Chapitre troisième
Utilisation avancée

5 Des boîtes

TEX, et donc LATEX qui est basé dessus, ramène la plus grande partie de son travail de composition de texte à la manipulation des boîtes, ou plus exactement à la manipulation des listes, qui une fois figées deviendront des boîtes.

Schématiquement, TEX travail en deux modes : soit en mode horizontal, où les différents éléments de la liste seront mis les uns à côté des autres, soit en mode vertical, où les éléments seront mis les uns sous les autres.

Ce que nous allons voir ici, après avoir vu les bases de la manipulation des boîtes avec LATEX, ce sont quelques packages qui tournent autour de ce thème là : comment faire des encadrés par exemple, ou des versions améliorées des commandes et environnements de LATEX.

5.1 Manipulation des boîtes par LATEX

Une boîte est toujours caractérisée par trois grandeurs clef : sa hauteur, au dessus de la ligne porteuse, sa profondeur, en dessous de la ligne porteuse, et sa largeur.

LATEX fait des boîtes à peu prés tout le temps, sans te le dire. Ainsi, la page est constituée d'une grande boîte qui contient l'en-tête (une boîte), un blanc, puis les flottants du haut de la page (autant de boîtes) séparés par des blancs, puis le corps du texte (encore une boîte), puis les flottants du bas de la page (encore des boîtes) séparés par des blancs, puis les notes de pied de page (encore une boîte). La page est donc une grande boîte verticale.

Paradoxalement, seules les boîtes verticales ont une largeur *intrinsèque*, une largeur *a priori*. En effet, une boîte horizontale à la largeur de tous les éléments qu'elle contient, mais une boîte verticale à une largeur *a priori*. Cette largeur, `\hsize`, que l'on ne manipule jamais de manière directe avec LATEX, indique sur quelle largeur il faudra justifier un paragraphe s'il s'en trouve un dans la boîte.

Les deux exemples classiques avec LATEX de boîtes verticales sont la commande `\parbox` et l'environnement `minipage`. Ces deux constructions, qui sont sensiblement équivalentes, créent une boîte verticale et précisent sa largeur a priori. Notes bien, c'est la largeur *a priori*, c'est-à-dire que si LATEX détecte que tu commences un paragraphe, il justifiera ce paragraphe sur cette largeur là. Sinon, pour le reste, cette largeur n'est pas vraiment prise en compte.

Un bon moyen de comprendre ce qui se passe est d'analyser l'exemple suivant. TEX ne commence un paragraphe que quand il rencontre quelque chose qui lui fait quitter de force le mode vertical, c'est le cas par exemple d'un caractère, ou de la commande `\leavevmode` (comme son nom l'indique). Il termine un paragraphe, et donc retourne au mode vertical, quand il rencontre la commande `\par`, ou une ligne blanche (ce qui est équivalent). La commande `\hbox` est la primitive fournie par TEX pour fabriquer une boîte horizontale. La commande `\mbox` est l'équivalent LATEX. La seule différence entre les deux est que `\mbox` appelle `\leavevmode`, et que donc elle commence un paragraphe s'il n'y en avait pas déjà un en cours.

Le contenu d'une `\hbox` est en mode horizontal, mais TEX ne change pas de mode quand il croise une `\hbox` : s'il était en mode vertical juste avant de la rencontrer, il retournera à

ce mode-là juste après. À l'opposé, quand il croise un caractère pendant qu'il est en mode vertical, ça le fait passer en mode horizontal pour faire un paragraphe.

Le pramètre `\overfullrule` demande à TEX de dessiner un trait noir de cette épaisseur là à gauche des textes qui dépassent quand il fait sa justification (pratique pour repérer les erreurs du premier coup d'œil dans la marge). Je m'en sert ici pour te montrer qu'avec `\hbox`, il n'a pas vu de dépassement, alors qu'avec `\mbox`, il en a vu un. Voyons l'exemple :

```
{\overfullrule=5pt
\parbox{3cm}{\hbox{Trop long. \LaTeX\ ne dit rien.}
             Ceci est un texte normal qui sera coup\'e.\par
             \mbox{Trop long. \LaTeX\ proteste:}}
}
```

Et ici le résultat :

Trop long. LaTeX ne dit rien.
Ceci est un texte
normal qui sera
coupé.
Trop long. LaTeX proteste :

La commande `\parbox` demande la création d'une boîte verticale dont la largeur *a priori* est de trois centimètres. Au début de cette boîte, alors qu'on est encore en mode vertical, je mets une `\hbox`, celle-ci est simplement ajoutée à la liste, mais ne fait pas sortir TEX du mode vertical. Il ne cherche donc pas à faire des paragraphes. Il ne cherche donc pas à justifier les lignes sur la larger a priori. Cette première `\hbox` dépasse donc au delà des trois centimètres sans que TEX s'en émeuve.

La deuxième ligne lui fait quitter le mode vertical, puisqu'elle commence par un caractère (un vrai, un qu'il faut imprimer). Elle se termine par `\par`. Il va dont transformer tout ça en un paragraphe de trois centimètres de large (le résultat n'est pas très beau, mais trois centimètres, ce n'est pas beaucoup).

La troisième ligne contient un appel à `\mbox`. Qui fait exactement comme `\hbox`, c'est-à-dire crée une boîte horizontale *que TEX ne pourra pas couper tout seul par la suite*. Simplement, la commande `\mbox` commence par faire appel à `\leavevmode`. TEX est donc en train d'essayer de construire un paragraphe, et ce paragraphe contient un seul gros objet, trop gros pour tenir dans une ligne. Du coup, forcément, ça dépasse[138]. Et cette fois-ci, TEX trouve que ce n'est pas normal.

