## Chapitre troisième
## Utilisation avancée

# 4 Mise en page

Je vais maintenant te parler de mise en page. Au sens strict des typographes d'autrefois, la mise en page est l'opération qui consiste, une fois que le texte est composé en une longue série de lignes, à faire des pages à partir de cela : prendre la bonne quantité de texte, les divers autres éléments (notes, figures, etc.) et les assembler en une page avec quelques éléments autour (titre courant, pied de page, etc.).

La notion clef pour la manipulation de la mise en page avec LaTeX est la notion de style de page, je commencerai donc par celle-là. Après une rapide présentation, on verra quelques packages permettant d'influer sur celle-ci, en commençant par le plus complet et en finissant par le plus simple.

La mise en forme du titre était, du temps des typographes anciens, dévolue au metteur en page, nous aborderons donc le sujet ici.

Ensuite, nous verrons comment influer sur le choix des marges, soit en respectant certaines règles considérées comme de bon goût, soit en ayant un contrôle total.

Enfin, nous verrons comment règler l'espacement vertical du document (là encore, cette tâche était dévolue au metteur en page, puisque lui seul avait une vision page par page du document) et comment jouer avec les changements de page.

## 4.1 Style de page

Quatre styles de page classiques existent pour LaTeX $2_\varepsilon$ :

1) Le style `empty` : rien. Ni en-tête, ni pied de page. Le numéro de page ne figure donc nulle part. Classiquement employé pour les lettres ou les couvertures de rapport.

2) Le style `plain` : c'est le style de page par défaut. Pas d'en-tête et le numéro de page est centré en pied de page.

   Ce style de page est utilisé par LaTeX pour la première page de chaque chapitre. C'est une assez mauvaise idée : si on souhaite avoir une mise en page différente, on est obligé de redéfinir ce style, ce qui peut avoir des conséquences fâcheuses. Nous verrons comment y remédier.

3) Le style `headings` : avec en-tête. Le numéro et le nom du chapitre en cours sont reportés en en-tête à gauche des pages de droite et le numéro et le nom de la section sont reportés en en-tête à droite des pages de gauche (*i.e.* les titres courants sont à l'intérieur). Le numéro de page est en haut à droite des pages de droite et en haut à gauche des pages de gauche (à l'« extérieur », pour aider à la recherche d'une page précise). Un document « recto » est composé exclusivement de pages de droite. On demande un document « recto-verso » en passant l'option `twoside` au `\documentclass`. Les pages de droite se reconnaissent au fait qu'elles ont des numéros impairs, comme l'imposent les règles élémentaires de la typographie classique [153].

   Avec ce style de page, les premières pages de chapitre seront composées en `plain` pour éviter qu'un en-tête ne se retrouve au dessus du titre du chapitre. On pourra souhaiter

```
\documentclass{expl}
\def\texte{Texte un peu long
repris encore et toujours pour
remplir. }
\begin{document}
\sloppy

\section{Section}
\texte \texte \par
\subsection{Essai}
\texte \texte \par
\texte \texte \par
\texte \texte \par
\end{document}
```

# 1 Section

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

## 1.1 Essai

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore

1

FIG. 4.1: Le style de page plain

---

que les chapitres commencent toujours sur des pages de droite[90], cela s'obtient en mettant l'option openright du `\documentclass`.

4) Le style myheading : personnalisé. Les possibilités de personnalisation de LaTeX sont tellement peu intéressantes, par rapport à ce que nous verrons dans les packages, que je les passerai sous silence.

Deux commandes existent pour changer de style de page :
- `\thispagestyle` : pour changer le style de la page en cours ;
- `\pagestyle` : pour changer le style des pages jusqu'à nouvel ordre et à partir de la prochaine.

La figure 4.1 présente un exemple du style de page par défaut, plain : pas de titre courant, le numéro de page au centre du pied de page. C'est le style par défaut pour LaTeX et c'est le style retenu pour les premières pages de chapitre.

La figure 4.2 page suivante présente le même document que précédemment, en recto-verso, mais cette fois-ci dans le style headings : le titre de la section courante est repris en haut des pages de gauche et le titre de la sous-section courante est repris en haut des pages de droite. Si le document contenait des chapitres (par exemple de la classe **report** ou **book**) alors les titres de chapitre seraient repris sur les pages de gauche et les titres de section sur les pages de droite.

LaTeX ne permet pas beaucoup d'adaptations sur le style de page, sauf à se plonger dans du codage complexe. Les packages que nous allons maintenant étudier permettent d'améliorer ça.

## 4.2 Comprendre et utiliser les marques

La mise en page est un art qui, pour être parfaitement maîtrisé avec TeX, et donc avec LaTeX, demande de comprendre comment TeX gère les pages.

TeX travaille un peu comme travaillaient les typographes de l'ancien temps : ils faisaient composer des presque-pages par des compositeurs, qui amenaient tout ça au metteur en page.

[90] Aussi appelées « belles pages » par les typographes.

```
\documentclass
    [twoside]{expl}
\pagestyle{headings}
% \texte comme d'habitude
\begin{document}
\setcounter{page}{4}
\sloppy

\section{Section}
\texte \texte \par
\subsection{Essai}
\texte \texte \par
\texte \texte \par
```

4 *1 SECTION*

**1 Section**

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

**1.1 Essai**

Texte un peu long repris encore et toujours

*1.1 Essai* 5

pour remplir. Texte un peu long repris encore et toujours pour remplir.

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et

FIG. 4.2: Le style de page headings

---

Lui prenait la quantité de matériel nécessaire pour chaque page en tâchant de respecter un certain nombre de règles qui semblent simples : le début du texte d'une note doit apparaître sur la même page que l'appel de la note, ne pas laisser la dernière ligne d'un paragraphe seule sur une page, veiller à ce que deux pages en vis-à-vis soient visuellement de la même longueur, etc. Ensuite seulement le metteur en page plaçait autour de cette page les titres courants, les notes, les marges, etc.

TEX procède de manière très similaire : il accumule suffisament de texte jusqu'à avoir de quoi composer une page, puis il appelle la routine chargée de faire la mise en page. Pendant qu'il est en train d'accumuler du texte, il n'a qu'une idée assez vague de là où il en est : il ne traite pas ligne à ligne mais paragraphe par paragraphe ; et il y a un certain nombre de constructions qu'il ne peut traiter que d'un bloc (les tableaux, par exemple).

Ainsi, il peut très bien avoir lu les titres des deux premières sections avant de chercher à faire la mise en page, puis se rendre compte, ce faisant, que le second titre de section ira, tout compte fait, sur la prochaine page et pas sur celle-ci.

Il ne suffit donc pas de mémoriser simplement le titre de la section quand on la commence pour savoir sur quelle page elle finira. Par exemple, ce bout de code là poserait des problèmes :

```
\section{Un}
\def\sectioncourante{Un}
bla bla bla...
\section{Deux}
\def\sectioncourante{Deux}
% ICI, \TeX cherche a faire la mise en page
la suite du texte
```

Si finalement il s'avère que le titre de la seconde section finit en page 2, on ne peut pas se fier au contenu de `\sectioncourante` pour savoir quel titre il faut mettre dans le titre courant en haut de la page. Dans notre exemple, quand la routine chargée de faire la mise en page sera appelée pour la page 1, notre commande contiendra `Deux`, alors que seule la section 1 sera effectivement commencée sur la page.

### 4.2.1 Les marques

TEX résoud ce problème là en fournissant quelques primitives : `\mark` permet de mémoriser un texte pour le reprendre lors de la mise en page. Durant la mise en page, `\firstmark` permet de reprendre la valeur du premier texte mémorisé sur la page, `\botmark` permet de reprendre la valeur du dernier texte mémorisé sur la page et `\topmark` permet de reprendre soit le dernier texte mémorisé sur une page précédente, soit le premier de la page (c'est souvent ce texte-là qu'on veut reprendre en haut de page, d'où son nom). Quand il n'y a pas de marque du tout sur une page, les trois valent donc la même chose que le `\botmark` de la page précédente.

Cette mécanique basée sur les primitives de TEX est utilisée par LATEX pour gérer les titres courants.

Je ne te détaillerai pas comment les marques de LATEX sont construites avec celles de TEX.

LATEX prévoit que tu puisses utiliser dans ta mise en page `\leftmark` et `\rightmark`. Il suppose que `\rightmark` change le plus souvent, puisque dans la majorité des livres, le titre courant des pages de gauche est le chapitre, et celui des pages de droite est la section dans le chapitre.

Tu peux positionner la valeur des deux marques en faisant appel à la commande `\markboth` :

```
\markboth{gauche}{droite}
```

Tu peux positionner la valeur de la marque de droite en faisant appel à `\markright` :

```
\markright{droite}
```

Quand tu utilises `\leftmark` dans une mise en page, LATEX te donne la dernière valeur de la marque de gauche sur la page en cours (l'équivalent du `\botmark` de TEX). Quand tu utilises `\rightmark`, il te donnera la première marque de droite de la page[91].

### 4.2.2 Ce que LATEX fait avec les marques

Quand tu utilises une commande de sectionnement, `\chapter` par exemple, LATEX fait appel à `\chaptermark` en lui passant comme paramètre le titre du nouveau chapitre. Dans les classes de document standard, par exemple dans **report**, `\chaptermark` fait appel à `\markboth`.

La définition ressemble[92] à ça[93] :

```
\newcommand{\chaptermark}[1]{%
  \markboth{\MakeUppercase{Chapitre \thechapter. #1}}{}
}
```

Si tu veux que le titre de chapitre ne soit pas en majuscules dans ton titre courant, il suffit de faire appel dans le préambule de ton document à :

```
\renewcommand{\chaptermark}[1]{%
  \markboth{Chapitre \thechapter. #1}{}
}
```

Bien entendu, il en va de même avec les `\section`, `\subsection`, etc.

[91] Ne me demande pas pourquoi ce choix parfaitement idiot, je ne le comprends pas et les gens qui assurent la maintenance de LATEX depuis plus de 10 ans ne le comprennent pas non plus. Nous verrons plus tard quelques packages qui permettent d'améliorer le comportement de LATEX sur ce point.

[92] La vraie définition est un peu plus subtile, en particulier elle n'écrit pas directement chapitre, mais un mot qui dépend de la langue en cours, et ne l'écrit que si les chapitres dans ce document là sont numérotés. Cette définition est en fait mise en place par le style de page headings si le document est recto-verso.

[93] Si tu ne connais pas encore, l'utilisation de `\newcommand` est expliquée au chapitre neuvième page 571.

| Commande | Rôle |
|---|---|
| `\lhead` | Partie gauche de l'en-tête |
| `\chead` | Partie centrale de l'en-tête |
| `\rhead` | Partie droite de l'en-tête |
| `\lfoot` | Partie gauche du pied de page |
| `\cfoot` | Partie centrale du pied de page |
| `\rfoot` | Partie droite du pied de page |

TAB. 4.1: Les six commandes simples de `fancyhdr`

```
* \usepackage{fancyhdr}
* \pagestyle{fancy}
* \lhead{\leftmark}
* \rhead{\thepage}
* \chead{} \cfoot{}
* \lfoot{} \rfoot{}
  \section{Essai}
  \texte\par
  \subsection{Test}
  \texte\texte\texte\par
  \texte\texte\texte\par
```

1 ESSAI 4

**1 Essai**

Texte un peu long qui se répète encore et toujours pour remplir.

**1.1 Test**

Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se ré-

1 ESSAI 5

pète encore et toujours pour remplir. Texte un peu long qui se répète encore et toujours pour remplir.

Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se répète

FIG. 4.3: Utilisation simple de `fancyhdr`

---

## 4.3 Style de page personalisé : le package `fancyhdr`

Ce package est assez ancien. Autrefois, il s'appelait `fancyheadings` ([210, 209]) et fonctionnait avec LaTeX2.09 (avant 1994). Nous le devons à Piet VAN OOSTRUM [211].

Le package `fancyhdr` considère que l'en-tête du document (dit aussi titre courant, dans la littérature) et le pied de page sont décomposés en trois parties indépendantes (gauche, centre, droite). Il permet de spécifier chacune des six (trois en haut, trois en bas) parties indépendament.

### 4.3.1 Le style de page fancy

En première approche, on peut utiliser les six commande du tableau 4.1, elles permettent de définir rapidement le style de page fancy. Ainsi l'exemple de la figure 4.3 nous présente une tentative (maladroite et ratée) de reproduire le comportement de headings.

Cette tentative est ratée essentiellement parce qu'on n'a pas tenu compte de l'aspect recto-verso du document : les côtés droite et gauche doivent être échangés selon qu'on est sur une page de droite (impaire) ou de gauche (paire).

| **Selecteur** | **Signification** |
|---|---|
| `H` | En tête (header)[1] |
| `F` | Pied de page (footer)[1] |
| `O` | Page impaire (odd) |
| `E` | Page paire (even) |
| `L` | Partie gauche (left) |
| `C` | Partie centrale (center) |
| `R` | Partie droite (right) |

[1] Ces deux sélecteurs n'ont de sens que pour la commande `\fancyhf`, `H` est pré-sélectionné pour `\fancyhead`, de même `F` l'est pour `\fancyfoot`.

TAB. 4.2: Sélecteurs pour les commandes `\fancyfoot`, `\fancyhead` et `\fancyhf`.

`\lhead[`⟨*page gauche*⟩`]{`⟨*page droite*⟩`}`

Une première façon de faire ça est d'utiliser l'argument optionnel des six commandes de base. Si l'argument optionnel (entre crochets) n'est pas précisé, alors les pages de gauche seront identiques aux pages de droite.

Ainsi, on peut règler notre problème avec ces appels un peu long :

```
\lhead[\thepage]{\leftmark}
\rhead[\leftmark]{\thepage}
```

Ça marche, mais ça pose souci : il faut rentrer les informations utiles deux fois (c'est une fois de trop) ; et pourtant c'est un cas très classique d'échanger droite et gauche dans les en-têtes ou les pieds de pages.

`\fancyfoot[`⟨*options*⟩`]{`⟨*texte*⟩`}`

`\fancyhead[`⟨*options*⟩`]{`⟨*texte*⟩`}`

Le package `fancyhdr` fournit une autre série de commandes permettant de faire ça plus simplement, ce sont les deux commandes `\fancyfoot` et `\fancyhead`, qui prennent comme premier argument optionnel (entre crochets, donc) une combinaison des sélecteurs du tableau 4.2.

Ainsi, notre appel devient :

```
\fancyhead[LO,RE]{\leftmark}
\fancyhead[LE,RO]{\thepage}
```

`\fancyhf[`⟨*options*⟩`]{`⟨*texte*⟩`}`

Enfin, il existe la forme la plus générale de ces commandes, `\fancyhf`, qui utilise les trois variétés de sélecteurs du tableau 4.2.

L'exemple de la figure 4.4 page suivante reprend ces fonctionnalités pour définir un style de page plus convenable pour du recto-verso, style assez voisin de ce que fait headings (sauf que les en-têtes ne sont pas en italique).

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhf{}
\fancyhead[LO,RE]{\leftmark}
\fancyhead[LE,RO]{\thepage}
\section{Essai}
\texte\par
\subsection{Test}
\texte\texte\texte\par
\texte\texte\texte\par
```

4 1 ESSAI

**1 Essai**

Texte un peu long qui se répète encore et toujours pour remplir.

**1.1 Test**

Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se ré-

1 ESSAI 5

pète encore et toujours pour remplir. Texte un peu long qui se répète encore et toujours pour remplir.

Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se répète

FIG. 4.4: Mise en page recto-verso avec `fancyhdr`

---

### 4.3.2 Personnaliser encore

Nous venons de voir, globalement, comment on peut intervenir sur le *contenu* des en-têtes et des pieds de page de nos documents. `fancyhdr` offre quelques autres pistes intéressantes.

La première piste que nous allons voir consiste à séparer les en-têtes et pieds de page du corps du texte par une ligne. En fait, pour `fancyhdr`, cette ligne est toujours présente, et son épaisseur est règlable. Par défaut son épaisseur est nulle dans le style plain, et de 0,4 points dans le style fancy.

On peut modifier ça en faisant ces appels là :

```
\renewcommand\headrulewidth{.4pt}
\renewcommand\footrulewidth{.4pt}
```

Comme tu t'en souviens peut-être, LaTeX peut gérer des notes dans les marges. La deuxième piste que nous allons creuser est de savoir si ces notes sont comptées dans la largeur de l'en-tête et du pied de page ou pas. Par défaut, elles ne le sont pas. En fait, il existe une longueur, `\headwidth`, qui contient la largeur de l'en-tête. Par défaut, c'est la même que pour le corps du texte. Si tu veux que l'en-tête recouvre également les notes en marge, tu peux utiliser cette construction là :

```
\addtolength{\headwidth}{\marginparwidth}
\addtolength{\headwidth}{\marginparsep}
```

L'exemple de la figure 4.5 page suivante présente une utilisation de ces possibilités. Note bien que, malgré son nom qui pourrait prêter à confusion, la longueur `\headwidth` indique la largeur de l'en-tête *et* du pied de page. Ainsi, sur une mise en page comme celle de l'exemple, si on supprime la ligne de séparation du pied de page, le numéro de page, centré en bas, aura l'air mal centré : ton œil va juger le centrage par rapport au texte, alors que ce centrage aura été calculé par rapport au texte *et* aux notes en marge.

### 4.3.3 Définir un style de page

Tout ce que nous venons de voir pour le style de page fancy est en fait applicable à n'importe quel style de page, soit un qui existe déjà et qu'on peut redéfinir (très pratique

```
% fancyhdr, comme avant
\cfoot{\thepage}
\renewcommand\headrulewidth{.4pt}
\renewcommand\footrulewidth{.4pt}
\addtolength{\headwidth}
            {\marginparwidth}
\addtolength{\headwidth}
            {\marginparsep}
% \texte, comme avant
\section{Essai}
\texte\par
\subsection{Test}
\texte\marginpar{Note 1}
\texte\marginpar{Note 2}
\texte\par
\texte\ \texte\
\texte\par
```

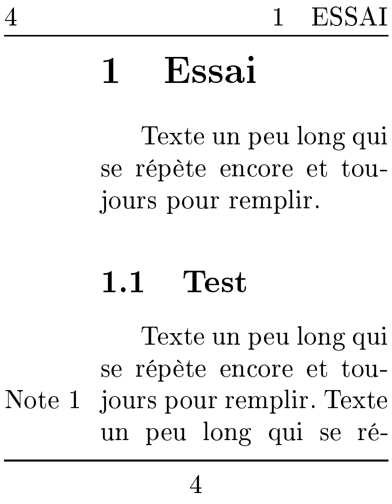



FIG. 4.5: Mise en page avec des en-têtes plus larges

---

de pourvoir redéfinir le style plain, puisqu'il est utilisé par LaTeX sans te demander ton avis pour les têtes de chapitre[94]) ou bien en définir un nouveau (pratique, par exemple, pour que le style de page soit différent entre le texte et l'index, parce qu'il est rare d'utiliser des notes en marge dans l'index).

`\fancypagestyle{⟨style⟩}{⟨définitions⟩}`

L'idée est simple : tout ce que tu aurais pu vouloir personnaliser avec `fancyhdr` peut s'appliquer à un autre style que fancy si c'est dans le deuxième argument de `\fancypagestyle`. Le premier argument étant le style que l'on redéfinit.

Ainsi, par exemple, pour refaire le style plain pour que le numéro de page soit à gauche des pages de droite, ou à droite des pages de gauche :

```
\fancypagestyle{plain}{
  \fancyhf{}
  \fancyfoot[RO,LE]{\thepage}
}
```

## 4.4 Style de page personnalisé : le package `titlesec`

Ce package, que nous devons à Javier BEZOS [17], a pour but premier de changer la mise en forme des titres de section. Cette fonctionnalité principale est longuement décrite au paragraphe 6.11 page 341. Mais il offre également, si on le charge avec la bonne option, une alternative intéressante à `fancyhdr`.

Le tableau 4.3 page suivante présente les deux options utiles pour activer cette fonctionnalité étendue du package `titlesec`.

[94]Si tu veux réserver le style plain pour un autre usage et en utiliser un autre, que tu peux définir avec `fancyhdr`, pour les chapitres, le package `sectsty` propose une alternative intéressante. Le package `sectsty` est décrit au paragraphe 6.8 page 336.

```
* \usepackage{fancyhdr}
* \pagestyle{fancy} \fancyhf{}
* \fancyhead[LO,RE]{\leftmark}
* \fancyhead[LE,RO]{\thepage}
* \renewcommand
*     {\headrulewidth}{.4pt}
* \fancypagestyle{plain}{
*   \fancyhf{}
*   \fancyfoot[LE,RO]{\thepage}
*   \renewcommand
*     {\headrulewidth}{0pt}
*   }
* % \texte comme avant
  \section{Essai}
  \thispagestyle{plain}
  \texte\par
  \subsection{Test}
  % contenu comme avant
```

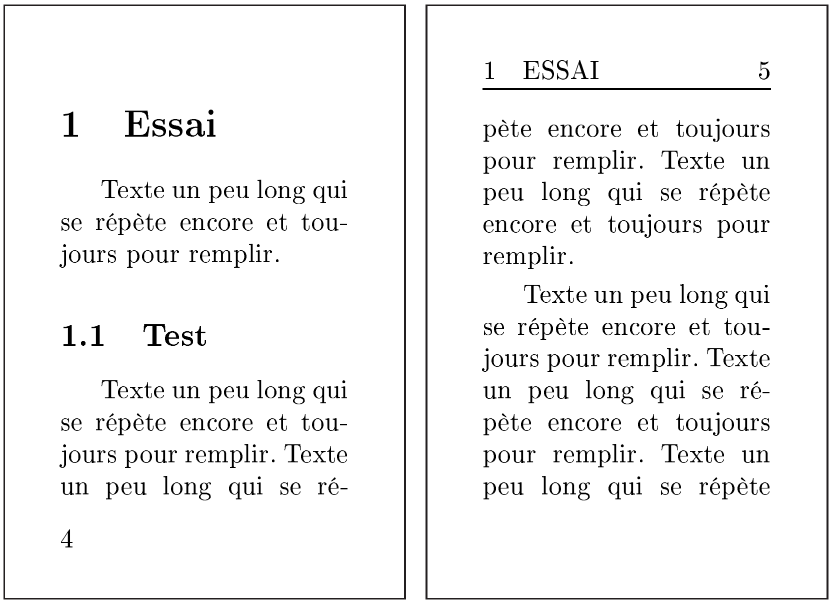



FIG. 4.6: Redéfinition de plain avec `fancyhdr`

| Option | Description |
|---|---|
| pagestyles | Active la gestion du style de page par le package qui, sans cette option, se cantonne à la mise en forme des titres de section. |
| floatps | Active la gestion différenciée des styles de pages pour les pages ayant des flottants. Cette fonctionnalité n'est pas activée par défaut car elle modifie certaines commandes internes de LaTeX et pourrait donc être incompatible avec d'autres packages. |

TAB. 4.3: Options du package `titlesec` pour la mise en page

---

Le principe est relativement simple : tout est centré sur le fait de définir ou redéfinir des styles de page, un peu comme on définirait des commandes. Ensuite, tout est question de ce qu'on place dans la définition du style de page. Là où `fancyhdr` sait faire à peu près tout ce qu'on peut vouloir, ce package se cantone à faire des choses classiques : il fait des en-têtes et des pieds de page en trois parties (gauche, centre, droite). Si tu veux faire plus complexe, il vaut mieux utiliser `fancyhdr`.

### 4.4.1 Définition de styles de page

`\newpagestyle{`⟨*nom*⟩`}[`⟨*commandes globales*⟩`]{`⟨*commandes*⟩`}`

`\renewpagestyle{`⟨*nom*⟩`}[`⟨*commandes globales*⟩`]{`⟨*commandes*⟩`}`

La commande `\newpagestyle` (et sa petite sœur `\renewpagestyle` qui sert à redéfinir) définit un nouveau style de page. Le premier paramètre indique le nom du nouveau style de page (ou de celui qu'on redéfinit), le deuxième (optionnel) donne des commandes fragiles qui

```
\usepackage[pagestyles]
               {titlesec}
\renewpagestyle{plain}{%
  \setfoot*{}{}{\thepage}
}
% \texte comme d'habitude
\pagestyle{plain}
\texte \texte \texte \par
\texte \texte \texte \par
% etc
```

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore

4

et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

5

FIG. 4.7: Exemple d'utilisation de `titlesec`

---

seront éxécutées à chaque en-tête ou pied de page dans ce style-là et le troisième argument donne les commandes caractérisant le nouveau style. Toutes les commandes que je t'indiquerai par la suite sont supposées, a priori, aller dans ce troisième argument pour indiquer comment tu veux construire tes en-têtes et tes pieds de page.

`\setheadrule`{⟨*épaisseur*⟩}

`\setfootrule`{⟨*épaisseur*⟩}

La commande `\setheadrule` permet d'indiquer qu'il faut faire une ligne sous l'en-tête et d'en préciser l'épaisseur. Le raccourci `\headrule` indique de faire une ligne de l'épaisseur habituelle par défaut (0,4 points). Les commandes `\setfootrule` et `\footrule` font la même chose pour le pied de page. À utiliser dans l'argument {⟨*commandes*⟩} de `\newpagestyle`.

En vrai, ces commandes servent à redéfinir les commandes `\makeheadrule` et `\makefootrule` qui se chargent de dessiner ces filets. Bien entendu, tu peux redéfinir toi-même ces commandes si la définition par défaut ne te plaît pas. La définition par défaut de `\makeheadrule` est :

```
\renewcommand\makeheadrule{%
    \rule[-.3\baselineskip]{\linewidth}{.4pt}%
}
```

`\sethead`[⟨*verso gauche*⟩][⟨*verso centre*⟩][⟨*verso droite*⟩]{⟨*recto gauche*⟩}
{⟨*recto centre*⟩}{⟨*recto droite*⟩}

`\sethead*`{⟨*recto gauche/verso droite*⟩}{⟨*centre*⟩}{⟨*recto droite/verso gauche*⟩}

La commande `\sethead` permet d'indiquer le contenu (et la mise en forme, bien entendu) des trois parties de l'en-tête. Si tu n'utilises pas les arguments optionnels, alors les pages paires et impaires seront sur le même modèle ; par contre, si tu utilises un argument optionnel, il faut utiliser les trois. La commande `\sethead*`, qui ne prend pas d'argument optionnel, compose les pages paires (verso) en miroir des pages impaires (recto). Les commandes `\setfoot` et `\setfoot*` fonctionnent de la même manière pour les pieds de page.

Ainsi, tu peux obtenir le même effet que le package `rplain` (avoir le numéro de page en bas côté marge extérieure) en appelant :

```
\setfoot[\thepage][][]{}{}{\thepage}
% ou plus simplement
\setfoot*{}{}{\thepage}
```

La figure 4.7 page précédente présente cet exemple très simple d'utilisation de `\renewpagestyle` et de `\setfoot`.

Dans les arguments de `\sethead` (et des autres), tu peux utiliser des informations qui ont été placées dans les marques par les titres de section (ces titres sont redéfinis par le package `titlesec`, puisque c'est quand même son boulot principal, mais le rendu par défaut est identique à celui de LaTeX). L'accès est assez simple : `\thesection`, par exemple, ne te renvoit pas le numéro de la dernière section commencée, mais le numéro de la section qui est dans la marque, ce qui est nettement mieux. Bien entendu, ça fonctionne avec les autres niveaux de sectionnement (`\thechapter` pour les chapitres, `\thesubsection` pour les sous-sections, etc.). De la même manière, `\sectiontitle` te renvoit le titre de la section, en le prenant dans la marque ; et bien entendu, il y a aussi `\chaptertitle` pour les chapitres, `\subsectiontitle` pour les sous-sections, etc.

`\ifthechapter`{⟨*si vrai*⟩}{⟨*si faux*⟩}

Plus amusant, la commande `\ifthechapter` permet de savoir s'il y a ou non un titre de chapitre courant. Bon, d'accord, pour les chapitres c'est pas fondamental. Par contre, pour les sections c'est intéressant : s'il y a dix pages de texte entre le début du chapitre et la première section, pendant ces dix pages, la commande `\ifthesection` retournera faux, ce qui permet d'adapter la mise en page, par exemple en reprennant le titre du chapitre à la place du titre de section quand il n'y a pas encore de section. Ces commandes sont définies pour tous les niveaux de sectionnement et sont utilisables pour les deux niveaux de sectionnement qui sont reportés dans les marques.

`\setmarks`{⟨*premier niveau*⟩}{⟨*second niveau*⟩}

Cette commande, `\setmarks`, indique quels sont les deux niveaux de sectionnement qui doivent être disponibles dans les marques. Si tu utilises cette commande en dehors de `\newpagestyle`, elle définit les deux niveaux pour les marques par défaut, qui s'appliqueront à tous les styles de page qui ne le précisent pas. Si tu t'en sers *dans* la commande `\newpagestyle`, alors la définition est valable quand tu es dans ce style de page. Ça permet d'avoir des comportements différents entre le corps du texte et les annexes, par exemple.

La figure 4.8 page suivante présente un exemple d'utilisation du package `titlesec` pour définir un style de page qui place dans l'en-tête, sur les pages de gauche, le titre de la section, et sur les pages de droite, le titre de la sous-section, s'il y en a une, et s'il n'y en a pas le titre de la section. Cet exemple reprend l'usage classique de `\setmarks`, de `\ifthesubsection`, de `\sectiontitle`, et de `\thesection`.

```
\usepackage[pagestyles]
                      {titlesec}
\newpagestyle{JMPL}
[\outertitlemarks]
{\setmarks{section}{subsection}
 \sethead[\thepage][]
  [\thesection\quad
                 \sectiontitle]
  {\ifthesubsection
   {\thesubsection\quad
              \subsectiontitle}
   {\thesection\quad
                \sectiontitle}}
  {}{\thepage}
}
% \texte comme d'habitude
\pagestyle{JMPL}
\section{Un un}
\texte\par
\section{Deux deux}
\texte \texte \par
\texte \texte \texte \par
% etc
```

4 1 Un un

1 **Un un** Texte un peu long repris encore et toujours pour remplir.

2 **Deux deux** Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et

2 Deux deux 5

toujours pour remplir.

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore et toujours pour remplir.

Texte un peu long repris encore et toujours pour remplir. Texte un

```
\usepackage[pagestyles]
                      {titlesec}
% \newpagestyle{JMPL} comme
% ci-dessus.
% \texte comme d'habitude
\pagestyle{JMPL}
\section{Un un}
\texte\par
\section{Deux deux}
\texte\par
\subsection{Ss-un}
\texte \par
\subsection{Ss-deux}
\texte \texte \texte \par
% etc
```

4 1 Un un

1 **Un un** Texte un peu long repris encore et toujours pour remplir.

2 **Deux deux** Texte un peu long repris encore et toujours pour remplir.

2.2 Ss-deux 5

**2.1 Ss-un**

Texte un peu long repris encore et toujours pour remplir.

**2.2 Ss-deux**

Texte un peu long repris encore et toujours pour remplir. Texte un peu long repris encore

Ces deux exemples montrent l'intérêt d'utiliser la commande `\ifthesection` et le résultat auquel on peut arriver.

La mise en forme des titres de section correspond au format `drop` du package `titlesec`. C'est expliqué dans la partie sur la mise en forme des titres de section, au paragraphe 6.11 page 341.

FIG. 4.8: Exemples d'utilisation de `titlesec`

| Option | Description |
|---|---|
| outermarks | Choisi les marques externes par défaut. Prévu pour les usages en recto/verso. |
| innermarks | Choisi les marques internes par défaut. Prévu pour les usages en recto/verso. C'est le mode de fonctionnement par défaut de LaTeX et donc du package. |
| topmarks | Reprend dans les marques l'équivalent du `\topmark` de TeX (le premier titre de ce type là sur la page, ou à défaut le dernier précédent). Prévu pour les document recto. |
| botmarks | Reprend dans les marques l'équivalent du `\botmark` de TeX (le dernier titre de ce type là sur la page, ou à défaut le dernier précédent). Prévu pour les document recto. |

TAB. 4.4: Options du package `titlesec` sur les marques

`\widenhead[`⟨*gauche verso*⟩`][`⟨*droite verso*⟩`]{`⟨*gauche recto*⟩`}{`⟨*droite recto*⟩`}`

`\widenhead*{`⟨*intérieure*⟩`}{`⟨*extérieure*⟩`}`

Dans certaines mises en page, par exemple quand les notes en marges sont nombreuses et volumineuses, on souhaite élargir l'en-tête, et le pied de page, de manière à ce qu'ils considèrent que les notes en marge font partie de la page. C'est ce que font ces deux commandes. Les arguments sont des longueurs et indiquent de combien il faut agrandir les en-êtes dans la marge correspondante. La version sans étoile, `\widenhead`, permet d'élargir l'en-tête et le pied de page de manière parfaitement asymétrique. La version avec étoile, `\widenhead*` utilise pour les pages verso le contraire de ce qu'on fait sur les pages recto, garantissant la symétrie une fois que le document sera en recto/verso.

### 4.4.2 La gestion des marques

Le package `titlesec` prévoit également quelques options pour ce qui concerne la façon dont les marques doivent être gérées. Le mode de fonctionnement par défaut des styles de page peut être choisi en utilisant une des options du package indiquées au tableau 4.4. Ces choix par défaut peuvent être modifiés dans la définition d'un style de page en utilisant les commandes *ad-hoc*.

La commande `\bottitlemarks` indique que dans ce style de page, on doit utiliser les marques du bas. S'il y a plusieurs titres du niveau intéressant (disons `\section`), `\sectiontitle` sera le titre de la dernière section de la page. S'il n'y en a pas sur la page, ce sera le titre de la dernière section avant cette page.

La commande `\toptitlemarks` indique que dans ce style de page, on doit utiliser les marques du haut. S'il y a plusieurs titres de section sur la page, il prendra le premier, sinon il prendra le titre de la dernière section avant la page.

La commande `\outertitlemarks` prends les marques externes. Il faut comprendre ça sur une double page. Sur la page de gauche, il prendra la marque du haut. Sur la page de droite, il prendra la marque du bas. Il prendra donc les titres les plus à l'extérieur de la double page, si tant est qu'il y ait des titres des bons types sur la double page.

La commande `\innertitlemarks` procède exactement dans l'autre sens. Le package prendra alors les titres les plus vers l'intérieur de la double page. C'est un comportement curieux, mais c'est le comportement par défaut de LaTeX.

Avec la commande `\nexttoptitlemarks`, on passe carrément dans le curieux. Cette commande n'est valable que et uniquement pour les titres de la classe straight[95]. Avec cette commande-là, ce que tu récupères, c'est la marque du haut de la prochaine page. Pour être précis, c'est ce que *contiendra la marque du haut si la prochaine page ne commence pas par un titre du même niveau.* En la comparant avec la marque du bas de la page en cours, ça te permet de voir si la section concernée se continue en haut de la prochaine page, ou si au contraire elle se termine en bas de cette page. Amusant pour ajouter un texte qui dise « (suite...) » ou « (fin) » selon le cas. En effet, tu peux très bien utiliser plusieurs de ces commandes et récupérer, par exemple, un titre de section à chaque fois pour faire des comparaisons.

Enfin, la commande `\firsttitlemarks`, je n'en comprends pas bien le fonctionnement... Pour être précis, elle fait des calculs assez savants et je ne vois pas dans quel cas le résultat serait différent de celui de `\toptitlemarks`.

`\newtitlemark{`⟨*commande*⟩`}`

`\newtitlemark*{`⟨*registre*⟩`}`

La commande `\newtitlemark` permet d'indiquer que tu souhaites pouvoir utiliser la commande que tu lui passes en paramètre dans les marques. Le package sauvegardera la valeur de cette commande à chaque fois que tu commenceras une section, quel que soit son niveau. Ainsi, si tu as tout au long de ton document une commande `\OuEnSuisJe` qui change régulièrement[96], tu peux demander à ce que sa valeur soit conservée dans les marques.

Si cette commande correspond à un registre TeX[97], il faut utiliser la commande `\newtitlemark*`, elle sauve la valeur du registre (ce qui change régulièrement) plutôt que sa définition (qui ne change en général jamais).

### 4.4.3 Style de page et flottants

Enfin, un des raffinements agréables de ce package est qu'il est capable de tenir compte des flottants dans les styles de page. Il fournit pour cela des commandes qui, comme d'habitude, sont à utiliser dans `\newpagestyle`. Ce raffinement n'est disponible que si l'option floatps est utilisée lors du chargement du package.

L'idée est relativement simple : changer l'en-tête et/ou le pied de page selon qu'il y a ou pas des flottants sur la page.

`\setfloathead[`⟨*gauche verso*⟩`][`⟨*centre verso*⟩`][`⟨*droite verso*⟩`]{`⟨*gauche recto*⟩`}{`⟨*centre recto*⟩`}{`⟨*droite recto*⟩`}[`⟨*types*⟩`]`

`\setfloathead*{`⟨*gauche recto/droite verso*⟩`}{`⟨*centre*⟩`}{`⟨*droite recto/gauche verso*⟩`}[`⟨*types*⟩`]`

La commande `\setfloathead` fonctionne sur le même modèle que `\sethead`, simplement elle s'appliquera pour les en-têtes des pages qui contiennent des flottants d'un des types

[95] C'est-à-dire la quasi-totalité des titres, tous sauf ceux qui commencent une page, comme `\chapter`, ou qui produisent une page à eux tout seuls, comme `\part`.

[96] Ce ne sont pas les usages possibles qui manquent. Par exemple, le nombre d'exercices dans un livre de maths, ou le nombre de morts dans un roman policier, les applications sont nombreuses.

[97] Les longueurs LaTeX et TeX ont la même forme et sont effectivement des registres TeX. Par contre, pour les compteurs, il en va autrement. Le compteur LaTeX `page` correspond au compteur TeX `\c@page`. Sur l'art de manipuler les macros avec un @ dans leur nom, reporte-toi au chapitre neuvième page 571 qui traite de la programmation.

énumérés dans `[⟨types⟩]`. `\setfloathead*` est similaire à `\sethead*`. Il existe également les deux commandes correspondantes, `\setfloatfoot` et `\setfloatfoot*` pour les pieds de page. Par défaut, pour les en-têtes, les types concernés sont `t` et `p`, c'est-à-dire que ce style particulier sera adopté pour les pages qui contiennent des flottants en haut de la page ou sur les pages qui ne sont faites que de flottants (le type `b` n'est pas autorisé pour les en-têtes, ce que je trouve triste). Pour les pieds de page, la valeur par défaut de `[⟨types⟩]` est `bp`, indiquant les pages qui contiennent des flottants en bas, ou qui ne contiennent que des flottants. Bien entendu, tu peux totalement différencier, par exemple en faisant :

