwidth{1.55em}
```

4.2.1. Ajout d'une entrée après un `\chapter*`

Ces commandes ne génèrent pas d'entrée dans la table des matières. On peut y remédier de plusieurs manières :

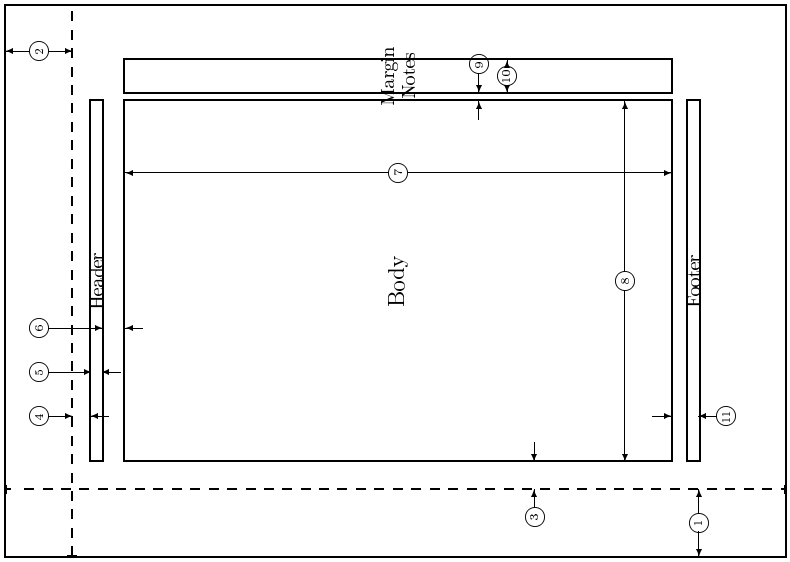
- Redéfinir `\@schapter`
- ajouter
  `\addcontentsline{toc}{chapter}{Introduction}`
  Le *package* `minitoc` permet d'avoir des mini-tables par chapitres, par parties, ... La documentation est disponible sur le LN.

4.3. Personnalisations : marges

Les paramètres d'une page peuvent être modifiés et la mise en page observée à l'aide du *package* `layout` en exécutant la commmande `\layout` :

```
\documentclass[a4paper,12pt]{article}
\usepackage{layout}
\begin{document}
\layout
\end{document}
```

dont le résultat est :



Les marges peuvent aussi être modifiées via le *package* `vmargin`.

L'une des commandes est `\setmargins` et sa syntaxe est :

`\setmargins`{ ⟨*leftmargin*⟩ } { ⟨*topmargin*⟩ } { ⟨*textwidth*⟩ } { ⟨*textheight*⟩ }
{ ⟨*headheight*⟩ } { ⟨*headsep*⟩ } { ⟨*footheight*⟩ } { ⟨*footskip*⟩ }

Par exemple, une mise en page typique de journal scientifique peut être satisfaite avec :

```
\usepackage{vmargin}
\setpapersize{A4}
\setmargins{1.0in}{1.25in}{6.8in}{8.8in}{0pt}{0mm}{0pt}{0mm}
```

4.4. Personnalisations : en-têtes

4.4.1. Commandes de bas niveau

Le *style d'une page* consiste en les hauts et bas de page. Chaque style a un nom et il lui correspond une commande `\ps@...`. Par exemple, au style 'headings', il correspond la commande `\ps@headings`. Celle-ci est déclenchée lorsque l'on écrit `\pagestyle{headings}`.

`\ps@headings` est défini dans la classe, par exemple `book.cls` et la définition dépend du mode de mise en page (`oneside` ou `twoside`).

Une définition typique (mais simplifiée) est :