Ça te donnes une première indication utile : quand on utilise les primitives de TEX, parfois, il se met à avoir des comportements qui ne sont pas immédiatement compréhensibles. C'est plus rare avec les commandes LaTeX. Il vaut donc mieux s'en tenir aux commandes LaTeX.

5.1.1 Boîtes horizontales

`\mbox{texte}`

Les boîtes horizontales, pour l'utilisateur du moins, ne sont pas courantes dans LaTeX. En général, on s'en sert pour définir un symbole dont on ne veut pas qu'il soit coupé, ou pour empêcher la coupure d'un mot. Pour un nom propre par exemple.

[138] Oui, *ça dépend*, ça dépasse. C'est forcé.

Par exemple, quand on veut saisir un nom de famille qui risque d'être coupé, si on veut interdire totalement et définitivement cette coupure, au risque que ça dépasse dans la marge, il suffit d'utiliser la commande `\mbox` :

```
M.~\mbox{\textsc{Vermillon}}
```

`\makebox[`⟨*largeur*⟩`][`⟨*align.*⟩`]{texte}`

Une version un peu plus évoluée est la commande `\makebox`. Elle prend trois arguments dont les deux premiers sont optionnels et sont donc entre crochets. Le premier fixe la largeur de la boîte. Si tu ne le précise pas, la boîte sera de la largeur exacte du texte. Le second précise son centrage, il peut valoir `l` pour indiquer que le texte doit être à gauche (left) de la boîte, `r` pour indiquer qu'il doit être à droite (right), `c` pour indiquer qu'il doit être centré ; ou `s` pour indiquer qu'il doit être étiré (shrink) à cette longueur là.

Attention, quand LaTeX va vouloir étirer ton texte, il doit avoir de quoi le faire, sinon, il va protester (un warning à l'écran, une histoire de `Underfull \hbox`). Tu peux lui préciser comment étirer, par exemple, en utilisant la commande `\hspace` avec des grandeurs étirables (ce qu'en jargon TeX on appelle des glues). Ca peut donner ça :

```
\makebox[10cm][s]{gauche\hspace{0cm plus 1fill}droite}
```

pour produire :

gauche droite

Dans l'argument qui donne la largeur, tu peux utiliser `\width` pour signifier la largeur naturelle de la boîte[139] ; `\height` pour signifier sa hauteur naturelle, `\depth` pour signifier sa profondeur naturelle, et `\totalheight` pour indiquer sa hauteur naturelle totale (hauteur plus profondeur).

Par exemple, pour demander à agrandir la boîte de trois centimètres, on peut faire comme ça :

```
\makebox[\width+3cm][s]{gauche\hspace{0cm plus 1fill}droite}
```

pour produire :

gauche droite

Par contre, rien ne te permet de fixer la hauteur de la boîte. Ta boîte sera de la hauteur du plus haut des éléments qu'elle contient, et de la profondeur du plus profond des éléments qu'elles contient.

5.1.2 Les traits

Presqu'à chaque fois que tu croises un trait dans LaTeX, il s'agit en fait d'une boîte horizontale qui contient un trait vertical[140].

[139] Ainsi, `2\width` demande de doubler la largeur de la boîte, ou `\width+2cm`, si tu as utilisé le package `calc`, demande d'allonger la boîte de deux centimètres.

[140] Une boîte horizontale ne peut contenir que des traits verticaux. C'est comme ça. C'est TeX qui veut ça. Ceci dit, la notion de vertical pour un trait est très abstraite. Il peut très bien être de nature verticale mais être beaucoup plus large que haut, et donc sembler horizontal à l'œil.

C'est une des droleries de TeX, il a deux type de traits, les traits horizontaux, qui ne peuvent être que dans des boîtes verticales, et les traits verticaux qui ne peuvent être que dans des boîtes horizontales. Très troublant.

Pour s'en souvenir, facile : les traits verticaux forment les côtés d'un cadre, et sont à gauche et à droite de la partie encadrée, ils sont dont dans la boîte *horizontale* qui contient le premier trait, le truc à encadrer, et le second trait. Par contre, les traits horizontaux forment le dessus et le dessous du cadre, ils sont donc dans la boîte verticale qui contient le premier trait (du haut), puis le truc à encadrer, puis le second trait (du bas).

`\rule[`⟨*décal. haut*⟩`]{`⟨*largeur*⟩`}{`⟨*hauteur*⟩`}`

La commande LaTeX pour produire un trait est la commande `\rule`. Elle prend trois arguments, dont le premier est optionnel. Le second fixe la largeur, et le troisième fixe la hauteur. Le premier, optionnel, indique de combien il faut décaler, vers le haut, le trait résultant. Ce trait-là : ▬ à donc été produit par :

```
\rule[4pt]{1cm}{1pt}
```

`\framebox[`⟨*larg.*⟩`][`⟨*align.*⟩`]{texte}`

La commande `\framebox` fonctionne exactement comme `\makebox`, mais en placant un cadre autour de la boîte. Ce cadre est d'épaisseur `\fboxrule` (par défaut, 0,4 point), et est séparé du texte de tous les côtés par un blanc de `\fboxsep` (par défaut, 3 points). Ces grandeurs sont utilisées comme paramètres par d'autres commandes fournies par LaTeX ou par des packages, il faut donc être prudent quand on les modifies, ça peut avoir des effets innattendus.

`\frame{texte}`

La commande `\frame`, pour sa part, encadre son contenu, en le plaçant dans une boîte horizontale. Elle diffère fondamentalement de `\framebox` parce qu'elle choisit comme épaisseur du cadre une grandeur qui dépend de la fonte, et qu'elle ne place pas d'espace autour de ce qu'elle encadre. C'est elle que j'ai utilisé dans les exemples précédents d'utilisation de `\makebox`.