```
\newpagestyle{monstyle}{%
  % Pages normales:
  \sethead...
  \setfoot...
  % Pages avec flottants en haut:
  \setfloathead...[t]
  % Pages avec flottants en bas:
  \setfloatfoot...[b]
  % Pages entierement en flottants:
  \setfloathead...[p]
  \setfloatfoot...[p]
}
```

Très pratique dans certains cas, par exemple pour enlever les en-têtes des pages de flottants où l'en-tête pourrait prèter à confusion.

`\nextfloathead[`*⟨gauche verso⟩*`][`*⟨centre verso⟩*`][`*⟨droite verso⟩*`]{`*⟨gauche recto⟩*`}{`*⟨centre recto⟩*`}{`*⟨droite recto⟩*`}[`*⟨types⟩*`]`

`\nextfloathead*{`*⟨gauche recto/droite verso⟩*`}{`*⟨centre⟩*`}{`*⟨droite recto/gauche verso⟩*`}[`*⟨types⟩*`]`

De la même manière, tu peux utiliser dans le corps de ton texte, et non pas dans `\newpagestyle`, la commande `\nextfloathead` juste avant un flottant. La même commande existe bien entendu pour les piedes de page : `\nextfloatfoot`. Les variantes avec des étoiles, `\nextfloathead*` et `\textfloatfoot*`, fonctionnent comme on s'y attend.

Si ce flottant se retrouve être tout en haut de la page et que tu as précisé `t` dans `[⟨types⟩]` pour `\nextfloathead` alors c'est ce style-là qui sera utilisé pour l'en-tête ; si le flottant se retrouve être tout en bas de la page et que tu as précisé `b` dans `[⟨types⟩]` pour `\nextfloatfoot` alors le style sera utilisé ; enfin si le flottant se retrouve seul sur une page et que tu as indiqué `p` dans `[⟨types⟩]` pour `\nextfloathead` et/ou `\nextfloatfoot`, alors ce style sera utilisé.

C'est très pratique quand, par exemple, un flottant devient dur à lire (ou source de confusion avec l'en-tête) quand il est tout en haut de la page : tu peux en pareil cas, plutôt que de modifier lourdement ton document, dire à LaTeX que si ce flottant se retrouve en haut de page, il faut changer l'en-tête.

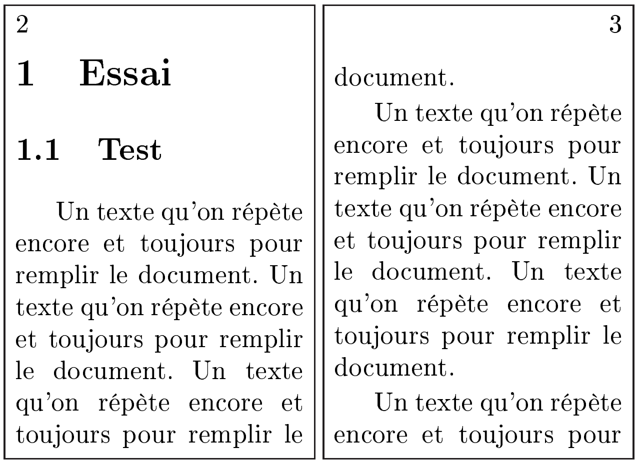
## 4.5 Mise en page simple : le package pageno

Le package `pageno`, que nous devons à Rowland McDonnel [152], d'après une idée (le package `rplain`) d'Harald Axel Sommerfeldt [199], traite également de la mise en page. Mais il traite des problèmes nettement plus simples. En effet, les solutions complètes pour la

| **Option** | **Signification** |
|---|---|
| centerfoot | Folio centré en pied de page. |
| outsidefoot | Folio à l'extérieur en pied de page. |
| insidefoot | Folio à l'intérieur en pied de page. |
| centerhead | Folio centré en tête de page. |
| outsidehead | Folio à l'extérieur en tête de page. |
| insidehead | Folio à l'intérieur en tête de page. |

TAB. 4.5: Options du package `pageno`

```
* \usepackage[outsidehead]
*                     {pageno}
  \section{Essai}
  \subsection{Test}
  \texte\ \texte\ \texte\par
  % ...
```



2

1 Essai

1.1 Test

Un texte qu'on répète encore et toujours pour remplir le document. Un texte qu'on répète encore et toujours pour remplir le document. Un texte qu'on répète encore et toujours pour remplir le

3

document.

Un texte qu'on répète encore et toujours pour remplir le document. Un texte qu'on répète encore et toujours pour remplir le document. Un texte qu'on répète encore et toujours pour remplir le document.

Un texte qu'on répète encore et toujours pour

FIG. 4.9: Exemple d'utilisation du package `pageno`

personnalisation de la mise en page sont très raisonnables pour un ouvrage conséquent (un livre, par exemple), mais un peu lourdes pour un besoin simple.

Par défaut, un document LaTeX utilise le style de page plain, comme on l'a déjà vu. Ce style, tel qu'il est défini dans LaTeX, place simplement le numéro de page (le folio, comme disent les typographes) en pied de page, au centre. Le package pageno permet de modifier le style plain pour placer le folio où on veut.

L'usage en est très simple : on utilise le package avec l'option choisie. Les options possibles font l'objet du tableau 4.5 page précédente. Le comportement par défaut, si on ne passe pas d'option, est le même que celui de LaTeX, à savoir celui correspondant à l'option centerfoot.

La figure **??** page ?? présente un exemple d'utilisation de ce package.

## 4.6 Le package `rplain`

Ce package, ridiculement court, nous est offert par Harald Axel SOMMERFELDT [199]. Il change le style de page plain : le numéro de page, au lieu de se retrouver au milieu en bas de la page, se retrouve à droite en bas de la page. Ce qui permet de localiser une première page de chapitre assez simplement quand on feuillette un document recto-verso.

Il ne prend aucune option, et fonctionne tout à fait silencieusement.

C'est un cas particulier de ce que permet le package pageno, qui lui-même est un cas particulier de ce que permet `fancyhdr`.

## 4.7 En tête souligné : le package `underlin`

Ce microscopique package, que nous devons à Rainer SCHÖPF [192], définit deux nouveaux styles de page : underline[98] et myunderline.

Ils sont similaires aux headings et myheadings de LaTeX, sauf qu'un filet horizontal est ajouté sous l'en-tête. Bien entendu, on peut faire largement aussi bien avec `fancyhdr`, mais `underlin` est nettement plus petit et plus léger.

La figure 4.10 page ci-contre te présente un exemple de ce style de page. Pour ce qui est de myunderline, je t'ai déjà déconseillé de te service de myheadings, je ne reviens pas sur ma parole : évite également myunderline. Si tu as besoin de quelque chose de différent, passe directement à `fancyhdr`.

## 4.8 Positionnement absolu : le package `textpos`

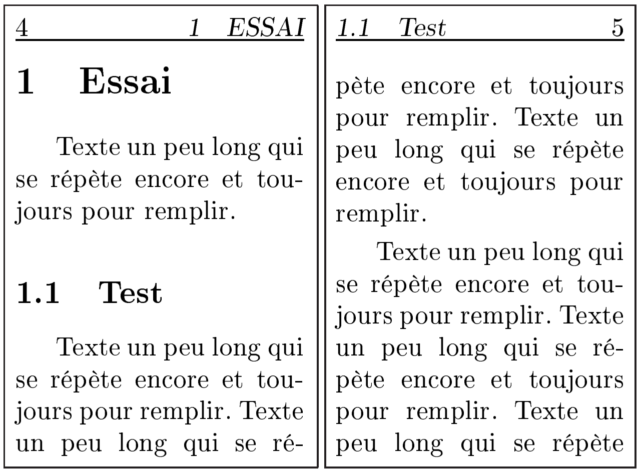
Ce petit package, que nous devons à Norman GRAY [74], permet de positionner du texte de manière absolue sur la page.

Sur le principe, ce package est fait pour mettre en forme des posters : de grandes pages sur lesquelles des zones de texte sont positionnées, par exemple pour présenter un projet. Ça, c'est son usage normal, celui pour lequel Norman GRAY l'a écrit. Mais le package offre bien plus de fonctions que ça, il permet en fait de positionner des blocs de texte de manière statique. Soit tous par rapport au coin en haut à gauche du texte, soit tous par rapport au coin en haut à gauche de la feuille.

Si on l'utilise avec un positionnement absolu par rapport au coin en haut à gauche d'une grande feuille, on peut très facilement faire un poster intéressant. Si on l'utilise avec un positionnement relatif, alors on peut assez facilement mettre en forme un pavé dans le texte, par exemple pour faire une figure.

[98] Le package s'appelle `underlin`, et non pas `underline`, pour que son nom fasse 8 lettres, parce que certains ordinateurs utilisant des systèmes d'exploitation de mauvaise qualité ont du mal avec les noms de fichier de plus de 8 lettres. Déplorable.

```
* \usepackage{underlin}
* \pagestyle{underline}
  \section{Essai}
  \texte\par
  \subsection{Test}
  \texte\texte\texte\par
  \texte\texte\texte\par
```


4 1 ESSAI

# 1 Essai

Texte un peu long qui se répète encore et toujours pour remplir.

## 1.1 Test

Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se ré-

1.1 Test 5

pète encore et toujours pour remplir. Texte un peu long qui se répète encore et toujours pour remplir.

Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se répète encore et toujours pour remplir. Texte un peu long qui se répète

FIG. 4.10: Exemple d'utilisation du package `underlin`

---

Un autre usage simple peut être de placer un élément à une position absolue sur une page, par exemple mettre le numéro de page dans la marge. Pour ça, il suffit d'utiliser le package dans l'en-tête ou le pied de page, de manière à ce que le numéro de page soit convenablement renseigné et que le texte apparaisse bien sur chaque page.

Le package divise la page sur laquelle il va trailler en une grille qui, par défaut, comporte 16 cases de coté, il appelle ces cases des modules. La largeur d'un module est `\TPHorizModule`, sa hauteur est `\TPVertModule`. Tu peux changer ce découpage en appelant la commande `\TPGrid`, qui prend deux paramètres qui indiquent le nombre de modules à prévoir sur la grille. L'appel par défaut est :

```
\TPGrid{16}{16}
```

La commande `\TPGrid` prend un paramètre optionnel qui indique de réserver des marges. Par exemple un appel à :

```
\TPGrid[1cm,2cm]{10}{12}
```

indique à `textpos` de découper sa page en une grille de 10 modules de large et de 12 modules de haut, en réservant une marge d'un centimètre à gauche et à droite, et une marge de deux centimètres en haut et en bas de la page. Ainsi, il pourra calculer la nouvelle taille des modules :

$$\begin{cases} \texttt{\textbackslash TPHorizModule} = \dfrac{\texttt{\textbackslash paperwidth} - 2 \times 1\text{cm}}{10} \\ \texttt{\textbackslash TPVertModule} = \dfrac{\texttt{\textbackslash paperheight} - 2 \times 2\text{cm}}{10} \end{cases}$$

Tout le fonctionnement du package tourne autour de l'environnement `textblock`. Cet environnement prend deux paramètres (et un troisième optionnel) : le premier, entre accolades, indique la largeur du texte, le second, entre parenthèses, indique sa position. Ces trois paramètres sont exprimés en nombre de module, ainsi l'appel à :

```
\begin{textblock}{2}(1,0)
```

indique qu'on fait un bloc de texte qui occupe la largeur de deux modules (donc, une largeur de deux fois `\TPHorizModule`) qui doit être placé à un module à gauche depuis le point de référence et en hauteur au niveau du point de référence. Quand le package fonctionne avec un mode de positionnement absolu, le point de référence est le coin en haut à gauche de la feuille.

Supposons (pour simplifier l'explication) qu'un module fait cinq centimètres de coté, et qu'on est en positionnement absolu. Notre bloc de texte fera donc dix centimètres de large, il sera positionné tout en haut de la page, et à 5 centimètres du bord gauche. Plus précisément : le coin en haut à gauche de notre texte sera placé à cet endroit. Ainsi, un bloc que tu positionnes en `(0,0)` sera effectivement placé dans le coin en haut à gauche.

Par contre, si tu veux mettre du texte au milieu de la page, ce ne sera pas simple, il va falloir faire plein de calculs pour déterminer à quelle distance, en nombre de modules, du bord gauche de la feuille il faut mettre ton bloc de texte, en fonction de sa largeur, pour qu'il soit centré. Il faudra faire le même calcul en hauteur.

Pour simplifier ce genre de calcul, l'environnement `textblock` offre un argument optionnel qui indique quel est le point de référence dans ton bloc de texte qui est utilisé pour le positionnement. Par défaut, cet argument optionnel vaut `[0,0]` indiquant le coin en haut à gauche de ton texte (l'unité utilisée est la taille de ton bloc). En lui passant une valeur de `[1,0]` tu dis que le point de référence est en haut à *droite*. Et donc, en disant `[0.5,0.5]`, tu indiques que le point de référence est le centre de ton bloc.

Ainsi, si tu as une grille de dix modules de coté, un appel à :

```
\begin{textblock}{3}[0.5,0.5](5,5)
```

te permet de placer un bloc de texte au centre de la page.

Si ça ne t'arrange pas de considérer ta page comme une grille de modules, et que tu préfères utiliser des dimensions absolues, tant pour la largeur du bloc que pour sa position sur la page, tu peux utiliser l'environnement `textblock*` qui est fait pour ça. Ainsi, sur une page A4, l'appel précédent devient[99] :

```
\begin{textblock*}{5cm}[0.5,0.5](21cm/2,29.7cm/2)
```

Pour que tu puisses voir comment s'utilise le package, voici un exemple complet. Le résultat est présenté sur les pages suivantes. L'une avec le texte effectivement positionné de manière absolue, la suivante sans le texte, mais avec des boîtes permettant de visualiser le positionnement, qui pour ce second exemple ne sera pas absolu.

Voilà le source utilisé pour le premier exemple, page suivante :

```
\TPGrid[1cm,1cm]{10}{10}
\begin{textblock}{3}(0,0)
Ce texte est en haut \`a gauche de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock}
\begin{textblock}{3}[1,0](10,0)
Ce texte est en haut \`a droite de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock}
\begin{textblock}{3}[0,1](0,10)
Ce texte est en bas \`a gauche de la feuille,
```

[99]Dans cet exemple, je suppose que tu utilises également le package `calc` qui permet de faire des calculs directement, comme par exemple ma division par 2. Ce package est décrit au chapitre neuvième, au paragraphe 7.5 page 593.

```
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock}
\begin{textblock}{3}[1,1](10,10)
Ce texte est en bas \`a droite de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock}
\begin{textblock}{3}[0.5,0.5](5,5)
Ce texte sera centr\'e sur la page, quel que
soit le contenu de la page. Ainsi, si ce texte
\'etait tr\`es long, ou sur une tr\`es grande
largeur, il pourrait d\'epasser de la page sans
m\^eme provoquer un message d'avertissement de
la part de \LaTeX.
\end{textblock}
Cette page est r\'eserv\'ee \`a \TidxPak{textpos} pour
qu'il s'amuse.
\newpage
```

On termine la page ici, et on retrouve `textpos` à la page suivante.

Ce texte est en haut à gauche de la feuille, en respectant une marge d'un centimètre tout autour de la feuille.

Ce texte est en haut à droite de la feuille, en respectant une marge d'un centimètre tout autour de la feuille.



Cette page est réservée à `textpos` pour qu'il s'amuse.

Cette page présente le fonctionnement nominal de `textpos` avec mes cinq blocs de texte aux endroits où je les ai demandés.

J'utilise ici l'option absolute du package `textpos` pour demander un positionnement absolu des blocs.

Ce texte sera centré sur la page, quel que soit le contenu de la page. Ainsi, si ce texte était très long, ou sur une très grande largeur, il pourrait dépasser de la page sans même provoquer un message d'avertissement de la part de LaTeX.

Ce texte est en bas à gauche de la feuille, en respectant une marge d'un centimètre tout autour de la feuille.

Ce texte est en bas à droite de la feuille, en respectant une marge d'un centimètre tout autour de la feuille.

Le deuxième exemple utilise un positionnement relatif. Dans mon exemple, j'ai mis les appels à `textblock*` en début de page, les positionnement est donc relatif au début de la page (en haut à gauche du corps du texte de la page). Dans cet exemple, j'ai choisi d'utiliser `textblock*` pour faire un positionnement avec des mesures liées à la taille du corps du texte, et pas à la taille de la page.

Voilà le source utilisé :

```
\begin{textblock*}{6cm}(1cm,1cm)
Ce texte est en haut \`a gauche de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock*}
\begin{textblock*}{6cm}[1,0](\textwidth-1cm,1cm)
Ce texte est en haut \`a droite de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock*}
\begin{textblock*}{6cm}[0,1](1cm,\textheight-1cm)
Ce texte est en bas \`a gauche de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock*}
\begin{textblock*}{6cm}[1,1](\textwidth-1cm,\textheight-1cm)
Ce texte est en bas \`a droite de la feuille,
en respectant une marge d'un centim\`etre tout
autour de la feuille.
\end{textblock*}
\begin{textblock*}{6cm}[0.5,0.5](\textwidth/2,\textheight/2)
Ce texte sera centr\'e sur la page, quel que
soit le contenu de la page. Ainsi, si ce texte
\'etait tr\`es long, ou sur une tr\`es grande
largeur, il pourrait d\'epasser de la page sans
m\^eme provoquer un message d'avertissement de
la part de \LaTeX.
\end{textblock*}

Cette page est r\'eserv\'ee \`a \TidxPak{textpos} pour
qu'il s'amuse.

\vspace{3cm}

Cette page quasiment blanche pr\'esente le fonctionnement...
```

On termine cette page ici, pour voir le résultat page suivante.

Cette page est réservée à `textpos` pour qu'il s'amuse.

Cette page quasiment blanche présente le fonctionnement de `textpos` avec mes cinq blocs aux endroits où je les ai demandés, mais sans le texte, et avec des rectangles pour indiquer les blocs.

Ça correspond au fait de ne pas utiliser l'option absolute pour avoir un positionnement relatif, et d'utiliser les options showboxes pour que les contour des blocs soient tracés et noshowtext pour que le texte ne soit pas affiché.

| Option | Signification |
|---|---|
| showboxes | Affiche les rectangles autour des blocs de texte que `textpos` dessine. Très pratique quand on prépare un poster pour voir le rendu et juger de l'alignement des différents blocs. |
| noshowtext | Masque le contenu des blocs. Si cette option est utilisée sans la précédente, `textpos` n'affichera plus rien. |
| absolute | Demande un positionnement absolu des blocs de texte. C'est le mode d'utilisation le plus intéressant pour un poster. |
| overlay | Par défaut, en mode positionnement absolu, `textpos` écrit les blocs de texte *sous* le contenu de la page (pratique, par exemple, pour mettre une image de fond sous ta page). Si tu souhaites le contraire, l'option overlay est là pour ça. |
| quiet | Par défaut le package raconte un peu ce qu'il est en train de faire, par exemple ses divers calculs. Cette option le rend silencieux. |
| verbose | Le contraire de quiet. Cette option est activée par défaut. |

TAB. 4.6: Options du package `textpos`

| Paramètre | Signification |
|---|---|
| `\TPHorizModule` | Largeur d'un module tel que définit par appel à `\TPGrid` et servant d'unité de mesure pour les largeurs par `textblock`. |
| `\TPVertModule` | Hauteur d'un module. |
| `\TPboxrulesize` | Épaisseur du trait utilisé avec l'option showboxes, par défaut c'est 0,4 points, comme tous les traits habituels de LaTeX. |
| `\showtextsize` | Cette commande est définie par défaut comme `\normalsize`, mais tu peux la redéfinir : ça permet de changer la taille du texte dans les blocs. Pratique pour écrire gros sur un poster. Ceci dit, tu peux changer de taille de manière classique dans un bloc de texte. |

TAB. 4.7: Paramètres du package `textpos`

---

Les différentes options du package font l'objet du tableau 4.6. Tu as déjà vu, dans les deux exemples précédents, les options les plus intéressantes.

Les différents paramètres pris en compte par le package sont présentés dans le tableau 4.7. Tu as déjà vu les plus intéressants.

La commande `\textblockcolor` permet d'utiliser le package `color` pour fixer la couleur de fond des blocs suivants. Bien entendu, les commandes habituelles du package `color` fonctionnent normalement également.

Enfin, la commande `\textblocklabel` peut être utilisée dans les environnement `textblock` et `textblock*`. En mode normal, son paramètre est ignoré. Quand on l'utilise avec l'option noshowtext, son paramètre est utilisé à la place du texte du bloc. Pratique pour voir un identifiant court à la place du texte réel quand on prépare un poster.

| Option | Signification |
|---|---|
| draft | Cette option classique est prise par défaut, elle active l'affichage de la grille. Elle est reconnue par un grand nombre de package et par les classes de document. |
| final | Cette option classique également désactive le package, elle permet d'indiquer que tu es en train de faire la version finale (destinée à être diffusée, par exemple) de ton document. Elle pourra avantageusement être utilisée comme option de la classe de documents, ainsi elle désactivera tous les packages du même type. |
| columns | Cette option prend une valeur (columns=5 par exemple) pour indiquer au package combien de colonnes il doit dessiner. Sa valeur peut être modifiée plus tard. |

TAB. 4.8: Options du package `typogrid`

## 4.9 Grille d'alignement : le package `typogrid`

Ce petit package, que nous devons à Harald HARDERS [76], permet de dessiner une grille sur les pages du document en cours de préparation pour juger de l'alignement horizontal d'un certain nombre d'éléments.

Les options du package (peu nombreuses) font l'objet du tableau 4.8. L'idée est simple : si tu charges le package avec l'option draft, il tracera un cadre autour de la zone de texte principale et divisera cette zone en un certain nombre de colonnes. Tu peux spécifier ce nombre via l'option columns, qui vaut 12 par défaut.

Tu peux à tout moment changer ce nombre de colonne par un appel à :

```
\typogridsetup{column=8}
```

La page suivante montre un usage possible de ce package[100].

[100] Pour pouvoir activer et désactiver le package à volonté au cours du document, ce qui n'est pas prévu, j'ai dû ruser un peu. En fait, ce package se base sur `eso-pic` pour dessiner la grille. Il utilise `\AtBeginDocument` pour lancer le dessin de sa grille. En fait, dans son appel à `\AtBeginDocument`, il appelle `\AddToShipoutPicture` qui est une commande fournie par `eso-pic`. Cette commande ajoute le bout de dessin (ici, la grille) à la commande `\ESO@HookI`. Il ne reste qu'à intercepter cette commande pour pouvoir activer ou désactiver `typogrid`.

Ainsi, j'ai chargé le package comme ça :

```
\makeatletter
\usepackage{eso-pic}
\let\ESO@HookI@sans@typogrid\ESO@HookI
\usepackage[draft,columns=6]{typogrid}
\let\ESO@HookI@avec@typogrid\ESO@HookI
\makeatother
```

Il ne me reste plus qu'à appeler ça pour désactiver le package :

```
\makeatletter
\let\ESO@HookI\ESO@HookI@sans@typogrid
\makeatother
```

Et le contraire pour le réactiver. Bon, en vrai, j'ai fait tout ça avec le package `contextsw`, parce que ça simplifie le travail, mais le principe reste le même.

## Exemple sur une colonne

Ce texte idiot sert simplement à remplir la page. Je vais utiliser ici quelques constructions qui ont des alignements horizontaux différents, le package `typogrid` m'aidera à déterminer si les alignements sont bons.

D'abord, une équation mathématique simple et célèbre chez les fans d'Astérix :

$$a + b = x$$

La même numérotée :

$$a + b = x \tag{4.1}$$

## Exemple sur deux colonnes

Encore du texte, mais cette fois-ci en deux colonnes. On va pouvoir regarder un peu l'alignement des équations dans ce cas là. La première :

$$a + b = x$$

qui n'est pas numérotée, comme tout à l'heure. Et une deuxième, numérotée :

$$a + b = x \tag{4.2}$$

## Exemple sur trois colonnes

Encore du texte, mais cette fois-ci en trois colonnes. On va pouvoir regarder un peu l'alignement des équations dans ce cas là. La première :

$$a + b = x$$

qui n'est pas numérotée, comme tout à l'heure. Et une deuxième, numérotée :

$$a + b = x \tag{4.3}$$

## Exemple sur six colonnes

Encore du texte, mais cette fois-ci en six colonnes. On va pouvoir regarder un peu l'alignement des équations dans ce cas là. La première :

$$a + b = x$$

qui n'est pas numérotée, comme tout à l'heure. Et une deuxième, numérotée :

$$a + b = x \tag{4.4}$$

```
\def\texte{Texte un peu long qui se
r\'ep\`ete encore et toujours pour
remplir.}
\author{%
  Benjamin \textsc{Bayart}%
                   \thanks{Moi.}%
  \and
  Toi \textsc{Aussi}%
                 \thanks{Pas moi.}}
\date{\today}
\title{Le titre}
\maketitle
\texte\texte\texte\par
```

Le titre

Benjamin Bayart*

Toi Aussi†

5 mars 2005

Texte un peu long qui se répète encore et toujours pour remplir.Texte un peu long qui se répète encore et toujours pour remplir.Texte un peu long qui se répète encore et toujours pour remplir.

*Moi.
†Pas moi.

1

Fig. 4.11: Exemple de page de titre de la classe **article**

## 4.10 Page de titre : le package `titlefoot`

Ce package, que nous devons à Brett Presnell[101], non documenté (enfin si, quelques lignes de commentaire au début du package) modifie la page de titre de la classe article.

Pour rappel, la page de titre normale de la classe **article** est présentée à la figure 4.11.

Les package `titlefoot` propose quelques commandes supplémentaires. La commande `\authorfootnote` te permet d'ajouter du texte en note de pied de page. La commande `\keywords` ajoute les mots clefs en note également. De même `\runningtitle` permet d'ajouter des infos en notes (je ne comprends pas bien ce que Brett Presnell a voulu faire avec cette commande), enfin il prévoit une commande `\amssubj` qui n'intéressera que les auteurs d'articles pour l'$\mathcal{AMS}$.

Les commandes qu'il crée ajoutent un texte devant le texte de la note en pied de page (il se trouve que pour `\authorfootnote` ce texte est vide). Ce texte est, par défaut, en anglais. Pour le personnaliser, il faut re-définir les commandes suivantes :

- `\runningtitlename`, pour la commande `\runningtitle` ;
- `\keywordsname`, pour `\keywords`,
- `\authorfnname`, pour `\authorfootnote`,
- `\amssubjname`, pour `\amssubj`.

L'exemple de la figure 4.12 page ci-contre te montre l'utilisation du package. Dans cet exemple, j'ai francisé, via le package `babel`, le texte de ces commandes. Si tu souhaites utiliser

[101] `presnell@stat.ufl.edu`

```
\usepackage{titlefoot}
\addto\extrasfrench{%
\renewcommand\runningtitlename{%
  \textit{Titre courant}: }%
\renewcommand\keywordsname{%
 Mots et phrases clefs: }%
}
\selectlanguage{frenchb}
\author{%
  Benjamin \textsc{Bayart}%
                   \thanks{Moi.}%
  \and
  Toi \textsc{Aussi}%
               \thanks{Pas moi.}}
\authorfootnote{%
  Un \textsc{Dernier}: essai}
\authorfootnote{%
  Un autre \textsc{Auteur}}%
\authorfootnote{%
  Encore un \textsc{Auteur}}%
\keywords{Exemple, titlefoot.}%
\runningtitle{titre courant}%
\date{\today}
\title{Le titre}
\maketitle
\texte\texte\texte\par
```

Le titre

Benjamin BAYART*

Toi AUSSI†

5 mars 2005

Texte un peu long qui se répète encore et toujours pour remplir.Texte un peu long qui se répète encore et toujours pour remplir.Texte un peu long qui se répète encore et toujours pour rem-

*Moi.
†Pas moi.
Un DERNIER : essai
Un autre AUTEUR
Encore un AUTEUR
*Titre courant* : titre courant
Mots et phrases clefs : Exemple, titlefoot.

1

FIG. 4.12: Exemple de page de titre avec `titlefoot`

---

plusieurs langues dans un article avec `titlefoot`, il te faudra produire les autres traductions pour ces langues, y compris l'anglais. En effet, ce que j'ai utilisé dans l'exemple redéfinit la commande `\keywordsname` lors du passage en français. Mais ne fait rien lors du passage en anglais, du coup, au début du document le texte est en anglais, jusqu'au premier changement de langue vers le français (dans l'exemple, ce changement est juste après la redéfinition) ; ensuite le texte reste en français puisque les autres changements de langue n'ont pas d'effet.

## 4.11 Page de titre : le package `titling`

Ce package, que nous devons à Peter WILSON [224], redéfinit entièrement la commande `\maketitle` des classes de document de LaTeX et la remplace par une version infiniment plus paramètrable.

La première avancée est que tu peux désormais composer entièrement le style du titre, qu'il soit sur une page pour lui seul ou pas, éventuellement en ajoutant des informations (pratique, par exemple, dans une classe de documents). La seconde avancée est que tu as accès dans le document aux informations qui ont été entrées dans le titre (ce qui n'est pas le cas avec LaTeX) et que tu peux ré-utiliser la commande `\maketitle`.

| **Commande** | **Rôle** |
|---|---|
| `\pretitle` | Indique ce qu'on doit placer avant le titre (changement de fonte, de taille, alignement, etc). |
| `\posttitle` | Ce qui se place après le titre. |
| `\preauthor` | Ce qui se place avant les noms des auteurs. |
| `\postauthor` | Ce qui se place après les noms des auteurs. |
| `\predate` | Ce qui se place avant la date. |
| `\postdate` | Ce qui se place après la date. |

TAB. 4.9: Commandes de mise en forme de `titling`

---

### 4.11.1 Mise en forme du titre

Pour faire simple, la nouvelle commande `\maketitle` est définie comme ça[102] :

```
\renewcommand{\maketitle}{%
  \vspace*{droptitle}
  \maketitlehooka
  {\pretitle \title \posttitle}
  \maketitlehookb
  {\preauthor \author \postauthor}
  \maketitlehookc
  {\predate \date \postdate}
  \maketitlehookd
}
```

Le package `titling` ajoute les commandes du tableau 4.9 pour définir les grandes lignes de la mise en forme d'un titre. Lors du chargement du package, ces commandes sont appelées avec des paramètres qui produisent un résultat identique à ce que font les classes de document standard de LaTeX :

```
\pretitle{\begin{center}\LARGE}
\posttitle{\par\end{center}\vskip 0.5em}
\preauthor{\begin{center}
           \large \lineskip 0.5em
   \begin{tabular}[t]{c}}
\postauthor{\end{tabular}\par\end{center}}
\predate{\begin{center}\large}
\postdate{\par\end{center}}
```

Comme tu peux le voir, ces commandes fixent l'alignement et les choix typographiques de l'ensemble des éléments. En règle générale, si tu utilises une des commandes (par exemple `\pretitle`), tu dois utiliser également sa commande complémentaire (`\posttitle`) pour avoir un résultat cohérent. En effet, il n'y a pas de commande simple[103] pour *ajouter* à `\pretitle`.

[102]Enfin, elle ressemble à ça, parce que, en vrai, elle utilise les *valeurs* des paramètres, alors que ce que je te montre utilise les paramètres. En effet, la vraie commande n'utilise pas `\title`. C'est l'auteur (toi) qui utilise cette commande, la commande `\maketitle` utilise la valeur que tu as passée à `\title`.

[103]Il existe bien une solution compliquée, mais elle n'est pas très recommandable, parce qu'elle fait appel à des commandes internes du package, que tu n'es pas censé utiliser (ces commandes pourraient changer de

```
* \usepackage{titling}
* \pretitle{\begin{center} \LARGE
*           \bfseries}
* \posttitle{\par\end{center}}
  \title{Le titre}
  \author{Benjamin \textsc{Bayart}}
  \date{\today}
  \maketitle
  \section{D\'ebut}
  Texte de mon article.\par
  Le titre est: \thetitle.\par
  La date est: \thedate.
```

**Le titre**

Benjamin BAYART

5 mars 2005

**1 Début**

Texte de mon article.
Le titre est : Le titre.
La date est : 5 mars 2005.

1

FIG. 4.13: Exemple d'utilisation de `titling`

---

Par défaut, dans les classes de document standard de LaTeX, après l'appel à `\maketitle`, les informations mises dans le titre (auteur, titre, date, etc.) sont perdues. Le package `titling` les conserve, et tu peux les ré-utiliser en appelant les commandes `\thetitle`, `\theauthor` et `\thedate`.

L'exemple de la figure 4.13 te montre un titre d'article personnalisé (le seul changement par rapport à la norme est qu'il est mis en gras) et l'usage de `\thetitle` et de `\thedate`.

Si tu veux retrouver le comportement de LaTeX, il te suffit de faire appel à `\killtitle` qui effacera ce contenu stocké et annulera tous tes appels à `\pretitle` et compagnie. La commande `\keepthetitle` efface la mise en forme et ne permet pas de rappeler `\maketitle`, mais les informations sur le titre sont stockées et toujours accessibles dans le corps du document.

Le fonctionnement de la commande `\thanks` est un peu particulier : cette commande ajoute son argument dans une liste, et cette liste est ajoutée au pied de la page. Si tu appelles deux fois `\maketitle` alors que ton titre utilise `\thanks` (comme dans l'exemple de la figure 4.14 page suivante), tu vas voir les « thanks » en double en bas de la deuxième page. Pour éviter ça, dans l'exemple, j'ai appelé `\emptythanks` qui vide cette liste.

---

rôle dans la prochaine version du package) :
`\addto{\@bspretitle}{\bfseries}`
par exemple passe le titre en gras. La commande `\addto` est fournie par le package `babel` (il l'utilise abondament pour complèter les informations sur les langues). C'est une commande qui rajoute son second argument à la définition de la macro passé en premier argument. Très utile même quand on n'a pas besoin de langues autres que l'anglais, ça devrait être dans le noyau LaTeX, selon moi.

