Exposés sur LaTeX

Cours 6

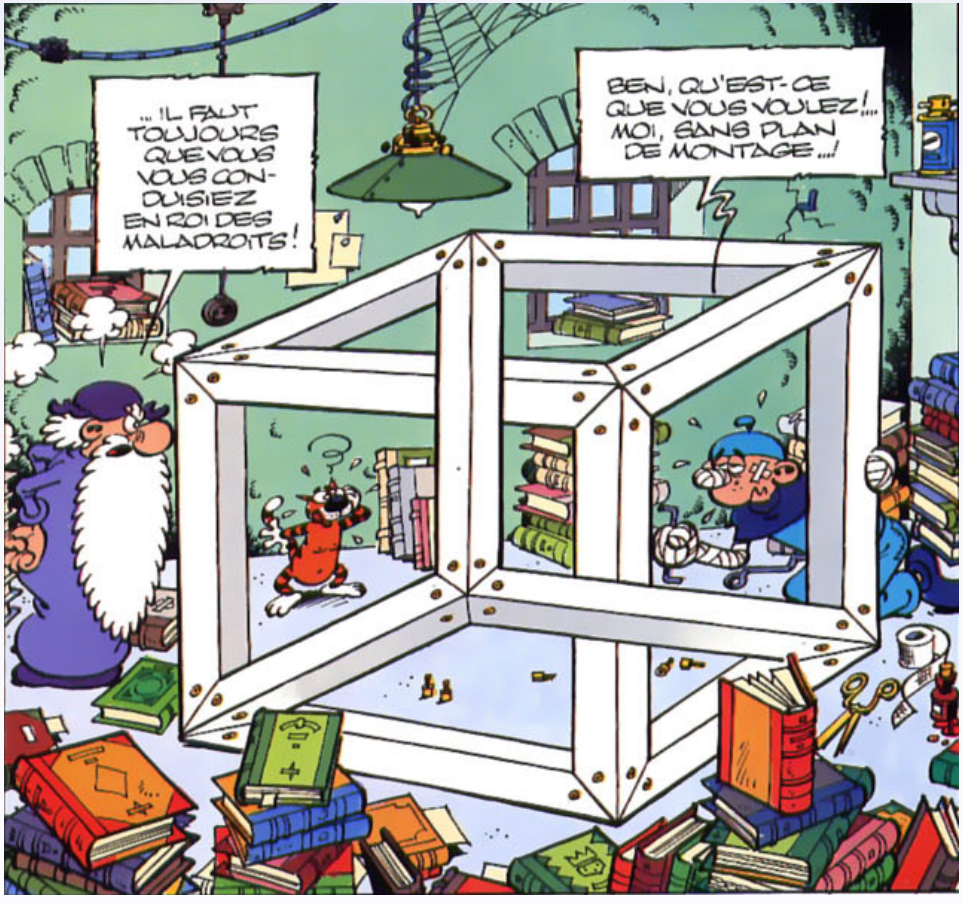
Des *packages* pour sauver les apparences

Thierry Masson

Centre de Physique Théorique
Campus de Luminy, Marseille

Cours 6 – Des *packages* pour sauver les apparences

- Des outils techniques utiles
- La gestion de la couleur
- La gestion des tableaux
- La gestion des listes
- La mise en page
- Autres éléments du document

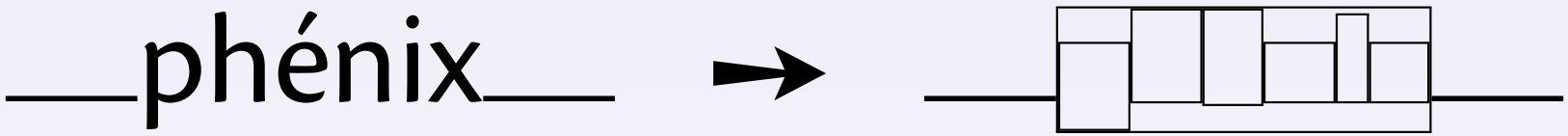


Des outils techniques utiles

Où l'on apprend à se simplifier la vie en s'aidant de commandes bien utiles…

La mise en boite

Dans son principe fondamental, LaTeX gère des boîtes : chaque lettre est une boite, chaque mot est un ensemble de boites (les lettres), les paragraphes sont des grosses boites…



Chaque boite est définie par sa largeur (`width`), sa hauteur au dessus de la ligne de base (`height`) et sa profondeur en dessous de la ligne de base (`depth`).

Ainsi, lorsque LaTeX a terminé de composer un tableau, ce dernier devient une boite dont il ne retient que les attributs de dimensions afin de la placer dans la page.

Dans de nombreuses commandes (comme celles qui suivent), il est possible d'utiliser les dimensions naturelles des objets LaTeX sur lesquelles elles s'appliquent :

- `\width` désigne la largeur,
- `\height` désigne la hauteur,
- `\depth` désigne la profondeur,
- `\totalheight` désigne la hauteur totale `\height`+`\depth`.

La commande `\phantom{-}` n'imprime pas son contenu tout en créant la boite associée :
`a\phantom{phénix}b` ➞ a b
Cette commande permet de dépanner dans certaines mises en page difficiles.

La commande \makebox

La commande \makebox[*largeur*][*position*]{-} crée une boite de largeur finale *largeur* dont le contenu est positionné selon *position*=c,l,r,s.

Dans les exemples suivants, un cadre est ajouté pour visualiser la boite finale.

\makebox{phénix}b → a phénix b (permet de créer des blocks insécables)
\makebox[5em][r]{phénix}b → a phénix b
\makebox[1.5em][l]{phénix}bcdef → a phébcdef
\makebox[5em][c]{phénix}b → a phénix b
\makebox[8em][s]{A B C D}b → a A B C D b
\makebox[2\width][l]{phénix}b → a phénix b
\makebox[10\height][c]{phénix}b → a phénix b

\framebox est comme \makebox avec un cadre en plus.
Les longueurs \fboxrule et \fboxsep désignent l'épaisseur du trait et la séparation intérieure avec le texte.