5.1.3 Les boîtes verticales

`\parbox[`⟨*align. externe*⟩`][`⟨*hauteur*⟩`][`⟨*align. interne*⟩`]{`⟨*larg.*⟩`}{texte}`

On a déjà survolé la commande `\parbox`, qui est le plus simple des moyens de faire une boîte verticale. En fait, cette commande prend cinq arguments dont les trois premiers sont optionnels. On a déjà vu le quatrième argument, qui est la largeur de la boîte à produire, et le cinquième, qui est le contenu de la boîte.

Le premier argument optionnel indique comment la boîte doit se centrer verticalement par rapport à la porteuse. S'il vaut `t` la porteuse externe à la boîte sera alignée sur la porteuse de la première ligne de la boîte[141] ; s'il vaut `b`, alors la porteuse externe sera alignée avec la porteuse de la dernière ligne de ta boîte ; et s'il vaut `c`, la boîte sera centrée verticalement sur la porteuse externe.

Cet argument est particulièrement utile quand tu veux aligner deux boîtes entre elles, et éventuellement avec un texte sur les côtés. Dans les tableaux, par exemple, il est fréquent que je mette côte à côte un tableau et un texte de commentaire, ou deux tableaux. Pour faire en sorte qu'ils soient alignés sur le haut, ce qui est de loin le plus joli, j'utilise l'option `t` sur les deux boîtes correspondantes.

Les deux arguments optionnels suivants jouent un rôle similaire à celui des arguments optionnels de `\makebox` : le premier précise la hauteur de la boîte à produire (la hauteur naturelle du contenu, s'il n'est pas précisé) et le second son alignement. Cet alignement pouvant être `t` pour indiquer que le texte est aligné en haut de la boîte ; `b` pour indiquer que le texte est aligné en bas de la boîte ; `c` pour indiquer que le texte est centré verticalement dans la boîte ; ou `s` pour indiquer que le texte doit être étiré sur toute la hauteur de la

[141] En fait, elle sera alignée sur la porteuse du premier élément de la boîte. Si ta boîte commence par un texte encadré ou un tableau, par exemple, il y a de grande chances que l'alignement se fasse avec le cadre plutôt qu'avec la première ligne.

boîte. Si tu choisis `s`, il faut que la boîte contienne suffisament de glue(s) pour permettre cet étirement. Tu peux en ajouter en utilisant `\vspace`.

`\begin{minipage}[`⟨*align. externe*⟩`][`⟨*hauteur*⟩`][`⟨*align. interne*⟩`]{`⟨*larg.*⟩`}`

L'environnement `minipage` prend les mêmes arguments que `\parbox` et fonctionne de manière très similaire. Sauf que c'est un environnement. Ce qui fait qu'un certain nombre de fonctionnalités qui sont interdites dans les arguments des commandes peuvent y être utilisées (la commande `\verb`, par exemple). Un autre intérêt est que cet environnement prévoit explicitement l'utilisation des notes de pied de page, mais qu'il les placera au pied de l'environnement, et non au pied de la page (ce qui en fait véritablement une mini page).

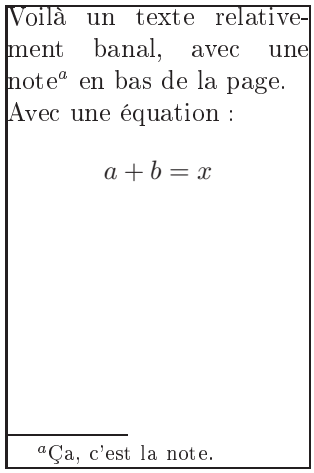
C'est le seul moyen que tu ais à ta disposition pour mettre des notes de bas de page dans un flottant, les notes normales, LaTeX les refusera : en effet, il ne sait pas sur quelle page arrivera ton flottant, donc il ne sait pas sur quelle page il faudrait placer les notes, qui devraient rester attachées avec le flottant. La meilleure solution pour que les deux restent attachés est de placer le tout dans un `minipage`.

Voyons rapidement un exemple de `minipage`. Je l'ai encadré avec `\frame`, pour que tu en vois bien les contours, mais, bien entendu, si tu fais ça, comme l'ensemble de l'environnement se trouve être en argument d'une commande, tu ne peux plus y faire ce qui est interdit dans les arguments de commandes (genre `\verb`). L'exemple :

```
\begin{minipage}[t][6cm][s]{4cm}
Voil\‘a un texte relativement banal,
avec une note\footnote{\c{C}a, c'est
la note.} en bas de la page.

Avec une \'equation:
\[ a + b = x \]
\vspace{0pt plus 1fill}
\end{minipage}
```

qui produit :



5.1.4 Boîtes sauvegardées

LaTeX est capable de mémoriser, et de recopier, le contenu d'une boîte. Au premier abord, ça peut sembler inutile, mais dans la pratique c'est parfois utile.

`\newsavebox{`⟨*commande*⟩`}`

`\savebox{`⟨*commande*⟩`}{texte}`

`\usebox{`⟨*commande*⟩`}`

Pour créer une nouvelle variable capable de contenir une boîte, il faut utiliser `\newsavebox`. Ensuite, tu peux lui affecter un contenu avec `\savebox`, qui produit une boîte horizontale

(un peu comme \mbox). Bien entendu, cette boîte peut contenir ce que tu veux, y compris une autre boîte, par exemple issue de \parbox.