```
* \usepackage{titling}
* \pretitle{\begin{center}\LARGE
*           \bfseries}
* \posttitle{\par\end{center}}
  \title{Le titre}
  \author{%
       Benjamin \textsc{Bayart}%
       \thanks{Moi}}
  \date{\today}
  \maketitle
  R\'esum\'e expliquant que cet
  article de \theauthor\ est
  int\'eressant.
  \pretitle{\begin{center}\LARGE
            }
  \emptythanks
  \maketitle
  \section{D\'ebut}
  Texte de mon article.\par
  Le titre est: \thetitle.\par
  La date est: \thedate.\par
  L'auteur: \theauthor.\par
```


**Le titre**

Benjamin Bayart*

5 mars 2005

Résumé expliquant que cet article de Benjamin Bayart est intéressant.

*Moi

1

Le titre

Benjamin Bayart*

5 mars 2005

**1 Début**

Texte de mon article.
Le titre est : Le titre.
La date est : 5 mars 2005.
L'auteur : Benjamin Bayart.

*Moi

2

Fig. 4.14: Exemple d'utilisation de `titling`

---

La longueur `\droptitle` permet de changer la position verticale du titre. Elle s'utilise comme les autres longueurs :

```
\setlength{\droptitle}{2cm} ou
\addtolength{\droptitle}{1cm} ou
\addtolength{\droptitle}{-1cm}
```

### 4.11.2 Nouveaux champs

Les quatre commandes `\maketitlehooka` à `\maketitlehookd` te permettent, soit de personaliser la mise en forme (mais tu iras plus vite en utilisant `\pretitle` et `\posttitle`, par exemple), soit d'ajouter facilement un nouveau champ dans ton titre.

Dans la documentation du package [224], Peter Wilson donne cet exemple : quand un article est republié dans un journal, on donne parfois, avant le titre, une référence à la publication originale. Ça peut se faire en créant une command `\published` :

```
\newcommand\published[1]{\gdef\puB{#1}}
\let\puB\empty
\renewcommand\maketitlehooka{%
  \ifx\puB\empty\else
    \par\noindent
    \puB\par
    \vspace*{1em}
  \fi
}
```

Tu pourras l'utiliser, par exemple, comme ça :

```
\published{Pr\'ec\'edemment publi\'e dans
          \textit{Un journal s\'erieux}%
   \thanks{Republi\'e avec autorisation de
   l'\'editeur.}}
```

### 4.11.3 Page de titre

Par défaut, dans la classe **article**, LaTeX met le titre en haut de la première page. Si tu passes l'option titlepage, alors le titre produit par `\maketitle` est mis sur une page seule et la page suivant le titre est numérotée 1 (ce qui pose problème dans une mise en page recto-verso, il manque une page blanche). Tu peux également utiliser l'environnement `titlepage` qui produit un titre (entièrement composé à la main) avant la page 1. Par contre, tu ne peux pas combiner l'environnement avec la commande. Ainsi, cette construction là ne marche pas :

```
\begin{titlepage}
Un texte pr\'ec\'edent le titre.
\vspace{3cm}
% \maketitle va changer de page
\maketitle
un texte apr\`es le titre.
\end{titlepage}
```

En effet, `\maketitle` change *aussi* de page : tu aurais alors un titre sur deux pages, ce qui n'est pas l'effet souhaité. Le package `titling` propose un environnement `titlingpage` qui permet ce type de mélange. La commande `\titlingpagestyle` permet de fixer le style de page de la page de titre[104].

Tu pourras, par exemple, utiliser cette construction là :

```
\begin{titlingpage}
\titlingpagestyle{plain}
\setlength{\droptitle}{2cm}
\maketitle
\begin{abstract}
Le r\'esum\'e de mon article
\end{abstract}
\end{titlingpage}
\refsetcounter{page}{2}
\cleardoublepage
```

L'environnement crée une page (qui sera numérotée 1, a priori), puis quand il se termine, change de page, repasse au style de page normal de ton document et numérote cette nouvelle

[104]Du moins, c'est ce que prétend le mode d'emploi du package, mais, au moins pour la version 2.1b, ce n'est pas vrai. En effet, dans cette version là, la commande n'est pas implémentée. Le package `fixtitling`, diffusé à l'URL :
`http://jmpl.fr.eu.org/JMPL/fixtitling.sty`
implémente la commande manquante, si elle est effectivement manquante.

Si la version que tu as de `titling` corrige ce problème, le package `fixtitling` est inutile, mais ne devrait cependant pas poser de problème.

Suite à un petit mail de ma part, Peter Wilson, l'auteur du package, a sorti une version 2.1c qui corrige le problème : plutôt que d'implémenter la fonctionnalités qui manquait, il l'a supprimée de la documentation. Du coup le package `fixtitling` conserve tout son intérêt.

```
\usepackage{fixtitling}
%Titre en gras comme avant
%Definition de \published
%comme vu plus tot.
%\title, \author et \date
%comme avant
\published{Publi\'e auparavant
  dans:
  \begin{center}
  \textit{L'exemple
          pr\'ec\'edent}
  \end{center}}
\begin{titlingpage}
\titlingpagestyle{plain}
\maketitle
R\'esum\'e expliquant que cet
article de \theauthor\ est
int\'eressant.
\end{titlingpage}
\pretitle{\begin{center}\LARGE
          }
\emptythanks
\let\maketitlehooka\empty
\maketitle
\section{D\'ebut}
Texte de mon article.\par
Le titre est: \thetitle.\par
La date est: \thedate.\par
L'auteur: \theauthor.\par
```

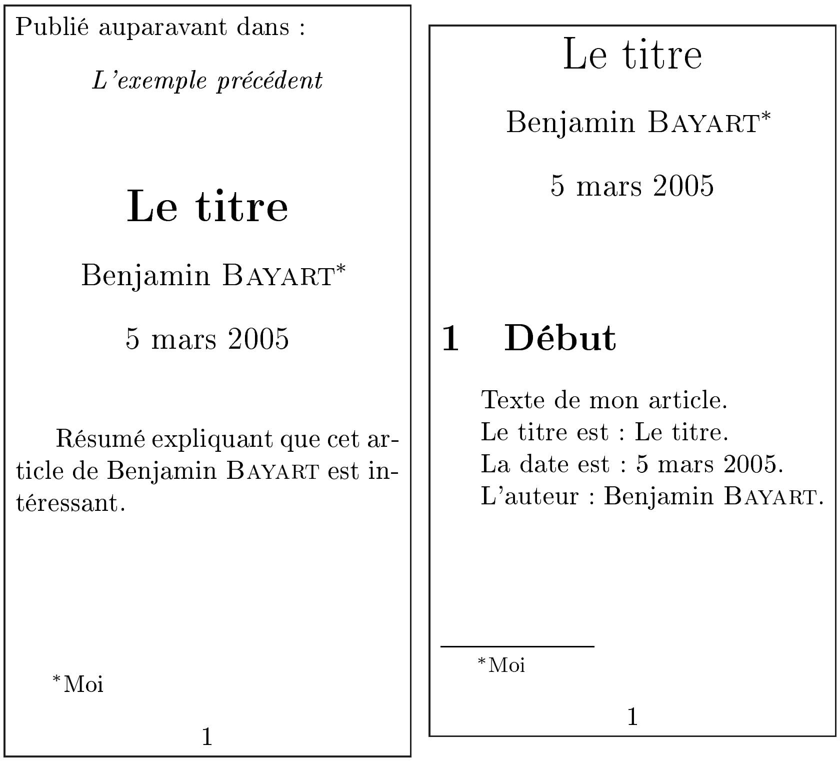



Fig. 4.15: Exemple d'utilisation de `titling`

---

page 1. Si tu prépares un document recto-verso, avoir deux pages 1 est une mauvaise idée, cela va inverser toutes les pages paires et impaires (les belles pages vont se retrouver à gauche et les fausses pages vont se retrouver à droite). Les deux lignes à la fin de l'exemple précédent évitent cet inconvénient.

L'exemple de la figure 4.15 te montre une utilisation possible de l'environnement `titlingpage` : on constate bien qu'on obtient deux pages portant le folio 1, ce qui décalera toutes les pages en cas d'impression recto-verso.

### 4.11.4 Mise en forme des notes

Comme tu l'as déjà compris, la commande `\thanks` permet de renvoyer en notes de pied de page certaines informations associées par exemple aux auteurs.

Le package `titling` permet également de personnaliser la façon dont ces éléments sont mis en page. Pour être précis, `titling` propose plus de commandes pour ce sujet que pour tout le reste réuni !

`\thanksheadextra{⟨gauche⟩}{⟨droite⟩}`

`\thanksfootextra{⟨gauche⟩}{⟨droite⟩}`

Les deux premières commandes utiles sont `\thanksheadextra` et `\thanksfootextra`. La première permet de modifier la façon dont est affiché l'appel de note. La seconde permet de modifier la façon dont le marqueur de note sera mis en forme en bas de page. Par exemple, pour mettre l'appel entre parenthèses et le marqueur en gras en bas de la page (l'utilité semble douteuse) :

```
\thanksheadextra{(}{)}
\thanksfootextra{\bfseries}{}
```

`\thanksmarkseries{⟨série⟩}`

La marque utilisée est en fait produite à partir d'un compteur (le compteur de notes de pied de page, pour être précis). Si tu as déjà lu le chapitre neuvième page 571 qui traite de la programmation, tu sais déjà qu'on peut changer la mise en forme d'un compteur. `titling` te permet de le faire directement en appelant `\thanksmarkseries`. L'argument de cette commande peut être :

- alph pour des lettres minuscules ;
- Alph pour des lettres majuscules ;
- roman pour des chiffres romains minuscules ;
- Roman pour des chiffres romains majuscules ;
- fnsymbol pour des symboles (par défaut en dehors de l'environnement `titlepage`) ;
- arabic pour des chiffres arabes (par défaut dans l'environnement `titlepage`).

`\thanksgap{⟨taille du blanc⟩}`

Les appels de note sont mis en forme sans prendre de place (*i.e.* ils occupent une largeur nulle), ce qui permet d'avoir un alignement impeccable quand ils sont utilisés en fin de ligne, par exemple à la fin du nom d'un auteur, mais posera problème s'ils sont utilisés en milieu de ligne (au niveau du prénom, par exemple) ou s'il y en a plusieurs. Pour cela, la commande `\thanksgap` permet d'ajouter un blanc (la taille de ce blanc est l'argument de la commande). Par exemple[105] :

```
\thanksgap{1ex}
```

Si tu veux réutiliser un des `\thanks`, par exemple quand plusieurs auteurs ont la même adresse (classique dans les articles scientifiques), tu peux utiliser la commande `\thanksmark` qui prend en argument le numéro de l'appel à `\thanks` à réutiliser.

L'exemple de la figure 4.16 page suivante reprend ces différentes commandes.

[105]La valeur `1ex` n'est pas choisie au hasard. Cette unité de mesure (`ex`) correspond à la hauteur de la lettre X-minuscule dans la fonte courante (ou à sa largeur, qui est sensiblement égale). Dans la pratique, les symboles utilisés par défaut (ou les chiffres arabes) ont cette largeur là.

```
* \usepackage{fixtitling}
* %Titre en gras comme avant
* %Definition de \published
* %comme vu plus tot.
* \thanksheadextra{(}{)}
* \thanksfootextra{\itshape}{)}
* \thanksmarkseries{alph}
  %\title, \author, \date
  %et \published comme avant
  \author{%
       Benjamin \textsc{Bayart}%
       \thanks{Moi}%
       \thanksgap{2ex}%
       \thanks{Mon adresse}
     \and
       Mon \textsc{Voisin}%
       \thanks{Pas moi}%
       \thanksgap{2ex}%
       \thanksmark{2}}
  %Contenu de l'article,
  %comme avant.
```

Publié auparavant dans :

*L'exemple précédent*

**Le titre**

Benjamin BAYART(a)(b)

Mon VOISIN(c)(b)

5 mars 2005

Résumé expliquant que cet article de Benjamin BAYART, Mon VOISIN est intéressant.

*a)* Moi
*b)* Mon adresse
*c)* Pas moi

1

Dans cet exemple, on place un blanc de `2ex` entre les appels à `\thanks` pour laisser un peu de place pour les parenthèses.

Le choix de mettre en forme les marques de manière différente entre l'appel et la note donne un résultat plus que discutable. C'est juste pour l'exemple. Ne pas faire pareil.

FIG. 4.16: Exemple d'utilisation de `titling`

## 4.12 À propos des marges

Si tu es habitué à un traitement de texte plus classique, de ces machins qui se manipulent avec une souris, tu dois te dire que le fait de régler les marges d'un document est une opération simple.

Aussi étrange que ça puisse paraitre, cette idée est fausse. Et ce pour plusieurs raisons. D'abord, la notion de marge est floue : la marge extérieure est-elle comptée avec ou sans les notes en marge ? La marge du haut est-elle comptée avec ou sans le titre courant ? Doit-on centrer le texte dans la page ?

En fait, chacune de ces questions mérite un peu d'attention et le plus souvent la réponse dépendra de l'esthétique de ton document. Par exemple, dans ce livre, je n'utilise à peu près jamais les notes en marge, j'ai donc choisi les marges sans prendre en compte ces notes. Dans d'autres ouvrages, ces notes sont fréquentes (à tel point qu'on en agrandit les marges pour pouvoir mettre toutes les notes) et il faut alors les compter comme faisant partie intégrante du texte de la page.

Il en va de même pour le titre courant et le pied de page. Dans ce livre, quasiment toutes les pages ont un titre courant, et presqu'aucune n'a de pied de page. De plus les pieds de page prennent très peu de place, quand ils sont présents (un numéro de page, rien de plus). Je peux donc négliger la place du pied de page (en gros, le laisser rogner dans la marge) alors que je ne peux pas le faire pour le titre courant.

Pour ce qui est du centrage de la page, les règles de l'art sont complexes. D'abord, il faut laisser suffisament de marge à l'intérieur de l'ouvrage pour la reliure (si elle prend de la place, par exemple les agrafes prennent de la place, alors qu'un dos collé n'en prend pas), mais il faut aussi laisser suffisament de place à l'extérieur pour qu'on puisse tenir le livre (compter la taille d'un pouce, en gros).

Sur ce sujet là, on peut lire dans [67] :

> « Lorsque la forme d'une page est dans les mêmes proportions que celle du papier plié selon le format auquel cette page appartient (ce qui doit toujours se faire si aucune considération particulière ne s'y oppose), le blanc du fond et celui de la tête doivent être les mêmes ; celui du pied et la marge extérieure doivent aussi être égaux entre eux. Les premiers sont ordinairement aux seconds dans le rapport approximatif de deux à trois.[106] »

Ce qu'on peut traduire dans un français de moins bonne qualité :
- la marge intérieure (le fond) est de la même taille que la marge du haut (la tête) ;
- la marge extérieure est de la même taille que la marge du bas (le pied) ;
- la marge intérieure (après prise en compte de la place perdue par la reliure) représente les deux-tiers de la marge extérieure.

Après avoir étudié le package `layouts`, qui permet de visualiser les marges d'un document et qui nous sera donc précieux dans les exemples, nous étudierons quelques packages permettant de régler les marges d'un document, soit en faisant tous les calculs à la main, soit en les laissant faire au package.

---

[106] Henri Fournier, *in* Traité de Typographie, tome 1.

## 4.13 Voir la mise en page : le package `layouts`

Ce package, que nous devons à Peter Wilson [221], permet de visualiser certaines des grandeurs utilisées par LaTeX pour faire sa mise en page. En particulier, l'usage le plus classique est de visualiser les marges d'un document.

Ce package est d'un usage double. Si on l'utilise sans afficher les paramètres, mais simplement la mise en forme, il permet de se donner une bonne idée de la mise en page qu'on prépare (c'est limite indispensable quand on prépare une classe de documents, par exemple). Si on lui dit d'afficher les paramètres mis en œuvre, alors c'est prodigieux pour faire un mode d'emploi : ça permet de montrer au lecteur les différentes grandeurs mises en œuvre dans tel ou tel aspect de la mise en forme du document.

### 4.13.1 Principe de fonctionnement

Peter Wilson classe les commandes fournies par `layouts` dans cinq grandes catégories, pour chaque type $x$ d'élément dont il permet de visualiser les paramètres de mise en page :

1. Les commandes de dessin, qui dessinent (dans un environnement `picture`, lire à ce sujet le chapitre quatrième page 473) un des éléments concernés. Ces commandes sont `\draw`$x$, `\`$x$`diagram`, et `\`$x$`design`.
2. Les commandes permettant de prendre en compte les paramètres du document courant. Ces commandes vont prendre (autant que faire se peut) les paramètres du document en cours pour produire les prochains dessins. Ce sont les `\current`$x$.
3. Les commandes affichant les valeurs utilisées, par exemple pour faire la légende du dessin. Certaines des grandeurs mises en œuvre ne peuvent qu'être estimées (par exemple quand elles sont codées en dur dans les macros ou environnements). Ce cas est indiqué en ajoutant des points d'interrogations à la valeur. Ces commandes sont les `\`$x$`values`.
4. Les commandes permettant de changer un paramètre précis pour les prochains dessins, par exemple pour mettre côte-à-côte plusieurs mises en forme possibles pour choisir. Ce sont les commandes `\try`$x$. Elles n'influent que sur le dessin, pas sur le document.
5. Les commandes de contrôle, de la forme `\`$x$`true` et `\`$x$`false`, qui permettent de modifier certains des comportements du package (afficher ou pas un titre, ou un élément du dessin, par exemple).

Le package `layouts` utilise un certain nombre de règlages globaux pour faire ses dessins. En particulier, il prend en compte l'indentation des paragraphes, ainsi que l'interlignage. De même, lorsqu'il dessine une mise en page, elle n'est en général pas à l'échelle, pour une raison simple : si tu dessines ta mise en page dans la page elle-même, à l'échelle, par définition, ça va dépasser dans la marge. Du coup, il dessine à une échelle généralement plus petite.

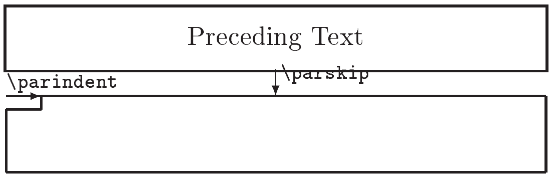
La commande `\setuplayouts`, qu'il appelle automatiquement au tout début du document, se charge de relever certaines grandeurs caractéristiques de ton document (l'interlignage, par exemple) et de positionner l'échelle des dessins à un-demi. Tu peux appeler cette commande à chaque fois que c'est utile (par exemple juste après avoir changé l'indentation des tes paragraphes). De même, tu peux changer l'échelle des dessins avec la commande `\setlayoutscale`. Ainsi, pour utiliser une échelle d'un-tiers, tu peux appeler :

```
\setlayoutscale{0.33}
```

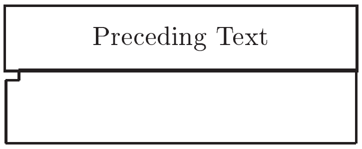
Pour chaque type d'élément $x$ à visualiser, le package définit trois commandes qui peuvent faire le dessin.

La première est `\`$x$`diagram`. Cette commande te présente une mise en forme relativement classique de cet élément (pas la mise en forme de ton document) : le diagramme te permet

Exemple d'un appel à \paragraphdiagram :




Exemple d'un appel à \paragraphdesign :




Lengths are to the nearest pt.

```
\parindent = 15pt        \parskip = 0pt
\baselineskip = 12pt     \linewidth = 400pt
```

Quelques éléments à retenir :
- quand layouts nomme une zone de texte, ici « Preceding Text », il la nomme en anglais et ce n'est pas facile à changer (il faut aller trifouiller à la main dans le package) ;
- le titre de la liste des paramètres est donné en anglais, et est tout aussi compliqué à aller changer ;
- dans ce livre, j'ai mis \parskip à zéro ;
- il y a une très légère erreur d'alignement entre le premier et le deuxième paragraphe ;
- la commande \paragraphdesign ne montre pas le rôle des différents paramètres, mais leur valeur.

FIG. 4.17: Différence entre *x*diagram et *x*design

---

de visualiser clairement le rôle joué par chacun des paramètres, mais ne te permet pas de juger de ta mise en forme.

La seconde est *x*design. Cette commande te présente la mise en forme de ton document. Elle n'indique pas spécialement quels paramètres sont utilisés. En fait, elle te permet de vérifier que la mise en forme est bien celle que tu voulais. De plus, elle affiche sous le dessin les valeurs des paramètres que tu es en train d'utiliser. Si tu appelles auparavant \printparametersfalse, ces paramètres ne sont plus affichés.

La troisième est *x*draw. Si elle est appelée après \drawparameterstrue, elle se comportera comme *x*diagram. Si elle est appelée après \drawparametersfalse, elle se comporte comme *x*design. Cette commande est considérée comme désuète.

La figure 4.17 présente, pour un élément assez simple, la différence entre *x*diagram et *x*design. L'élément choisi, et qui sera utilisé dans tous les exemples génériques, est le paragraphe. Il n'est régi que par quelques paramètres, et tous ne sont pas visualisés sur le diagramme[107].

[107] À noter un petit détail sur la façon d'utiliser le package, qui peut poser problème. Dans l'exemple de la figure 4.17, j'utilise *x*diagram *puis* *x*design. La commande \current*x* se charge de prendre en compte les paramètres courants pour un prochain appel à *x*design. Elle se charge également de faire la mise à l'échelle. La commande *x*diagram, pour sa part, remplace les paramètres mémorisés par des valeurs par défaut génériques qui assurent un bon rendu du diagramme. Ainsi, cette construction ne marche pas :

```
\currentparagraph
\setlayoutscale{0.33}
\begin{minipage}{.6\linewidth}
\paragraphdiagram
\paragraphdesign
\end{minipage}
```

En effet, il y a deux fautes : les paramètres collectés par \current*x* sont écrasés par *x*diagram ; et quand bien même j'inverserais l'ordre entre *x*diagram et *x*design (les paramètres collectés seraient alors pris en compte) le dessin ne serait pas à la bonne échelle, puisque c'est \current*x* qui fait la mise à l'échelle.

La construction suivante, qui peut sembler logique, ne fonctionne pas non plus :

```
\setlayoutscale{0.33}
\begin{minipage}{.6\linewidth}
\paragraphdiagram
\currentparagraph
```

Quand les paramètres sont affichés sur le dessin (par exemple avec `\xdiagram`), le corps de fonte utilisé pour écrire leur nom est celui spécifié par la commande `\setparametertextsize`. Par défaut, le package choisit[108] :

```
\setparametertextsize{\footnotesize}
```

Dans certains dessins, les zones de texte ont un nom, pour permettre de les identifier (malheureusement, ce nom est toujours en anglais et ce n'est pas facile à modifier). Tu peux choisir la fonte à utiliser et éventuellement la taille, en appelant :

```
\setlabelfont{\small\bfseries}
```

Dans les dessins produits par `\xdesign`, les dimensions clefs ne sont pas indiquées sur le dessin. On peut les faire apparaître avec `\drawdimensionstrue`. Par défaut, le package fait un appel à :

```
\drawdimensionsfalse
```

Sous les dessins produits par `\xdesign`, on trouve une table affichant les valeurs des grandeurs intéressantes pour la mise en forme de cet élément. On peut supprimer cette table en appelant `\printparametersfalse`. Par défaut, le package appelle :

```
\printparameterstrue
```

La figure 4.18 page ci-contre présente quelques règlages différents de ceux par défaut.

### 4.13.2 Visualiser une double page

Le package `layouts` propose une commande `\drawaspread`, qui prend 8 paramètres, dont le premier est optionnel, et qui peut servir à dessiner une double page avec une zone de texte à l'intérieur. D'un coté, elle est bien commode : elle utilise une seule longueur fixe (qu'il faut lui préciser explicitement) qu'elle considère être la largeur de la page imprimée et pour tout le reste, elle prend des ratios. Pour appliquer les règles que nous avons vues précédemment (extraites de [67]), ce sera pratique. Par contre, et ça rend cette commande presque inutile, on verra le rendu, mais on n'aura pas les grandeurs qu'elle a calculées (largeur de la zone de texte, dimensions des différentes marges, etc.).

---

```
\paragraphdesign
\end{minipage}
```

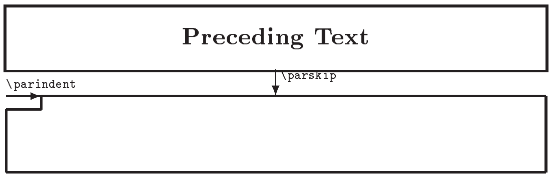
En effet, les dessins sont bien à l'échelle, et les paramètres utilisés par `\xdesign` sont bien ceux collectés, mais ils sont collectés dans l'environnement `minipage`, qui change la forme des paragraphes (il supprime l'indentation). Ainsi, on n'est pas plus avancé. La solution est la suivante :

```
\setlayoutscale{0.33}
\currentparagraph
\begin{minipage}{.6\linewidth}
{\paragraphdiagram}
\paragraphdesign
\end{minipage}
```

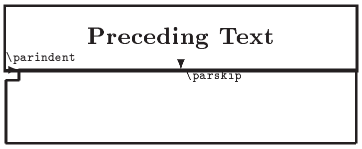
Les accolades autour de `\xdiagram` rendent son action locale. Ainsi, les paramètres sont collectés à l'échelle en dehors de l'environnement `minipage`, puis, sont écrasés par `\xdiagram`, puis sont restaurés après l'accolade fermante (le changement fait par `\xdiagram` est local aux accolades) pour être enfin utilisés par `\xdesign`. Compliqué. Logique, mais compliqué.

[108] Dans la version 2.6b du package, cela ne fonctionne pas. La commande a été nommée par erreur `\setparametertextfont`. C'est en effet une erreur, parce que cette commande ne permet pas de changer la fonte comme on veut (le package utilisera toujours `\texttt`). Cependant, la commande permet d'influer un peu sur la fonte (en fait, choisir tout sauf la famille). En toute logique, la documentation du package a raison, cette commande devrait être `\xtextsize`.

Exemple d'un appel à `\paragraphdiagram` :




Exemple d'un appel à `\paragraphdesign` :




Les règlages suivants ont été appliqués :
- changement de la fonte des noms des blocs (`\small` et `\bfseries`) ;
- changement de la taille pour les noms des longueurs dans les dessins (`\tiny`) ;
- affichage des dimensions clefs dans `\`*x*`design` en appelant : `\drawdimensionstrue`
- suppression de la table des valeurs dans `\`*x*`design` en appelant : `\printparametersfalse`

FIG. 4.18: Paramètrage de `\`*x*`diagram` et `\`*x*`design`

---

`\drawaspread[`⟨*ratio bas/intérieure*⟩`]{`⟨*largeur*⟩`}`
`{`⟨*ratio hauteur p./largeur p.*⟩`}`
`{`⟨*ratio hauteur/largeur*⟩`}`
`{`⟨*ratio intérieure/largeur p.*⟩`}`
`{`⟨*ratio haut/intérieure*⟩`}`
`{`⟨*ratio extérieure/intérieure*⟩`}`
`{`⟨*ratio goutière/intérieure*⟩`}`

Les paramètres sont, dans l'ordre :

1. Rapport de la marge du bas sur la marge intérieure. Paramètre optionnel. S'il est absent, sa valeur est calculée d'après les autres paramètres.
2. Largeur de la page : c'est la seule grandeur absolue.
3. Rapport de la hauteur de la page sur sa largeur. Une valeur de 1 indique donc une page carrée. Pour nos classiques papiers A$n$, le rapport est $\sqrt{2}$, soit approximativement 1,414.
4. Rapport de la hauteur du bloc de texte sur sa largeur.
5. Rapport de la marge intérieure sur la largeur totale de la page.
6. Rapport de la marge du haut sur la marge intérieure.
7. Rapport de la marge extérieure sur la marge intérieure.
8. Rapport de la gouttière entre les deux colonnes (si la page est en deux colonnes) sur la marge intérieure. Si ce rapport est négatif ou nul, une page en une seule colonne est dessinée.

Rappelons les recommandations de [67] :

1. Le bloc de texte dans les mêmes proportions que la page, soit, pour une page A$n$, 1,414.
2. Le rapport de la marge extérieure à la marge intérieure est de 3 pour 2, soit 1,5.
3. Le rapport de la marge du haut à la marge du bas est le même que celui de la marge intérieure à la marge extérieure. Ce qui, pour nous, revient à dire que la marge du haut est 1,414 fois la marge intérieure et que donc la marge du bas devrait être $1{,}5 \times 1{,}414 = 2{,}121$ fois la marge intérieure.

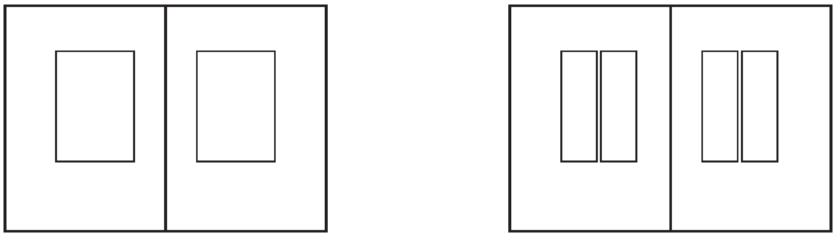


FIG. 4.19: Appels à `\drawaspread` avec les pramètres préconisés par [67]

Si on suppose une page de 21 millimètres (un dizième de la page réelle), ça nous donne donc ces deux lignes là, la première pour une mise en forme en une colonne, et la seconde pour une mise en forme en deux colonne[109] :