\framebox{phénix} → phénix (défaut : \fboxrule=0.4pt, \fboxsep=3.0pt)

```
\setlength{\fboxrule}{1.5pt}\setlength{\fboxsep}{1pt}
\framebox{phénix}
```
→ phénix

⚠ Les commandes précédentes ne peuvent accepter qu'une boite déjà formatée (quelques mots, un tableau, une image…) mais pas un paragraphe à mettre en forme.

La commande \parbox

La commande `\parbox[`*position*`][`*hauteur*`][`*pos. interne*`]{`*largeur*`}{-}` crée une boite dont le contenu est un paragraphe entier qu'il formate sur des lignes de largeur *largeur*.

La hauteur finale est fixée par *hauteur*, le contenu est aligné verticalement selon *pos. interne*=`t`,`c`,`b`,`s`, le paramètre *position*=`t`,`c`,`b` fixe l'alignement vertical de la boite finale par rapport au texte ambiant.

```
A\parbox{5em}{...}B
```
→ A a a a a a a a a a a B

```
A\parbox[t][\height][t]{5em}{...}B
```
→ A a a a a a a a a a a B

```
A\parbox[b][1.5\height][t]{5em}{...}B
```
→ A a a a a a a a a a a B

Il existe un environnement équivalent nommé `minipage` :

```
A\begin{minipage}[c][8ex][b]{5em}
...
\end{minipage}B
```
→ A a a a a a a a a a a B

Lorsqu'on souhaite utiliser des environnements pour formater le paragraphe (`center`, `flushleft`, `flushright`…) il est préférable d'utiliser `minipage` pour la gestion des espaces verticaux.

La commande \raisebox

La commande \raisebox{*distance*}[*hauteur*][*profondeur*]{-} relève son contenu d'une distance *distance* (+ vers le haut, − vers la bas), et produit une boite finale de hauteur *hauteur* et de profondeur *profondeur*.

Cette commande ne peut contenir qu'une boite déjà formatée.

```
a\raisebox{1ex}{phénix}b
```
→ a phénix b

```
a\raisebox{1ex}[1ex][0pt]{phénix}b
```
→ a phénix b

```
a\raisebox{-1ex}{phénix}b
```
→ a phénix b

```
a\raisebox{-1ex}[2ex][12pt]{phénix}b
```
→ a phénix b

```
a\raisebox{-1ex}[\height][\depth]{phénix}b
```
→ a phénix b

```
a\raisebox{1ex}[0pt][0pt]{phénix}b
```
→ a phénix b

En combinant les commandes \makebox et \raisebox, on peut produire une boite dont toutes les dimensions sont nulles :

```
a\raisebox{1.5ex}[0pt][0pt]{\makebox[0pt][c]{phénix}}b
```
→ phénix ab

```
a\raisebox{1.5ex}[0pt][0pt]{\makebox[0pt][l]{phénix}}b
```
→ phénix ab

Le *package* `calc`

Dans le langage TEX, les opérations arithmétiques élémentaires sont réalisées grâce aux commandes `\advance` et `\multiply` dans une syntaxe assez lourde.

Le *package* **`calc`** apporte de la souplesse en donnant la possibilité de réaliser des opérations avec des notations mathématiques plus habituelles.

```
\newcounter{local} \setcounter{local}{2}
\setcounter{local}{\value{local}*3 + 5}
```
La commande `\thelocal` donne alors la valeur 11.

```
\newlength{\lgrlocale}
\setlength{\lgrlocale}{0.75\textheight-\textwidth/2}
```
est accepté pour définir une dimension.

`a\makebox[2\width+1em][c]{phénix}b` ➟ a[phénix]b

Les opérations possibles sont : `+`, `-`, `*`, `/`, `\ratio{-}{-}` (rapport de deux longeurs) et `\real{-}` (nombre réel pour des multiplications).

`\widthof{`*texte*`}`, `\heightof{`*texte*`}`, `\depthof{`*texte*`}`, `\totalheightof{`*texte*`}` permettent de récupérer les dimensions de la boite définie par *texte* : `\setlength{\lgrlocale}{\widthof{AA}}` assigne la largeur de “AA” à `\lgrlocale`

Les commandes `\maxof{-}{-}` et `\minof{-}{-}` retournent respectivement les maxima et minima des expressions données dans leurs deux arguments (réels).

Le *package* `ifthen`

La comparaison de différents éléments est possible en TEX.
Le *package* **`ifthen`** simplifie les expressions conditionnelles.
Il définit la commande `\ifthenelse{-}{-}{-}` où :

1. le premier argument est une condition à tester ;
2. le second argument est le code à exécuter si la condition est vraie ;
3. le troisième argument est le code à exécuter si la condition est fausse.

Les conditions peuvent être construites avec des parenthèses `\(`, `\)` et les commandes `\and`, `\or` et `\not`. Les atomes élémentaires des conditions sont :

- ⟨*num1*⟩<⟨*num2*⟩, ⟨*num1*⟩=⟨*num2*⟩, ⟨*num1*⟩>⟨*num2*⟩,
- `\isodd{`⟨*num*⟩`}`,
- `\isundefined{\`*cmd*`}`,
- `\equal{`*texte1*`}{`*texte2*`}`,
- `\lengthtest{`⟨*dim1*⟩<⟨*dim2*⟩`}`, `\lengthtest{`⟨*dim1*⟩=⟨*dim2*⟩`}`, `\lengthtest{`⟨*dim1*⟩>⟨*dim2*⟩`}`,
- `\boolean{`*nom*`}`.

Les commandes `\newboolean{-}` et `\setboolean{-}{-}` permettent de créer des booléens et de leur assigner la valeur `true` ou `false`.
On peut manipuler des boucles avec la commande `\whiledo{`*test*`}{`*code*`}`.
Le *package* **`xifthen`** ajoute d'autres tests.