En gros, à l'utilisation, ça donne ça :

```
\newsavebox{\maboite}
\savebox{\maboite}{\parbox{3cm}{...}}
... plus loin:
\usebox{\maboite}
```

À chaque fois que tu utiliseras \usebox, une copie de ta boîte sera placée.

\begin{lrbox}{⟨*commande*⟩}

Une autre alternative existe, c'est l'environnement lrbox, qui fonctionne à peu près de la même façon. Par contre, il a l'avantage d'être un environnement. Ainsi, si tu veux encadrer une boîte qui contienne un appel à \verb, tu vas pouvoir faire ça, par exemple :

```
\begin{lrbox}{\maboite}
\begin{minipage}{3cm}
Ce que tu veux, y compris des appels
\`a \verb.\verb..
\end{minipage}
\end{lrbox}
Le r\'esultat: \frame{\usebox{\maboite}}.
```

Le résultat : Ce que tu veux, y compris des appels à \verb.

5.2 Des familles de boîtes : le package eqparbox

Ce package, que nous devons à Scott Pakin [167], permet de créer des familles de boîtes (des \parbox) qui ont toutes la même largeur ; cette largeur étant celle de la plus large des boîtes.

À la première lecture, ça peut sembler un peu opaque. L'idée est de faire des alignements, un peu comme un tableau, mais qui puisse avoir lieu sur l'ensemble du document, ou sur une partie du document, sans pour autant être dans un tableau.

Un exemple assez simple est celui-là : je veux des titres sur trois colonnes, les trois colonnes étant centrées entre elles. Une solution qui vient assez rapidement à l'idée est de faire ça :

```
\parbox{\linewidth}{%
  Gauche\hspace{0cm plus 1fill}%
  Centre\hspace{0cm plus 1fill}%
  Droite}
```

Mais le résultat n'est pas terrible, par exemple deux lignes avec des morceaux de longueurs différentes :

Gauche1 Centre1 long Droite1 véritablement long
Gauche2 franchement long Centre2 Droite2

Il saute aux yeux que tout ça n'a pas du tout l'air d'être aligné. L'idée est de mettre tous les morceaux *gauche* dans une boîte qui soit de la largeur du plus large d'entre eux.

Supposons qu'on définisse une largeur `\gauche` qui soit le plus large des morceaux de gauche, `\centre` qui soit le plus large des morceaux du centre, et `\droite` qui soit le plus large des morceaux de droite, on pourrait faire :

```
\parbox{\linewidth}{%
  \parbox{\gauche}{Gauche}\hspace{0pt plus 1fill}%
  \parbox{\centre}{\centering Centre}\hspace{0pt plus 1fill}%
  \parbox{\droite}{Droite}}
```

pour obtenir un alignement nettement plus raisonnable. Si en plus on centre la boîte du centre (comme c'est le cas dans l'exemple ci-dessus) alors le résultat sera presque joli.

Ce que le package `eqparbox` permet, c'est tout simplement de simplifier le calcul des grandeurs `\gauche`, `\centre` et `\droite`.

`\eqparbox[`⟨*align. externe*⟩`][`⟨*hauteur*⟩`][`⟨*align. interne*⟩`]{`⟨*famille*⟩`}{texte}`

Il propose en effet une commande `\eqparbox` qui fonctionne comme `\parbox`, sauf qu'à la place de la largeur on met un mot clef (le nom de la famille de boîte). Toutes les boîtes de cette famille seront de la même largeur, à savoir la largeur naturelle de la plus large d'entre elles. Voila ce que ça donne pour notre exemple :

```
\parbox{\linewidth}{%
  \eqparbox{gauche}{Gauche}\hspace{...}%
  \eqparbox{centre}{Centre}\hspace{...}%
  \eqparbox{droite}{Droite}}
```

ce qui, appliqué à notre exemple précédent, produit :

Gauche1	Centre1 long	Droite1 véritablement long
Gauche2 franchement long	Centre2	Droite2

Le principe est assez simple en fait : pendant la compilation du document, le package note la largeur de chacune des boîtes d'une famille donnée, ne retenant que la plus grande de ces largeurs. À la fin du document, il note cette valeur. Lors de la prochaine compilation, il s'en servira comme point de départ. Si, au cours de la compilation il constate que la valeur pour l'une des familles a changé (soit que tu as une nouvelle boîte plus large que les autres, soit que tu as réduit la boîte qui était avant la plus large, alors il change la valeur qu'il note, et il te dit :

```
LaTeX Warning: Rerun to correct eqparbox widths.
ou
LaTeX Warning: Rerun to correct the width of eqparbox ‘gauche’.
```

Une deuxième commande est disponible, `\eqboxwidth`, qui te donne la largeur d'une famille de boîtes. Ainsi, si j'ai besoin de mettre dans la partie `centre` un texte qui soit vraiment beaucoup plus large que les autres, mais dont je veux garantir l'alignement, il me suffit de faire :

```
\parbox{\eqboxwidth{centre}}
       {Texte vraiment plus long que les
        autres, que je veux quand m\^eme
        aligner.}
```

comme par exemple dans ce cas là :

Gauche1	Centre1 long	Droite1 véritablement long
Gauche	Texte vraiment plus long que les autres, que je veux quand même aligner.	Droite
Gauche2 franchement long	Centre2	Droite2

5.3 Nouvelles options : le package `parboxx`

Ce tout petit package, que nous devons à Jörg KNAPPEN, fournit deux nouvelles valeurs, pour le premier paramètre optionnel de `\parbox`.

Ces deux nouvelles valeurs sont `B` et `T` qui alignent les boîtes véritablement sur le bas (resp. le haut) et non pas sur la première ou dernière ligne, comme `b` et `t`.