```
\drawaspread[2.121]{21mm}{1.414}{1.414}{.2}{1.414}{1.5}{0}
\drawaspread[2.121]{21mm}{1.414}{1.414}{.2}{1.414}{1.5}{.2}
```

La figure 4.19 présente le résultat de ces deux commandes.

Nous aurons l'occasion, par la suite, d'utiliser cette commande pour visualiser d'autres mises en forme proposées par certains packages dont le but est de modifier les marges.

### 4.13.3 Visualiser la mise en page

Pour documenter les packages suivants, qui servent à règler les marges et le positionnement des différents blocs de texte sur la page, je vais intensivement utiliser le package `layouts`. Plus précisément, je vais utiliser les commandes qui permettent de voir la mise en forme d'une page complète.

Les commandes pour visualiser ça sont, fort logiquement, la commande `\pagediagram` pour la version générique et la commande `\pagedesign` pour la version correspondant au document en cours. Le dessin de la mise en page est contrôlé par les différents paramètres que nous avons vus précédemment pour afficher la liste des paramètres ou changer les fontes, mais également par d'autres, qui permettent de dire quels éléments on souhaite voir sur la page.

Le premier point important est de savoir si on veut voir une page de droite (impaire) ou une page de gauche. C'est surtout important pour les documents recto-verso, où les marges doivent être alternées. La commande `\oddpagelayouttrue`, prise par défaut, indique qu'on souhaite visualiser la mise en forme d'une page de droite, c'est-à-dire une page impaire. La commande opposée, indiquant qu'on veut visualiser une page paire (de gauche) est logiquement `\oddpagelayoutfalse`. Si tu veux visualiser une double page, il faut fort logiquement commencer par la page de gauche. Tu ne chercheras donc pas a visualiser côte à côte les pages 1 et 2 (elles sont au dos l'une de l'autre) mais les pages 2 et 3.

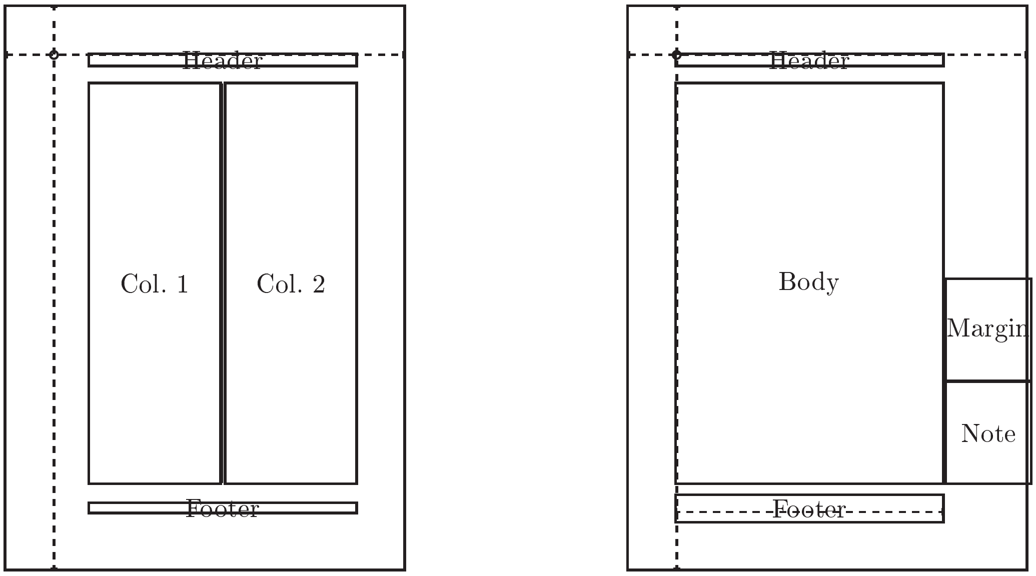
Presque toutes les classes de documents, au moins celles standard pour LaTeX (les sempiternelles **article**, **report** et **book**) peuvent être utilisées en une colonne (mode classique pour un livre) ou en deux colonnes (classique pour un article dans une revue scientifique). La

[109] J'ai choisi, arbitrairement, que la marge intérieure était de 20% de la largeur du papier et que l'espace entre les deux colonnes était de 20% de la marge intérieure, soit 4% de la largeur du papier (8,4 mm pour une page A4).

La marge intérieure représente 20% de la largeur de page. La marge extérieure 1,5 fois la marge intérieure, soit 30% de la page. La zone de texte occupe donc 50% de la largeur. Les proportions en hauteur sont les mêmes, donc 50%. La zone de texte occupe donc un quart de la surface totale de la page.

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.




À noter sur ces deux exemples :

- ils reprennent la mise en page de ce livre ;
- le dessin de droite est obtenu avec les paramètrages par défaut du package `layouts` pour la commande `\pagedesign` ;
- le dessin de gauche est obtenu en demandant une page de gauche, en supprimant les notes en marge, et en forcant le dessin en deux colonnes ;
- les dessins ne présentent pas les paramètres qu'ils dessinent (`\printparametersfalse`, que nous avons déjà vu), on aurait pu ne pas ajouter l'en-tête qui explique le rôle du cercle sur le dessin en appelant : `\printheadingsfalse` ;
- le pied de page de la page de gauche a été déformé en appelant : `\setfootbox{2\baselineskip}{5mm}`.

FIG. 4.20: Dessin de la mise en page

---

commande `\twocolumnlayoutfalse`, prise par défaut, indique que l'on veut voir la mise en page en une colonne. Bien entendu la commande contraire, `\twocolumnlayouttrue`, indique qu'on veut voir la version en deux colonnes.

LaTeX prévoit, indépendament de la classe de document, la gestion de notes en marge (voir à ce sujet le paragraphe 7.2.2 page 405). Si ton document en comporte, tu voudras les visualiser, par exemple pour être sûr qu'elles n'aillent pas déborder du papier[110]. La commande `\drawmarginparstrue`, prise par défaut, indique que l'on souhaite dessiner les notes en marge. Bien entendu, la commande `\drawmarginparsfalse` existe et stipule précisément le contraire.

La figure 4.20 présente l'effet de ces différentes commandes sur la commande `\pagedesign`.

[110] C'est le cas, par exemple, pour ce livre. Je n'y utilise à peu près jamais les notes en marge, donc j'ai volontairement négligé de leur réserver de la place. Du coup, leur position naturelle les amène à dépasser du papier, comme tu peux t'en convraincre en jetant un œil à la figure 4.20. Du coup, les rares fois où je me sers de telles notes, essentiellement dans des exemples, je m'assure qu'elles ne contiennent que peu de texte et que ce texte soit à l'opposé du bord du papier.

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.

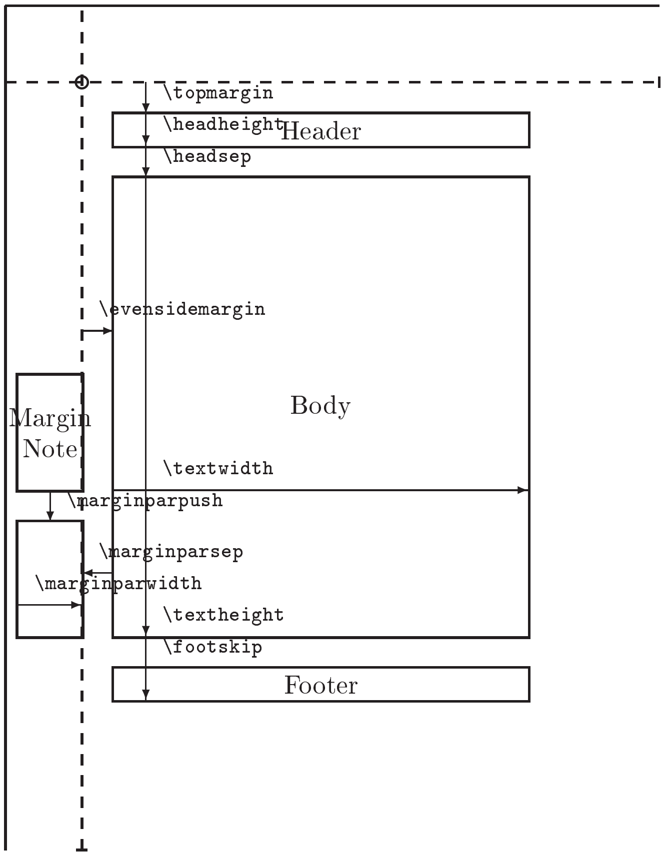



FIG. 4.21: Diagramme générique d'une mise en page

Elle présente la mise en page de ce livre, en une ou en deux colonnes, avec ou sans notes en marge et sans rappeler les paramètres utilisés.

La figure 4.21 présente la mise en page générique proposée par la commande `\pagediagram`. Les différents paramètres, que nous avons déjà vus (figure 4.1 page 23), sont ceux auxquels on s'intéressera pour les packages suivants.

Comme tu peux le constater sur la figure 4.21, il n'y a pas de paramètre qui précise à LaTeX la taille du pied de page. Il y a un paramètre (`\footskip`) qui indique à LaTeX la distance à laisser entre le bas du pied de page et le bas du texte. Le bas désigne ici la ligne porteuse (*i.e.* le bas d'un e, pas le bas d'un g). Ainsi, `layouts` ne sait pas trop à quoi ressemble ton pied de page et ne peut pas savoir quelle hauteur (et quelle profondeur, sous la ligne porteuse) il faut lui prévoir pour le dessiner.

Du coup, il existe la commande `\setfootbox` qui prend deux paramètres, le premier indiquant la hauteur et le deuxième indiquant la profondeur de ton pied de page. Ainsi, si tu as (c'est assez inhabituel) un pied de page de deux lignes de haut et qui descende de 5 millimètres sous la ligne porteuse, tu pourras appeler :

```
\setfootbox{2\baselineskip}{5mm}
```

| Commande | Usage |
|---|---|
| `\trypaperwidth` | Largeur du papier. |
| `\trypaperheight` | Hauteur du papier. |
| `\tryhoffset` | Offset horizontal, généralement zéro. |
| `\tryvoffset` | Offset vertical, généralement zéro. |
| `\tryoddsidemargin` | Marge des pages de droite (impaires). |
| `\tryevensidemargin` | Marge des pages de gauche (paires). |
| `\trymarginparwidth` | Largeur des notes en marge. |
| `\trymarginparsep` | Distance entre le bord du texte et les notes en marge. |
| `\trymarginparpush` | Distance verticale minimale entre deux notes en marge. |
| `\trytopmargin` | Marge du haut. |
| `\tryheadheight` | Hauteur de l'en-tête. |
| `\tryheadsep` | Distance de l'en-tête au corps du texte. |
| `\tryfootskip` | Distance du bas du corps du texte au bas du pied de page. |
| `\trytextheight` | Hauteur de la zone de texte principale. |
| `\trytextwidth` | Largeur de la zone de texte principale. |
| `\trycolumnsep` | Largeur de la gouttière entre les deux colonnes, quand on est en deux colonnes. |
| `\trycolumnseprule` | Épaisseur du filet entre les deux colonnes, au milieu de la gouttière. |

TAB. 4.10: Paramètres possibles pour la composition de la page

Si tu es en train de préparer la mise en page d'un document, par exemple pour définir ta propre classe de document, la solution qui consiste à corriger ta classe de document, puis tout recompiler, puis visualiser avec `layouts` l'effet de tes changements peut s'avérer très longue. Le package te propose de pouvoir afficher des pages plus ou moins déformées.

Ainsi, si tu veux tester quelques valeurs différentes pour la largeur des notes en marge, tu peux procéder comme ça :

```
\currentpage
\pagedesign % La version telle-quelle
\trymarginparwidth{1cm}
\pagedesign % Essai avec 1 cm
\trymarginparwidth{1.5cm}
\pagedesign % Essai avec 1 cm et demi
```

Ça te permet de produire les différentes variantes d'un seul coup et de bien visualiser la différence : ainsi tu pourras comparer et choisir. Pour toutes les grandeurs présentées sur chacun des diagrammes, il en va de même, que ce soit pour les pages ou pour les autres éléments dont `layouts` sait présenter la mise en forme.

Le tableau 4.10 te présente l'ensemble des commandes `\try`*x* en rappelant brièvement leur usage.

### 4.13.4 La classe memoir

La classe de documents **memoir**, dont nous reparlerons au paragraphe **??** page ??, utilise des paramètres très différents de ceux des classes de document standard de LaTeX pour sa mise en page.

Dashed lines represent the actual page size after trimming the stock.

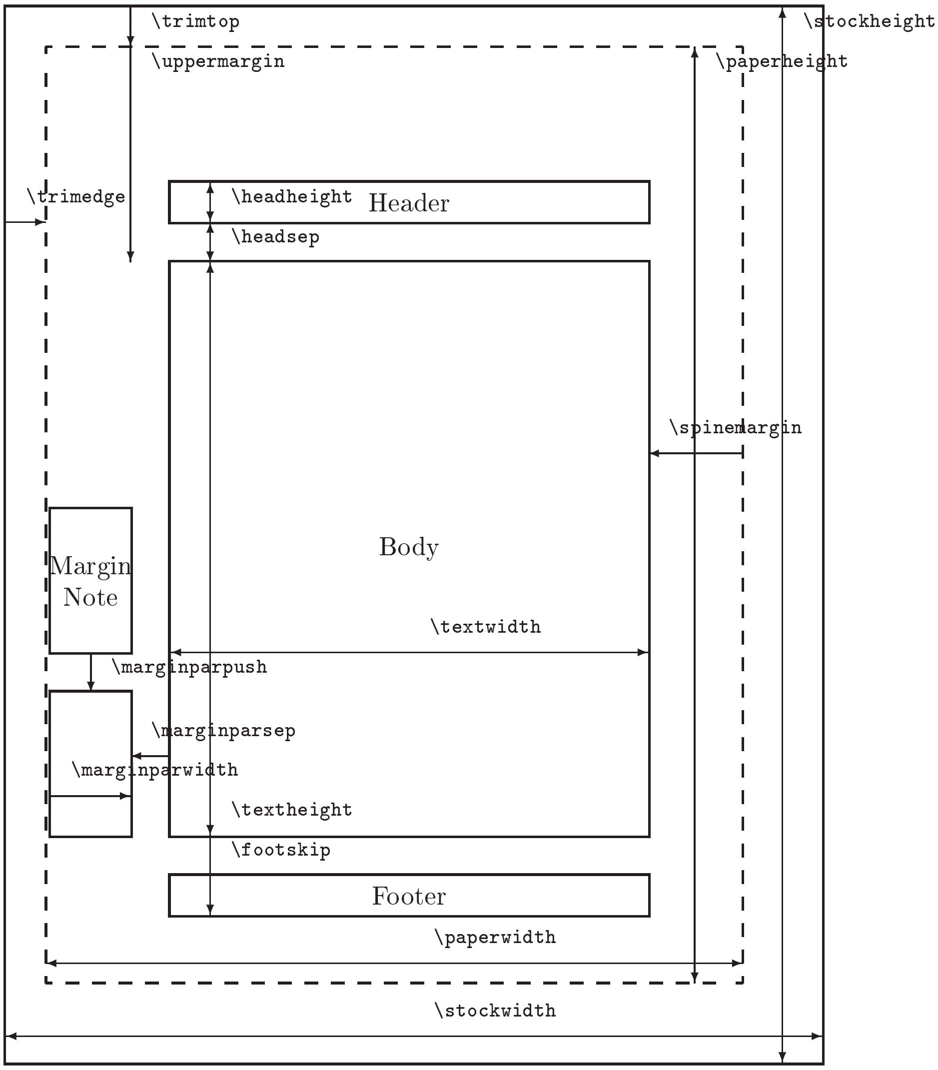
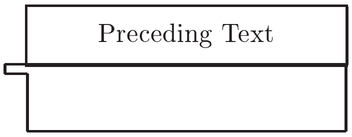



Fig. 4.22: Diagramme générique pour la classe **memoir**

| Commande | Usage |
|---|---|
| `\tryparindent` | Indentation du paragraphe. |
| `\tryparskip` | Blanc entre deux paragraphes. |
| `\trybaselineskip` | Distance entre deux lignes. |
| `\trylinewidth` | Largeur du texte. |

TAB. 4.11: Paramètres possibles pour la forme des paragraphes




Lengths are to the nearest pt.

```
\parindent = -28pt        \parskip = 0pt
\baselineskip = 12pt      \linewidth = 400pt
```

FIG. 4.23: Un paragraphe avec une indentation négative

---

Les commandes `\stockdesign` et `\stockdiagram` permettent de les visualiser. La figure 4.22 page précédente te présente sommairement les paramètres utiles.

Un des aspect intéressant de ce mode de paramètrage est qu'il prévoit deux tailles de papier : la taille du papier sur lequel on imprime et la taille du papier qu'on lit. Le premier étant (ça semble évident) plus grand que le second. C'est très pratique, quand on prépare un ouvrage destiné à partir vers un imprimeur : les imprimeurs utilisent toujours du papier un peu plus grand, qu'ils coupent par la suite aux dimensions voulues.

La classe **memoir** prévoit ça assez bien.

### 4.13.5 Voir la forme des paragraphes

Tu as déjà vu ces commandes, puisque ce sont celles que j'ai utilisées dans les tous premiers exemples.

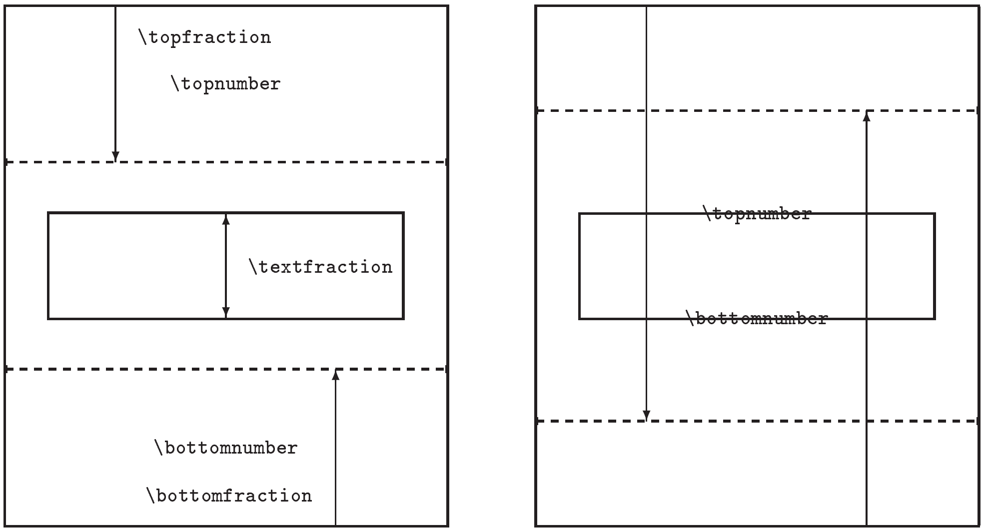
Les quatre grandeurs clefs de la mise en forme des paragraphes peuvent être modifiées, pour faire des essais, avec les commandes du tableau 4.11.

La figure 4.23 te présente un exemple obtenu avec[111] :

```
\currentparagraph
\begin{figure*}[t!]
\setlayoutscale{.3}
\tryparindent{-1cm}
\paragraphdesign
\caption{Un paragraphe avec une indentation n\'egative}
\label{layouts-paragraph}
\end{figure*}
```

| Commande | Usage |
|---|---|
| `\trytopfraction` | Fraction maximale de la page que peuvent occuper les flottants du haut. |
| `\trybottomfraction` | Idem pour les flottants du bas. |
| `\trytextfraction` | Le texte doit occuper au moins cette fraction là de la page, sinon LaTeX retirera un flottant. |
| `\trytopnumber` | Nombre maximum de flottants en haut de page. |
| `\trybottomnumber` | Nombre maximum de flottants en bas de page. |
| `\trytotalnumber` | Nombre maximum de flottants sur une page. |

TAB. 4.12: Paramètres possibles pour la forme générale des flottants




```
\topnumber = 4        \topfraction = 0.800
\bottomnumber = 0     \bottomfraction = 0.800
\totalnumber = 4      \textfraction = 0.199
```

Tu trouves ici, sur la gauche, le diagramme général indiquant les paramètres pris en compte pour la construction d'une page contenant des flottants. Ce diagramme est obtenu avec la commande `\floatpagediagram`.

Sur la droite, un appel à la commande :
`\floatpagedesign`
permet de visualiser les paramètres que j'utilise dans ce livre.

À noter, l'erreur d'arrondi sur le paramètre :
`\textfraction`
dont la vraie valeur est `.2`. Curieusement, le package s'amuse à faire tout un tas de calculs pour dessiner la valeur (ça, c'est logique) et il affiche le résultat de son calcul dans le tableau, plutôt que la vraie valeur (ça, c'est moins logique).

FIG. 4.24: Mise en forme générale des flottants

### 4.13.6 Voir la forme des flottants

Les flottants, nous en reparlerons plus tard, au titre 7 page 403. En gros, ce sont les éléments hors-texte, comme les figures ou les tableaux.

[111] Il est important de noter que l'appel à `\currentparagraph` a eu lieu en dehors de l'environnement `figure*`. Ainsi ce sont bien les réglages du texte qui sont utilisés et modifiés. Ceux *dans* l'environnement `figure*` sont différents.

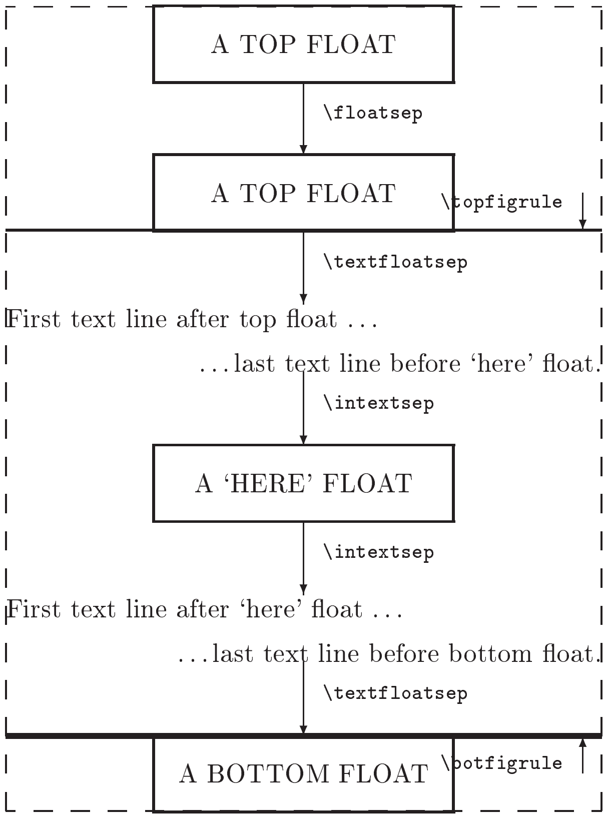



FIG. 4.25: Mise en forme des flottants

---

Le package `layouts` te permet d'en visualiser le placement. Soit dans les grandes lignes, en te montrant la plus grande zone utilisable par les flottants en haut de page comme en bas de page et la zone de texte minimale, soit de manière plus détaillée.

Les commandes pour ça sont `\floatpagediagram`, pour voir une version générique, qui indique l'utilisation des différentes grandeurs et `\floatpagedesign` pour voir les paramètres du document en cours. Bien entendu, la commande `\currentfloatpage` se charge de collecter les paramètres utiles[112].

La figure 4.24 page précédente te présente un exemple d'utilisation de ces commandes. Le tableau 4.12 page ci-contre reprend l'ensemble des commandes `\try`$x$ sur ce thème.

La mise en forme plus détaillée des flottants peut être obtenue avec les commandes `\floatdiagram` et, comme d'habitude, `\floatdesign` pour visualiser le placement des flottants entre eux, ainsi que les paramètres concernés.

---

[112]Dans sa version 2.6b, le package comporte un bug sur ce point. Pour une raison que je ne m'explique pas, la commande `\currentfloatpage` ne prend pas les paramètres du document pour `\topfraction`, `\bottomfraction` et `\textfraction`. Du coup, pour produire mon exemple, j'ai du faire comme ça :

```
\currentfloatpage
\trytopfraction{\topfraction}
\trybottomfraction{\bottomfraction}
\trytextfraction{\textfraction}
\floatpagedesign
```

Une prochaine version du package corrigera peut-être ce point.

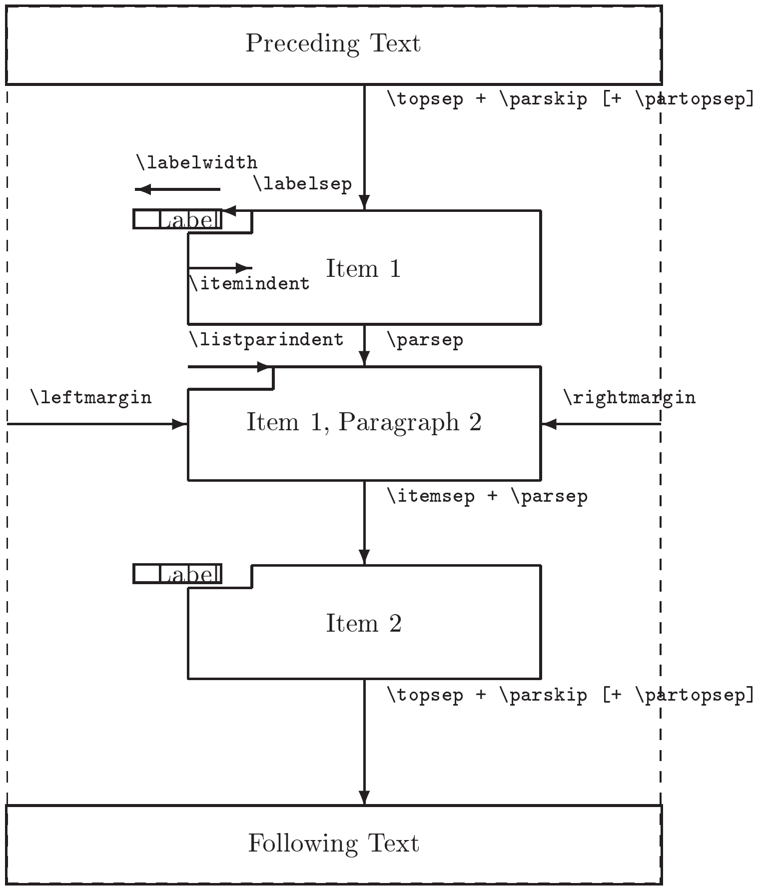



FIG. 4.26: Forme générale d'une liste

---

La figure 4.25 page précédente présente les différents paramètres pris en compte[113].

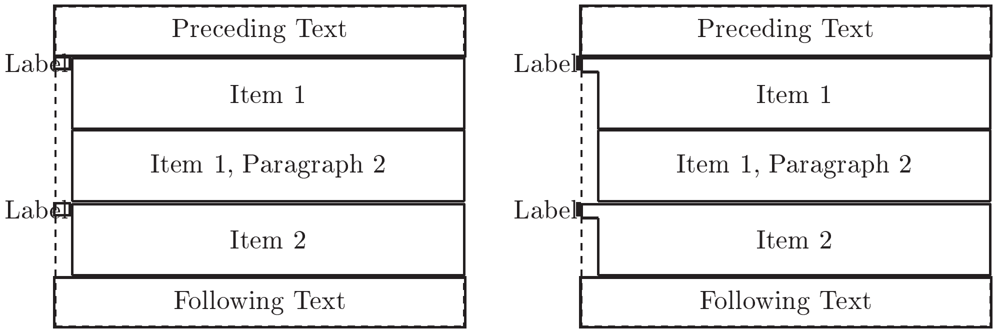
### 4.13.7 Voir la forme des listes

S'il y a bien un élément dans LaTeX qui soit complexe à paramètrer, ce sont bien les listes. La mise en forme d'une liste dépend d'un très grand nombre de paramètres, qui ne sont pas simples à comprendre.

Le package `layouts` permet de visualiser tous ces paramètres, et de faire des essais de plusieurs valeurs. C'est très pratique quand on veut préparer une classe de document avec des listes un peu différentes de la normale. Sur le principe, c'est très simple : les commandes utiles

[113]Toujours dans la version 2.6b du package `layouts`, pour une raison que je ne m'explique pas bien, les commandes `\floatdiagram` et `\floatdesign` supportent très mal d'être utilisées dans l'environnement `minipage`.

De même, les paramètres `\topfigrule` et `\botfigrule` sont normalement des commandes, chacune produisant le trait correspondant (pour séparer les flottants du texte). Normalement, dans les classes standard de LaTeX, ces commandes ne font rien (elles sont vides). J'ai constaté que quand on les modifie, `layouts` ne sait pas les prendre en compte. Au mieux, il sait dessiner un trait sur toute la largeur de la page, dont tu peux spécifier l'épaisseur avec `\trytopfigrule` et `\trybotfigrule`. Il me semble que ça doit être considéré comme un bug.




Lengths are to the nearest pt.

```
\leftmargin = 25pt      \rightmargin = 0pt
\itemindent = 0pt       \labelwidth = 20pt
\labelsep = 5pt         \listparindent = 0pt
\topsep = 3pt           \parskip = 0pt
\partopsep = 2pt        \parsep = 2pt
\itemsep = 2pt
```

Lengths are to the nearest pt.

```
\leftmargin = 25pt      \rightmargin = 0pt
\itemindent = -25pt     \labelwidth = 0pt
\labelsep = 5pt         \listparindent = 0pt
\topsep = 3pt           \parskip = 0pt
\partopsep = 2pt        \parsep = 2pt
\itemsep = 2pt
```

Le dessin de gauche présente la mise en forme des listes `enumerate` de ce livre. Ce dessin te permet de constater que si tu utilises un label trop grand (par exemple en utilisant le package `enumerate`), il dépassera dans la marge.

Le dessin de droite te présente une liste `description`. Sur ce dessin, le package `layouts` se trompe : il dessine le label comme dépassant à gauche du début de l'`\item`, ce qui, pour cette liste, n'est pas vrai. Il a une excuse : ça ne peut pas se deviner, en examinant les paramètres auxquels il s'intéresse.

FIG. 4.27: Mise en forme des listes `enumerate` et `description`

sont `\currentlist`, pour prendre en compte les paramètres, `\listdiagram` pour dessiner le rôle de chacun des paramètres et `\listdesign` pour représenter les paramètres collectés, soit par `\currentlist`, soit via des appels à `\try`$x$.

La figure 4.26 page ci-contre te présente le résultat de `\listdiagram`, c'est-à-dire les paramètres que LaTeX prend en compte dans la mise en forme des listes. Le tableau 4.13 page suivante présente les commandes `\try`$x$ disponibles pour les listes.

En fait, le seul point un peu subtil à comprendre est l'usage de la commande `\currentlist`. En effet, l'endroit où tu vas appeler cette commande est très important : les paramètres concernant une liste donnée sont définis lorsqu'on rentre dans l'environnement de liste concerné. Du coup, si on appelle cette commande dans le texte, en dehors de toute liste, elle va collecter un peu n'importe quoi comme valeurs. Un usage type pourrait être :

```
\begin{itemize}
\item \currentlist % Ici, je suis dans un itemize, ce sont
                   % donc les paramètres d'itemize qui
                   % seront pris en ligne de compte.
\listdesign
\end{itemize}
```

Bon, sur le principe, c'est facile à comprendre. Mais ça devient plus complexe si tu cherches à produire quelque chose comme l'exemple de la figure 4.27. En effet, cette figure te présente la mise en forme de deux listes classiques et, pourtant, il n'y a pas de liste dedans.

Pour récupérer les paramètres, il *faut* placer la commande `\currentlist` dans un environnement de liste. Or, dans mon exemple, je refuse de *voir* l'environnement de liste en

| Commande | Usage |
|---|---|
| `\tryitemindent` | Renfoncement du texte de la première ligne, sans prendre en compte le label. |
| `\trylabelwidth` | Largeur supposée du label, si la liste utilise la commande `\item` classique (ce n'est pas le cas de `description`). |
| `\trylabelsep` | Taille du blanc entre le label et le texte. |
| `\tryleftmargin` | Renfoncement de la liste par rapport à la marge gauche du texte alentour. |
| `\tryrightmargin` | Idem par rapport à la marge de droite. |
| `\trylistparindent` | Indentation des paragraphes dans la liste, sauf le paragraphe qui commence par `\item`. |
| `\trytopsep` | Blanc en haut de liste dans tous les cas. |
| `\tryparskip` | Blanc ajouté en haut de la liste (utilisé ailleurs également). |
| `\trypartopsep` | Blanc ajouté en haut de liste si elle n'est pas la suite directe du paragraphe précédent. |
| `\tryparsep` | Blanc entre deux paragraphes de la liste (ajouté également entre deux items) |
| `\tryitemsep` | Blanc ajouté entre deux items. |

TAB. 4.13: Paramètres possibles pour la forme des listes

question. Le moyen le plus simple est de se livrer à un petit bricolage qui utilise certaines notions du chapitre neuvième page 571. Le principe est simple, mais comme il manipule des notions avec lesquelles tu n'es peut-être pas encore familier, je vais détailler un peu.

Le premier morceau intéressant à comprendre est qu'on peut placer à peu près n'importe quoi et surtout ce qui nous intéresse, à savoir une liste, dans une boîte. Et que, si on fait ça, la boîte en question n'apparaît pas dans le texte, elle est mémorisée pour un usage futur. On peut donc faire ça :

```
\newsavebox{\ajeter}
\sbox{\ajeter}{
\begin{minipage}{\linewidth}
\begin{enumerate}
\item Ma liste inutile.
\end{enumerate}
\end{minipage}
}
```

Ce faisant, je n'ai *rien* affiché. Je peux donc, dans cette liste qui ne s'affiche pas, récupérer les paramètres. Mais mon problème n'est pas pour autant résolu. En effet, la commande `\currentlist` fait une affectation locale des paramètres : quand on sera sorti de la liste, ils retrouveront leur valeur par défaut et tout ce travail n'aura servi à rien. Qu'à cela ne tienne : dans la liste, on va sauver, dans une autre boîte, le dessin qu'on veut produire (celui de `\listdesign`). Là encore, l'affectation entre la boîte et son contenu (le dessin) est, *a priori* locale et sera donc défaite en sortant de la liste. Il faut donc préciser à LaTeX que *cette* affectation est globale (*i.e.* doit survivre après la fin du groupe en cours). Ça donne ça :

```
\newsavebox{\ajeter}
\newsavebox{\sauvelayout}
\sbox{\ajeter}{%
```

| Commande | Usage |
|---|---|
| `\trybeforeskip` | En valeur absolue, la taille du blanc vertical avant le titre. |
| `\tryafterskip` | En valeur absolue, la taille du blanc après le titre. Si positif, le blanc est vertical et le titre est seul sur sa ligne. Si négatif, le blanc est horizontal et le titre est sur la même ligne que le texte qui suit. |
| `\tryindent` | Taille du blanc horizontal avant le titre. |

TAB. 4.14: Paramètres possibles pour la forme des titres

---

```
\begin{minipage}{\linewidth}
\begin{enumerate}
\item Ma liste inutile.
  \currentlist
  \global\sbox{\sauvelayout}{%
    \begin{minipage}{\linewidth}
    \listdesign
    \end{minipage}%
  }
\end{enumerate}
\end{minipage}
}
```

Il ne restera[114] plus alors qu'à utiliser le contenu de la boite `\sauvelayout` à l'endroit où on veut voir apparaître le dessin en appelant :

```
\begin{center}
\usebox{\sauvelayout}
\end{center}
```

### 4.13.8 Voir les têtes de section

Ce n'est pas forcément l'aspect le plus immédiatement utile du package `layouts`, mais on peut aussi s'en servir pour travailler sur les différents paramètres pris en compte dans la mise en forme des titres de section.

Comme tu le verras au titre 6 page 319, la majorité des titres de section de LaTeX sont mis en forme avec la commande `\@startsection`. Cette commande prend six paramètres, dont quatre qui servent à la mise en forme.

Dans ce cas précis, le package ne peut pas aller chercher les valeurs réellement utilisées par le document, il faut donc, si tu veux visualiser tes titres, que tu utilises les commandes du tableau 4.14.

Les deux commandes habituelles, `\headingdiagram` pour faire un dessin générique, et `\headingdesign` pour présenter simplement un essai de paramètres, prennent cette fois-ci un paramètre : la fonte à utiliser.

Pour un appel à `\headingdesign`, le package prendra en compte les valeurs que tu as indiquées pour les différents paramètres pour déterminer si le titre est seul sur sa ligne ou

[114] Un subtil et dernier détail à noter cependant : la première boîte, celle qui contient la liste, est bel et bien de la largeur du texte, mais pas la deuxième. En effet, la liste a légèrement réduit la longueur de la ligne. Précisément, elle l'a réduit de `\leftmargin` sur le coté gauche et de `\rightmargin` sur le coté droit. Du coup, pour récupérer vraiment la largeur du texte, j'ai dû indiquer, pour la deuxième boite :
`\begin{minipage}{\linewidth+\leftmargin+\rightmargin}`

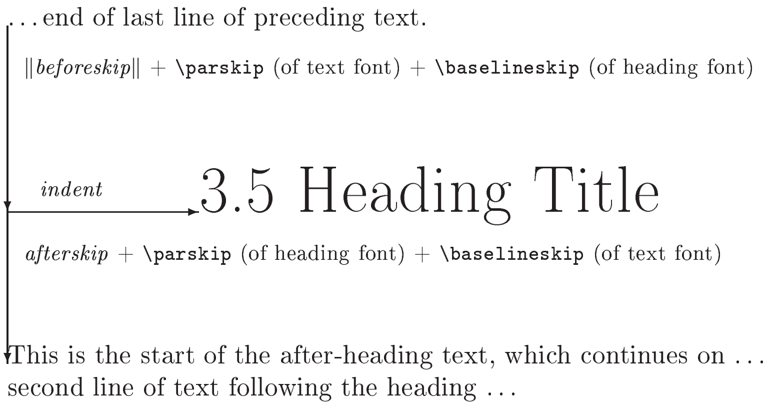



FIG. 4.28: Forme générale d'un titre de section seul sur sa ligne

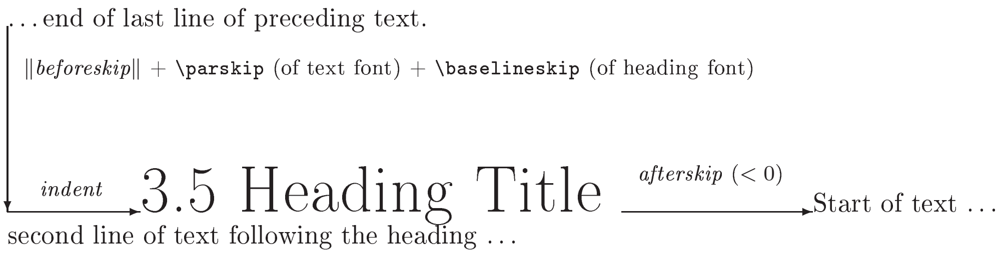



FIG. 4.29: Forme générale d'un titre de section dans le corps du texte

non. Pour la commande `\headingdiagram`, il ne peut pas faire comme ça. Tu peux donc lui spécifier `\runinheadfalse`, qui est la valeur par défaut, pour lui dire que le titre est seul sur sa ligne. Pour lui dire le contraire, il faut appeler `\runinheadtrue`.

Ainsi, pour produire les deux exemples des figure 4.28 et 4.29 de la présente page j'ai utilisé le source suivant :