Le *package* snapshot

Le *package* **snapshot** répond à une question pratique : quels fichiers fournir ou conserver avec le document source pour que le tout compile correctement ?
La réponse dépend bien sûr de la machine sur laquelle compiler le document.
snapshot s'occupe de collecter dans un fichier .dep (dépendances) tous les fichiers appelés lors de la compilation, ainsi que leur version.
Pour l'utiliser, il faut le placer avant tout le reste :

```
\RequirePackage{snapshot}
\documentclass[12pt]{article}
...
```

Après compilation, le fichier .dep contient des informations du type

```
*{application}{TeX}      {1990/03/25 v3.x}
*{format} {LaTeX2e}      {2009/09/24 v2.e}
*{class}  {article}      {2007/10/19 v1.4h}
*{file}   {size12.clo}   {2007/10/19 v1.4h}
*{package}{inputenc}     {2008/03/30 v1.1d}
*{file}   {utf8.def}     {2008/04/05 v1.1m}
*{package}{textcomp}     {2005/09/27 v1.99g}
*{file}   {ts1enc.def}   {2001/06/05 v3.0e}
*{package}{amsmath}      {2000/07/18 v2.13}
```

Le moteur bundledoc : tirer profit de snapshot

On peut utiliser les informations fournies par **snapshot** dans le fichier .dep pour en faire automatiquement une archive, grâce au script PERL appelé bundledoc.

La commande

```
bundledoc --verbose snapshot.dep
```

construit une archive qui contient tous les fichiers utiles à la compilation, y compris les images et les fichiers de style personnels.

Il est donc possible de réaliser un “instantané” des fichiers utilisés dans un projet pour pouvoir le recompiler plus tard (des années ?).

De nombreuses options sont disponibles.
On peut utiliser un fichier de configurations, on peut inclure des types de fichiers particuliers, on peut exclure des dossiers.

```
bundledoc --exclude=/usr/local/texlive/ ...
```

exclut les fichiers déjà présents dans l’installation TeXLive.
➟ très utile pour envoyer les documents sur les serveurs en ligne, comme **arXiv**.

La gestion de la couleur

Où l'on apprend à colorier son document…

Le *package* `color`

Le *package* **`color`** permet de gérer les couleurs dans LaTeX.

Comme pour les graphiques, `tex` ne gère pas directement les couleurs : ce sont les drivers qui s'en occupent. Il faut donc préciser quel chemin de production on utilise :

`\usepackage[dvips]{color}` charge les spécificités du driver `dvips`.

Autres options possibles : `dvipdfm`, `pdftex`, `xetex`…

Une couleur est définie à partir d'un modèle et d'une spécification :

red-green-blue	`rgb`	3 nombres compris entre 0 et 1 `0.5,0.2,0.8` →
cyan-magenta-yellow-black	`cmyk`	4 nombres compris entre 0 et 1 `0.5,0.2,0.8,0.3` →
Échelle de gris	`gray`	1 nombre compris entre 0 et 1 `0.8` →
Nom explicite	`named`	nom d'une couleur prédéfinie `magenta` →

La commande
`\definecolor{`*nom*`}{`*modèle*`}{`*spécification*`}`
définit une couleur nommée *nom* dans le modèle choisi.

`\definecolor{light-blue}{rgb}{0.8,0.85,1}` →

`\definecolor{mygrey}{gray}{0.75}` →

Utilisation des couleurs

Plusieurs commandes sont définies pour utiliser les couleurs.

- `\color{`*couleur*`}` installe la couleur dont le nom est dans l'argument.
  `Noir {\color{magenta} couleur} et noir` ➟ Noir couleur et noir
- `\textcolor{`*couleur*`}{`*texte*`}` imprime le texte dans la couleur sélectionnée.
  `Noir \textcolor{magenta}{couleur} et noir` ➟ Noir couleur et noir
- `\colorbox{`*couleur*`}{`*texte*`}`, `\fcolorbox{`*couleur1*`}{`*couleur2*`}{`*texte*`}` produisent des boîtes de fond coloré.
  `\colorbox{magenta}{test}` ➟ test
  `\fcolorbox{blue}{magenta}{test}` ➟ test
  Les longueurs `\fboxrule` et `\fboxsep` gèrent l'épaisseur du trait et de la séparation avec le texte (mêmes paramètres que pour `\framebox`).
- `\pagecolor{`*couleur*`}` colore le fond de la page.
- Ces commandes acceptent une variante qui spécifie la couleur par un modèle :
  `\color[`*modèle*`]{`*spécification*`}`
  `\textcolor[`*modèle*`]{`*spécification*`}{`*texte*`}`
  `\colorbox[`*modèle*`]{`*spécification*`}{`*texte*`}`
  `\fcolorbox[`*modèle*`]{`*spécification1*`}{`*spécification2*`}{`*texte*`}`
  `\pagecolor[`*modèle*`]{`*spécification*`}`

Le *package* xcolor

Le *package* **xcolor** étend sur de nombreux points le *package* **color** en fournissant d'autres modèles et une procédure pour créer des couleurs par mélanges.
Il utilise essentiellement les mêmes options que **color**.

Les nouveaux modèles sont :

cyan-magenta-yellow	`cmy`	3 nombres compris entre 0 et 1 `0.5,0.2,0.8` →
hue-saturation-brightness	`hsb`	3 nombres compris entre 0 et 1 `0.5,0.2,0.8` →
Longueur d'onde en nm	`wave`	1 nombre compris entre 363 et 814 `650.57` →
Red-Green-Blue	`RGB`	3 nombres entiers compris entre 0 et 255 `100,150,200` →
RRGGBB	`HTML`	1 nombre hexadécimal entre 000000 et FFFFFF `43ADF9` →
Hue-Saturation-Brightness	`HSB`	3 nombres entiers entre 0 et 240 `10,100,200` →
Niveau de gris	`Gray`	1 nombre entier entre 0 et 15 `10` →

Il reprend et étend les commandes définies par **color** : `\color`, `\textcolor`, `\colorbox`, `\fcolorbox` et `\pagecolor`.

Les couleurs prédéfinies et les mélanges

Le *package* **xcolor** définit les couleurs suivantes :

black	white	red	green	blue
cyan	magenta	yellow	orange	pink
brown	olive	purple	teal	violet
gray	darkgray	lightgray		

Les options `dvipsnames`, `svgnames` et `x11names` de **xcolor** chargent de nombreuses autres couleurs prédéfinies. Consulter la documentation de **xcolor**.