Un exemple rapide suffira à voir la différence :

```
\def\bx#1{\parbox[#1][\totalheight][#1]{1cm}{ici\\pas l\‘a}}
aa \bx{t} bb \bx{b} cc \bx{T} dd \bx{B} ee
```

produit :

aa ici pas là bb ici pas là cc ici pas là dd ici pas là ee

Curieusement, pour que ces nouvelles options aient un effet, il faut les passer à la fois au premier argument (où elles sont équivalentes à `t` et `b`) mais aussi au troisième argument. C'est curieux, en effet, parce que le premier argument indique l'alignement vis à vis de l'extérieur, alors que le troisième argument indique normalement l'alignement à l'intérieur de la boîte. Or, ces deux options, qui n'ont d'influence que sur l'alignement extérieur, sont prises en compte véritablement dans le troisième argument. Voilà. C'est curieux. Mais ça marche.

5.4 Généralisation de `minipage` : le package `genmpage`

Ce package très prometteur[142], que nous devons à Thomas LOTZE [128], remplace l'environnement `minipage` de LaTeX par une version généralisée qui est capable de traiter quelques points supplémentaires.

`\begin{minipage}[`⟨*align. externe*⟩`][`⟨*hauteur*⟩`][`⟨*align. interne*⟩`]{`⟨*larg.*⟩`}`
`[`⟨*options*⟩`]`

Les options acceptées par l'environnement `minipage` ainsi re-définit permettent de préciser les mêmes informations que les autres arguments optionnels, mais permettent également d'apporter de nouvelles fonctionnalités. Ces options font l'objet du tableau 5.1 page suivante.

En elles mêmes, ces options n'apportent rien de nouveau, parce que tout ce qu'elles font, tu savais déjà le faire. Par exemple, l'appel à :

```
\begin{minipage}{5cm}[fshape=it,raggedright]
```

[142] J'ai bon espoir d'en voir de nouvelles versions un jour, avec tout un tas de nouvelles fonctionnalités, et quelques améliorations.

Option	Description
flush, raggedright, RaggedRight, raggedleft, center	Ces options indiquent le mode de justification du contenu de l'environnement. flush ne fait rien. RaggedRight n'est disponible que si le package `ragged2e` est chargé. Ne requièrent pas de valeur.
ffamily, fseries, fshape	Permettent de préciser la fonte par défaut de l'environnement, en utilisant ces trois paramètres de NFSS. Prennent une valeur respectivement nom de famille, de série, ou de forme.
resetfont	Ne requiert pas de valeur. Indique simplement que le changement de fonte se fait par rapport à `\normalfont` et non pas par rapport à la fonte précédente.
fsize	Indique le corps de la fonte à utiliser. La veleur doit être le nom d'une commande de changement de taille (par exemple `small`).
width, height, outer, inner	Correspondent aux arguments habituels de `minipage`. Si ces arguments sont précisés, ils prennent le pas sur ceux passés à l'environnement.
widtharg, heightarg, outerarg, innerarg	Indiquent de prendre en compte les arguments de l'environnement et de négliger les options précédentes.
parindent, keepparindent	Permettent de régler l'indentation des paragraphes dans l'environnement. Par défaut LaTeX supprime l'indentation des `minipage`. parindent prend une valeur qui est l'indentation. keepparindent indique de garder la même indentation qu'à l'extérieur.

TAB. 5.1: Options de l'environnement `minipage` ajoutées par `genmpage`

```
...
\end{minipage}
```

est équivalent à :

```
\begin{minipage}{5cm}
\itshape\raggedright
...
\end{minipage}
```

Quel intérêt, alors ? En fait, le vrai intérêt, c'est la commande `\defineMPstyle`.

`\defineMPstyle{`⟨*style*⟩`}{`⟨*définition*⟩`}`

Cette commande permet de créer des styles de `minipage`, qu'on peut ensuite ré-utiliser dans tout le document. Par exemple, pour moi qui ait l'habitude d'utiliser des `minipage` pour faire des commentaires dans les figures, je peux procéder comme ça :

```
\defineMPstyle{commentaire}
    {fsize=footnotesize,
     outer=t,
     parindent=12pt,
     width=.6\linewidth}
```

Ce qui me garanti que tous mes environnements seront bien mis en forme de la même façon ; et si je décide de changer cette façon de faire, je n'ai qu'à le faire à un seul endroit. Un exemple du style commentaire que je viens de définir :

Dans cet environnement, je n'ai même pas pris la peine de préciser la largeur du texte à produire, puisque c'est dans la définition du style.

Amusant : j'ai imbriqué deux environnement dans le même style.

Comme la largeur de l'environnement est définie en fonction de `\linewidth`, c'est-à-dire la longueur d'une ligne avant de commencer l'environnement, on peut jouer aux poupées russes.

Le source ressemble à ça :

```
\begin{center}
\begin{minipage}{}[commentaire]
Dans cet environnement, ...
\begin{center}
\begin{minipage}{}[commentaire]
Amusant: j'ai imbriqu\'e ...
\end{minipage}
\end{center}
\end{minipage}
\end{center}
```

À noter que le package ne prévoit pas d'option pour revenir à un style justifié si on est, par exemple, en raggedright. Il est donc passablement dangereux d'utiliser l'une de ces options dans un style : tu ne pourras pas faire de cas particulier sur ce style-là.

5.5 Largeur minimale multi-ligne : le package `pbox`

Ce micro package, de Simon Law [120], définit une nouvelle commande `\pbox` autour du thème de `\parbox`.

L'idée est relativement simple. `\parbox` est prévue pour faire des paragraphes, des textes entiers, dont on fixe la largeur par avance, pour que le texte soit justifié dans sa boîte. Bien souvent, on s'en sert abusivement sitôt qu'on a des textes de plus d'une ligne, et qu'on souhaite déterminer pleinement leur alignement. Dans ce cas là, quand on ne cherche pas vraiment à mettre en forme des paragraphes, mais simplement un texte multi-lignes, `\parbox` s'avère être une commande qui n'est pas tout à fait adaptée.