```
\setlayoutscale{1}
\headingdiagram{\large\bfseries}
... et ...
\setlayoutscale{1}
\runinheadtrue
\headingdiagram{\large\bfseries}
```

### 4.13.9 Voir la forme de la table des matières

Comme la mise en forme des titres de section, la mise en forme de la table des matières peut être ajustée avec un certain nombre de paramètres. Pour les titres de section, ces paramètres sont cachés dans les commandes de sectionnement (`\section` et ses copains),

| Commande | Usage |
|---|---|
| `\trytocindent` | Indentation du numéro de section par rapport à la marge gauche. Paramètre de `\@dottedtocline`. |
| `\trytocnumwidth` | Largeur réservée pour le numéro de section. Paramètre de `\@dottedtocline`. |
| `\trytoclinewidth` | Largeur de la ligne (en général la même que pour le texte). |
| `\trytocrmarg` | Largeur de la fin du titre à la marge gauche. |
| `\trytocpnumwidth` | Largeur réservée pour le numéro de page. |
| `\trytocdotsep` | Séparation entre deux points successifs dans le pointillé entre le titre et le numéro de page. C'est une valeur sans unité (LaTeX utilisera tout seul l'unité `mu`, un dix-huitième de la taille de corps en cours). |

TAB. 4.15: Paramètres possibles pour la forme de la table des matières

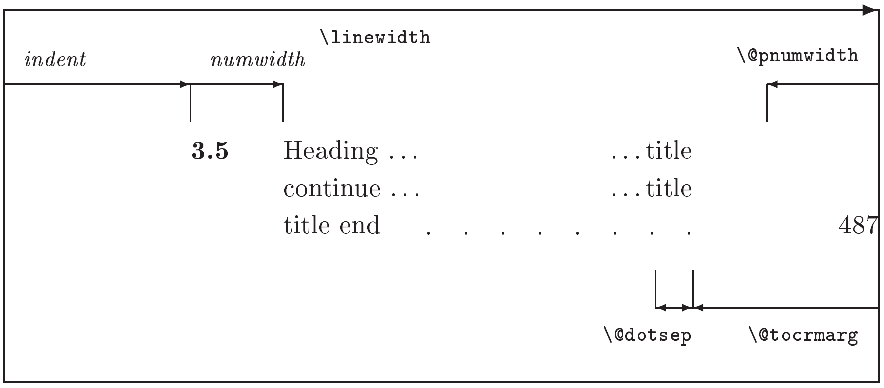



FIG. 4.30: Forme générale d'une entrée de la table des matières

---

quand elles font appel à `\@startsection`. Pour la table des matières, il en va de même. Une entrée de la table des matières correspondant à `\section` est mise en forme par la commande `\l@section`, qui elle-même utilise `\@dottedtocline`. C'est cette commande qui prend 3 paramètres utiles à la mise en page. L'un est le niveau de profondeur de l'entrée en question, qui est utilisé pour décider s'il faut ou pas afficher cette entrée. Les deux autres correspondent aux commandes `\trytocindent` et `\trytocnumwidth`.

Les deux commandes habituelles, `\tocdiagram` et `\tocdesign`, permettent de faire soit le dessin générique, soit le dessin prenant en compte les paramètres. La figure 4.30 montre le résultat d'un appel à `\tocdiagram`. À noter que sur ce type de dessin l'espacement des points de suspension peut être légèrement faux : le package `layouts` fait des calculs pour la mise à l'échelle ; or il se trouve que ces calculs, dans ce cas précis, peuvent mener à des erreurs d'arrondis.

Comme pour les autres dessins, la commande `\currenttoc` permet de récupérer les paramètres du document en cours. Enfin, les paramètres qui ne sont pas des arguments de `\@dottedtocline`. Pour ces deux-là, il prendra des valeurs qu'il juge convenables. Tu peux les forcer, de même que tous les autres paramètres, avec les commandes du tableau 4.15.

| **Commande** | **Usage** |
|---|---|
| `\tryfootins` | Blanc entre le texte et les notes. |
| `\tryfootnotesep` | Blanc entre deux notes successives. |
| `\tryfootnotebaseline` | Interligne dans les notes. |
| `\tryfootruleheight` | Hauteur du filet de séparation des notes. |
| `\tryfootrulefrac` | Fraction de la largeur du texte couverte par le filet (un quart, soit `.25` par défaut). |

TAB. 4.16: Paramètres possibles pour la forme des notes de pied de page

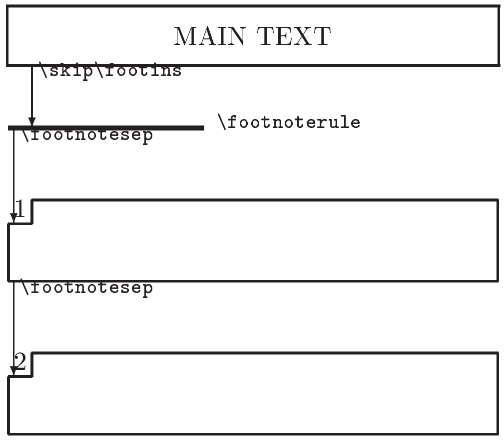



FIG. 4.31: Paramètres pour la mise en forme des notes de pied de page

---

### 4.13.10 Voir la forme des notes de pied de page

Là encore, tu ne t'es sans doute jamais posé la question, mais, oui, la mise en page des notes de pied de page peut se personnaliser. LaTeX ne propose que très peu de paramètres par défaut, mais le package `layouts` te propose tout de même de les visualiser avec les commandes habituelles : `\footnotediagram` pour voire la version générique qui expose le rôle des paramètres et `\footnotedesign` pour tester des paramètres.

Le tableau 4.16 récapitule les commandes permettant de tester la valeur des paramètres et la figure 4.31 présente graphiquement leur rôle en appelant la commande `\footnotediagram`.

Enfin, une fois de plus, la commande `\currentfootnote` n'est pas capable d'aller prendre tous les paramètres. En particulier, elle ne peut pas savoir quelle fonte tu utilises dans tes notes de pied de page (trop complexe à déterminer, même si c'est probablement la taille `\footnotesize` de la fonte par défaut de ton document). De même, elle ne peut pas déterminer facilement si tu as personnalisé la commande `\footnoterule` pour produire un filet différent. Du coup, elle ne prend en compte que `\skip\footins` et la séparation entre les notes, `\footnotesep`.

### 4.13.11 Visualiser des boites

Enfin, une dernière fonctionnalité intéressante du package `layouts` est qu'il permet de visualiser l'alignement d'à peu près n'importe quelle boite TeX ou LaTeX. Tu n'en auras probablement jamais besoin, mais c'est tout de même pratique pour voir l'alignement de

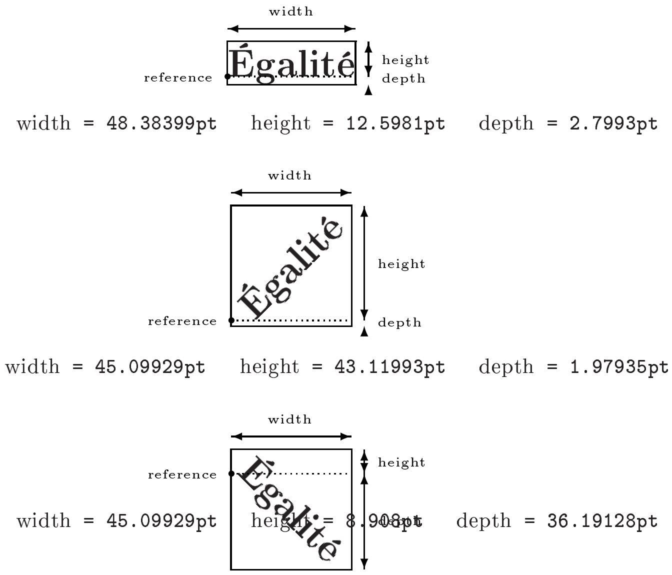



FIG. 4.32: Visualisation d'un texte après rotation

---

certains objets dans des cas très précis.

Le principe est assez simple. La première commande intéressante est `\drawfontframe` qui place simplement un cadre autour de son argument en indiquant la position de la ligne porteuse, comme ça : Égal.

La seconde commande intéressante est `\drawfontframelabel`, qui affiche le même type de chose, mais en ajoutant les différentes longueurs en jeux. La figure 4.32 en donne un bon exemple. Elle permet, dans ce cas précis, de voir les changements induits par une rotation de 45° ou de -45°.

La commande `\drawfontframelabel` peut ne pas afficher les paramètres sous le dessin, en utilisant la commande habituelle :

```
\printparametersfalse
```

## 4.14 À propos des exemples pour les marges

Les packages dont je vais te parler par la suite portent sur l'art et la manière de paramètrer les différents éléments de mise en page. Pour quasiment tous ces packages, je ne peux pas appliquer les exemples que je souhaite de montrer sur le corps de ce livre, ça en changerait totalement la mise en page.

Du coup, je vais, le plus souvent, reprendre des exemples externes, comme on les a déjà rencontrés. Simplement, cette fois-ci, on s'intéressera aux dispositions que ça permet.

La première mise en page que je te propose de regarder est celle par défaut de la classe **article** quand on lui passe l'option `a4paper`. Elle servira de référence sur deux points : c'est cet exemple que nous allons déformer avec les packages suivants et c'est également le seul cas où je te montrerai le source complet de l'exemple. En effet, dans les exemples suivants, je t'indiquerai seulement ce qui change par rapport à cet exemple-là. Pour ce qui est du corps de l'exemple, ce sera toujours à peu près le même, il n'est pas intéressant de le répéter.

Ce premier exemple de référence est celui de la figure 4.33 page suivante

```
\documentclass[a4paper]{article}
\usepackage{layouts}
\begin{document}
\printheadingsfalse
\drawparametersfalse
\printparametersfalse
\setlabelfont{\small\ttfamily}
\thispagestyle{empty}
\setlayoutscale{.25}
\currentpage
\drawpage
\end{document}
```

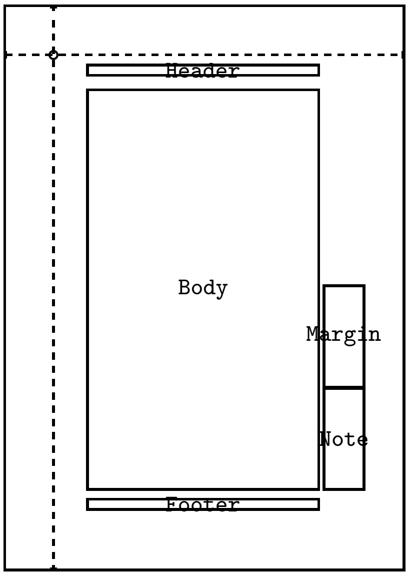



FIG. 4.33: Mise en page de la classe **article**

---

## 4.15 Mises en pages classiques : le package typearea

Ce petit package, qui fait partie de l'ensemble KOMA-Script [110], reprend, pour la mise en page, certaines anciennes traditions. Les principes qu'il applique sont simples :
- la zone de texte a les mêmes proportions que la page visible ;
- dans un document en recto, les marges de gauche et de droite sont de la même taille, dans un document recto-verso la marge intérieure est la moitié de la marge extérieure ;
- la marge du haut est la moitié de la marge du bas.

Pour arriver à ce résultat là, c'est très simple. Le point de départ est la taille du papier. Il retire, en largeur, la place nécessaire à la reliure (tu peux lui spécifier cette place). Il en déduit donc la taille de la page visible (le papier moins la reliure).

Ensuite, il divise la page visible en une grille de $n \times n$ rectangles, $n$ étant paramètrable, qui ont donc tous les mêmes proportions que la page visible. Si le papier a une taille $L \times H$, chacun de ces rectangles a la taille $l \times h$ où $l = \frac{L}{n}$ et $h = \frac{H}{n}$. La marge du haut sera de $h$ et celle du bas de $2h$.

Dans un document recto, la marge gauche et la marge droite seront de $l$. Dans un document recto-verso, la marge intérieure sera de $l$ et la marge extérieure sera de $2l$.

En y réfléchissant un peu, on arrive à la conclusion que la valeur minimale pour le paramètre $n$ est 4 : en effet, si on trace une grille de $4 \times 4$ sur la page, que la bande de gauche, les deux bandes de droites et idem en hauteur, ce sont les marges, il reste un seul rectangle au centre qui soit la page : on ne peut pas faire moins avec cette méthode, et elle donne une zone de texte minuscule qui est un seizième de la page.

Symétriquement, si on prend une valeur très élevée de $n$, alors les marges seront presque nulles et il se peut qu'il n'y ait même plus la place pour le titre courrant ou pour le pied de page.

Le paramètrage est simple : le package accepte deux types d'option. Le premier type sert à dire quelle place il faut réserver pour la reliure (par défaut, il ne réserve pas de place),

| Option | Signification |
|---|---|
| dvips | Transmet la taille du papier à `dvips`. |
| pdftex | Transmet la taille du papier à `pdftex`. |
| pagesize | Transmet la taille du papier à `pdftex` ou à `dvips` selon le mode de compilation en cours (permet l'auto-détection de `pdftex`). |
| a4paper | Papier A4. |
| a5paper | Papier A5. |
| letterpaper | Papier *letter* américain (11 pouces sur 8,5). |
| legalpaper | Papier *legal* américain (14 pouces sur 8,5). |
| executivepaper | Papier *executive* américain (10,5 pouces sur 7,25). |
| landscape | Document en mode paysage. |
| a2paper | Papier A2, en fait cette option est disponible pour tous les formats A$n$. |
| b2paper | Papier B2, disponible pour tous les B$n$. |
| c2paper | Papier C2, disponible pour tous les C$n$. |
| d2paper | Papier D2, disponible pour tous les D$n$. |

TAB. 4.17: Options du package `typearea` pour définir la taille du papier

| Option | Signification |
|---|---|
| headinclude | Considère que l'en-tête fait partie de la zone de texte. |
| headexclude | Considère que l'en-tête est dans la marge du haut. Option prise par défaut. |
| footinclude | Considère que le pied de page est dans la zone de texte. |
| footexclude | Considère que le pied de page est dans la marge du bas. Option pris par défaut. |
| mpinclude | Considère que les notes en marge sont dans la zone de texte. |
| mpexclude | Considère que les notes en marge sont dans la marge extérieure. Option prise par défaut. |

TAB. 4.18: Options du package `typearea` pour définir la zone de texte

il s'appelle BCOR[115]. Le second type de paramètre indique la valeur de $n$ et s'appelle DIV. Ainsi, pour indiquer que la reliure prend 11 millimètres et qu'on veut utiliser une valeur de 8 pour $n$, on indique :

```
\usepackage[BCOR11mm,DIV8]{typearea}
```

Une autre forme classique de mise en page, qu'on retrouve dans certains ouvrages du moyen-âge, est de considérer que la hauteur de la zone de texte doit être égale à la largeur de la page visible. Ce mode de fonctionnement est possible avec l'option DIVclassic.

Le package `typearea` permet également de fixer la taille du papier de manière un peu plus riche que ce qui est proposé par LaTeX par défaut. Il propose pour cela les options du tableau 4.17.

La figure 4.34 page suivante donne un exemple avec une valeur de $n$ de 24, sur un document recto-verso, ce qui est très élevé pour une page A4. Pour ce type d'usage-là, ou tout simplement si tu considères que l'en-tête fait partie de ton bloc de texte, il vaut peut-être mieux utiliser une des options du tableau 4.18.

La figure 4.35 page 255 donne un exemple plus raisonnable, en recto, avec une valeur de $n$ de 12. Ça correspond à un document avec de petites marges, c'est raisonnable pour un

[115]Pour « Binding CORrection ». Binding, en anglais, c'est reliure.

| Option | Signification |
|---|---|
| BCOR | Permet d'indiquer la place à conserver pour la reliure. |
| DIV | Indique la taille de la grille à utiliser pour calculer les marges. |
| DIVcalc | Indique de calculer automatiquement la taille de la grille pour obtenir une longueur de ligne raisonnable. |
| DIVclassic | Indique d'utiliser le format dit classique où la hauteur de texte est égale à la largeur de la page visible. |
| headlines | Permet d'indiquer le nombre de lignes de l'en-tête, par exemple 3headlines pour 3 lignes. |

TAB. 4.19: Options du package `typearea` pour définir la mise en page

```
* \documentclass
*     [a4paper,twoside]
*             {article}
* \usepackage{layouts}
* \usepackage[DIV24]{typearea}
  \currentpage
  \drawpage
```

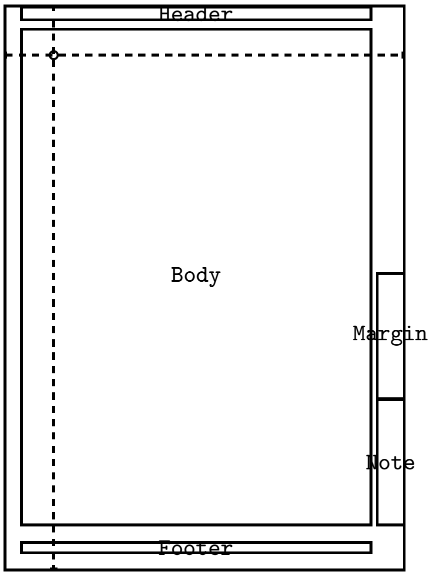



FIG. 4.34: Exemple d'utilisation de `typearea`

document en corps 12 et ça fait des lignes un peu longues pour un document en corps 10. Parfait pour un document en deux colonnes.

L'exemple suivant, figure 4.36 page suivante, correspond à ce qui est préconisé par l'auteur du package pour une page A4 en corps 10, à savoir une valeur de $n$ de 8. Ça fait un texte relativement restreint, mais on n'est pas très loin des recommandations d'Henri FOURNIER [67] que nous avons déjà vues.

L'exemple suivant, figure 4.37 page 256 est plutot un exemple de ce qu'il ne faut pas faire : il utilise une valeur de $n$ de 4, ce qui est le minimum possible. Cet exemple est tellement petit que pour pouvoir afficher le dessin produit par `layouts` j'ai dû en réduire l'échelle (sinon, il ne tenait pas dans la zone de texte ridiculement petite du document).

Enfin, le package `typearea` propose deux valeurs spéciales pour l'option DIV. La première, DIVcalc, demande au package de choisir une valeur raisonnable pour $n$ en fonction de la taille du papier et du corps de fonte, pour que la longueur de la ligne corresponde à quelque chose de lisible. Le package prend en compte le fait que le document soit en deux colonnes ou pas, mais seulement via l'option twocolumn de `\documentclass` ; il ne prend pas en compte

```
\documentclass
    [a4paper,twocolumn]
              {article}
\usepackage{layouts}
\usepackage[DIV12]{typearea}
\currentpage
\drawpage
```

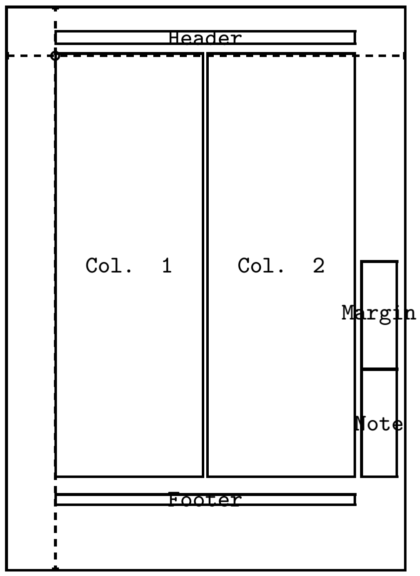



FIG. 4.35: Exemple d'utilisation de `typearea`

```
\documentclass
    [a4paper,twoside]
            {article}
\usepackage{layouts}
\usepackage[DIV8]{typearea}
\currentpage
\drawpage
```

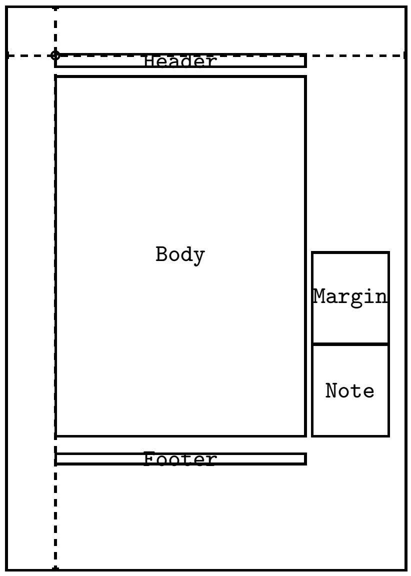



FIG. 4.36: Exemple d'utilisation de `typearea`

```
\documentclass
    [a4paper,twoside]
            {article}
\usepackage{layouts}
\usepackage[DIV4]{typearea}
\setlayoutscale{.2}
\currentpage
\drawpage
```

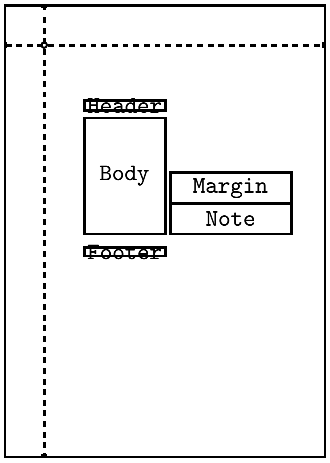



FIG. 4.37: Exemple à ne pas suivre d'usage de `typearea`

```
\documentclass{article}
\usepackage{layouts}
\usepackage
    [DIVclassic,
     d4paper,
     5headlines,
     headinclude]{typearea}
\currentpage
\drawpage
```

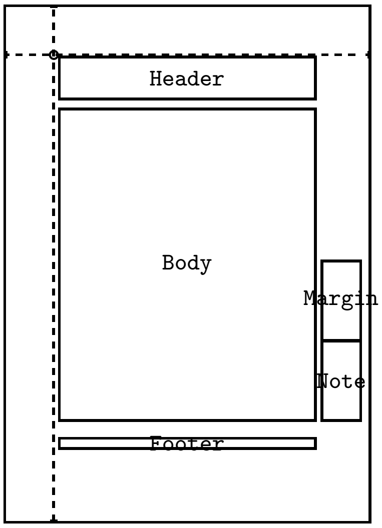



FIG. 4.38: Exemple d'utilisation de l'option DIVclassic du package `typearea`

---

l'environnement `multicols` du package `multicol`. La seconde, que nous avons déjà vue, est DIVclassic.

L'exemple de la figure 4.38 illustre d'utilisation de cette option DIVclassic sur un document D4 ayant cinq lignes d'en-tête et dont on considère que l'en-tête est dans la zone de texte.

Si tu changes de fonte dans le préambule de ton document, par exemple pour utiliser du Times, les calculs fait par `typearea` peuvent s'avérer faux. Ce sera le cas, par exemple, si tu utilises une classe de document qui appelles `typearea` et que tu charges par la suite un package de changement de fonte : lorsque `typearea` a fait ses calculs (en particulier pour l'option DIVcalc), il ne savait pas que tu changerai de fonte et donc son estimation de la longueur convenable pour une ligne s'en trouve faussée. Tu peux lui dire de refaire ses calculs plus tard, mais de préférence dans le préambule (en effet, changer les paramètres de la mise en page dans le corps du document peut avoir des effets des plus curieux et le plus souvent ne marchera pas).

$\boxed{\texttt{\textbackslash typearea[}\langle\textit{reliure}\rangle\texttt{]\{}\langle n\rangle\texttt{\}}}$

Pour lui faire refaire ses calculs, il suffit d'utiliser la commande `\typearea`. Elle prend deux paramètres. Le premier, optionnel, indique la zone à réserver pour la reliure. Le second, obligatoire, indique la valeur de $n$. Tu peux par exemple faire cet appel là :

```
\typearea[11mm]{8}
```

À la place du paramètre correspondant à $n$, tu peux utiliser quelques valeurs normalement interdites. `1` correspond à l'option DIVcalc ; `3` correspond à l'option DIVclassic. De même, des valeurs symboliques son possibles. `calc` correspond à l'option DIVcalc, `classic` correspond à DIVclassic, `current` reprend la valeur courrante de $n$, `default` prend la valeur par défaut pour ce format de papier et pour le corps de fonte en cours et enfin `last` reprend la valeur du dernier appel. Je ne comprends pas bien le rôle de `current` par rapport à `last`.

De même, tu peux utiliser `current` comme valeur pour le paramère optionnel, si tu ne souhaites pas le modifier.

| Option | Type | Description |
|---|---|---|
| includehead | B | L'en-tête est pris dans la zone de texte. |
| ignorehead | B* | L'en-tête est pris dans la marge. |
| includefoot | B | Le pied de page est pris dans la zone de texte. |
| ignorefoot | B* | Le pied de page est pris dans la marge. |
| includeheadfoot | B | includehead et includefoot. |
| ignoreheadfoot | B | ignorehead et ignorefoot. |
| includemp | B | Les notes en marge sont prises dans la zone de texte. |
| ignoremp | B* | Les notes en marge sont prises dans la marge. |
| includeall | B | includeheadfoot et includemp. |
| ignoreall | B | ignoreheadfoot et ignoremp. |

* Option prise par défaut.

Types pour les options :

**B** Booléen ; valeurs possibles : false et true. Par défaut, si l'option est présente sans valeur, true est supposé.

TAB. 4.20: Options de `geometry` définissant la zone de texte

## 4.16 Régler les marges à la main : le package `geometry`

Le package `geometry`, que nous devons à Hideo UMEKI [208], te permettra, assez facilement, de contrôler tous les paramètres de la mise en page de ton document.

Plus exactement, il s'intéressera à tout ce qui touche aux marges.

Le mode de fonctionnement du package est assez simple : il prend des options (il en existe une grande variété) lors de son chargement qui précisent la mise en page voulue. Si le package a été chargé avec des options qui ne te conviennent pas, par exemple par la classe de documents que tu utilises, tu peux à tout moment changer ces options en appelant la commande `\geometry` en lui passant les options qui te conviennent. Bien entendu, en dehors du préambule, un appel à cette commande n'a pas de sens.

Ainsi, si ta classe de document utilise le package `geometry` en lui passant des paramètres qui ne te plaisent pas, tu peux appeler la commande `\geometry` ensuite pour changer ces options. L'option reset permet d'ailleurs d'annuler toutes les options précédentes.

Le package `geometry` utilise les options qui lui sont passées, puis celles qui sont passées à la commande `\geometry`, puis, enfin, les options globales du document (celles de `\documentclass`).

Ainsi, dans un appel comme celui-là :

```
\documentclass[landscape]{article}
\usepackage[width=12cm]{geometry}
\geometry{height=8cm,reset,scale=0.8}
```

ne survivront que les options scale et landscape.

### 4.16.1 Découpage de la page

Le package `geometry` s'intéresse essentiellement au positionnement de la zone de texte dans la page. Il considère que la page se découpe horizontalement en trois parties : gauche (left), largeur de la zone de texte (width) et droite (right). Dans le cas d'un document recto-verso, left désigne la marge intérieure, c'est-à-dire la marge de gauche des belles pages (pages impaires, recto) et donc la marge de droite des fausses pages (pages paires, verso). En hauteur

| Option | Description |
|---|---|
| paper | Indique la taille du papier. Peut prendre les valeurs suivantes : a0paper, a1paper, a2paper, a3paper, a4paper, a5paper, a6paper, b0paper, b1paper, b2paper, b3paper, b4paper, b5paper, b6paper, letterpaper, executivepaper ou legalpaper.<br>Un appel à a4paper est équivalent à paper=a4paper. De même un appel à a4paper=XXX quelle que soit la valeur passée à l'option est équivalent à a4paper. |
| screen | Une valeur spéciale qui correspond aux proportions d'un écran d'ordinateur classique (225 mm sur 180 mm). |
| paperwidth | Donne la largeur exacte du papier. |
| paperheight | Donne la hauteur exacte du papier : ainsi a4paper est un raccourci pour dire paperwidth=21cm et paperheight=29.7cm. |
| papersize | Donne la taille exacte du papier : ainsi a4paper est une abréviation pour papersize={21cm,29.7cm}. |
| landscape | Indique une orientation en paysage. |
| portrait | Indique une orientation en portrait. Option prise par défaut. |

TAB. 4.21: Options de `geometry` définissant la taille du papier

---

il découpe également la page en trois parties : le haut (top), la hauteur de la zone de texte (height) et le bas (bottom).

L'en-tête peut être considéré comme étant pris dans la marge (ignorehead) ou comme étant pris dans la zone de texte (includehead). De même que le pied de page peut être dans la marge (ignorefoot) ou dans la zone de texte (includefoot). Horizontalement, il en va de même, soit il considère que les notes en marge sont effectivement prises sur la marge (ignoremp), soit il considère qu'elles sont prises sur le coté de la zone de texte (includemp). L'ensemble de ces options est repris de manière synthétique dans le tableau 4.20 page ci-contre.

### 4.16.2 Dimensions du papier

Dans tous ses calculs, le package `geometry` suppose qu'il connaît la taille réelle du papier. Pour lui préciser la taille du papier, tu peux utiliser les options du tableau 4.21.

Les options donnant directement la taille du papier, comme a4paper et landscape, peuvent être passées à la classe de document, `geometry` en tiendra compte.

### 4.16.3 Dimensions de la zone de texte

Spontanément, si on ne lui dit rien, `geometry` va considérer que la zone de texte représente 70% de la page dans chaque direction, *i.e.* que la largeur du texte est 70% de la largeur de la page et idem pour sa hauteur[116]. Ça correspond à utiliser l'option :

```
\geometry{hscale=0.7,vscale=0.7}
\geometry{scale={0.7,0.7}}
\geometry{scale=0.7}
```

Ces trois appels étant équivalent.

[116]Ce qui fait que le texte occupe $0{,}7 \times 0{,}7 = 0{,}49$ soit 49% de la surface du papier. Avec des marges raisonnables classiques, le texte représente moins de la moitié de la surface du papier !

| **Option** | **Type** | **Description** |
|---|---|---|
| scale, hscale, vscale | P/R*, R, R | Définit le ratio entre la taille du papier et la taille de la zone de texte. Les options hscale et vscale spécifient le rapport de la largeur width sur paperwidth et height sur paperheight. |
| width, totalwidth | L | Définit la largeur de la zone de texte. C'est la même que textwidth si ignoremp est vrai. Ces deux options sont synonymes. |
| height, totalheight | L | Idem en hauteur. Prend en compte, ou non, les en-têtes et les pieds de page, selon la définition de la zone de texte. |
| total | PL | Indique width et height en une seule option. |
| textwidth | L | Indique la longueur d'une ligne de texte. Peut être moins que width si includemp. |
| textheight | L | Indique la hauteur du texte pur. Peut être moins que height si includefoot et/ou includehead. |
| text, body | PL | Indique textwidth et textheight en une seule option. Ces deux options sont synonymes. |
| lines | E | Indique textheight comme un nombre de lignes de texte. |
| heightrounded | B | Indique que la valeur de textheight doit être arrondie au nombre entier de lignes le plus proche. |

* Option prise par défaut. scale=0.7.

Types pour les options :

**R** Ratio. Nombre à virgule, par exemple `0.8` indiquant 80%.

**P/R** Ratio ou paire de ratios, par exemple `0.8`, équivalent à `{0.8,0.8}`.

**L** Longueur. Indique une longueur directe, ou calculée, par exemple `12.6cm` ou `0.6\paperwidth+2cm`.

**PL** Paire de longueurs. Par exemple `{12cm,16.5cm}`.

**E** Entier.

**B** Booléen, valeurs possibles false et true. Par défaut, si l'option est présente sans valeur, true est supposé.

TAB. 4.22: Options de `geometry` définissant la taille de la zone de texte

Tu peux préférer lui donner directement les dimensions (dans n'importe quelle unité comprise par TEX ou en faisant de savants calculs) de ta zone de texte. Pour bien comprendre ce que tu fais, il faut te souvenir de la différence entre la largeur de la zone de texte et la longueur d'une ligne de texte. En effet, par défaut, ces deux notions coïncident. Mais si tu actives l'option includemp, alors elles ne coïncident plus : la zone de texte inclut les notes en marge ; sa largeur, que tu peux spécifier avec l'option width comprend alors la longueur de la ligne (option textwidth), la séparation entre texte et notes en marge (marginparsep) et la largeur des notes en marge (marginparwidth).

La même distinction existe pour ce qui est de la hauteur. La hauteur de la zone de texte (height) est égale, par défaut, à la hauteur du texte proprement dit. Si tu actives l'option includefoot, alors elle comprend également la hauteur du pied de page (footskip). Et si tu actives l'option includehead, alors la hauteur de l'en-tête (headheight) ainsi que le blanc entre l'en-tête et le texte (headsep) sont comptés dans height en plus de textheight.

Ainsi, si tu actives les trois options, par exemple en utilisant includeall, tu te trouves dans ce cas là :

$$\left\{\begin{array}{lcl} \mathsf{width} & = & \mathsf{textwidth} + \mathsf{marginparsep} + \mathsf{marginparwidth} \\ \mathsf{height} & = & \mathsf{textheight} + \mathsf{headheight} + \mathsf{headsep} + \mathsf{footskip} \end{array}\right.$$

Certaines options permettent de régler plusieurs grandeurs ensembles : ainsi total permet de fixer width et height ; ces deux lignes sont équivalentes :

```
\geometry{total={16cm,20cm}}
\geometry{width=16cm,height=20cm}
```

De même, l'option body (ou text qui est un synonyme) permet de positionner textwidth et textheight en un seul appel ; ces trois lignes là étant donc équivalentes puisque par défaut c'est excludeall qui prévaut :

```
\geometry{body={12cm,17cm}}
\geometry{text={12cm,17cm}}
\geometry{textwidth=12cm,textheight=17cm}
```

Enfin, en plus de spécifier un pourcentage (avec scale) ou une dimension exacte (avec body), tu peux préciser la hauteur du texte (textheight) comme étant un nombre de lignes. C'est pratique, pour certaines mises en page, en particulier quand on cherche à s'aligner sur la grille.

S'aligner sur la grille, c'est faire en sorte que, si on trace sur une grille des lignes séparées d'exactement l'interligne qu'on a choisi (mettons, pour l'exemple, 12 points), alors les lignes de texte tomberont toujours sur les lignes tracées. Ça peut sembler évident, mais ça ne l'est pas. Ainsi, ça suppose que tout les éléments qui sont plus haut que la ligne (les titres, les équations, les tableaux, les images, les flottants, etc.) aient tous une hauteur qui soit un multiple de l'interligne choisi.

TEX, spontanément, cherche à avoir des pages qui font toujours la même longueur (c'est plus joli, sur un document recto-verso), ainsi, il se trouve parfois obligé d'étirer un peu ses pages, ou de les condenser un peu. Pour ce faire, il utilise l'élasticité de certains blancs (souvent, le blanc autour des titres, autour des équations, etc., est un peu élastique, de même que le blanc entre deux paragraphes).

Or, dans une mise en page destinée à être alignée sur la grille ou plus simplement dans un texte brut (genre un roman, pas de titres, pas de dessins, etc.) dans lequel le blanc entre deux paragraphes n'est pas élastique, il n'y a pas d'élasticité. Du coup, en pareil cas, si ta page ne contient que du texte (ce qui n'est pas si rare) et que la hauteur de la zone de texte n'est pas un multiple exact du nombre de lignes, TEX va être triste. Et quand il est triste,

| Option | Type | Description |
|---|---|---|
| left, lmargin, inner | L | Taille de la marge intérieure (recto-verso) ou gauche (recto). |
| right, rmargin, outer | L | Taille de la marge extérieure (recto-verso) ou droite (recto). |
| top, tmargin | L | Taille de la marge du haut (du haut de la feuille à la zone de texte). |
| bottom, bmargin | L | Taille de la marge du bas (de la zone de texte au bas de la feuille). |
| hmargin, vmargin | P/L, P/L | Donne les deux marges horizontales (resp. verticales) en une seule fois. |
| margin | P/L | Donne hmargin et vmargin en une seule fois. margin=2cm fixe les 4 marges. |
| bindingoffset | L | Zone réservée à la reliure. |
| twoside | B | Indique qu'on est en recto-verso, échange les marges droite et gauche sur les pages verso. |
| asymmetric | B | N'échange pas les marges des pages verso, mais fait bien l'inversion pour la reliure (un document mis en page en recto mais imprimé et relié en recto verso). |

Types pour les options :

**L** Longueur. Indique une longueur directe, ou calculée, par exemple `12.6cm` ou `0.6\paperwidth+2cm`.

**P/L** Longueur ou paire de longueurs, par exemple `12cm`, équivalent à `{12cm,12cm}`.

TAB. 4.23: Options de `geometry` définissant la taille des marges

---

il râle. Ça se traduira par un warning à la compilation (`Underfull \vbox while \output is active`).

Pour éviter ce type de message, il suffit de dire que la hauteur de la page est un multiple exact du nombre de lignes, par exemple :