La commande `\definecolor` admet la même syntaxe que pour **color**.
La commande `\colorlet{`*couleur*`}{`*mélange*`}` définit une couleur à partir de mélanges :
`blue!85!black` ➟ , `blue!65!black` ➟ , `blue!45!black` ➟
`-red` ➟ (couleur complémentaire)
`red!50!green!60!yellow` ➟ , `red!50!green!20!yellow` ➟
`red!50!green!20` ➟ (dernière couleur : `white`)

Les commandes `\color`, `\textcolor`, `\colorbox`, `\fcolorbox` et `\pagecolor` acceptent directement des couleurs sous forme de mélanges :
`\textcolor{blue!85!black}{test}` ➟ test

Le *package* **xcolor** fournit beaucoup d'autres fonctionnalités, en particulier la possibilité de transformer une couleur d'un modèle dans un autre.
L'option `gray` convertit toutes les couleurs en niveaux de gris ➟ impression en N & B.

La gestion des tableaux

Où on découvre comment réaliser des tableaux avec des colonnes de différentes formes…

Rappels sur les tableaux

L'environnement `tabular` définit les commandes suivantes pour le formatage des colonnes d'un tableau :

`|`	insère une ligne verticale
`l`	alignement à gauche
`r`	alignement à droite
`c`	alignement au centre
`p{`⟨*dim*⟩`}`	paragraphe de largeur donnée
`@{`*code*`}`	supprime l'espace entre colonnes et insère *code*
`*{`*num*`}{`*opts*`}`	répète *num* fois la déclaration *opts*

Les paramètres suivants déterminent l'aspect du tableau :

`\tabcolsep`	moitié de la largeur de l'espacement entre les colonnes	(`6pt`)
`\arrayrulewidth`	épaisseur des traits du tableau	(`0.4pt`)
`\doublerulesep`	séparation entre les doubles traits (`||`)	(`2pt`)
`\arraystretch`	fraction avec laquelle l'espace entre lignes est multiplié	(`1.0`)

Par exemple, `\renewcommand{\arraystretch}{1.5}` écarte les lignes de 50%.

Le *package* `array`

Le *package* **`array`** définit d'autres commandes de formatage :

`m{`⟨*dim*⟩`}`	colonne de largeur donnée, verticalement centrée
`b{`⟨*dim*⟩`}`	colonne de largeur donnée, verticalement en bas
`>{`*code*`}`	insère *code* avant les données d'une cellule
`<{`*code*`}`	insère *code* après les données d'une cellule
`!{`*code*`}`	insère *code* entre deux colonnes

La commande `\newcolumntype{-}{-}` permet de définir des nouveaux types de colonnes en utilisant la syntaxe des déclarations de colonnes.
La dimension `\extrarowheight` s'ajoute à la hauteur des cellules.

```
\setlength{\extrarowheight}{4pt}
\begin{tabular}{|>{\large}c|>{\bfseries}l|}
\hline A & B \\
\hline 100 & 50 \\ \hline
\end{tabular}
```

A	**B**
100	**50**

```
\newcolumntype{R}{>{$}r<{$}}
\begin{tabular}{|R!{=}>{\bfseries}l|}
\hline a_1 & A \\
\hline b_1 & B \\ \hline
\end{tabular}
```

a_1	=	**A**
b_1	=	**B**

Le *package* array : autres exemples

```
\begin{tabular}{p{2em}p{2em}}
\hline a a a a a a & b \\ \hline
\end{tabular}
```

a a a a a	b

```
\begin{tabular}{m{2em}m{2em}}
\hline a a a a a a & b \\ \hline
\end{tabular}
```

a a a a a	b

```
\begin{tabular}{b{2em}b{2em}}
\hline a a a a a a & b \\ \hline
\end{tabular}
```

a a a a a	b

```
\newcolumntype{z}{>{\raggedleft\arraybackslash}p{2em}}
\begin{tabular}{zz}
\hline a a a a a a a & b b b b b b b \\ \hline
\end{tabular}
```

a a a a a	b b b b
a a	b b b

Les commandes `\raggedright` et `\centering` peuvent aussi être utilisées.

⚠ Ces commandes redéfinissent `\\`, qu'il faut rétablir aussitôt avec `\arraybackslash`.

Le *package* tabularx

L'environnement `tabular*` admet un paramètre supplémentaire qui désigne la largeur finale du tableau. Pour atteindre ce but, LaTeX joue sur l'espace entre les colonnes.

Le *package* **tabularx** résout ce problème différemment : il définit l'environnement `tabularx` qui admet pour paramètre supplémentaire la largeur finale du tableau.
Pour atteindre cette largeur, un nouveau type de colonne est défini :

X colonne de largeur automatiquement calculée

```
\begin{tabularx}{10em}{|X|X|p{1.1em}|}
\hline a a a a a a a & b b b b b b b & c c c c \\ \hline
\end{tabularx}
```

a a a a	b b b b	c c
a a a	b b b	c c

Toutes les colonnes de type X dans un tableau auront la même largeur.

tabularx est compatible avec **array**.
On peut définir de nouveaux types de colonnes avec `\newcolumntype{-}{-}` :
`\newcolumntype{Y}{>{\raggedright\arraybackslash}X}`

Par définition, X est construit sur `p{`$\langle dim \rangle$`}` où $\langle dim \rangle$ est déterminée par l'algorithme de création du tableau. On peut redéfinir X pour utiliser `m{`$\langle dim \rangle$`}` ou `b{`$\langle dim \rangle$`}` :
`\renewcommand\tabularxcolumn[1]{m{#1}}` redéfinit X pour utiliser `m{`$\langle dim \rangle$`}`.

Couleurs dans les tableaux

Le *package* **colortbl** permet de colorer les tableaux.

- `\columncolor{`*couleur*`}` colore le fond d'une rangée ;
- `\rowcolor{`*couleur*`}` colore le fond d'une ligne entière ;
- `\cellcolor{`*couleur*`}` colore le fond d'une cellule ;
- `\arrayrulecolor{`*couleur*`}` colore les traits ;
- `\doublerulesepcolor{`*couleur*`}` colore l'espace entre les doubles traits.

```
\arrayrulecolor{green}\doublerulesepcolor{yellow}
\setlength{\arrayrulewidth}{2pt}
\begin{tabular}{|>{\columncolor{brown}}r||l|}
\hline \rowcolor{red} AA & BB \\ \hline
 CC & DD \\ \hline
 EE & \cellcolor{cyan} FF \\ \hline
\end{tabular}
```

AA	BB
CC	DD
EE	FF

On peut utiliser la commande `\newcolumntype{-}{-}` avec ces commandes.
L'option `table` de **xcolor** appelle **colortbl** et définit une commande
`\rowcolor[`*commandes*`]{`*ligne*`}{`*couleur1*`}{`*couleur2*`}`
où les deux couleurs concernent les lignes impaires et paires, *ligne* désigne la première ligne où commencer l'alternance et *commandes* les commandes entre les lignes (`\hline` par exemple).