```
\def\ps@headings{%
    \let\@oddfoot\@empty\let\@evenfoot\@empty
    \def\@evenhead{\thepage\hfil\slshape\leftmark}%
    \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
    }
```

- les bas de page impair (odd) et pair (even) sont vides ;
- le haut de page pair (even) comporte le numéro de page à gauche et la 'marque gauche' à droite, et en police oblique ('slanted') ;
- le haut de page impair (odd) comporte la 'marque droite' à gauche et le numéro de page à droite.

Les marques sont définies par des commandes comme `\chaptermark`, `\sectionmark` et éventuellement d'autres. La marque gauche est typiquement le premier chapitre apparaissant sur la page et la marque droite le dernier numéro de section.

On trouvera les définitions de `\chaptermark`, etc. par exemple dans `book.cls`. Les définitions ne sont pas nécessairement les mêmes pour tous les styles de page.

Exercice :

Définissez un nouveau style appelé 'perso' sur le modèle du style 'headings', où le bas de page gauche est identique au haut de page droit du style 'headings' et où le bas de page droit est identique au haut de page gauche du style 'headings'. Testez ce style sur un fichier que vous aurez rempli de quelques chapitres et sections.

4.4.2. Emploi du *package* `fancyhdr`

Le *package* `fancyhdr` de Piet van Oostrum permet de modifier facilement les styles d'en-têtes. La documentation de ce *package* est disponible sur le LN.

4.5. Personnalisations : listes

4.5.1. Environnement *enumerate*

Il y a quatre niveaux et chaque niveau a un compteur. Les noms des compteurs sont : `enumi`, `enumii`, `enumiii`, `enumiv`. La représentation d'un compteur se fait via `\theenum...` et l'affichage via `\labelenum...`. De plus, chaque niveau est muni d'un préfixe pour construire les références. Ce préfixe est dénoté par `\p@enum...`

En résumé :

Niveau	1	2	3	4
Compteur	`enumi`	`enumii`	`enumiii`	`enumiv`
Représentation	`\theenumi`	`\theenumii`	`\theenumiii`	`\theenumiv`
Par défaut	`\arabic{enumi}`	`\alph{enumii}`	`\roman{enumiii}`	`\Alph{enumiv}`
Label	`\labelenumi`	`\labelenumii`	`\labelenumiii`	`\labelenumiv`
Par défaut	`\theenumi.`	`(\theenumii)`	`\theenumiii.`	`\theenumiv.`
Exemple de num.	1.,2.	(a),(b)	i.,ii.	A.,B.
Préfixe	`\p@enumi`	`\p@enumii`	`\p@enumiii`	`\p@enumiv`
Par défaut	`{}`	`\theenumi`	`\theenumi (\theenumii)`	`\p@enumiii \theenumiii`
Exemple de réf.	1,2	1a,2b	1(a)i,2(b)ii	1(a)iA,2(b)iiB

(tableau extrait du *LaTeX Companion*)

Exercice :

Faites en sorte que tous les niveaux soient « numérotés » alphabétiquement (utiliser `\Alph` *comme dans le quatrième niveau) et que dans les références un niveau soit séparé du suivant par '–'.*

Le *package* `enumerate` permet de faciliter certaines des redéfinitions de l'environnement `enumerate`.

4.5.2. Environnement *itemize*

Les items de l'environnement `itemize` sont contrôlés par les commandes `\labelitemi`, `\labelitemii`, etc.

Si les redéfinitions ne doivent être faites que pour une liste, on les mettra juste après `\begin{itemize}`.

4.5.3. Environnement *description*

L'environnement `description` peut être paramétré en redéfinissant la commande `\descriptionlabel` qui crée le label.

```
\renewcommand{\descriptionlabel}[1]{\hspace{\labelsep}\textsf{#1}}
```

4.5.4. Autres listes

On peut définir de nouvelles listes via l'environnement '`list`' :

`\begin{list}{`⟨*default_label*⟩`}{`⟨*decls*⟩`}` ⟨*item_list*⟩ `\end{list}`

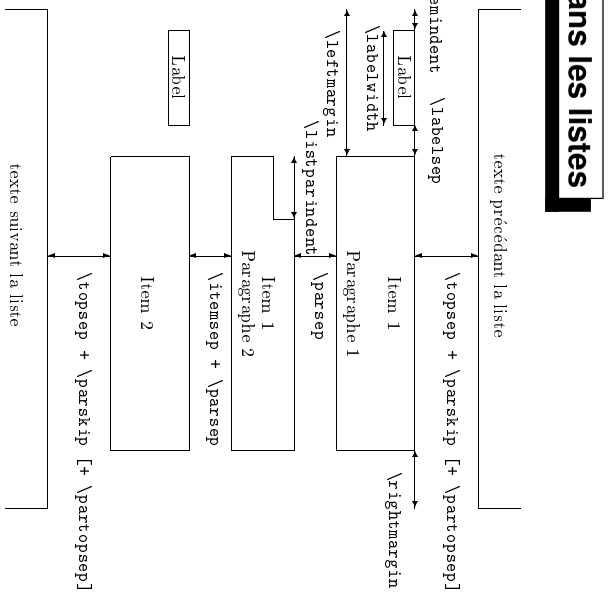
4.6. Personnalisations : placement des figures

4.6.1. Paramètres de placement

Le placement des figures obéit à des contraintes régies par les paramètres suivants :

- `topnumber` : nombre maximal de flottants au haut d'une page ;
- `bottomnumber` : nombre maximal de flottants au bas d'une page ;
- `totalnumber` : nombre maximal de flottants sur une page
- `\topfraction` : fraction maximale de la page occupée par des flottants en haut de page ;
- `\bottomfraction` : fraction maximale de la page occupée par des flottants en bas de page ;
- `\textfraction` : fraction minimale de la page devant être occupée par du texte ;
- `\floatpagefraction` : fraction minimale d'une page de flottants devant être occupée par des flottants ;
- `dbltopnumber` : analogue à `topnumber` pour des pages en deux colonnes ;
- `\dbltopfraction` : analogue à `\topfraction` pour des pages en deux colonnes ;
- `\dblfloatpagefraction` : analogue à `\floatpagefraction` pour des pages de flottants en deux colonnes ;

4.5.5. Paramètres dans les listes



Les valeurs par défaut des paramètres permettant le positionnement des figures sont définies dans `latex.ltx` :