```
\geometry{textheight=35\baselineskip+\topskip}
```

indiquera une page de 35 lignes (`\baselineskip` est l'interligne et `\topskip` est la hauteur supplémentaire de la première ligne du texte). Tu peux obtenir le même résultat avec :

```
\geometry{lines=35}
```

Cette méthode marche, mais tu es obligé de calculer toi-même le nombre de lignes que tu veux obtenir. Une autre alternative est de faire quelque chose comme ça :

```
\geometry{scale=.7,heightrounded}
```

qui calculera la hauteur du texte de manière habituelle (tu peux utiliser n'importe quelle autre méthode), puis l'arrondira au nombre entier de lignes le plus proche.

Le tableau 4.22 page 260 reprend l'ensemble de ces options de manière synthétique.

### 4.16.4 Taille des marges

Les marges, au sens « le blanc autour de la zone de texte », peuvent être définies selon deux grandes méthodes. Soit on spécifie des dimensions exactes, soit on spécifie des règles à respecter.

| Option | Type | Description |
|---|---|---|
| hmarginratio, vmarginratio, marginratio, | R, R, P/R* | Fixe le ratio de la marge gauche sur la marge droite (resp. haute/basse). marginratio fixe le même ratio en horizontal et en vertical. |
| hcentering, vcentering, centering | B, B, B | Demande de centrer la zone de texte. Équivalent à un ratio de `1:1`. |
| hdivide | 3 | Fixe la marge gauche, la largeur de la zone de texte, et la marge droite en une seule option. |
| vdivide | 3 | Idem en vertical. |
| divide | 3 | Fixe la même règle de division en horizontal et en vertical. |

* Option prise par défaut. marginratio=2:3 en recto-verso, et marginratio=1:1 en recto.
Types pour les options :

**R** Ratio. Indique le rapport de deux grandeurs entre elles : par exemple 2:3 indique que la première grandeur est deux-tiers de la seconde.

**P/R** Ratio ou paire de ratios : par exemple `2:3`, équivalent à `{2:3,2:3}`.

**B** Booléen ; valeurs possibles : false et true. Par défaut, si l'option est présente sans valeur, true est supposé.

**3** Longueurs, a priori trois. Celles laissées en blanc, ou avec une étoile, sont automatiquement calculées. Par exemple hdivide={2cm, ,3cm} laisse une marge gauche de deux centimètres, droite de trois, et calcule la largeur du texte en fonction de la taille du papier.

TAB. 4.24: Options de `geometry` définissant le rapport des marges

---

Dans un document recto-verso (spécifié avec l'option twoside) les marges sont alternées entre pages recto et pages verso, de telle sorte que la zone de texte apparaît toujours au même endroit du papier. On parle alors de marge intérieure (inner) et extérieure (outer). En fait, on indique la mise en page pour les pages de droite (impaires) et LaTeX se chargera d'échanger les marges pour les pages de gauche (paires). Dans ce cas là, il y a donc équivalence entre right et outer, qui sont en fait des synonymes. Dans un document recto, ces marges ne sont pas échangées, bien entendu.

Le package `geometry`, comme `typearea`, prévoit une zone réservée à la reliure (bindingoffset). Cette zone est réservée sur l'intérieur du document et n'est plus prise en compte dans le calcul des marges. Dans un document recto-verso, cette zone est réservée sur le côté intérieur (gauche pour les pages de droite, droite pour les pages de gauche). Dans un document recto, cette zone est réservée sur le côté gauche. Enfin, on peut choisir un mode hybride, dont l'utilité ne me parait pas évidente, que `geometry` appelle asymmetric. Dans ce mode là, les marges ne sont pas échangées (comme si le document était en recto), mais la zone réservée à la reliure, elle, est échangée entre pages paires et impaires. Ça correspondrait à un document recto imprimé en recto-verso.

Pour spécifier les marges, on peut utiliser les quatre grandeurs appropriées ou leurs synonymes : right désigne la marge de droite (alias rmargin ou outer) ; left la marge de gauche (alias lmargin ou inner) ; top la marge du haut (alias tmargin) et bottom la marge du bas (alias bmargin).

Tu peux également préciser ces grandeurs par paires, ainsi, ces lignes sont équivalentes :

```
\geometry{left=1cm,right=2cm,top=3cm,bottom=4cm}
\geometry{hmargin={1cm,2cm},vmargin={3cm,4cm}}
```

À noter que si tu donnes une seule grandeur à hmargin ou à vmargin, il considère que les deux marges ont la même taille. Il existe même une option margin. Si tu lui donnes une seule valeur, il s'en servira pour les quatre marges ; si tu lui en donnes deux, la première servira pour les marges horizontales et la seconde pour les marges verticales. Ainsi, pour laisser une marge uniforme de deux centimètres tout autour de ta page, tu peux faire :

```
\geometry{margin=2cm}
```

Cette approche pour les marges n'est pas très pratique, et peut t'amener à demander à geometry des choses contradictoires. Par exemple :

```
\geometry{a4paper,scale=0.6,margin=2cm}
```

En effet, tu dis que la largeur du papier est de 21 centimètres (a4paper), que le texte occupe 60% de cette largeur, soit 12,6 centimètres (restent 8,4 centimètres) et que toutes les marges font 2 centimètres. Il reste 4,4 centimètres en largeur dont on ne sait pas quoi faire...

Si tu veux spécifier toutes les grandeurs, comme dans l'exemple précédent, tu as tout intérêt à bien faire ton calcul ! Sinon, tu peux, plus raisonnablement, donner à geometry suffisamment d'informations pour qu'il comprenne ce que tu veux et lui laisser calculer le reste. Par exemple, si pour toi, le plus important c'est d'avoir une marge de deux centimètres sur une feuille A4, alors il suffit de dire :

```
\geometry{a4paper,margin=2cm}
```

Il se débrouillera tout seul pour en déduire la taille de la zone de texte.

Une autre approche pour donner la taille des marges est de donner ça sous forme d'un ratio, par exemple pour dire « les marges droites et gauche ont la même taille ». C'est le rôle de l'option hmarginratio, elle peut s'utiliser par exemple comme ça :

```
\geometry{a4paper,scale=.6,hmarginratio=1:2}
```

qui indique que la marge gauche est de la moitié de la marge de droite. geometry se chargera de calculer tout seul la taille exacte de chacune des marges, fonction de ce qu'il reste comme place sur le papier, puisque tu as précisé la taille de la zone de texte.

L'option équivalente en vertical est vmarginratio qui indique le rapport de la marge haute à la marge basse. Comme on l'a déjà vu et comme le recommande [67], il est de bon ton que le rapport de la marge gauche à la marge droite soit le même que celui de la marge haute à la marge basse. D'ailleurs, Henri FOURNIER, dans [67], nous recommande un rapport de deux à trois. Il recommande même que la marge du haut soit la même que la marge intérieure, d'ailleurs.

L'option hcentering correspond à un appel à l'option hmarginratio avec un rapport de un pour un (`1:1`). Les options vcentering et centering font la même chose.

Enfin, il reste une méthode un peu complexe qui permet de préciser toutes les informations utiles en seulement quelques options. Ça permet de préciser, en même temps, la taille de la zone de texte et les marges. Ces options sont hdivide et vdivide. Elles prennent trois paramètres : la taille de la marge gauche (resp. haute), la largeur de la zone de texte (resp. hauteur) et la taille de la marge droite (resp. basse). Si tu précises les 3 grandeurs, alors la taille de la zone de texte est ignorée et recalculée d'après la taille du papier et la taille des marges que tu as indiquée. Si tu ne précises que 2 des grandeurs (en laissant la troisième vide, ou en la remplaçant par $*$), il calcule le grandeur manquante (la taille du papier moins les deux grandeurs que tu as indiquées). Si tu ne précises qu'une des grandeurs, il calcule le reste à partir du marginratio et de la taille du papier. Enfin, si tu ne précises aucune des grandeurs (ce qui ne sert à rien), il applique ses règles par défaut, c'est-à-dire un scale de 70% et un marginratio de 2:3.

Tu peux aussi utiliser divide, qui appliquera ses trois grandeurs à hdivide et à vdivide, ce qui n'a de sens que si tu as du papier carré ou si tu ne précises que les marges.

Par exemple, les recommandations d'Henri FOURNIER peuvent s'écrire :

```
\geometry{divide={4cm,*,6cm}}
```

à savoir même marge à gauche qu'en haut ; même marge à droite qu'en bas et un ratio de 2 à 3.

L'ensemble de ces options est repris de manière synthétique dans les tableaux 4.23 page 262 et 4.24 page 263.

### 4.16.5 Parler directement en LaTeX

Toutes les grandeurs et options que nous venons de voir, pour le package `geometry`, ne correspondent pas aux grandeurs que LaTeX utilise pour définir sa page. En effet, ce sont des éléments utilisés par `geometry` pour définir ce que tu veux comme résultat et il traduira ensuite tout cela en termes de paramètres pour LaTeX.

Tu peux décider de ne pas passer par cette étape d'interprétation et préciser toi même les valeurs que LaTeX doit donner à ses différents paramètres. C'est assez délicat, parce qu'il faut bien comprendre ces paramètres. La figure 4.39 page suivante, que tu connais déjà, reprend l'ensemble de ces paramètres.

Pour utiliser directement les paramètres de LaTeX, tu peux faire appel aux options qui sont listées dans le tableau 4.25 page 267.

Pour ce qui est des marges droite/gauche, il faut comprendre comment fonctionne LaTeX. Comme te le montre la figure 4.39 page suivante, LaTeX n'a pas la notion de marge de droite, ni de taille de papier. Ça ne l'intéresse pas. Ce qui l'intéresse, c'est combien de blanc il faut laisser avant d'écrire et sur quelle largeur il faut écrire. La taille du papier, c'est un problème d'imprimante. Par contre, il sait que la taille du blanc à laisser (la marge de gauche), dans le cas particulier d'un document recto-verso, n'est pas la même pour les pages paires et pour les pages impaires.

Ainsi, dans un document recto, ce qui définit la taille de la marge gauche, c'est `\oddsidemargin` (littéralement « marge du côté pour les impaires »). Dans un document recto verso, il utilisera `\oddsidemargin` pour les pages impaires et `\evensidemargin` pour les pages paires. Ainsi, quand tu dis à `geometry`, pour un document recto-verso, d'utiliser une marge gauche de $x$ et une marge droite de $y$, il va le traduire en `\oddsidemargin=`$x$ et `\evensidemargin=`$y$[117]. Si ton document est en recto simple, LaTeX ne se servira pas de ta marge droite, jamais[118].

### 4.16.6 Choix du driver

TeX, et donc LaTeX qui se base dessus, n'a pas la notion de taille de papier, ni d'orientation. Pour lui, portrait ou paysage, ce n'est pas parlant, et ça ne le concerne pas. Par contre, ça va concerner très fort le logiciel qui se chargera de te montrer ton document à l'écran ou celui qui se chargera de te l'imprimer. Dans certains cas, quand on utilise des moteurs autres que TeX, ces moteurs peuvent avoir la notion de taille de papier et s'intéresser à l'orientation du papier. C'est par exemple le cas de pdfTeX qui peut générer du PDF directement : il faut qu'il écrive dans le fichier PDF le format exact du papier et l'orientation utilisée.

[117] La traduction n'est pas si directe, puisque, lui, il connaît la taille du papier et donc il va en déduire d'autres grandeurs, comme la largeur de ton texte.

[118] Mais, je l'ai déjà dit, `geometry`, lui, s'en servira dans ses calculs.

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.

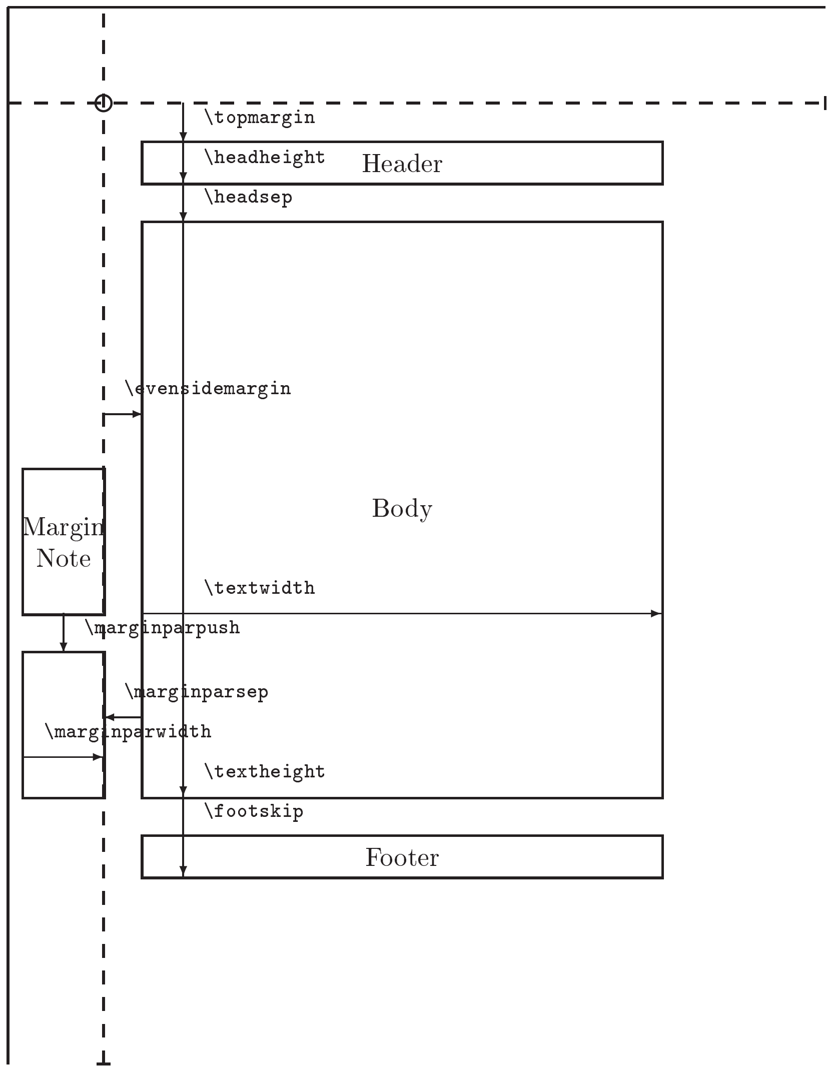



FIG. 4.39: Diagramme générique d'une mise en page

| Option | Type | Description |
|---|---|---|
| headheight, head | L | Hauteur de l'en-tête. |
| headsep | L | Blanc entre l'en-tête et le texte. |
| footskip, foot | L | Distance du bas du texte au bas du pied de page. |
| nohead | B | Supprime l'en-tête. |
| nofoot | B | Supprime le pied de page. |
| noheadfoot | B | Supprime l'en-tête et le pied de page. |
| footnotesep | L | Blanc entre le bas du texte et le haut des notes de bas de page. |
| marginparwidth, marginpar | L | Largeur des notes en marge. |
| marginparsep | L | Blanc entre les notes en marge et le texte. |
| nomarginpar | B | Ne réserve pas de place pour les notes en marge. Équivalent à marginpar=0pt et marginparsep=0pt. |
| columnsep | L | Blanc entre les deux colonnes, si on est en deux colonnes. |
| hoffset | L | Décale l'ensemble de la page. |
| voffset | L | Décale l'ensemble de la page. |
| offset | P/L | Décale l'ensemble de la page. |
| twocolumn | B | Indique que le document est en deux colonnes. Ne marchera que si la classe de document le prévoit. |
| twoside | B | Indique que l'on est en recto-verso (échangera les marges selon la parité de la page). Ne marchera que si la classe de documents le prévoit. |
| textwidth | L | Largeur du texte. |
| textheight | L | Hauteur du texte. |
| reversemp, reverse-marginpar | B | Fait apparaître les notes en marge dans la marge intérieure au lieu de la marge extérieure. |

Types pour les options :

**L** Longueur. Indique une longueur directe, ou calculée, par exemple `12.6cm` ou `0.6\paperwidth+2cm`.

**P/L** Longueur ou paire de longueurs, par exemple `12cm`, équivalent à {`12cm,12cm`}.

**B** Booléen ; valeurs possibles : false et true. Par défaut, si l'option est présente sans valeur, true est supposé.

TAB. 4.25: Options de `geometry` utilisant directement les paramètres de LaTeX

| Option | Description |
|---|---|
| dvips | Utilisation de `dvips` ou de tout autre programme compatible avec lui. Option par défaut. |
| vtex | Utilisation du moteur VTEX (commercial). |
| dvipdfm | Utilisation de `dvipdfm`. |
| pdftex | Utilisation du moteur pdfTEX. |

TAB. 4.26: Options de `geometry` pour le choix du driver

Pour communiquer avec ces différents outils (soit des outils externes, pour ce qui est de TEX, soit avec les commandes spécifiques pour les autres moteurs), le package `geometry` doit s'adapter, parce que, bien entendu, il y a autant de langues que de logiciels.

Le plus souvent, en fait, `geometry` est capable de détecter tout seul les cas particuliers connus, et ne tiendra même pas compte des indications que tu pourrais lui donner.

L'option la plus raisonnable est dvips. D'abord parce que c'est un programme très répandu et qui marche bien, mais aussi parce que la majorité des programmes de visualisation et/ou d'impression s'efforcent d'être compatible avec `dvips` qui est donc une espèce de norme de fait. En particulier, cette option est compatible avec `xdvi` et `dviout`, avec peut-être des warnings pour ce dernier.

La différence entre dvips et dvipdfm est que pour le premier, il effectue quelques règlages, si tu demandes landscape, pour que le document apparaisse à l'endroit dans un visualisateur de PostScript (ce qui n'est pas évident de prime abord, puisque ton imprimante ne l'entend a priori pas de cette oreille[119]). Pour dvipdfm ce petit règlage n'est pas utilisé puisque le PDF, au contraire du PostScript, inclut nativement cette notion de mode paysage.

Il est à noter que, dans la majorité des cas, si tu te trompes d'option, le package corrigera de lui-même. En effet, il est capable de détecter si tu utilises VTEX ou pdfTEX et si c'est le cas il activera de lui-même le nécessaire sans tenir compte de ton avis.

Le seul cas qu'il ne peut pas déterminer tout seul c'est celui-ci : tu utilises LATEX pour produire un fichier DVI, et ensuite tu vas le convertir. Il ne peut pas savoir si tu utiliseras `dvips` ou `dvipdfm`, c'est-à-dire si tu convertiras ton fichier DVI en PostScript ou en PDF. Dans ce cas-là, il ne peut pas y avoir de détection automatique, puisque ce n'est pas pendant la compilation LATEX que le choix se fait. Dans ce cas-là, le package fait donc entièrement confiance aux options dvips et dvipdfm : il te croit sur parole.

Ces quelques options sont reprises succintement dans le tableau 4.26.

Bien entendu, elles peuvent être passées à la commande `\geometry`, mais il semble plus raisonnable, puisque ce sont des options relativement globales et qui peuvent intéresser plusieurs packages (les noms en sont relativement standard, ils sont par exemple reconnus également par le package `graphics`) de les passer au `\documentclass`.

### 4.16.7 Agrandissement

TEX est capable de faire, tout seul, un document agrandi. Autrefois, ça pouvait être utile pour imprimer sur une imprimante à faible résolution (typiquement 240 dpi pour les imprimantes à aiguilles) et augmenter la résolution en faisant une réduction à la photocopieuse. De nos jours ce sont des choses à éviter : ça marche très bien, mais ça a des effets de bord

[119]Pour l'imprimante il faut faire tourner le texte de 90 degrés. Pour l'affichage à l'écran (du PDF par exemple) il faut changer la taille de la fenêtre. C'est ton logiciel de visualisation PDF qui ajoutera la rotation de 90 degrés lors de l'impression.

| Option | Description |
|---|---|
| verbose | Le package raconte sa vie à l'écran, en plus du fichier `.log`. |
| reset | Annule toutes les options précédentes et revient dans sa configuration par défaut. |
| mag | Pas toucher, dangereux et sans intérêt. |
| truedimen | Pareil. Pas toucher. |
| pass | Désactive entièrement le fonctionnement du package (il ne touche pas la mise en page) sauf verbose et showframe. Pour tester la mise en page d'un autre package. |
| showframe | Dessine les éléments clefs de la mise en page sur la première page du document. |
| compat2 | À donner en premier, pour qu'il prenne les options par défaut d'une vieille version. Intéressant seulement pour les gens qui l'utilisaient déjà dans de vieilles versions. |

TAB. 4.27: Options de contrôle de `geometry`

auxquels on ne s'attend pas.

Pour faire ça, TEX utilise une grandeur, `\magnification` qui vaut 1000 par défaut. Si tu lui donnes une valeur de 2000, ça double toutes les dimensions de ton document (y compris les tailles des fontes). L'option mag permet de fixer cette valeur.

Une fois que tu as, par exemple, indiqué la valeur 2000 alors *toutes* les grandeurs sont doublées. Ainsi, si tu dis que tu es en A4, il comprendra que ton papier fait 42 cm sur 59,4 cm, puisqu'il double *tout*. Si tu veux lui indiquer que le papier est *vraiment* en A4, du peux faire comme ça :

```
\geometry{mag=2000,paper={21truecm,29.7truecm}}
```

Pour indiquer à `geometry` que *toutes* les grandeurs que tu lui as données sont vraies et qu'il ne faut pas les agrandir, tu peux utiliser l'option truedimen.

Cependant, je te le redis : l'utilisation de ces deux options est à éviter autant que faire se peut.

### 4.16.8 Options de contrôle

Enfin, certaines options permettent de contrôler le travail du package, en dehors de ses calculs sur la mise en page. Ces options sont regroupées dans le tableau 4.27.

verbose est probablement l'option la plus immédiatement utile : elle indique au package d'afficher le résultat de ses savants calculs à l'écran pendant la compilation. C'est une option booléenne, par défaut à false. Quelle que soit la valeur de l'option, tu peux retrouver les résultats en question dans le fichier `.log`.

reset, par contre est une option plus raisonnable d'utilisation : comme on l'a déjà vu, dans certains contextes précis, ça sert de pouvoir annuler les options précédentes. Par exemple, si tu utilises une classe de document qui se croit maligne en paramétrant la mise en page et que tu veux changer la dite mise en page : tu peux annuler ses choix en utilisant l'option reset dans la commande `\geometry`.

showframe, pour sa part, permet de dessiner la mise en page obtenue, pour se faire une petite idée (elle trace quelques traits et cadres sur la première page du document). C'est mignon. À choisir, il vaut probablement mieux utiliser le package `layouts`.

Enfin, l'option pass n'a de sens que pour les gens qui ne veulent vraiment pas utiliser layouts : elle permet de dessiner des marges que geometry n'a *pas* calculées. Par exemple, tu utilises la classe **tralala** et tu veux savoir ce qu'elle fait comme mise en page, tu peux faire ça :

```
\documentclass{tralala}
\usepackage[pass,verbose,showframe]{geometry}
\begin{document}
Rien
\end{document}
```

Ça t'indiquera les paramètres choisis par la classe et ça te les dessinera en grandeur réelle sur la première page *sans* que geometry ne touche quoi que ce soit sur les marges (il n'applique même pas son propre réglage par défaut). Franchement, il vaut mieux faire appel à layouts pour ça.

## 4.17 Régler les marges à la main : le package vmargin

Nous devons à la générosité de Volker Kuhlmann [111], un joli package du nom de vmargin qui permet sans trop se fouler de changer les marges d'un document. Pas aussi puissant que geometry, mais plus léger.

Ce package traite les deux problèmes classiques : la taille du papier et la taille des marges.

La première commande vraiment utile est celle permettant de fixer la taille du papier : \setpapersize. Elle prend deux paramètres, dont le premier est optionnel. Le premier indique l'orientation et peut valoir portrait (par défaut) ou landscape. Le deuxième argument peut-être le nom de n'importe quel format de papier de A0 à A9, de B0 à B9 et de C0 à C9, ainsi que USletter, USlegal, et USexecutive. Un nom particulier, custom permet de spécifier à la main une taille de papier (dans ce cas là, la commande prend deux arguments de plus). Quelques exemples vite fait :

```
\setpapersize{A4}
\setpapersize[landscape]{A4}
\setpapersize[landscape]{custom}{7cm}{15cm}
```

Le package stocke dans \PaperWidth la largeur de ton papier et dans \PaperHeight sa hauteur. De plus, il crée le booléen[120] \ifLandscape qui peut être utilisé par la suite :

```
Mon papier fait
\ifLandscape
  \the\PaperHeight sur \the\PaperWidth (paysage)
\else
  \the\PaperWidth sur \the\PaperHeight (portrait)
\fi
```

Comme le suggère cet exemple, le package échange les valeurs de la hauteur et de la largeur du papier quand tu es en mode paysage, *i.e.* à part le texte entre parenthèses, les deux lignes produiront toujours le même résultat puisque le package aura, en mode paysage, échangé les deux grandeurs.

[120] C'est bien un booléen au sens de TeX, en ce sens que tu peux jouer avec \Landscapetrue et \Landscapefalse, simplement, ça ne changera que la valeur du booléen, mais pas l'orientation de la page. Tu aurais du t'en douter. Pour plus d'infos sur les booléens de TeX, reporte-toi au chapitre neuvième page 571.

| Commande | Options | Description |
|---|---|---|
| \setmargins | ltwh1234 | Définit entièrement la géométrie de la page et des en-têtes en mode *largeur-hauteur*. |
| \setmarginsrb | ltrb1234 | Idem en mode *marges*. |
| \setmargnohf | ltwh | Définit une page sans en-tête ni pied de page en mode *largeur-hauteur*. |
| \setmargnohfrb | ltrb | Idem en mode *marge*. |
| \setmarg | ltwh | Définit les marges sans toucher ni à l'en-tête ni au pied de page, en mode *largeur-hauteur*. |
| \setmargrb | ltrb | Idem en mode *marge*. |

Le package `vmargin` fonctionne en deux modes. Dans le premier, *largeur-hauteur*, tu spécifies les marges de gauche et du haut, puis la largeur et la hauteur de ton texte, comme LaTeX l'attend spontanément. Dans le mode *marge*, tu spécifies les quatre marges (gauche, haut, droite, bas) et il calculera tout seul la largeur et la hauteur du texte en fonction de la taille du papier.

Les options sont les suivantes :

- `l` (left) marge gauche
- `t` (top) marge du haut
- `r` (right) marge droite
- `b` (bottom) marge du bas
- `w` (width) largeur du texte
- `h` (height) hauteur du texte
- `1` hauteur de l'en-tête
- `2` blanc entre l'en-tête et le texte
- `3` hauteur du pied de page (LaTeX ne s'en sert pas en général, mais d'anciennes versions s'en servaient)
- `4` distance du bas du texte au bas du pied de page

Tab. 4.28: Commandes du package `vmargin`

| Option | Description |
|---|---|
| shiftmargins | Laisse `\hoffset` et `\voffset` à leur valeure initiale et décrémente donc les marges d'un pouce pour en tenir compte. |
| portrait | Portrait. |
| landscape | Paysage. |
| nohf | Pas d'en-tête ni de pied de page. |

Tous les formats de papier connus du package sont admis comme option.

Tab. 4.29: Options de chargement du package `vmargin`

---

Pour positionner les marges ? Eh bien voici la méthode la plus simple qui soit, une commande (`\setmarginsrb`) et huit paramètres :

```
\setmarginsrb{l}{t}{r}{b}{1}{2}{3}{4}
```

Pour la signification des paramètres, reporte toi au tableau 4.28.

Un petit exemple vite fait :

```
\setmarginsrb{1cm}{2cm}{3cm}{4cm}{12pt}{13pt}{12pt}{24pt}
```

qui se lit comme ça[121] : marge gauche (intérieure) d'un centimètre, marge haute de deux centimètres, marge droite (extérieure) de trois centimètres, marge basse de quatre centimètres

[121] Les valeurs ne sont pas choisies pour fournir un résultat intelligent, mais pour que tu puisses lire l'exemple. Dans un vrai document, je te recommande de choisir des valeurs moins idiotes.

| | | | |
|---|---|---|---|
| a4paper | a5paper | b5paper | letterpaper |
| legalpaper | executivepaper | landscape | |

L'option landscape, qui doit apparaître après celle indiquant la taille du papier, ne fait qu'échanger la largeur et la hauteur de celui-ci.

TAB. 4.30: Options du package `anysize` (tailles de papier)

(facile de se souvenir de l'ordre : commence à gauche et tourne dans le sens des aiguilles d'une montre), l'en-tête fait douze points de haut (une ligne, en gros), il y a un blanc de treize points entre l'en-tête et le texte, le pied de page fait douze points de haut[122] et le bas du pied de page est à vingt-quatre points du bas du texte (deux lignes, donc, une ligne de séparation entre le texte et le pied de page).

D'autres commandes existent pour définir les marges, qui fonctionnent à peu près sur le même principe. Elles sont toutes décrites dans le tableau 4.28. Certaines permettent de dire qu'il n'y a ni en-têtes ni pied de page sans avoir à indiquer quatre fois `0cm` pour les quatres derniers paramètres, on simplement qu'on ne souhaite pas modifier la place qui leur est réservée.

Ainsi, si tu veux juste faire du A4, centré, avec des marges de trois centimètres, sans toucher à ce que LaTeX fait par défaut avec les en-têtes et pieds de page (c'est pas si mal, après tout), ça donne ça :

```
\setpapersize{A4}
\setmargrb{3cm}{3cm}{3cm}{3cm}
```

Le package `vmargin` peut recevoir les options indiquées au tableau 4.29 page précédente lors de son chargement. Elles permettent en particulier de définir la taille du papier, mais ne permettent pas de définir la taille des marges. L'une d'entre elles mérite une petite explication : shiftmargins. Spontanément, TeX décale tout le document d'un pouce vers le bas et d'un pouce vers la droite (ce que `layouts` indique avec un petit rond en haut à gauche de ses dessins). TeX prévoit les deux dimensions `\hoffset` et `\voffset`, qui par défaut sont nulles, pour décaler le document par rapport à ce point de référence. Si on ne lui dit rien, le package `vmargin` va positionner des deux dimensions à `-1in`, annulant ainsi le décalage par défaut de TeX. Comme ça ses calculs s'en trouvent simplifiés : la marge du haut, par exemple, est directement celle que tu as indiquée, et non pas celle que tu as indiquée moins un pouce.

L'option shiftmargins lui dit de *ne pas* faire cette manipulation sur `\hoffset` et `\voffset`. Ça peut s'avérer pratique dans certains cas, par exemple si tu as pris l'habitude d'utiliser cette fonctionnalité de TeX pour autre chose (par exemple pour centrer un petit document sur une grande feuille).

Pour positionner les marges, c'est clairement mon package préféré : simple et efficace.

## 4.18 La méthode rustique : le package `anysize`

C'est un package microscopique, et pourtant ils s'y sont mis à trois : Michael SALZENBERG, Thomas ESSER[123] et Dirk HILLBRECHT [186].

[122]Dans les versions que tu croiseras de LaTeX, qui datent d'après 1994, ce paramètre n'est pas pris en compte. Dans les versions plus anciennes, il existait réellement et était superbement ignoré.

[123]Il a fait mieux depuis : il est principalement connu pour être l'auteur et le mainteneur de teTeX (le « te » signifiant Thomas ESSER), qui est la distribution de TeX qui fait référence.

Ultra simple. Le package ne propose que deux commandes. La première permet de donner la taille du papier :

```
\papersize{21cm}{29.7cm}
```

La deuxième permet de donner les marges dans l'ordre gauche, droite, haut, et bas :

```
\marginsize{3cm}{2cm}{2cm}{2cm}
```

Les valeurs que je viens de te donner dans ces deux exemples sont les valeurs prises par défaut par le package.

De plus, il reconnaît les options du tableau 4.30 page précédente pour fixer la taille du papier. Une autre option existe, un peu curieuse, qui s'appelle germanpar qui supprime l'indentation des paragraphes et ajoute du blanc (à peu près une demi-ligne) entre les paragraphes.

## 4.19 Reliquat historique : le package `a4`

Le package a4 qui est un des grands standard pour LaTeX a été réécrit pour LaTeX $2_\varepsilon$ par Nico Poppelier et Johannes Braams [175]. Il a l'air assez joli et semble prévoir tous les cas. On peut lui passer une option widemargins qui permet d'obtenir de grandes marges.

Fais gaffe, les marges sont déjà trop grandes à mon goût[124].

Ce package, qui avait historiquement un intérêt (dans les années 80, il permettait de produire avec LaTeX des documents pour du papier A4), n'en a vraiment plus aucun aujourd'hui. Il vaut mieux, de loin, utiliser quelque chose comme :

```
\usepackage[a4paper]{geometry}
```

## 4.20 Approche non-numérique : le package `rmpage`

Ce package, que nous devons à Rowland McDonnel [131], aborde le problème sous un angle complètement différent. Les packages qu'on a vus jusque là, comme `geometry` ou `vmargin`, partent du principe que, quand tu cherches à faire ta mise en page, tu te dis un truc du genre « Dieu que j'aimerais avoir une marge de 13,4 millimètres coté intérieur en réservant 9 millimètres pour la reliure ». `rmpage`, pour sa part, considère que ton problème est « Moui, faut que j'en fasse 20 pages, j'en suis à 21, faudrait élargir un poil le texte ».

Sur le principe, c'est assez intéressant : à aucun moment, à moins de faire de très gros efforts, tu ne préciseras une dimension explicite. Tu diras toujours des choses comme « ultra large » ou « un poil plus à gauche », etc. Seul défaut, tu devras le dire en anglais. Ça n'a l'air de rien, mais ça complique. Comme en plus, l'auteur du package utilise un anglais peu classique (sans être véritablement de l'argot, c'est au moins de l'anglais familier que je n'ai pas appris à l'école), même pour quelqu'un qui parle un peu anglais, c'est une source d'ennui[125].

---

[124]Le choix de marges aussi grandes est dû à un souci, louable, de lisibilité. Il a été montré par certaines études scientifiques qu'au delà d'une certaine longueur de ligne, un texte devenait plus difficile et plus fatigant à lire. La longueur optimale étant aux alentours de 2,5 alphabets dans la fonte concernée. Sur une page de format A4, 2,5 alphabets en 10 points, cela représente une faible largeur de texte, aussi les marges sont elles très grandes.

Tu remarqueras d'ailleurs que généralement, les livres ne sont pas publiés en format A4, ce qui justifie le fait que leurs marges soient plus réduites.

[125]Le package `rmpage-fr`, dont je te parlerai un peu plus loin, permet de traduire les options de `rmpage`.