Autres *packages* pour créer des tableaux

Les *packages* **supertabular** (voir aussi son extension **xtab**) et **longtable** permettent de créer des tableaux s'étendant automatiquement sur plusieurs pages.

hhline permet de gérer les intersections des lignes horizontales et verticales doubles.

arydshln permet d'utiliser des traits pointillés comme séparations de cellules.

multirow définit une commande pour regrouper des cellules d'une même colonne.

booktabs introduit les commandes `\toprule`, `\midrule` et `\bottomrule` qui gèrent des espacements verticaux moins resserrés que `\hline` :

```
\begin{tabular}{@{}llr@{}}
\toprule
\multicolumn{2}{c}{AAA} &
\multicolumn{1}{c}{BBB}\\ \midrule
CCC & DDD & EE\\ \midrule
aa & bb & cc \\
dd & ee & ff \\ \bottomrule
\end{tabular}
```

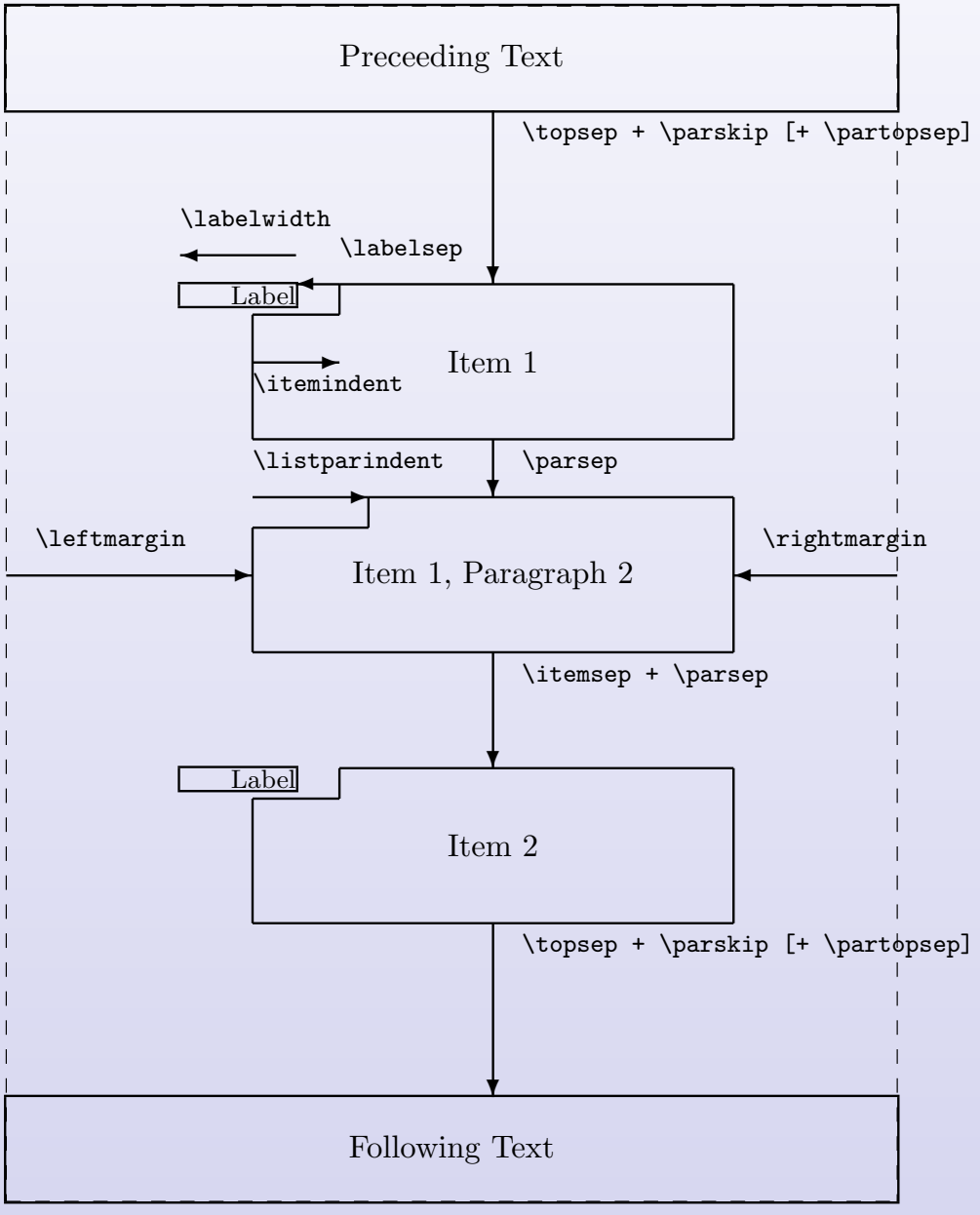
AAA		BBB
CCC	DDD	EE
aa	bb	cc
dd	ee	ff

Il est utile de consulter la documentation de tous ces *packages*.

La gestion des listes

Où l'on s'initie à l'art difficile de bien faire ses listes…

Les listes : la géométrie



Géométrie des listes

Les dimensions verticales sont :

- `\topsep`
- `\partopsep`
- `\parsep`
- `\itemsep`

Les dimensions horizontales sont :

- `\leftmargin`, `\rightmargin`
- `\labelwidth`
- `\labelsep`
- `\itemindent`
- `\listparindent`

→ pas facile à personnaliser !

Les listes : personnalisation

L'environnement `list` permet de définir des nouveaux types de listes, mais il faut maîtriser à la fois la géométrie décrite auparavant et les labels.
Une telle définition peut ressembler à ceci :

```
\newenvironment{Description}
  {\begin{list}{}{\let\makelabel\Descriptionlabel
     \setlength{\labelwidth}{40pt}%
     \setlength{\leftmargin}{\labelwidth+\labelsep}}}%
  {\end{list}}
\newcommand*{\Descriptionlabel}[1]{\textsf{#1:}\hfil}
```

La première partie définit l'environnement `Description` avec des dimensions spécifiques.
La commande `\Descriptionlabel` sert à placer les labels de la liste.