Par exemple, pour faire le haut d'un CV, où on veut mettre dans deux boîtes voisines son état civil et son adresse, `\parbox` n'est pas forcément idéal : il suppose qu'on connaisse d'avance la largeur des boîtes. Ça donnerait une construction comme ça :

```
\parbox[t]{largeur1}{%
Nom Pr\'enom\\
Date de naissance\\
\^Age\\
Statut marital}%
\parbox[t]{largeur2}{%
Adresse1\\
Adresse2\\
CP Ville\\
Tel\\
Mail}
```

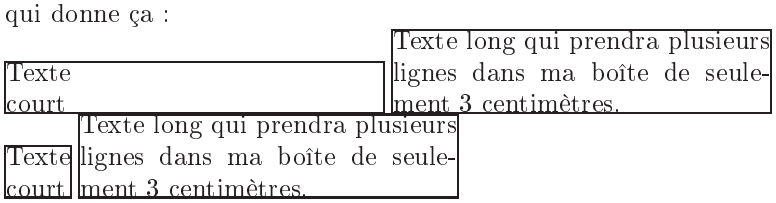
`\pbox[`⟨*align. externe*⟩`][`⟨*hauteur*⟩`][`⟨*align. interne*⟩`]{`⟨*largeur*⟩`}{texte}`

Le package `pbox` offre une nouvelle commande, avec une nouvelle approche pour mettre en forme ce type d'éléments. La syntaxe est la même que celle de `\parbox`, sauf que la largeur indiquée est la largeur *maximale*. Si le texte tient sur moins que ça en largeur, alors la boîte aura la largeur naturelle du texte.

Essayons rapidement un exemple, pour bien comprendre :

```
\frame{\parbox{3cm}{Text\\court}}
\frame{\parbox{3cm}{Texte long...}}\\
\frame{\pbox{3cm}{Texte\\court}}
\frame{\pbox{3cm}{Texte long...}}
```

qui donne ça :



Le package offre également quelques raccourcis pour pouvoir manipuler ce type de boîtes. Si je reprend l'exemple de tout à l'heure, le haut du CV : si tu veux mettre l'état civil sans changer de ligne (ce qui semble raisonnable), et que tu accepterais qu'il y ait des changements de ligne dans l'adresse, et que tu veux réserver de la place pour une photo, le problème peut être relativement complexe à résoudre :

- la largeur de ta page est `\linewidth`, tu veux réserver trois centimètres pour la photo ;
- on va se fixer une largeur minimale pour l'adresse, disons deux centimètres ;

– on va se fixer également un demi centimètre de blanc entre chaque élément ;
– il faut mettre en forme l'état civil dans une boîte qui n'excède pas `\linewidth` − 6 cm de largeur ;
– puis il faut mettre en forme l'adresse dans un boîte de la largeur restante (`\linewidth` moins la largeur de l'état civil, moins les trois centimètres de la photo).

La commande `\pbox` souffre de quelques très sérieuses limitations, en particulier, elle ne peut contenir que ce qu'on a le droit de mettre dans une colonne de type `l` dans un tableau. Par exemple, pas moyen d'y mettre une liste ou une équation hors-texte. Pour pouvoir faire ce genre de choses, il vaut mieux utiliser le package `varwidth` (voire à ce sujet en 5.6 page suivante).

`\settominwidth[`⟨*largeur*⟩`]{`⟨*variable*⟩`}{texte}`

La commande `\settominwidth` va nous aider : elle range dans la variable (une longueur) la largeur de la `\pbox` correspondante. Ainsi, on peut traiter notre problème comme ça[143] :

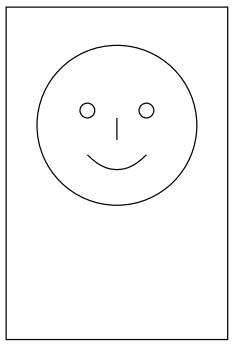
```
% Les variables
\newlength{\largeuretatcivil}
\newlength{\largeuradresse}
\newlength{\largtmp}
% Les calculs
\setlength{\largtmp}{\linewidth-6cm}
% \largtmp est la largeur max. possible pour
% l'etat civil
\settominwidth[\largtmp]
              {\largeuretatcivil}
              {Nom Pr\'enom\\...}
% \largeuretatcivil est la largeur reele de
% l'etat civil, on peut maintenant calculer la
% largeur de l'adresse
\setlength{\largeuradresse}%
          {\linewidth-4cm-\largeuretatcivil}
% Le texte
\noindent
\pbox[T][\noexpand\totalheight][T]
     {\largeuretatcivil}
     {Nom Pr\'enom\\...}%
\hspace{5mm}%
\parbox[T][\totalheight][T]
       {\largeuradresse}%
       {Adresse1\\..}%
\hspace{5mm}%
\parbox[T][\totalheight][T]
       {3cm}%
       {...photo...}
```

[143] Notes que j'utilise l'option `T` de la commande `\parbox`, option introduite par le package `parboxx`, pour que le haut du texte de l'état civil et de l'adresse vienne s'aligner sur le haut de la photo.

À noter également que `\pbox` fait des manipulations curieuses avec les arguments optionnels avant de les transmettre à `\parbox`, et du coup `\totalheight` est mal reconnu. Le `\noexpand` permet de résoudre ce problème.

Nom Prénom
Né le jj-mm-aaaa
xx ans
Célibataire

Adresse 1
Adresse 2
CPnnn Ville
Tel : 01.02.03.04.05
Mail : `moi@mondomaine.org`



5.6 Boîte plus étroite : le package `varwidth`

Nous devons ce package à Donald Arseneau. Il fournit un nouvel environnement, `varwidth`, qui fonctionne comme `minipage`, mais en produisant un texte très légèrement plus étroit que ce qui est demandé.