```
* \documentclass[a4paper,twoside]
*                     {article}
* \usepackage{rmpage}
* \usepackage{layouts}
  \currentpage
  \drawpage
```

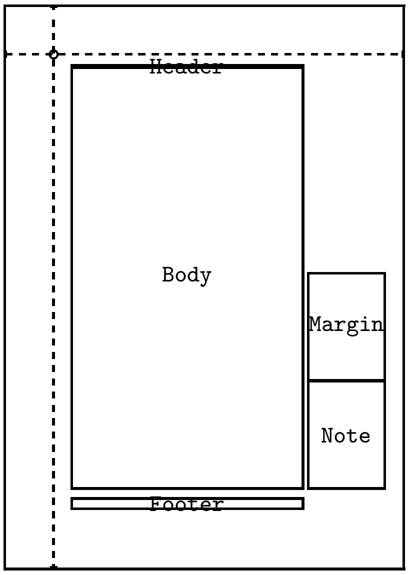



FIG. 4.40: Mise en page par défaut de `rmpage` sur du papier A4

### 4.20.1 La taille du bloc de texte

La première chose à comprendre en ce qui concerne `rmpage`, c'est qu'il a sa propre notion de l'esthétique et qu'il considère qu'il y a une mise en page donnée qui est la bonne. Toutes ses options tendent à exprimer à quel point tu n'es pas d'accord avec lui. Cette mise en page est celle de l'exemple de la figure 4.40. Ses choix sont assez raisonnables pour un document de base. À noter cependant qu'il suppose que ton document n'utilise pas d'en-têtes (après tout, c'est le comportement par défaut de LaTeX, si on ne lui dit rien, puisqu'il est en style de page plain).

Il détermine cette mise en page selon un certain nombre de critères : il réserve une zone, tout autour de la feuille, qu'il considère comme ne pouvant pas être imprimée[126] ; ensuite il estime la longueur d'une ligne de texte idéale pour la fonte que tu utilises (il existe à ce sujet quelques études savantes) ; ensuite il considère que, sur ta page, une zone de texte occupant 71,38 %[127] de la largeur serait du plus bel effet ; ensuite, de ces deux valeurs, il retient la plus petite. C'est cette valeur qu'il considère être la largeur *normale* de ton texte.

Il opère selon des critères un poil moins savants pour déterminer la longueur normale de ton texte (la longueur s'entend, pour lui, incluant l'en-tête et le pied de page). Pour lui, cette longueur idéale est de 78,7 % de la longueur de ton papier pour toutes les classes de document, sauf pour la classe **slides**, où il prévoit 85,96 %.

Les options du tableau 4.31 page suivante te permettront de lui dire de s'écarter plus ou moins de la taille de la zone de texte qu'il a jugée comme étant optimale. Les différentes options suivent une progression géometrique[128] et non pas linéaire. Chaque option ajoute à

[126]Toutes les imprimantes ont leur propre zone qui ne peut pas être imprimée. La zone qu'il réserve spontanément, de quelques millimètres, est raisonnable pour la pluspart des cas.
[127]Ne me demande pas d'où il sort ça. Ce n'est pas le nombre d'or (1,618), ni la moitié de celui-ci (0,809), ni la célèbre valeur $\frac{\sqrt{2}}{2}$ (0.707). Bref, ça devait être le bidule tel que ça plaise à l'auteur du package.
[128]Pour ceux qui ne savent pas ce que c'est qu'une progression géométrique par rapport à une progression linéaire, c'est assez simple à expliquer. Une progression linéaire, c'est un truc où on rajoute toujours la même quantité. Par exemple, une progression linéaire aurait été d'ajouter toujours 5 lignes pour chaque option. La

Du plus petit au plus grand :

| Largeur | Longueur |
|---|---|
| narrowest | shortest |
| narrower | shorter |
| narrow | short |
| narrowish | shortish |
| normalwidth | normallength |
| widish | longish |
| wide | long |
| wider | longer |
| widest | longest |
| **En plus/moins** | |
| touchwider | touchlonger |
| t@uchwider[*] | t@uchlonger[*] |
| touchnarrower | touchshorter |
| t@uchnarrower[*] | t@uchshorter[*] |

Options spécifiques :

| Option | Description |
|---|---|
| stdwidth | Comme LaTeX tout seul. |
| fullwidth | Toute la page. |
| oneinchmargins | Somme des marges d'un pouce. |
| halfinchmargins | Somme des marges d'un demi pouce. |
| stdlength | Comme LaTeX tout seul. |
| fulllength | Toute la page. |
| characterwidthset | La fonte fixe la largeur[1]. |
| paperwidthset | Le papier fixe la largeur[1]. |
| bothwidthset[†] | Le minimum des deux fixe la largeur[1]. |

[*] Cette option ne peut être utilisée que dans une classe de document ou un autre package.

[†] Cette option est prise par défaut.

[1] Le package commence par calculer la largeur qu'il juge être optimale par rapport à la fonte de ton document ; ensuite, il regarde ce que ta feuille de papier peut supporter comme largeur en conservant des marges raisonnables (il prend une certaine fraction de la largeur du papier). Par défaut, il prend, comme définition de normalwidth, le minimum de ces deux grandeurs, l'équivalent de bothwidthset. characterwidthset lui indique de prendre uniquement ce qu'indique la fonte (sans dépasser de la zone imprimable du papier). paperwidthset lui dit de ne pas prendre en compte la fonte.

Tab. 4.31: Options de `rmpage` pour la taille des marges

peu près 25 % en longueur et à peu près 7 % en largeur, sachant que l'option la plus petite correspond à un quart de la page approximativement (moitié de la largeur sur moitié de la longueur) et que la plus grande des options correspond à quasiment toute la page.

À chaque fois que `rmpage` propose une série d'options, comme par exemple celle qui va de shortest à longest, les options suivent une échelle graduée de 3 en 3. Les options comme touchlonger et touchshorter permettent de se décaler d'un cran sur cette échelle vers le haut ou vers le bas, ce qui permet de faire un réglage fin.

L'exemple de la figure 4.41 page suivante te présente la page la plus petite que `rmpage` produira pour du papier A4 avec la fonte par défaut de LaTeX (Computer Modern en 10 points), ça correspond à l'appel des otpions shortest et narrowest. L'exemple de la figure 4.42 page suivante te présente, dans les mêmes conditions, la page la plus grande qu'il propose sans qu'on le force, avec les options widest et longest. Enfin, la figure 4.43 page 277 te montre ce qu'on obtient quand on demande explicitement à `rmpage` de prendre toute la page, avec les marges les plus réduites possibles, avec les options fullwidth et fulllength.

```
* \usepackage[shortest,
*             narrowest]{rmpage}
  \currentpage
  \drawpage
```

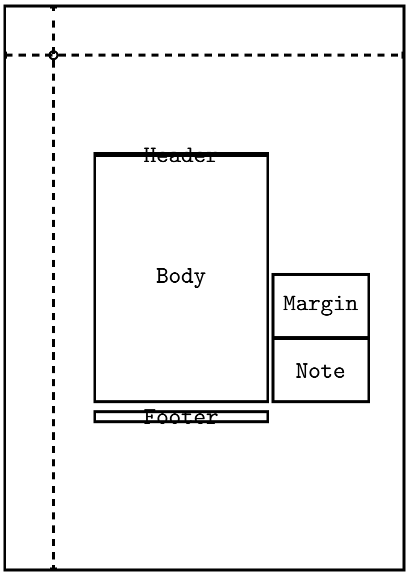



FIG. 4.41: La page la plus petite pour `rmpage`

```
* \usepackage[longest,widest]{rmpage}
  \currentpage
  \drawpage
```

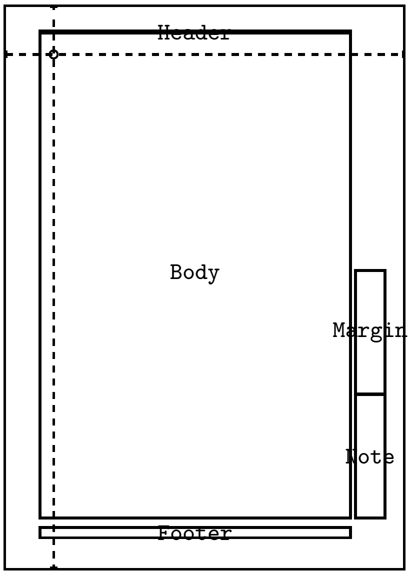



FIG. 4.42: La page la plus grande pour `rmpage`

```
* \usepackage[fullwidth,
*             fulllength]{rmpage}
  \currentpage
  \drawpage
```

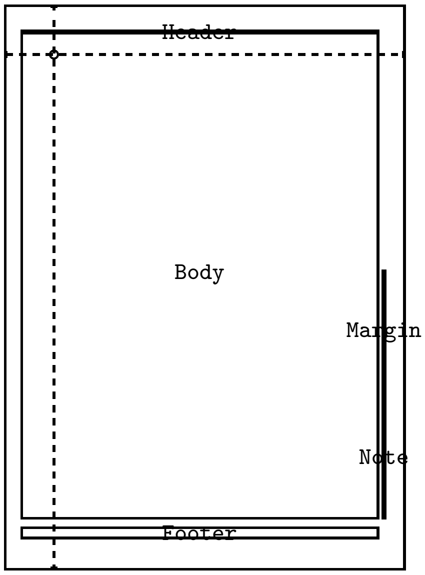



FIG. 4.43: Une page pleine pour `rmpage`

Du plus petit au plus grand :

| Séparation | |
|---|---|
| **en-tête** | **pied** |
| mostheadsep | mostfootskip |
| moreheadsep | morefootskip |
| moreishheadsep | moreishfootskip |
| normalheadsep | normalfootskip |
| lessishheadsep | lessishheadsep |
| lessheadsep | lessfootskip |
| leastheadsep | leastfootskip |
| **En plus/moins** | |
| touchmoreheadsep | touchmorefootskip |
| t@uchmoreheadsep* | t@uchmorefootskip* |
| touchlessheadsep | touchlessfootskip |
| t@uchlessheadsep* | t@uchlessfootskip* |

Options spécifiques :

| Option | Description |
|---|---|
| headers | Réserve de la place pour l'en-tête. |
| noheaders† | Pas d'en-tête. |
| footers† | Réserve de la place pour le pied de page. |
| nofooters | Pas de pied de page. |

*Cette option ne peut être utilisée que dans une classe de document ou un autre package.
†Cette option est prise par défaut.

TAB. 4.32: Options de `rmpage` pour les en-têtes et les pieds de page

---

### 4.20.2 Les en-tête et pied de page

Par défaut, comme on l'a vu, `rmpage` ne réserve pas de place pour l'en-tête, ce qui correspond à l'option noheaders. Pour lui dire de réserver de la place pour l'en-tête, il faut

progression géométrique, c'est d'ajouter toujours le même pourcentage. Par exemple, ajouter toujours 5 % en hauteur. C'est très différent. Sur une page de 100 lignes, ajouter 5 % c'est ajouter 5 lignes, alors que sur une page énorme de 1000 lignes, c'est en ajouter 50. Une progression géométrique, c'est une progression qui avance de plus en plus vite.

Du plus petit au plus grand :

| Horizontal | Vertical |
|---|---|
| | lowest |
| leastoffset | lower |
| lessoffset | low |
| lessishoffset | lowish |
| normaloffset | normalaltitude |
| moreishoffset | highish |
| moreoffset | high |
| mostoffset | higher |
| | highest |
| **En plus/moins** | |
| touchmoreoffset | touchhigher |
| t@uchmoreoffset* | t@uchhigher* |
| touchlessoffset | touchlower |
| t@uchlessoffset* | t@uchlower* |

Options spécifiques :

| Option | Description |
|---|---|
| stdmargins† | Marge extérieure la plus grande. |
| notstdmargins | Marge intérieure la plus grande. |
| centre | Centre le bloc de texte. |
| center | Idem. |
| notcentre | Ne centre pas le bloc de texte. |
| notcenter† | Ne centre pas le bloc de texte. |

*Cette option ne peut être utilisée que dans une classe de document ou un autre package.

†Cette option est prise par défaut.

À noter : stdmargins et notcenter sont pris par défaut, produisant des marges relativement classiques. Si tu précises leastoffset ton texte se retrouve centré. Si tu ajoutes encore touchlessoffset, tu obtiens un offset *négatif*, et donc un léger décalage *dans l'autre sens*.

Quand tu augmentes l'offset, tu accentues le décentrage dans le sens que tu as demandé (stdmargins ou notstdmargins).

TAB. 4.33: Options de `rmpage` pour le centrage de la zone de texte

utiliser headers. L'ensemble des options du tableau 4.32 page précédente servent à règler la présence ou l'absence, ainsi que la place réservée pour les en-têtes et/ou les pieds de page.

Tu remarqueras qu'il y a une série d'options permettant de préciser l'espacement entre l'en-tête et le corps du texte, mais rien pour spécifier la hauter du texte d'en-tête. L'auteur du package considère que c'est à toi de le préciser ou plus exactement que c'est ta classe de document qui devrait le faire. Il propose cette approche-là pour préciser la hauteur de l'en-tête :

```
\providecommand{\RMheadheight}{32pt}
\usepackage[headers]{rmpage}
```

Une solution dérivée de celle-ci est de donner la taille de ton en-tête en nombre de lignes. Normalement, une ligne, ça fait `\baselineskip` en hauteur, du coup, tu dois pouvoir faire ça :

```
\providecommand{\RMheadheight}{5\baselineskip}
\usepackage[headers]{rmpage}
```

pour préciser que ton en-tête fait cinq lignes.

On sait donc maintenant comment dire à `rmpage` comment choisir la taille de la zone de texte et comment choisir la taille des en-têtes et des pieds de page. Tout cela forme le bloc de texte. Il nous reste à savoir comment le centrer dans la page.

```
* \usepackage[narrowest,
*             largebasemaxmparwidth,
*             moreoffset]{rmpage}
  \currentpage
  \drawpage
```

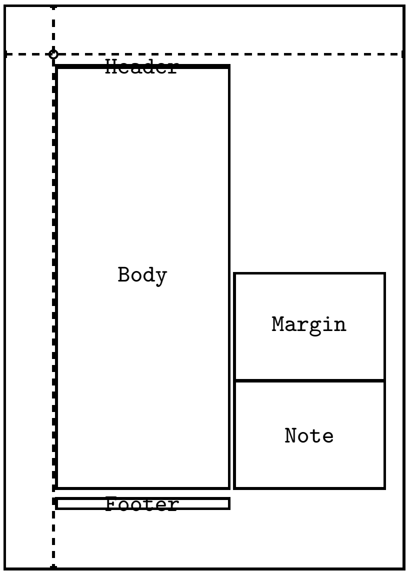



FIG. 4.44: Exemple de document avec beaucoup de notes en marge, avec rmpage

---

### 4.20.3 Le centrage du bloc de texte

Par défaut, il considère que tu veux un bloc de texte légèrement décentré sur la gauche (ce qui est classique) et que la marge gauche et la marge droite ont un rapport de deux pour trois, ce qui est classique également. D'ailleurs, si tu utilises l'option oneinchmargins, qui lui précise en fait la largeur du texte, puisque l'option signifie « marges d'un pouce en moyenne », il va répartir les deux pouces de marge entre gauche et droite en respectant le rapport de deux à trois : il mettra 0,8 pouce sur la gauche et 1,2 pouces sur la droite.

Les options du tableau 4.33 page ci-contre servent à changer le centrage du bloc de texte dans la page. Les options parlant de l'offset servent à lui indiquer à quel point il faut accentuer la dissymétrie entre les marges et les options portant sur l'altitude (higher/lower) indiquent le centrage vertical voulu (par défaut, c'est à peu près centré verticalement).

Les options stdmargins et notstdmargins lui indiquent dans quel sens faire le décentrage (comme d'habitude, c'est-à-dire sur la gauche, ou au contraire, sur la droite). L'option center, qui est équivalente à leastoffset indique de centrer le bloc de texte horizontalement.

### 4.20.4 Les notes en marge

Une fois que la taille du bloc de texte est déterminée et que son centrage est choisi, le package peut déterminer les quatre marges que tu souhaites appliquer. Dans la marge extérieure, il va placer, si ton document en contient, les notes en marge. Pour déterminer leur largeur, c'est assez simple : il s'est fixé un maximum qu'il juge raisonnable et tant que ce maximum n'est pas atteint, les notes occupent toute la largeur de cette marge, sauf évidemment un blanc entre le texte et les marges et un blanc sur le bord du papier.

Le blanc qui sépare le texte des notes en marge, c'est ce que le package appelle mparsep, le blanc à réserver entre les notes en marge et le bord du papier, c'est ce que le package appelle mparclearance et la largeur maximale des notes en marge, c'est ce que le package appelle maxmparwidth.

Du plus petit au plus grand :

| | Notes en marge | |
|---|---|---|
| **séparation** | **réservation** | **largeur max.** |
| leastmparsep | leastmparclearance | leastmaxmparwidth |
| lessmparsep | lessmparclearance | lessmaxmparwidth |
| lessishmparsep | lessishmparclearance | lessishmaxmparwidth |
| normalmparsep | normalmparclearance | normalmaxmparwidth |
| moreishmparsep | moreishmparclearance | moreishmaxmparwidth |
| moremparsep | moremparclearance | moremaxmparwidth |
| mostmparsep | mostmparclearance | mostmaxmparwidth |
| | **En plus/moins** | |
| touchmoremparsep | touchmoremparclearance | touchmoremaxmparwidth |
| t@uchmoremparsep* | t@uchmoremparclearance* | t@uchmoremaxmparwidth* |
| touchlessmparsep | touchlessmparclearance | touchlessmaxmparwidth |
| t@uchlessmparsep* | t@uchlessmparclearance* | t@uchlessmaxmparwidth* |

Options spécifiques :

| Option | Description |
|---|---|
| normalbasemaxmparwidth† | Les largeurs max. des notes en marge sont celles par défaut de `rmpage`. |
| largebasemaxmparwidth | Les largeurs max. des notes en marge proposées sont doublées par rapport à la normale. |
| normalbasemparclear† | Les valeurs de réservation entre les notes et le bord du papier sont celles par défaut de `rmpage`. |
| largebasemparclear | Les valeurs de réservation sont doublées. |

*Cette option ne peut être utilisée que dans une classe de document ou un autre package.
†Cette option est prise par défaut.

Le package `rmpage` choisit les largeurs des notes en marge en fonction d'un certain nombre de critères. Le premier, mparsep est le blanc qui sépare le texte des notes en marge. Le second, mparclearance est la réservation, le blanc qu'il doit absolument y avoir au minimum entre les notes en marge et le bord de la feuille de papier. Enfin, il y a la largeur maximale pour les notes en marge, maxmparwidth. Il fera des notes en marge aussi larges que possible, tout en restant dans ces trois contraintes.

TAB. 4.34: Options de `rmpage` pour les notes en marge

Les options du tableau 4.34 te permettent de régler tout ça. La figure 4.44 page précédente te donne un exemple d'une mise en page dans laquelle on a réservé énormément de place pour les notes en marge.

### 4.20.5 Plusieurs colonnes de texte

Tu l'as vu tout à l'heure, `rmpage` choisit, par défaut, de ne pas avoir des lignes trop longues. Pour ce faire, il se base sur la largeur de la fonte que tu vas utiliser pour ton document. En fait, il choisit comme largeur par défaut pour ton texte le minimum entre cette largeur supposée optimale et un pourcentage donné de la largeur de ton papier.

Du plus petit au plus grand :

| Séparation | Nombre |
|---|---|
| leastcolsep | onecolumnwidth |
| lesscolsep | twocolumnwidth |
| lessishcolsep | threecolumnwidth |
| normalcolsep | fourcolumnwidth |
| moreishcolsep | fivecolumnwidth |
| morecolsep | sixcolumnwidth |
| mostcolsep | sevencolumnwidth |
| | eightcolumnwidth |
| | ninecolumnwidth |
| | tencolumnwidth |
| **En plus/moins** | |
| touchmorecolsep | |
| t@uchmorecolsep* | |
| touchlesscolsep | |
| t@uchlesscolsep* | |

Options spécifiques :

| Option | Description |
|---|---|
| adaptativecolumnsep | (1) |
| noadaptativecolumnsep† | (2) |
| onecolumn† | (3) |
| twocolumn | (4) |

(1) Calcule l'espacement entre les colonnes d'après la largeur de la fonte utilisée.
(2) Utilise les options standard pour déterminer l'espacement entre les colonnes.
(3) Cette option est destinée à `\documentclass` et est l'option standard de LaTeX.
(4) Option également destinée à `\documentclass`.
Les autres options, XXXcolumnwidth, sont réservées à `rmpage` pour lui indiquer que tu vas faire plusieurs colonnes avec un package du style de `multicol`.

*Cette option ne peut être utilisée que dans une classe de document ou un autre package.
†Cette option est prise par défaut.

`rmpage` fait des calculs savants en se basant sur la largeur de la fonte que tu utilises pour déterminer la largeur qu'il jugera optimale de ton texte (celle proposée par normalwidth). Si tu lui indiques que tu vas faire plusieurs colonnes, il réduira les marges en conséquences.

TAB. 4.35: Options de `rmpage` pour le multi-colonnage

Bien entendu, si ton texte est en deux colonnes, il faut qu'il compte le minimum entre deux fois la largeur optimale (plus l'espacement entre les deux colonnes), au lieu de simplement la largeur optimale. Les options indiquées au tableau 4.35 permettent de paramètrer tout ça.

Notes bien que les options onecolumn et twocolumn sont à passer à `\documentclass`, parce que les classes standard de LaTeX savent faire des documents en deux colonnes. Les autres options, celles à propos de columnwidth, n'intéressent pas les classes de document. Elles servent juste à indiquer au package `rmpage` que tu vas mettre ton texte en forme en plusieurs colonnes, pour qu'il en tienne compte. Ça te sera utile, par exemple, si tu utilises le package `multicol` pour mettre en forme ton document.

En règle générale, il vaut mieux prendre le nombre de colonnes *dominant*, c'est-à-dire celui qui représente la plus grande partie du document, comme option du package. Tu peux encore, bien entendu, influer sur les marges avec les options classiques portant sur la largeur de la zone de texte.

### 4.20.6 La taille du papier

Un autre des paramètres dont on a parlé à plusieurs reprises à propos de ce package, c'est la taille du papier. Sur ce sujet-là, il est beaucoup plus savant que LaTeX, il connaît tous les formats du tableau 4.36 page suivante.

Par contre, il y a une ombre au tableau : comme je te l'ai indiqué dans le tableau 4.36, le package se trompe sur certaines tailles de papier. Ce n'est pas bien grave, parce que ce ne sont pas des tailles de papier trop courantes.

| **Option** | **Taille** |
|---|---|
| letterpaper | 8,5 in × 11,0 in |
| legalpaper | 8,5 in × 14,0 in |
| executivepaper | 8,5 in × 10,5 in |
| dlpaper | 11,0 cm × 22,0 cm |
| letter4paper[1] | 21,0 cm × 8,5 in |
| a0paper | 84,1 cm × 118,9 cm |
| a1paper | 59,4 cm × 84,1 cm |
| a2paper | 42,0 cm × 59,4 cm |
| a3paper | 29,7 cm × 42,0 cm |
| a4paper | 21,0 cm × 29,7 cm |
| a5paper | 14,8 cm × 21,0 cm |
| a6paper | 10,5 cm × 14,8 cm |
| a7paper | 7,4 cm × 10,5 cm |
| a8paper | 5,2 cm × 7,4 cm |
| a9paper | 3,7 cm × 5,2 cm |
| a10paper | 2,6 cm × 3,7 cm |
| b0paper | 100,0 cm × 141,4 cm |
| b1paper | 70,7 cm × 100,0 cm |
| b2paper | 50,0 cm × 70,7 cm |
| b3paper | 35,3 cm × 50,0 cm |
| b4paper | 25,0 cm × 35,3 cm |
| b5paper | 17,6 cm × 25,0 cm |
| b6paper | 12,5 cm × 17,6 cm |
| b7paper | 8,8 cm × 12,5 cm |
| b8paper | 6,2 cm × 8,8 cm |
| b9paper | 4,4 cm × 6,2 cm |
| b10paper | 3,1 cm × 4,4 cm |
| c0paper | 91,7 cm × 129,7 cm |
| c1paper | 64,8 cm × 91,7 cm |
| c2paper | 45,8 cm × 64,8 cm |
| c3paper | 32,4 cm × 45,8 cm |
| c4paper | 22,9 cm × 32,4 cm |
| c5paper | 16,2 cm × 22,9 cm |
| c6paper | 11,4 cm × 16,2 cm |
| c7paper | 8,1 cm × 11,4 cm |
| c7/6paper | 8,1 cm × 16,2 cm |

| **Option** | **Taille** |
|---|---|
| no10envelopepaper | 4,12 in × 9,5 in |
| bspopseedenvelopepaper[2] | 10,2 cm × 15,2 cm |
| bspopnonisoenvelopepaper[2] | 10,2 cm × 22,9 cm |
| bsbrochureenvelopepaper[2] | 17,8 cm × 25,4 cm |
| bslegalenvelopepaper[2] | 21,6 cm × 27,0 cm |
| bslargelegalenvelopepaper[2] | 12,7 cm × 30,5 cm |
| bscalendarenvelopepaper[2] | 25,4 cm × 38,1 cm |
| foolscappaper[3] | 8,5 in × 13,5 in |
| foolscapfoliopaper[3] | 8,5 in × 13,5 in |
| foolscapquartopaper | 6,75 in × 8,5 in |
| foolscapoctavopaper | 4,25 in × 6,75 in |
| crownfoliopaper | 10,0 in × 15,0 in |
| crownquartopaper | 7,5 in × 10,0 in |
| crownoctavopaper | 5,0 in × 7,5 in |
| royalfoliopaper | 12,5 in × 20,0 in |
| royalquartopaper | 10,0 in × 12,5 in |
| royaloctavopaper | 6,25 in × 10,0 in |
| imperialfoliopaper | 15,5 in × 22,0 in |
| imperialquartopaper[4] | 11,0 in × 15,0 in |
| imperialoctavopaper | 7,5 in × 11,0 in |
| largecrownoctavopaper | 5,25 in × 8,0 in |
| demyoquartopaper | 8,75 in × 11,25 in |
| demyoctavopaper | 5,625 in × 8,75 in |
| mediumquartopaper | 9,5 in × 12,0 in |
| mediumoctavopaper | 6,0 in × 9,5 in |
| ra0paper | 96,0 cm × 127,0 cm |
| ra1paper[5] | 96,0 cm × 127,0 cm |
| ra2paper[5] | 96,0 cm × 127,0 cm |
| sra0paper | 90,0 cm × 128,0 cm |
| sra1paper[6] | 84,0 cm × 90,0 cm |
| sra2paper | 45,0 cm × 64,0 cm |
| metricdoublecrownpaper | 50,5 cm × 77,0 cm |
| metricquadcrownpaper | 77,0 cm × 101,0 cm |
| metriclargequadcrownpaper | 82,0 cm × 106,0 cm |
| metricquaddemypaper | 89,0 cm × 103,0 cm |
| metricsmallquadroyalpaper | 96,0 cm × 127,0 cm |

[1] Destiné à être un format bi-compatible pour a4paper et letter. Ceci dit, les dimensions ne correspondent pas à celles qui sont indiquées dans la documentation du package.
[2] Format d'enveloppe visiblement classique dans un pays qui n'est pas la France. Probablement l'Angleterre.
[3] Le format foolscapfoliopaper est supposé être du foolscappaper utilisé en *in-folio* (c'est-à-dire plié en deux), je ne comprends pas qu'ils soient tous les deux de la même taille. L'auteur du package confond peut-être *in-folio* et *in-plano*.
[4] Je ne comprends pas pourquoi l'*in-quarto* n'est pas la moitié de l'*in-folio* dans ce cas là.
[5] Je ne comprends pas pourquoi c'est pas divisé par deux...
[6] Devrait être 64 cm de large...

TAB. 4.36: Options de `rmpage` pour la taille du papier

| Chargement avec | sans | Description |
|---|---|---|
| cmrwidth† | loadcmr | Fontes Computer Modern. |
| avantwidth | loadavant | Avant Garde. Basé sur le package `avant`. |
| bookmanwidth | loadbookman | Bookman. Basé sur le package `bookman`. |
| chancerywidth | loadchancery | Zapf Chancery. Basé sur le package `chancery`. |
| helvetwidth | loadhelvet | Helvetica. Basé sur le package `helvet`. |
| newcentwidth | loadnewcent | New Century Schoolbook. Basé sur le package `newcent`. |
| palatinowidth | loadpalatino | Palatino. Basé sur le package `palatino`. |
| timeswidth | loadtimes | Times. Basé sur le package `times`. |
| utopiawidth | loadutopia | Utopia. Basé sur le package `utopia`. |
| lucasualwidth | loadlucasual | Lucida casual. Basé sur le package `lucasual`. |
| courierwidth | loadcourier | Courier. Basé sur aucun package. |
| concretewidth | loadconcrete | Concrete. Basé sur le package `beton`. |
| thisfountwidth | | Ne charge rien, suppose que la fonte en cours est la bonne. |

†Cette option est prise par défaut.

TAB. 4.37: Options de `rmpage` pour la fonte

Par contre, c'est signe que le package a été relativement peu utilisé et donc qu'il contient peut-être d'autres bugs.

Une taille de papier est un peu particulière : letter4paper qui se veut être un mix entre letterpaper qui est classique en Amérique du Nord, et a4paper qui est classique en Europe. Le A4 est plus étroit (de 5,9 millimètres), alors que le papier *letter* est plus court (de 17,6 millimètres). Ce que dit le package, c'est que cet hybride est donc de la largeur du A4 et de la longueur du *letter*, de manière à être compatible avec les deux. Mais l'auteur du package s'est trompé, dans la doc, comme dans le package : il a cru que le papier *letter* faisait 8,5 pouces de long (alors que c'est 11 pouces). Du coup il s'est trompé sur la taille de son papier letter4paper.

### 4.20.7 La taille de la fonte

Le dernier paramètre dont on a parlé à plusieurs reprises et qu'on n'a pas encore détaillé, c'est le fait que le package se base sur la largeur de ta fonte pour déduire une bonne longueur de ligne. On pourrait croire, de prime abord, qu'il fera ses petits calculs tout seul. En fait ce n'est pas toujours le cas. Certains packages de changement de fonte ne font leur changement qu'au début du document. Du coup, même si tu les appelles avant `rmpage`, celui-ci verra encore comme fonte courante la fonte par défaut de LaTeX (Computer Modern).

Du coup, il y a deux approches classiques : soit tu te débrouilles pour charger ta fonte toi même et c'est une des fontes que le package connaît et tu n'as qu'à lui dire laquelle et il fera ses calculs en fonction de ça. Ce sont les options XXXwidth. L'autre approche classique c'est carrément de lui déléguer le fait de charger le package en question. Ce sont les options loadXXX. Enfin, s'il ne connaît pas ta fonte, alors tu dois te débrouiller pour que celle-ci soit active *avant* le chargement de `rmpage` et lui préciser l'option thisfountwidth. À noter que l'option loadcmr ne fait rien, puisque LaTeX est par défaut en Computer Modern. Toutes ces options sont regroupées dans le tableau 4.37.

Le support du package `beton` est relativement complexe, en effet, celui-ci peut recevoir des options, par exemple pour changer l'interligne ou pas, ou pour faire des modifs sur le

| Option | Description |
|---|---|
| usdate | Modifie la commande `\today` pour faire une date à l'américaine. |
| ukdate | Idem pour faire une date dans un style anglais plus ancien. |
| verbose | Indique au package d'être spécialement bavard. |
| silent | Indique au package d'être spécialement silencieux. |
| errorshow | Comme silent. Ne parle qu'en cas d'erreur. |
| warningshow | Indique également les warnings. |
| infoshow | Raconte tout ce qu'il fait. |
| debugshow | Raconte également les détails, pour chercher d'éventuels bugs. |

TAB. 4.38: Autres options de `rmpage`

mode maths. `rmpage` accepte donc les trois options standard-baselineskip qui indique de ne pas agrandir l'interligne (`beton` agrandit l'interligne, par défaut, pour obtenir un résultat plus joli), oldstyle-equation-numbers qui lui dit de numéroter les équation avec `\textos` et concrete-maths qui indique à `beton` de modifier les fontes mathématiques également. Ces trois options son passées au package `beton` si tu utilises loadconcrete. Ça permet à `rmpage` de tenir compte du changement d'interligne (il en a besoin parce qu'il fait en sorte que la longueur de la page soit toujours un nombre entier de lignes).

### 4.20.8 Les autres options

Le package `rmpage` propose d'autres options, qui sont reprises dans le tableau 4.38. Elles ne sont pas spécialement intéressantes. D'autres options existent, que je ne mentionne pas ici, qui ne sont pas spécialement passionnantes non plus.

## 4.21 Une page la tête en bas : le package `rotpages`

Ce package, que nous devons à Sergio CALLEGARI [28], permet de faire tourner de 180 degrés le texte d'une page.

Comme ça, de prime abord, ça à l'air inutile. En fait, l'idée de l'auteur de ce package est de pouvoir présenter, dans un livre d'exercices, les solutions à l'envers pour qu'elles ne soient pas directement lisibles.

Le principe d'utilisation est assez simple. Tout ce qui est dans l'environnement `rotboxpages` sera écrit la tête en bas. Mais, une fois ce principe simple posé, il reste quelques limitations sévères.

La page suivante est un exemple d'utilisation de ce package : une page à l'envers, comme si elle contenait les réponses à des exercices, par exemple.

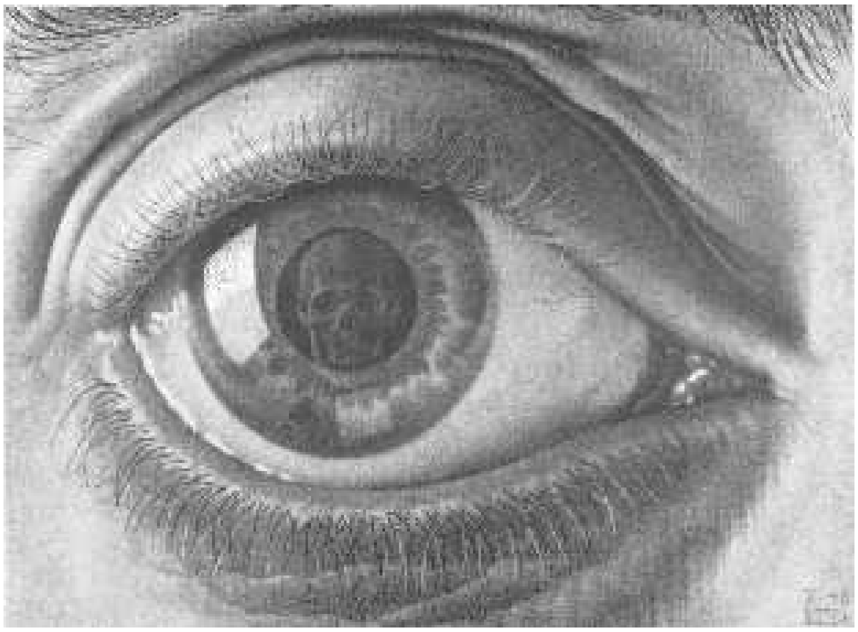


Fig. 4.45: Une figure qui va finir à l'envers

Cette page, présentant la solution à tous nos problèmes, est présentée avec le texte à l'envers[129].

Ici, un exemple de note en marge. Exemple

Tout cela s'obtient à partir du source suivant :

```
\begin{rotboxpages}
Cette page, pr\'esentant la solution \`a tous nos
probl\`emes, est pr\'esent\'ee avec le texte \`a
l'envers\footnote{Un exemple de note de pied de
  page}.