On a déjà vu que le *package* **`pifont`** définit les listes
`\begin{dinglist}{`⟨*num*⟩`}` ... `\end{dinglist}`
`\begin{dingautolist}{`⟨*num*⟩`}` ... `\end{dingautolist}`
où ⟨*num*⟩ est le numéro d'un des symboles de la police Zapf Dingbats.
La seconde liste est une énumération, elle incrémente ⟨*num*⟩ à chaque `\item`.
`pifont` définit aussi les listes suivantes qui acceptent n'importe quelle famille de police :
`\begin{Pilist}{`*fmy*`}{`⟨*num*⟩`}` ... `\end{Pilist}`
`\begin{Piautolist}{`*fmy*`}{`⟨*num*⟩`}` ... `\end{Piautolist}`

Le *package* enumitem

Le *package* **enumitem** aide à modifier certains paramètres des 3 listes usuelles `itemize`, `enumerate` et `description`.

```
\begin{enumerate}[label=\emph{\alph*}), parsep=2ex]
    \item ...
    \item ...
\end{enumerate}
```

Il est possible de définir de nouvelles listes :

```
\newlist{maliste}{enumerate}{3}
\setlist[maliste,1]{label=\alph*),font=\sffamily\bfseries}
\setlist[maliste,2]{label=\arabic*),font=\sffamily\bfseries}
```

Les dimensions modifiables sont `topsep`, `partopsep`, `parsep`, `itemsep`, `leftmargin`, `rightmargin`, `listparindent`, `labelwidth`, `labelsep`.
On peut personnaliser la police avec `font=...` et le label avec `label=...`.

⚠ Ce *package* n'est pas compatible avec l'option `french` de **babel**.
Après l'appel de **babel**, il faut utiliser la commande
`\frenchbsetup{StandardLists=true}`
pour que `french` ne change pas les listes de LaTeX.

Le *package* `paralist`

Le *package* **`paralist`** permet de modifier facilement les items des listes usuelles à l'aide d'une syntaxe très simple :

```
\begin{enumerate}[{exemple} a)]...
\begin{enumerate}[{A}-1]...
\begin{enumerate}[\bfseries {Item} I]...
```

qui produisent :
exemple a), exemple b), exemple c)…
A-1, A-2, A-3…
Item I, **Item II**, **Item III**…

D'autres environnements de listes sont définis :

- `inparaenum`, `inparaitem`, `inparadesc` qui produisent des listes dans le même paragraphe ;
- `compactenum`, `compactitem`, `compactdesc` qui produisent les listes usuelles en versions plus compactes ;
- `asparaenum`, `asparaitem`, `asparadesc` qui produisent des listes où chaque item commence un nouveau paragraphe (indentation ordinaire d'un début de paragraphe).

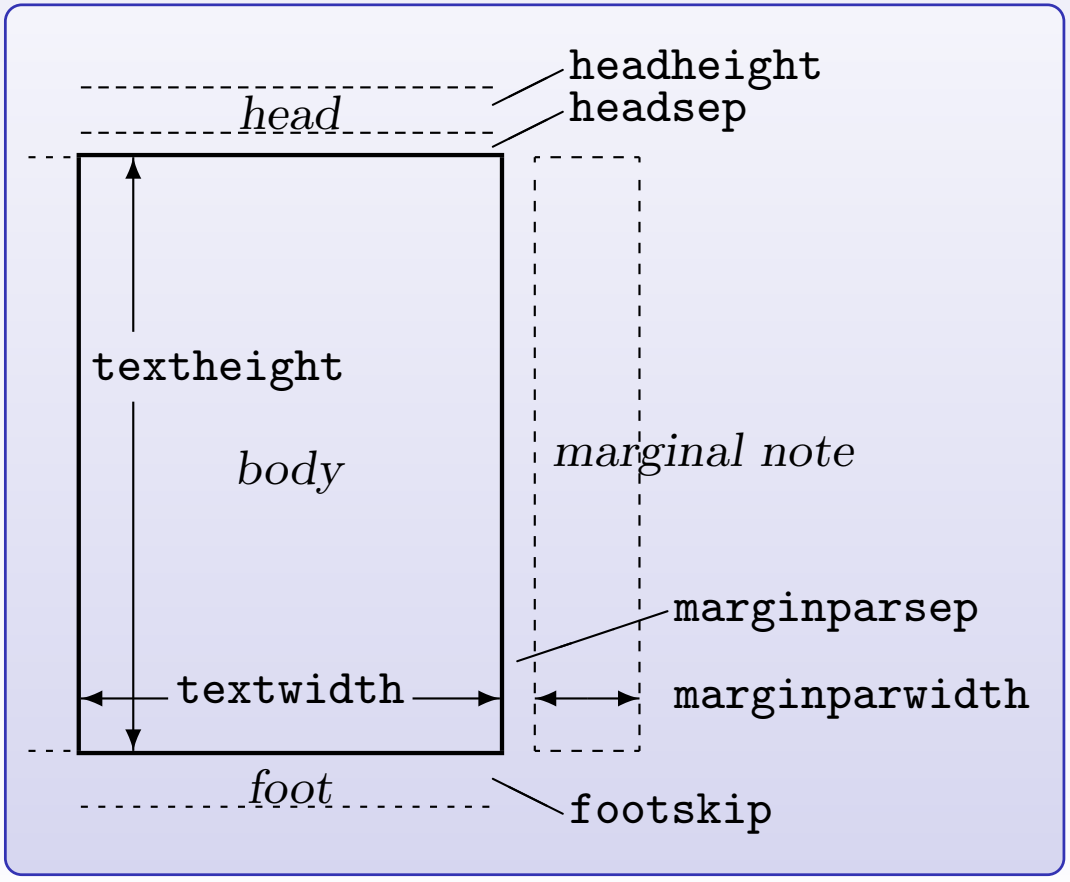
Ce *package* comporte bien d'autres commandes de personnalisation utiles.