`\begin{varwidth}[`⟨*align. externe*⟩`][`⟨*hauteur*⟩`][`⟨*align. interne*⟩`]{`⟨*largeur*⟩`}`

Quand le texte est effectivement moins large que la largeur demandée en argument, alors le comportement de l'environnement est très comparable à ce que fait la commande `\pbox` (package `pbox`). Si par contre le texte prend bien toute la largeur de la boîte, il va produire un résultat très légèrement plus étroit que l'environnement `minipage` (à peine visible à l'œil nu).

Le package propose également une commande `\narrowragged` qui, quand elle est utilisée dans le contexte de l'environnement, produit un texte comme `\raggedright`, c'est-à-dire non justifié, mais en ayant tendance à remplir la dernière ligne comme les autres, quitte à produire un texte plus étroit. C'est particulièrement agréable dans un tableau, ça permet de gagner un peu en largeur sans changer la hauteur du tableau.

Du coup, le package propose également un nouveau type de colonne pour les tableaux, le type `V`, qui prend en paramètre la largeur (approximative) de la colonne. Le texte de cette colonne là sera mis dans un environnement `varwidth`, en appelant la commande `\narrowragged`.

À retenir : la commande `\pbox` est très fragile, elle ne peut pas contenir n'importe quoi (pas de liste, par exemple) ; alors que l'environnement `varwidth` est véritablement solide et peut contenir à peu près n'importe quoi sans problème (quelques soucis cependant avec les équations numérotées, et avec les environnements du package `amsmath`).

5.7 Fond grisé : le package `shadbox`

Ce package, que nous devons à D.A. Glazkov, permet de faire des boîtes avec un fond grisé, et ce *sans utiliser autre chose que TEX*.

À noter tout de suite : le package `shadbox` contient vraiment un grand nombre d'erreurs, dont certaines assez graves. Une version corrigeant ces erreurs est disponible à cet endroit là :
`\http://jmpl.fr.eu.org/JMPL/myshadbox.sty`

Le package dont je parlerai ici est donc en fait `myshadbox`.

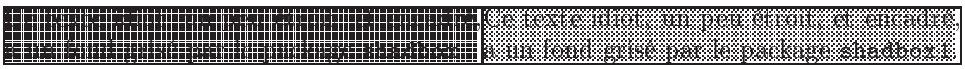
`\shadbox{`⟨*rapport*⟩`}{texte}`

La commande `\shadbox` existe en deux variantes, qui ont la même syntaxe (la deuxième s'appelle `\shadboxalt`). Toutes deux produisent un fond grisé en dessinant de petits carrés noirs régulièrement espacé sous le texte passé comme deuxième argument. Chacun des ces petits carrés fait `\deltablack` de côté (par défaut 0,3 point). Le premier argument dit quelle distance doit se trouver entre deux carrés successifs. S'il vaut 1, le fond sera uniformément noir, s'il vaut 2, il y aura exactement `\deltablack` entre deux carrés successifs. La première variante, `\shadbox` dispose les carrés selon une grille. La deuxième selon un forme d'échiquier (le gris obtenu est plus agréable).

Un exemple d'utilisation, pour voir ce que ça donne :

```
\noindent\shadbox{3}{%
\frame{%
\begin{minipage}{.45\linewidth}
Ce texte idiot, un peu \'etroit, et encadr\'e,
a un fond gris\'e par le package \TidxPak{shadbox}.
\end{minipage}%
}%
}
\shadboxalt{3}{%
\frame{%
\begin{minipage}{.45\linewidth}
Ce texte idiot, un peu \'etroit, et encadr\'e,
a un fond gris\'e par le package \TidxPak{shadbox1}.
\end{minipage}%
}%
}
```

qui donne ça :



Et maintenant, le même en positionnant `\deltablack` à un dixième de point[144] avec :

```
\deltablack=.1pt
```

produit :

À noter que ce package génère des fichiers énormes, à cause du grand nombre d'éléments qui sont dans le fichier (tous les petits rectangles), cela se traduit d'ailleurs par un temps d'impression tout à fait déraisonnable. Donc, si ton système te permet de le faire, ce qui est très fortement probable, alors *il ne faut pas te servir de ce package*, il vaut mieux utiliser les choses que l'on verra au chapitre quatrième page 467.

Exemple de `boiteepaisseavecuntitre` :

Le titre

Ceci est un texte peu utile, mais qui est encadré, comme ça on peut voir que le cadre est bel et bien coupé entre les deux colonnes de mon exemple. Je m'attends à un résultat curieux, voire laid.

Exemple de `boitenumeroteeavecunedoublebarre` :

42 Ceci est un texte peu utile, mais qui est encadré, comme ça on peut voir que le cadre est bel et bien coupé entre les deux colonnes de mon exemple. Je m'attends à un résultat curieux, voire laid.

Exemple de `boiteavecunelignequiondulesurlecote` :

Ceci est un texte peu utile, mais qui est encadré, comme ça on peut voir que le cadre est bel et bien coupé entre les deux colonnes de mon exemple. Je m'attends à un résultat curieux, voire laid.

Exemple de `boitecoloriee` :

Ceci est un texte peu utile, mais qui est encadré, comme ça on peut voir que le cadre est bel et bien coupé entre les deux colonnes de mon exemple. Je m'attends à un résultat curieux, voire laid.