Ici, un exemple de note en marge\marginpar{Exemple}.

Tout cela s'obtient \`a partir du source suivant:
...

\end{rotboxpages}
```

[129] Un exemple de note de pied de page

Ces limitations viennent du mode de fonctionnement du package. En effet, il intervient lorsque LaTeX a complètement fini de mettre en forme la page, et qu'il s'apprête à la placer dans le document. La page ne comporte pas encore les en-têtes ni les pieds de page, mais elle comporte déjà les figures, tableaux et notes de bas de page. Toutes les pages qui seront préparées par LaTeX pendant qu'on est dans l'environnement `rotboxpages` seront mémorisées par le package, puis, une fois sorti de l'environnement, elles seront placées dans le document la tête en bas, *en ordre inverse* (pour qu'elles se lisent dans le bon ordre).

D'où une première limitation évidente : si tu prévois d'utiliser cette technique pour un très grand nombre de pages, par exemple pour mettre les solutions de tes 8.000 exercices à la fin d'un livre, tu peux t'attendre à ce que LaTeX râle parce qu'il n'a pas assez de mémoire pour tout retenir, ça se terminera par une message d'erreur peu clair de TeX te disant :

```
If you really absolutely need more capacity,
you can ask a wizard to enlarge me.
```

Une deuxième limitation est liée à la gestion des flottants par LaTeX. Quand le package intervient, pour mettre la page tête en bas, LaTeX a déjà terminé sa mise en page. Les flottants dont LaTeX aura jugé qu'ils auraient très bien été sur l'une de ces pages, il les y a mis. Même si ce sont des flottants qui viennent d'avant la partie qui est tête en bas.

Ainsi, si tu as 100 pages dans le bon sens, suivies de 10 pages à l'envers, et qu'à la fin des 100 pages dans le bon sens, il reste à LaTeX une figure à placer, il y a de bonnes chances pour qu'il place cette figure en haut de la prochaine page, pour lui, la $101^{\text{ième}}$. Du coup, la figure en question va se retrouver en haut de la première page à l'envers, et elle-même mise à l'envers, alors qu'elle n'était *pas* dans l'environnement `rotboxpages`. Cette page sera la première de la zone à l'envers, elle apparaîtra donc en $110^{\text{ième}}$ position, mettant la figure tant attendue 10 pages plus loin.

C'est ce cas que j'ai produit avec la figure 4.45 page précédente : le source de la figure *n'est pas* dams l'environnement `rotboxpages`, et pourtant la figure se retrouve dans la page à l'envers.

Le même problème va se poser si tu as des notes de pied de page très longues sur la dernière page à l'endroit. Si une de ces notes est tellement longue que LaTeX décide de la couper en deux, la deuxième partie va finir dans le texte à l'envers.

Un bon moyen de se prémunir de tout ça est de placer, tout simplement, un appel à `\clearpage` juste avant le début de l'environnement et juste avant la fin également. Cette commande change de page en prenant bien soin qu'il ne reste plus de flottant (figures, notes de pied de page, etc.) à traiter. Comme de toutes façons, la zone écrite à l'envers force déjà un changement de page, ça ne posera pas de problème.

Une troisième limitation vient du fait que le package intercepte les page *de texte*, c'est-à-dire les pages qui contiennent au moins un peu de texte, en plus des flottants. Du coup, les pages qui ne contiennent que des flottants (par exemple pour un très gros tableau) sont, soit perdues, soit mises à l'endroit à la fin. Contre ça, rien à faire.

Une dernière limitation porte sur les notes en marge. L'auteur du package n'en prend pas spécialement soin, et du coup, parfois ça marche, parfois ça ne marche pas. Mais, dans tous les cas, tu ne retrouveras pas tes notes dans la marge attendue ou alors par coup de chance : quand LaTeX met en forme la page (et les notes en marge avec), il pense qu'elle va apparaître tout de suite dans le document. Or, en fait, elle n'apparaîtra que plus tard (selon la longueur de la partie qui est à l'envers). Il n'a donc pas de moyen fiable, puisque le package ne fait pas le nécessaire, de savoir de quel côté il faut mettre les notes pour qu'elles continuent à apparaître comme dans le reste du document, dans la marge qui est prévue pour les recevoir.

En dehors de ces limitations, qui correspondent tout de même à des cas d'utilisation relativement peu fréquents, le package fonctionne très bien et il a le bon goût d'être facile à utiliser.

## 4.22 Obtenir du landscape

On cherche à utiliser du landscape généralement pour trois classes de problèmes différents. Il y a donc trois types de réponses :

1. Tu cherches à faire tout ton document en landscape, par exemple pour des transparents pour une soutenance.
2. Tu cherches à alterner régulièrement entre landscape et portrait selon les passages du document.
3. Tu cherches à mettre un tableau ou un schéma en landscape parce qu'il est trop grand pour la page.

Ces trois problèmes sont relativement différents. On relèvera surtout le troisième qui est plus proche des problèmes de flottants que des problèmes de mise en page. Donc il sera traité au paragraphe 7.6 page 409 qui traite du package `rotating`.

Le premier problème est relativement simple. Il suffit de spécifier à LaTeX que tu fais du landscape, soit avec l'option prévue spécialement dans le `\documentclass`, soit avec le package `vmargin` (ou `geometry`). Attention, l'option de `\documentclass` ne change pas les marges ni largeur du texte, c'est à toi de le faire. C'est pourquoi je te recommande de passer par un package.

Il ne reste plus alors qu'à expliquer à `dvips`[130] que tu imprimes en landscape. Pour ce faire, il suffit d'appeler `dvips` au clavier et de lui donner les options suivantes :

```
dvips -t a4land toto
```

pour imprimer, ou en rajoutant encore l'option `-o` pour obtenir le fichier PostScript associé. Le package `geometry` se chargera tout seul de transmettre les bonnes instructions à `dvips`.

Le second problème, à savoir alterner entre landscape et portrait dans un même document, est relativement simple. Il te suffit de connaître le package `lscape` que nous devons à David Carlisle [30]. Ce package fonctionne en collaboration avec `graphics`. Il crée un environnement `landscape` dans lequel toutes les pages sont en landscape. Enfantin. Pour obtenir un document principal en landscape avec certaines pages en mode portrait, tu auras deviné tout seul : il faut combiner les deux. Tu utilises la première méthode (correction de la largeur du texte et des marges, et appel de `dvips` avec les bonnes options) pour spécifier que tout le document est en landscape et tu fais appel à l'environnement fourni par le package `lscape` pour repasser en mode portrait pour certaines pages.

Ce passage en trois colonnes et en mode landscape a été produit par l'environnement `landscape` du package `lscape`.

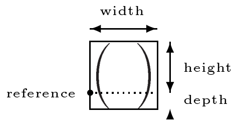
[130] Plus précisément, à ton logiciel d'impression, s'il en est capable. Sinon il faudra retourner la feuille dans l'imprimante.

## 4.23 Changement de ligne, changement de page, espacement

### 4.23.1 Interlignage

Un des derniers paramètres d'une mise en page auquel on ne s'est pas encore intéressé est l'interlignage. Normalement, il est intimement lié à la fonte que tu utilises. En règle générale, on considère qu'il faut un interligne de 120% de la hauteur de la fonte. Ainsi, dans un texte en corps 10, un interligne de 12 points est une bonne valeur par défaut.

Mais il faut savoir que le corps de la fonte est une notion qui est mal définie. Autant, quand on dit d'un humain qu'il mesure 2 mètres, c'est facile, on sait quelle partie de l'humain on mesure : du bas du talon au haut du crane. Par contre, quand on dit d'une fonte qu'elle fait 10 points, il n'y a pas de critère de mesure. Certains peuvent sembler intéressants, comme par exemple la hauteur d'une parenthèse.




width = `25.2353pt` height = `18.62059pt` depth = `6.18608pt`

Ici, on peut voir que notre parenthèse à 18,62 points de hauteur et 6,18 points de profondeur, ce qui pourrait nous faire penser à une fonte de corps 24,80 points. Alors que la fonte prétend, elle, pour sa part, être de corps 24.88. La différence n'est pas grande, mais elle est importante : elle montre que la mesure des parenthèses ou tout autre mesure aussi hasardeuse, ne donne pas à coup sûr le corps de la fonte.

On distingue bien deux éléments : le corps de la fonte, et *l'œil*[131] de la fonte. En gros, pour deux fontes en corps 10, une fonte « gros œil » a des caractères plus gros qu'une fonte « petit œil ». En règle générale, quand deux fontes ont le même corps, les capitales dans ces fontes ont à peu de choses près la même taille. Une fonte « petit œil » présente des minuscules nettement plus petites que ses capitales ; alors qu'une fonte « gros œil » présente des minuscules d'une taille plus voisine de ses capitales. La fonte que tu es en train de lire ici a plutot un petit œil, alors que par exemple la fonte `Courrier` a un gros œil.

Par exemple, les fontes classiques en PostScript sont un peu plus grosses que les fontes Computer Modern, ce qui fait qu'elles se mélangent assez mal, a priori. D'ailleurs, les fontes PostScript qui sont chargées par les différents packages pour se mélanger convenablement avec du Computer Modern sont parfois réduites de 5 % (option scaled du package `helvet`).

Du coup, la règle simple de « la taille (corps) plus 20 % » pour définir l'interligne est un peu trop rapide. Il faut en fait tenir compte de la taille réelle (œil) de la fonte, pour que les lettres montantes et les lettres descendantes ne risquent pas de se toucher, ainsi que de la noirceur de la fonte, pour obtenir un niveau de gris du texte qui soit convenable.

La commande de changement de taille de fonte, celle qui se cache derrière `\large` ou `\small`, c'est-à-dire `\fontsize`, prend d'ailleurs deux paramètres : en premier, le taille de fonte qu'on souhaite et en second, la taille de l'interligne voulu. Ainsi, un appel à :

```
\fontsize{10}{12}
```

indique qu'on demande une fonte en taille (corps) 10 et un interligne de 12 points.

L'interligne correspond à la grandeur `\baselineskip`. LaTeX laissera cette distance *au moins* entre deux lignes porteuses successives. Il laissera cependant parfois plus, par exemple si une ligne contient un gros caractère ou une formule de mathématiques trop haute.

[131] Ces deux termes viennent de l'imprimerie au plomb. Dans ce contexte-là, le corps est la hauteur du support en plomb de la lettre, et l'œil est la hauteur de la lettre elle même. Ainsi on peut avoir des petites lettres sur un gros corps. En général, ce qui fixait la hauteur du corps, c'était la hauteur des capitales.

Cette grandeur n'est prise en compte qu'à un seul moment : lorsque LaTeX assemble un paragraphe. Donc, à la fin d'un paragraphe. La valeur de `\baselineskip` qui est en vigueur à la fin du paragraphe servira à fixer l'interlignage de l'ensemble du paragraphe. Ainsi, dans cet exemple, c'est l'interligne de la commande `\footnotesize` qui est pris en compte pour le premier paragraphe :

```
Pour le moment, c'est un paragraphe normal.
...
{\footnotesize Ces quelques mots sont
\'ecrits plus petit.

D\'ebut du paragraphe suivant.} La fin du
deuxi\`eme paragraphe est de taille normale
\'egalement...
```

les lignes du premier paragraphe seront donc trop petites, comme le prouve le résultat de cet exemple :

> Pour le moment, c'est un paragraphe normal, de la même taille que tout le reste du livre. Cependant, son interlignage sera faussé à cause du fait que la fin du paragraphe a lieu pendant que l'interligne est plus réduit. Ces quelques mots sont écrits plus petit.
>
> Début du paragraphe suivant. La fin du deuxième paragraphe est de taille normale également. Ça se voit assez bien, les lignes n'ont pas l'air de se toucher.

Un cas particulier : la première ligne d'une page. En effet, pour cette ligne là, la notion d'interligne ne s'applique pas. La porteuse de la première ligne de la page est en fait à `\topskip` du haut de la page. En gros, tu peux considérer, en première approximation, que `\topskip` est la hauteur de la ligne elle-même et que `\baselineskip` est la distance entre deux lignes, la différence entre les deux étant à peu près le blanc qu'on trouve entre deux lignes.

Un autre cas particulier : la dernière ligne d'une page. Pour elle non plus, la notion d'interligne ne joue pas tellement. Plus subtilement cependant : sa position (enfin, la position de la porteuse de la dernière ligne) est déterminée comme étant, a priori, `\baselineskip` après l'avant-dernière ligne. Ce qui veut dire que, si cette porteuse coïncide avec le bas de la page (LaTeX fait en sorte que ce soit le cas le plus souvent possible, en étirant la page si nécessaire), ça veut dire que la profondeur de la dernière ligne est prise dans le blanc qui sépare le bas de page du pied de page (ou des flottants et notes en bas de page). Ce qui n'est acceptable que jusqu'à une certaine limite : si c'est la pronfondeur normale d'un texte tout bête, ça ne pose pas de problème. Mais si, pour une raison ou une autre, la dernière ligne contient un objet avec une très grande profondeur (par exemple à la suite d'une rotation de 90 degrés), ça va beaucoup trop dépasser.

Un paramètre permet de limiter ça : `\maxdepth`, qui indique la valeur maximale que peut prendre la profondeur de la dernière ligne d'une page. Si la ligne est plus profonde que `\maxdepth`, alors LaTeX la remonte au-dessus de sa porteuse jusqu'à ce qu'elle ait la profondeur indiquée par `\maxdepth`.

Du coup, si tu souhaites que ta page fasse un nombre entier de ligne, mettons 25, il faudra que sa longueur soit `\topskip` (pour la première ligne) plus 24 fois `\baselineskip`. C'est exactement le calcul que fait l'option lines du package `geometry`. C'est également l'arrondi que fait `rmpage` pour avoir une longueur de page qui soit un nombre pile de lignes.

Pour modifier l'interlignage, en gros, il y a deux solutions proposées par LaTeX. La solution lourde : redéfinir ta propre classe de document, qui définira les commandes de changement de taille et donc, dans celles-ci, choisira l'interligne qui te plaît. La solution légère est d'utiliser la commande `\linespread` dans le préambule de ton document. Elles prend un argument qui est une grandeur par laquelle seront multipliés tous les interlignes. Ainsi, si tu appelles :

```
\linespread{1.1}
```

l'interligne qui par défaut était de 12 points sera de 13,2 points. C'est une technique possible, mais un peu brutale, pour obtenir un double interligne.

Brutale parce que, souvent, pour un double interligne, pour un corps 10, on souhaiterait avoir un interligne de 20 points, c'est donc `1.66` qu'il faut passer à `\linespread` et non pas 2. Brutal aussi parce que ça va changer *tous* les interlignes, y compris ceux des notes en marge ou des notes de bas de page ou des titres de chapitres.

Mais, en première approche, ça peut dépanner.

### 4.23.2 Changement de ligne, césure

Plusieurs méthodes existent pour changer de ligne.

La première est de changer de paragraphe (cf. 9.5 page 40). La seconde est d'imposer un changement brutal comme
ici avec la commande `\newline`.

Enfin, il est possible d'autoriser un changement de ligne bizarre à l'aide de la commande `\linebreak[n]` où $n$ peut avoir les valeurs suivantes :

1 ⟶ changement de ligne autorisé ;

2 ⟶ changement de ligne acceptable ;

3 ⟶ changement de ligne bon ;

4 ⟶ changement de ligne préférable ;

5 ⟶ équivalent à un `\linebreak` sans paramètre ; changement de ligne imposé. Attention, contrairement au `\newline`, le `\linebreak` laisse la justification du paragraphe se faire.

On peut interdire un changement de ligne avec `\nolinebreak` qui admet les mêmes options que `\linebreak`.

Dans certains cas précis, LaTeX ne trouve pas un point de césure dans un mot, on peut alors les lui indiquer :

```
fonc\-tion\-nai\-re
```

NB : le troisième, entre le `i` et le `r`, n'est toléré par la typographie française que dans les cas extrêmes puisqu'il renvoit seulement 2 lettres à la ligne.

On peut aussi, pour des mots fréquents, les ajouter à son dictionnaire d'exceptions avec la commande

```
\hyphenation{fonc-tion-nai-re}
```

Attention, LaTeX différencie majuscules et minuscules, il faudra donc plus vraissemblablement entrer :

```
\hyphenation{fonc-tion-nai-re Fonc-tion-nai-re}
```

### 4.23.3 Espacement

Un espacement par défaut est ajouté entre deux paragraphes, on le trouvera parfois insuffisant, par exemple pour séparer deux séries de paragraphes sur deux idées différentes (thèse et antithèse dans une dissertation). On aura alors recours à l'une des trois commandes suivantes :

- `\smallskip` laisse un blanc étroit ;
- `\medskip` laisse un blanc moyen ;
- `\bigskip` laisse un grand blanc (à peu près une ligne).

Tu voudras parfois, pour d'autres raisons, laisser plus de blanc encore. Tu auras alors recours à une commande plus générale :

```
\vspace{5cm}
```

Commande à laquelle on peut spécifier la distance à laisser en `cm`, `mm`, `in` (pouce), `pt` (72,27`pt`=1`in`=2,54`cm`), `bp` (big point, ou point PostScript, 72`bp`=1`in`) et bien d'autres unités encore.

`\hspace` permettra d'obtenir un espacement horizontal d'une taille choisie.

Note bien que le blanc produit par `\hspace` et `\vspace` est de la même nature que celui qui est entre deux mots (ou entre deux lignes), c'est-à-dire qu'il disparaît en fin de ligne (pour `\hspace`) ou en fin de page (pour `\vspace`). Pour qu'il en soit autrement, il te faudra utiliser `\hspace*` et `\vspace*`.

C'est d'ailleurs également le cas pour `\smallskip` et consor. Si tu veux laisser une ligne blanche à tout prix quite à ce qu'elle se retrouve hideusement en haut ou en bas de page, il faut faire comme ca :

```
\vspace*{\bigskipamount}
```

### 4.23.4 Changement de page

De même que pour le changement de ligne, il existe plusieurs commandes. `\newpage`, `\pagebreak` et `\nopagebreak` sont similaires à leurs homologues pour les lignes, étudiés plus haut.

Il existe en sus `\clearpage` qui fait l'équivalent d'un `\newpage` puis s'assure que tous les flottants en attente soient placés. `\cleardoublepage` fait la même chose en veillant en plus — quite à laisser une page blanche — à ce que l'on reparte bien sur une page de droite.

### 4.23.5 Changement de page amélioré

L'une des grandes difficultés avec LaTeX dans ses vieilles version était de lui expliquer où il avait le droit de changer de page. En effet, dans un texte un peu compliqué (contenant des tableaux ou des blocs de texte insécables[132]) LaTeX ne savait jamais vraiment bien où couper sa page, et il avait tendance à faire des pages pleines de vide alors qu'une petite ligne de plus aurait bien arrangé ses affaires.

Il y a maintenant une solution :

```
\enlargethispage{longueur}
\enlargethispage*{longueur}
```

[132] Textes encadrés, tableaux, images, dessins, etc.

Ce texte est présenté avec un interligne double (il utilise l'environnement `doublespace`). Pour s'en convaincre il suffit de regarder. Un exemple de formule hors texte[a] :

$$a + b = x$$

La fin du texte.

[a]Une note de pied de page, pour voir ce que ça fait sur plusieurs lignes.

Ce texte là est encore plus espacé, puisqu'il utilise l'environnement `spacing` avec comme argument `2.5`. Une formule hors texte[a] :

$$a + b = x$$

La fin du texte.

[a]Une note de pied de page, pour voir ce que ça fait sur plusieurs lignes.

FIG. 4.46: Exemples d'utilisation du package `setspace`

qui permettent d'allonger la page de la longueur spécifiée[133] pour faire tenir une ligne de texte en plus ou pour en retirer une (longueur négative) de manière à arranger un peu les choses.

La forme avec une `*` impose à LaTeX de réduire au strict minimum toutes les longueurs élastiques mises en œuvre dans l'espacement vertical, c'est-à-dire qu'il comprime au maximum la page pour en faire tenir le plus possible dessus.

## 4.24 Double interlignage : le package `setspace`

Ce petit package, que nous devons à Geoffrey TOBIN [206], sert à changer l'interlignage d'un document.

L'usage type est celui d'un document destiné à la relecture, et pour lequel on souhaite avoir un interligne très large pour pouvoir noter des choses entre les lignes. La demande de double interligne est classique, par exemple, dans les universités, pour les thèses, même si le résultat est parfois épouvantablement moche.

Le principe d'utilisation est simple. Le package fournit trois commandes de base, qui sont a priori à utiliser dans le préambule de ton document, pour changer complètement l'interlignage de celui-ci. `\singlespacing`, qui en gros ne devrait rien changer (on reste en interligne simple) et est pris par défaut. `\doublespacing` qui prend un interligne double (en fait, un interligne de 1,67 fois approximativement celui de base, pour arriver à un interligne de deux fois le corps de la fonte) et une commande `\onehalfspacing` qui prend en gros un interligne d'une fois et demi le corps de fonte (1.25 fois la normale, à peu près). Une quatrième commande permet d'utiliser une autre valeur que les trois prévues par défaut : `\setstretch` qui prend cette valeur en argument. Trois options, singlespacing, doublespacing, et onehalfspacing, peuvent être passées au package et ont le même effet que d'appeler les

[133]Une longueur rigide.

commandes correspondantes.

Seul le corps du texte est changé, pas les flottants ni les notes en bas de page.

À tout moment, dans ton document, tu peux changer l'interlignagne en faisant appel à l'un des environnements : `singlespace`, `onehalfspace` et `doublespace`. L'environnement `spacing` permet de choisir l'espacement qu'on souhaite.

La figure 4.46 page précédente présente deux exemples d'utilisation de ces environnements.

## 4.25 Le package `afterpage`

Ce package est une vaste bidouille écrite par David Carlisle [33] (décidément, un habitué) qui permet des petits trucs assez cool au niveau de la mise en page de documents un peu bizarres. L'idée est de stocker des commandes qui seront appliquées juste après la fin de la page courante. Par exemple, tu as un tableau dont tu sais pertinemment qu'il sera grand. Tu as la possibilité de couper la page pour insérer ton tableau puis de reprendre ton texte après, mais ça a le désavantage de laisser une page non-pleine avant le tableau. C'est pas beau. Tu vas alors décider de mettre ton tableau en flottant. Oui, mais Dieu seul sait quand il va apparaître, peut-être bien à des pages et des pages de là.

Alors que si tu intimes l'ordre à LaTeX 2ε de le placer juste après la fin de la page courante, tu pourras aisément dire dans ton texte : « le tableau en haut de la page suivante vous indique... », ce qui est très joli.

On peut aussi, pour éviter que tous les flottants d'un chapitre ne viennent s'entasser à la fin du chapitre, indiquer un `\clearpage` juste après la fin de la page. Ça ne changera pas ta page, mais ça viendra vider tous les flottants encore en mémoire avant de continuer dans le texte. C'est parfois pratique.

Comment on utilise ce package ? Facilement :

```
\afterpage{%
\label{ex-util-afterpage}
\framebox{%
  \normalfont
  Texte en haut de la page~\pageref{ex-util-afterpage}
  (exemple de \Cde{afterpage})}
}
```

**Attention !** Ce package est une version béta[134]. La commande n'est pas particulièrement robuste et pourrait probablement introduire des erreurs.

De plus, ce package pose problème quand il est utilisé avec `multicol` puisque `multicol` détourne la gestion des pages pour gérer ses colonnes. Il ne fonctionnera donc que dans les passages en une seule colonne ou plus exactement le texte apparaîtra au prochain changement de page se faisant en une seule colonne.

Enfin, il faut noter que la fonte utilisée pour composer le contenu d'`\afterpage` sera celle en cours lorsque le changement de page se produira. A priori, tu ne peux pas prédire la fonte en question. Du coup, il est plus prudent, comme dans l'exemple ci-dessus, de fixer soi-même la fonte à quelque chose de raisonnable, en commençant par `\normalfont`. Une pareille précaution peut ne pas être suffisante, par exemple si le changement de page peut se produire dans un passage en russe ou en arabe.

[134] Cette remarque, toujours vraie, date de 1994. Dire que c'est une version béta, pour un package LaTeX, c'est une façon de dire qu'on n'a pas envie de le ré-écrire au propre et que s'il ne marche pas, tant pis. Bref, voilà, ce package, il marche. Et il peut aussi produire des erreurs très curieuses.

| Option | Description |
|---|---|
| pscoord | Option par défaut. Origine en bas à gauche. |
| texcoord | Origine en haut à gauche. |
| grid | Demande qu'une grille soit dessinée sur la page. |
| gridunit | Fixe l'unité de graduation de la grille. Valeurs possibles : mm, in, bp et pt. |
| gridcolor | Couleur des lignes principales de la grille, par défaut noir. |
| subgridcolor | Couleur des lignes secondaires de la grille, par défaut noir. |
| subgridstyle | Style utilisé pour les lignes secondaires. Peut être solid ou dotted. Attention, dotted est lent. Par défaut, solid. |
| colorgrid | Raccourci pour grid, gridcolor=red et subgridcolor=green. |
| dvips | Indique que le logiciel d'impression est `dvips`. Si c'est le cas, le package `eepic` est chargé pour accélérer le dessin des lignes pointillées (option dotted pour subgridstyle). |

Tab. 4.39: Options du package `eso-pic`

## 4.26 Dessiner sur la page : le package `eso-pic`

Texte en haut de la page 295 (exemple de `\afterpage`)

Ce package, que nous devons à Rolf Niepraschk [162], permet de dessiner ce qu'on veut en toile de fond de la page. Soit de toutes les pages, soit de la page en cours.

C'est basé sur deux commandes. La première, `\AddToShipoutPicture`, ajoute un élément (ou plusieurs) à toutes les pages. La seconde, `\AddToShipoutPicture*`, ajoute un élément (ou plusieurs) à la page en cours uniquement.

La command `\ClearShipoutPicture` permet de vider complètement le dessin qui a été rempli par les deux commandes précédentes.

Le dessin se fait avec les commandes habituelles de l'environnement `picture` (tu apprendras à utiliser ça au chapitre quatrième page 473). L'origine de l'environnement `picture` est en bas à gauche de la feuille de papier par défaut (option pscoord). Tu peux demander à ce qu'il soit en haut à gauche, ce qui est plus classique avec LaTeX, en utilisant l'option texcoord.

Les options possibles pour le package `eso-pic` sont reprises dans le tableau 4.39.

Un premier exemple d'utilisation peut être de placer une image en fond de la page, par exemple en faisant ça[135] :

```
\AddToShiptoutPicture*{%
  \put(0,0){\makebox(\LenToUnit{\paperwidth},%
                     \LenToUnit{\paperheight})%
                    {\includegraphics[width=\paperwidth,
                                      height=\paperheight]
                                     {clear-day}}}
}
```

Le package fournit certaines commandes utilitaires pour simplifier la saisie des différents éléments de ton dessin. La plus utile est probablement la commande `\LenToUnit`, que tu

[135] Sur une impression noir et blanc, le rendu sera assez mauvais, donnant l'impression d'une mauvaise photocopie. Sur un écran en couleur (je n'ai pas de quoi tester l'impression couleur chez moi) ça dessine un ciel clair avec quelques nuages d'altitude.

```
\usepackage[colorgrid]{eso-pic}
% \texte, comme avant
\section{Essai}
\texte\par
\subsection{Test}
\texte\marginpar{Note 1}
\texte\marginpar{Note 2}
\texte\par
\texte\ \texte\
\texte\par
```

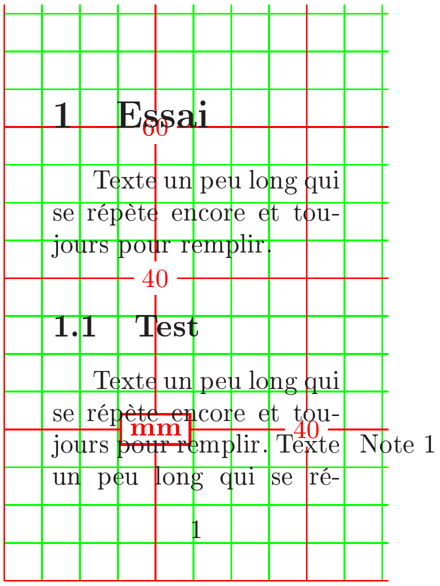



FIG. 4.47: Document avec l'option colorgrid de `eso-pic`

```
\usepackage{calc,eso-pic}
% \texte, comme avant
% Pour aligner sur la grille:
\parskip=0pt
% \AddToShipoutPicture, comme vu
\texte\par
\texte\ \texte\
\texte\par
\texte\ \texte\
\texte\par
```

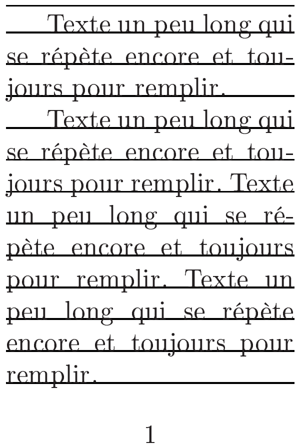



FIG. 4.48: Tracé de la grille typographique avec `eso-pic`

---

peux d'ailleurs utiliser dans n'importe quel environnement `picture`, qui te permet d'indiquer des grandeurs plutôt que des unités (l'environnement `picture` est normalement utilisé en indiquant les coordonnées et les longueurs en termes de `\unitlength`). Les autres commandes sont présentées dans le tableau 4.40 page suivante.

Une autre application possible est de tracer une grille, mais pas l'espèce de papier millimétré proposé par le package spontanément, comme celle montrée par l'exemple de la figure 4.47, plutôt une grille typographique. Pas non plus celle de `typogrid`, mais en fait simplement une grille qui présente l'endroit où *devraient* être les lignes si l'interlignage était respecté sur toute la page.

En fait, certains puristes de la typographie expliquent qu'on devrait toujours aligner le

| Commande | Signification |
|---|---|
| `\AtPageUpperLeft` | Place un élément en haut à gauche de la page. |
| `\AtPageLowerLeft` | Idem en bas à gauche. |
| `\AtPageCenter` | Idem au centre de la page. |
| `\AtTextUpperLeft` | Place un élément en haut à gauche du bloc de texte. |
| `\AtTextLowerLeft` | Idem en bas à gauche. |
| `\AtTextCenter` | Idem au centre. |

TAB. 4.40: Commandes annexes du package `eso-pic`

---

texte sur cette grille (ce qui n'est pas le cas des mises en page par défaut de LaTeX). C'est-à-dire que tous les éléments devraient avoir une hauteur qui soit un multiple de l'interligne, comme par exemple les équations hors texte, les tableaux, les titres, etc.

Une telle grille est assez longue à construire, mais pas très compliqué. Pour ça, je vais me réserver deux registres qui me serviront à faire mes calculs de coordonées[136]. Ensuite, je vais poser un `\label`, puis aller chercher le numéro de page correspondant à ce `\label`[137]. Une fois que la macro `\tmp` est définie comme contenant effectivement le numéro de page, via cette sombre bidouille[138], je détermine comment il faut caller ma grille horizontalement, en utilisant la marge des pages paires ou impaires, selon le cas.

```
\newskip\esoX
\newskip\esoY
\label{eso-pic-grid}
\expandafter\ifx\csname r@eso-pic-grid\endcsname\relax
  \def\tmp{0}
\else
  \edef\tmp{\expandafter\expandafter
            \expandafter\@secondoftwo
            \csname r@eso-pic-grid\endcsname}
\fi
\ifodd\tmp
    \setlength\esoX{1in+\voffset+\oddsidemargin}
\else
    \setlength\esoX{1in+\voffset+\evensidemargin}
\fi
```

[136]La majorité des techniques que je vais utiliser sont exposées dans le chapitre sur la programmation : chapitre neuvième page 571.

[137]La construction ésotérique autour de `\expandafter` et de `\tmp` sert à ça. À un label `toto` correspond une commande `\r@toto` qui contient {x}{y} où `x` est le numéro auquel on fait référence (numéro de figure ou de tableau, par exemple) et `y` est le numéro de la page.

La macro `\@secondoftwo`, comme son nom l'indique, prend deux paramètres et retourne le second, sa définition ressemble à ça :

`\def\@secondoftwo#1#2{#2}`

Bien entendu, la construction que j'utilise là est très dangereuse, parce qu'elle suppose que personne n'a touché à la façon dont LaTeX gère les références, ce qui n'est pas toujours vrai. En particulier, ça n'a quasiment aucune chance de marcher avec le package `hyperref` qui, lui, stocke 5 valeurs dans la commande en question. Il faudrait une macro `\@secondoffive`, qui est d'ailleurs fournie par le package `hyperref`.

[138]Souviens-toi, je t'ai déjà indiqué que de simplement regarder la valeur de `\thepage` ne suffit pas : LaTeX ne l'incrémente que lors qu'il éjecte une page terminée, or il lui arrive de lire quelques lignes d'avance sur la page d'après : dans ces quelques lignes, la valeur de `\thepage` n'est pas bonne.

Ensuite, c'est assez simple : on fait le même calcul pour le positionnement vertical, puis on place un trait horizontal sur chaque ligne porteuse. Un premier trait tout en haut de la page, puis un second `\topskip` plus bas (pour la première ligne), puis ensuite un trait tous les `\baselineskip` pour les lignes suivantes.

```
\AddToShipoutPicture*{%
  \setlength\esoY{\paperheight-1in-\voffset-\topmargin-\headheight-\headsep}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY}){\line(1,0){12}}
  \addtolength\esoY{-\topskip}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY}){\line(1,0){12}}
  \addtolength\esoY{-\baselineskip}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY}){\line(1,0){12}}
  \addtolength\esoY{-\baselineskip}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY}){\line(1,0){12}}
  \addtolength\esoY{-\baselineskip}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY}){\line(1,0){12}}
  \addtolength\esoY{-\baselineskip}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY}){\line(1,0){12}}
}
```

Une solution plus rapide est de positionner tous les traits sauf le premier en une seule commande, en utilisant `\multiput`.

```
  \setlength\esoY{\paperheight-1in-\voffset
                 -\topmargin-\headheight-\headsep}
  \put(\LenToUnit{\esoX},\LenToUnit{\esoY})
      {\line(1,0){\LenToUnit{\textwidth}}}
  \addtolength\esoY{-\topskip}
  \multiput(\LenToUnit{\esoX},\LenToUnit{\esoY})
           (0,\LenToUnit{-\baselineskip}){15}
   {\line(1,0){\LenToUnit{\textwidth}}}
```

Le résultat est présenté dans l'exemple de la figure 4.48 page 296. Une telle grille est un outil très pratique, si tu veux faire une typographie *alignée sur la grille*. En dehors de ça, ça ne sert à rien, bien entendu.