La mise en page

Où l'on apprend à changer l'aspect de différents éléments de la page de texte…

Rappels sur les éléments d'une page et d'un document

Une page composée par LaTeX est constituée de différents éléments :



- l'entête, où figurent souvent le rappel des titres de chapitres et des sections ;
- le pied de page, où figurent souvent le numéro de la page ;
- des notes de bas de page (`\footnote{-}`) ;
- des notes dans la marge (`\marginpar{-}`) ;
- et le corps du texte qui contient à son tour des paragraphes, divisions par chapitres et sections, *etc.*

Le style des pages

LaTeX introduit la commande `\pagestyle{-}` pour installer un style qui va gérer les entêtes et pieds de page.
La commande `\thispagestyle{-}` n'affecte que la page en cours.

Par défaut, 4 styles sont prédéfinis :

`empty` L'entête et le pied de page sont vides.

`plain` L'entête est vide, le pied de page contient le numéro de la page centré.

`headings` L'entête contient les chapitres et sections (ou sous divisions) ainsi que le numéro de la page, le pied de page est vide.

`myheadings` L'entête contient le numéro de la page et des informations fournies par l'utilisateur, le pied de page est vide.

Dans la classe **book**, un chapitre commence dans le style `plain`, les autres pages sont dans le style `headings` par défaut.

Les commandes `\markboth{`*mark1*`}{`*mark2*`}` et `\markright{`*mark3*`}` servent à remplir l'entête dans les styles `headings` et `myheadings`.
L'esprit de ces commandes est que *mark1* correspond à une entrée principale, et *mark2* et *mark3* à des entrées secondaires.
Les commandes `\chapter`, `\section`… utilisent ces commandes (ou plutôt des commandes équivalentes) en y insérant le libellé des chapitres, sections…

Les commandes \leftmark et \rightmark

Il est possible de récupérer des données insérées dans \markboth{*mark1*}{*mark2*} et \markright{*mark3*} : on dispose des commandes \leftmark et \rightmark.

Une fois la page terminée, la situation est la suivante :

- \leftmark contient la valeur *mark1* du *dernier* \markboth rencontré sur la page ➟ entrée principale.
- \rightmark contient la valeur *mark2* ou *mark3* du *premier* \markboth ou \markright rencontré depuis le début de la page, sinon du dernier de la page précédente ➟ entrées secondaires.

En plus clair (?), dans le cas d'un livre : lorsque la page se termine, \leftmark récupère le nom du chapitre qui a été créé en dernier, alors que \rightmark récupère le nom de la première section créée en début de page (ou de la dernière section créée sur la page précédente).

Le *package* **extramarks** définit des nouvelles commandes pour récupérer les autres entrées des commandes \markboth et \markright :

- \firstleftmark récupère *mark1* du *premier* \markboth rencontré sur la page ;
- \lastrightmark récupère *mark2* ou *mark3* du *dernier* \markboth ou \markright rencontré sur la page.

Ces commandes peuvent servir à définir de nouveaux styles de pages.

Le *package* fancyhdr

Le *package* **fancyhdr** définit un nouveau style de page appelé `fancy`.
Ce style est composé de 6 parties : les parties gauches, centrées et droites de l'entête et du pied de page, qu'on peut personnaliser à l'aide des 2 commandes :
`\fancyhead[-]{-}` et `\fancyfoot[-]{-}`.
L'argument optionnel utilise les lettres `L`, `C`, `R` pour désigner l'une des trois zones, et `O`, `E` pour désigner les pages impaires ou paires.
On peut définir l'aspect des lignes sous l'entête et au dessus du pied par les commandes `\headrule` et `\footrule`, et leur épaisseur par les commandes `\headrulewidth` et `\footrulewidth`.

```
\pagestyle{fancy} \fancyhead{} \fancyfoot{}
\fancyhead[LO]{\slshape\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\fancyfoot[LO,RE]{\thepage}\fancyfoot[RO,LE]{\today}
\renewcommand{\headrule}{\dotfill}
\renewcommand{\footrulewidth}{0pt}
```

La commande `\fancypagestyle{-}{-}` permet de redéfinir un style déjà défini.
Si on n'utilise pas l'option `twoside`, on peut utiliser les 6 commandes `\lhead{-}`, `\chead{-}`, `\rhead{-}`, `\lfoot{-}`, `\cfoot{-}` et `\rfoot{-}`.

Lire la documentation pour plus d'informations.

Le *package* `titlesec` : le style des pages

Le *package* **`titlesec`** permet de (re)définir des styles de page.

La commande `\newpagestyle{`*nom*`}[`*options globales*`]{`*code*`}` et son équivalent `\renewpagestyle` (re)définissent un style nommé *nom* selon le code donné par *code*. Les *options globales* s'appliquent aux entêtes et aux pieds. Le code *code* repose sur deux commandes `\sethead[-][-][-]{-}{-}{-}` et `\setfoot[-][-][-]{-}{-}{-}`. Ces lots de trois cases correspondent aux parties gauches, centrées et droites des entêtes et des pieds. Les parties optionnelles concernent les pages paires (voir les arguments comme sur un livre ouvert sur deux pages).

De nombreuses commandes permettent d'insérer dans ces arguments des données diverses :

- `\headrule`, `\footrule`, et variantes… pour des filets ;
- `\chaptertitle`, `\sectiontitle`… pour les titres de chapitres…
- `\ifthechapter{-}{-}`, `\ifthesection{-}{-}`…

Des options du *package* permettent de choisir le type des *marks* :
`outermarks`, `innermarks`, `topmarks`, `botmarks`.
On peut aussi directement désigner ces *marks* grâce aux commandes :
`\bottitlemarks`, `\toptitlemarks`, `\firsttitlemarks`, `\nexttoptitlemark`, `\outertitlemarks`, `\innertitlemarks`.