```
\setcounter{topnumber}{2}
\setcounter{bottomnumber}{1}
\setcounter{totalnumber}{3}
\newcommand\topfraction{.7}
\newcommand\bottomfraction{.3}
\newcommand\textfraction{.2}
\newcommand\floatpagefraction{.5}
\setcounter{dbltopnumber}{2}
\newcommand\dbltopfraction{.7}
\newcommand\dblfloatpagefraction{.5}
```

Des valeurs moins contraignantes sont par exemple :

```
\renewcommand{\topfraction}{.8}
\renewcommand{\bottomfraction}{.8}
\renewcommand{\textfraction}{.2}
\renewcommand{\floatpagefraction}{.8}
```

Il y a au maximum 18 figures en attente à un moment donné.

4.6.2. Contrôle amélioré du positionnement

`\suppressfloats`

La commande `\suppressfloats` permet d'interdire des objets flottants d'un certain type sur une page donnée.

Si l'on ne souhaite pas de flottant au haut d'une page sur laquelle se trouve le début d'une section, on peut (re)définir `\section` de la manière suivante :

```
\newcommand{\section}{\suppressfloats[t]%
                      \@startsection{section}{...}{...}{...}...
                     }
```

Emploi de ' !'

L'emploi de ' !' en conjonction avec les paramètres `h`, `t` ou `b` relache les contraintes de placement.

Package `afterpage`

```
\afterpage{\clearpage}
```

Package `flafter`

Ce *package* a pour effet d'éviter qu'un flottant n'apparaisse avant l'endroit où il est défini dans le texte.

Package `float`

Ce *package* permet de définir de nouveaux styles de flottants. Il est documenté sur le LN.

5. Quelques *packages* utiles

Nous parcourons ici en vrac quelques *packages* qui peuvent s'avérer utile pour des personnalisations.

5.1. letterspace : espacement de lettres d'un mot

5.2. ulem : soulignements divers

5.3. xspace : espaces après les commandes

En chargeant le *package* `xspace` et en définissant `\ie` comme suit :

```
\newcommand{\ie}{i.e.,\xspace}
```

on peut écrire « a i.e. b » avec `a \ie b`. 'i.e.' et 'b' ne sont pas collés.

5.4. Styles de thèse : exemple TheseCRIN

La classe `TheseCRIN` realisée pour les thèses du CRIN, donne un exemple de nombreux paramétrages typiques. La documentation de cette classe est disponible sur le LN.

6. Quelques notions de programmation avancée

6.1. Commandes robustes et fragiles

Problème : certaines commandes sont employés dans des conditions telles qu'elles peuvent être évaluées prématurément.

On remédie à cela en précédant la commande de `\protect` ou en déclarant une nouvelle commande avec `\DeclareRobustCommand`.

6.2. Résolution d'interactions entre *packages*

Il arrive qu'un *package* soit incompatible avec un autre *package* et que le problème n'est le fait que de quelques commandes qui ne devraient par exemple pas être redéfinies.

On peut dans ce cas sauvegarder l'ancienne valeur d'une commande le temps de l'exécution de la nouvelle.

```
\let\olddef=\cmd
\def\cmd{nouvelle definition}
...
\let\cmd=\olddef % on revient a l'ancien etat
```

6.3. Caractères actifs

Tous les caractères apparaissant dans un fichier TEX reçoivent un code. Il y a 16 codes différents, de 0 à 15. Le code pour les lettres est 11. Celui pour la plupart des autres caractères est 12. Par défaut, les caractères ont le code que l'on attend d'eux ; par exemple, la lettre 'a' a par défaut le code 11. Mais il est possible de changer ce code et ceci peut perturber le fonctionnement de certaines commandes.

```
x  % code 11
\catcode`\x=13
x  % code 13
\catcode`\x=11
x  % code 11
```

Typiquement, si le code de 'a' est changé, il n'est plus possible d'utiliser une commande comportant cette lettre (comme `\catcode`) puisque...ce n'est plus une lettre dans le sens du code 11.

Ainsi, les commandes bien connues `\makeatletter` et `\makeatother` sont équivalentes à :

```
\catcode`\@=12  % @ devient `autre' et ne peut plus etre utilise'
                % dans une commande
```

et

```
\catcode`\@=11  % @ devient une lettre et peut etre utilise' dans une commande
```

6.4. Compteurs

Le compteur 'a' est représenté en LaTeX par la variable entière `\c@a`.

- `\newcounter`
- `\setcounter`
- `\addtocounter`

6.5. Programmation TeX

- booléens : `\newif`, `\if...`
- compteurs : `\newcount`, `\ifnum`
- dimensions : `\newdimen`, `\ifdim`
- ...

Pour écrire une classe ou un *package*, voir les fichiers `*guide.tex` dans la distribution standard, en particulier `clsguide.tex`. Tous ces guides sont disponibles formatés sur le LN.

Pour aller plus loin, lire le TeXbook.

7. Les messages d'erreur

Comment réagir face à un message d'erreur ?

Lire tout le message, localiser le lieu et la cause de l'erreur.

En général, c'est simple.

Si le contexte est insuffisant, augmenter `\errorcontextlines` (ou le compteur 'errorcontextlines').

Pour voir le déroulement des macros, faire `\tracingmacros=1` puis (plus loin) `\tracingmacros=0`.