Le *package* `titlesec` : exemples de styles

Rédéfinition du style `plain` pour insérer la date et mettre le numéro de page à l'extérieur de la page, le tout en gras et dans la famille sans-sérif :

```
\renewpagestyle{plain}
  [\normalfont\sffamily\bfseries\mathversion{bold}]
  {\setfoot[\today][][]{}{}{\today}
   \sethead[\usepage][][]{}{}{\usepage}}
```

Définition d'un nouveau style :

```
\newpagestyle{main}
  [\normalfont\sffamily\bfseries\mathversion{bold}]
  {\headrule
   \setfoot[\today][][]{}{}{\today}
   \sethead[\usepage][][\ifthechapter%
     {\chaptertitlename\ \thechapter\ - \chaptertitle}
     {\chaptertitle}]
   {\ifthesection{\thesection\ - \sectiontitle}
     {\chaptertitle}}%
   {}{\usepage}}
```

Le *package* `titlesec` : les commandes de section

La finalité première du *package* **`titlesec`** est de personnaliser l'apparence des commandes de type sections.
Deux commandes sont définies pour ça :

- `\titleformat{-}[-]{-}{-}{-}{-}[-]` pour décrire comment rendre un commande du type `\chapter`, `\section`…
- `\titlespacing*{-}{-}{-}{-}[-]` pour définir les espacements avant et après les commandes `\chapter`, `\section`…

Les arguments de `\titleformat` désignent des points précis comme : la forme générale du label (nouveau paragraphe, en ligne…), le formatage général d'un point de vue typographique, le formatage du label seul, le formatage du titre seul, les espaces horizontaux entre le label et le titre…
Les arguments de `\titlespacing*` concernent les espacements horizontaux à gauche et à droite, et les espacements verticaux avant et après.

Des commandes pour la justification et pour placer des filets sont fournies.
Des commandes plus légères sont aussi fournies pour personnaliser plus simplement les commandes `\chapter`, `\section`…

➟ Consulter la documentation pour des explications plus fournies et où de nombreux exemples sont donnés.
À noter que le *package* compagnon **`titletoc`** permet de personnaliser la table des matières dans le même esprit.

Le *package* `titlesec` : exemples

Redéfinition du style des chapitres :

```
\titleformat{\chapter}[display]
  {\raggedright\normalfont\sffamily\LARGE\bfseries%
       \mathversion{bold}}
  {\chaptertitlename\ \thechapter}{0pt}{\huge}

\titlespacing*{\chapter}{0pt}{20pt}{4ex plus 1ex minus 1ex}
\assignpagestyle{\chapter}{empty}
\newcommand{\chapterbreak}{\cleardoublepage}
```

Redéfinition du style des sections :

```
\titleformat{\section}
  {\normalfont\sffamily\large\bfseries\mathversion{bold}}
  {\thesection}{1em}{}

\titlespacing*{\section}{0pt}{3ex plus 2.5ex minus .8ex}
  {2ex plus .5ex}
```

Autres éléments du document

Où l'on apprend à personnaliser la page de garde et à encadrer de façon amusante du texte…

La page de garde

Dans la classe `article`, la commande `\maketitle` produit une page de garde avec des données renseignées auparavant par l'auteur à l'aide des commandes `\title{-}`, `\author{-}`, `\date{-}` et `\thanks{-}`.

On peut créer soi-même une mise en page convenable pour réaliser une page de garde.
L'environnement `titlepage` de LaTeX créé une page de titre de style vide, et la page suivante commence avec le numéro 1.
À l'intérieur de cet environnement on peut faire ce qu'on veut.

Le document **Some Examples of Title Pages** fournit un grand nombre d'exemples de pages de gardes, avec le code pour les réaliser.

Le *package* `titling` permet de personnaliser le rendu de la commande `\maketitle`.
Il définit des commandes `\pretitle{-}`, `\posttitle{-}`, `\preauthor{-}`, `\postauthor{-}`, `\predate{-}`, `\postdate{-}` qui insèrent du code avant et après les éléments spécifiés : `{\pretitle... \title... \posttitle}`.

```
\pretitle{\begin{flushright}\LARGE\sffamily\bfseries}
\posttitle{\par\end{flushright}\vskip 0.5em}
```

Il est même possible d'insérer du code supplémentaire entre les éléments du titre par des commandes du type `\maketitlehooka{-}`… `\maketitlehookd{-}`.
Il définit l'environnement `titlingpage` qui compose la page de garde sur une page à part.

Le *package* fancybox

Par défaut, LaTeX définit une commande pour encadrer du texte :

`\fbox{-}` → **Primum movens** (Voir aussi `\framebox` avec ses options.)

Le *package* **fancybox** définit une série de commandes et d'environnements destinés à encadrer des éléments de la page : du texte, des listes, des flottants, des formules de mathématique, et même toute la page !

Les boites définies grâce à **fancybox** sont :

`\shadowbox{-}` → **Primum movens**

`\doublebox{-}` → **Primum movens**

`\ovalbox{-}` → **Primum movens**

`\Ovalbox{-}` → **Primum movens**

De nombreux paramètres permettent de gérer l'aspect de ces encadrements.

Remarque technique : Pour les amateurs de programmation, **fancybox** définit aussi des commandes pour la gestion des environnements de type `verbatim`…

On a déjà vu que le *package* **empheq** pouvait utiliser les boites de **fancybox** en mode mathématique.

Packages en vrac

Le *package* **lscape** définit l'environnement landscape qui bascule son contenu en format paysage, tandis que l'entête et le pied restent dans le format portrait. Le contenu peut parcourir plusieurs pages.

Le *package* **soul** définit des commandes pour modifier l'espacement entre les lettres, passer en petites capitales, souligner, barrer, surligner et mettre en valeur du texte.
Il est possible de personnaliser le comportement des commandes définies.

Le *package* **lastpage** permet d'accéder au numéro de la dernière page.

Le *package* **numprint** propose de formater les grands nombres, avec notation scientifique, et gestion des unités physiques. Il permet aussi d'imprimer des longueurs et des compteurs.

Le *package* **relsize** définit des commandes pour gérer la taille du texte relativement à la taille de l'environnement : `\relsize{-}`, `\smaller[-]`, `\larger[-]`…

Le *package* **setspace** définit des commandes pour gérer l'espacement entre les lignes.

Les *packages* **showlabels** et **showkeys** permettent d'afficher les labels attribués par l'auteur. Utile dans la phase préparatoire d'un document. Nombreuses options disponibles.

Le *package* **listings** permet d'imprimer des codes informatiques, avec coloration syntaxique, numérotation des lignes, encadrement… Il reconnaît de nombreux langages informatiques.