TAB. 5.2: Environnements fournis par le package `boites_exemples`

5.8 Faire des cadres : le package `boites`

Ce package, que nous devons à Vincent ZOONEKYND [228], permet de faire quelques constructions intéressantes avec les boîtes. Il se base sur le package `eclbkbox` de Hideki ISOZAKI.

L'idée de base est assez simple : faire des boîtes encadrées qui puissent être coupées entre plusieurs pages. En fait, il va utiliser l'environnement générique `breakbox` du package `boites` dans le package `boites_exemples` pour produire quelques cas types intéressants.

Voyons rapidement le résultat de `breakbox` :

Ceci est un texte peu utile, mais qui est encadré, comme ça on peut voir que le cadre est bel et bien coupé entre les deux colonnes de mon exemple. Je m'attends à un résultat curieux, voire laid.

J'ai mis l'exemple en deux colonnes, ce qui permet de mettre en évidence le fait que cette boîte est encadrée *et* qu'elle peut être coupée entre deux pages. Ça, c'est la base de ce que fournit le package `boites`.

Son petit frère, `boites_exemples` offre pour sa part quelques autres environnements, qui sont repris dans le tableau 5.2.

Le package `boites` peut être utilisé pour définir d'autres boîtes, mais ça relève clairement de la programmation LaTeX, il vaut donc mieux que tu ailles lire le chapitre neuvième page 565 avant de te lancer là-dedans (au moins le début du chapitre).

En fait, l'environnement `breakbox` utilise un certain nombre de macros pour dessiner la boîte. Il suffit de modifier ces macros pour changer l'allure de la boîte.

[144] Cette valeur, sur une imprimante laser 1200dpi, donne un gris absolument parfait pour `\shadboxalt`.

Regardons rapidement le fonctionnement de `breakbox`. Le principe est relativement simple. Il commence par mettre en forme ton texte dans l'équivalent d'une `\parbox` (ou d'une `minipage`), ensuite, en utilisant la commande `\vsplit`, il va découper cette boîte en morceaux le plus petit possible. Ce qui fait qu'il va découper ça en lignes. La première et la dernière ligne vont être traitées spécialement (elles portent respectivement le haut et le bas du cadre). Chacune des lignes du milieu est flanquée d'un bout de cadre à gauche et à droite (avec quelques ajustements pour que tout ça s'aligne comme il faut). Le résultat est une suite de boîtes, une par ligne, entre lesquelles il n'y a pas de blanc[145] mais entre lesquelles LaTeX a le droit de changer de page.

De là, on comprend rapidement le sens des commandes à re-définir pour faire une boîte bien à nous.

`\bkvz@before@breakbox` contient ce qui devra apparaître avant la boîte. En gros, ça vérifie que l'éventuel paragraphe en cours est terminé, et ça ajoute un blanc de `\breakboxskip`, par défaut :

```
\def\bkvz@before@breakbox{%
  \ifhmode\par\fi
  \vskip\breakboxskip\relax
}
```

`\bkvz@left` est ce qui sera mis à gauche de toutes les lignes, par défaut ça fait un trait vertical de `\fboxrule` d'épaisseur, puis ça laisse un blanc de `\fboxsep` :

```
\def\bkvz@left{%
  \vrule \@width\fboxrule
  \hskip\fboxsep
}
```

`\bkvz@right` fait la même chose pour la droite, en échangeant bien entendu le blanc et le trait :

```
\def\bkvz@right{%
  \hskip\fboxsep
  \vrule \@width\fboxrule}
```

`\bkvz@top` dessine le haut du cadre, qui sera rattaché à la première ligne. C'est par exemple lui qui fait le titre. Par défaut, pour un cadre tout simple, c'est un trait horizontal de `\fboxrule` d'épaisseur :

```
\def\bkvz@top{%
  \hrule\@height\fboxrule
}
```

`\bkvz@bottom` dessine le bas du cadre, qui sera rattaché à la dernière ligne :

```
\def\bkvz@bottom{%
  \hrule\@height\fboxrule
}
```

Ça peut ne pas sembler évident de prime abord, mais la largeur du texte *n'est pas* celle du texte en dehors de la boîte. Si ton document à une longueur de ligne de dix centimètres, c'est la boîte qui sera dix centimètres de large. Le texte, pour sa part, sera un peu moins

[145] C'est important qu'il n'y ait pas de blanc. S'il y en avait, LaTeX pourrait s'autoriser à agrandir ces blancs pour pouvoir allonger les pages et les aligner en longueur, ça ferait des cadres en pointillés.

large. Pour être précis, il sera moins large de deux `\fboxrule` et de deux `\fboxsep` : les traits et blancs qui sont ajoutés à gauche et à droite. En fait, il faut retirer de la longueur de la ligne la largeur des éléments que tu places à gauche et à droite de chaque ligne. Comme tu peux re-définir ces éléments, tu peux re-définir la longueur de la ligne de texte.

`\bkvz@set@linewidth` est appellée au début de `breakbox` et est supposée calculer la nouvelle valeur de `\linewidth` à partir de l'ancienne. Voilà la définition par défaut, celle qui correspond aux définitions par défaut ci-dessus :

```
\def\bkvz@set@linewidth{%
  \advance\linewidth -2\fboxrule
  \advance\linewidth -2\fboxsep
}
```

Tout cela nous permet de définir notre propre cadre. On va choisir un exemple similaire à ce que fait déjà le package, mais en adaptant quelques détails. On veut faire une boîte avec un cadre fin, un cadre épais comme d'habitude (0,4 points), avec un titre dans la barre du haut, mais décentré (à deux centimètres à peu près du bord gauche), et faire en sorte que la barre du haut arrive à une hauteur raisonnable par rapport au texte (et non pas au pied du texte).

Pour la hauteur du trait du haut, c'est assez facile, il y a une unité exprès, `ex` qui est à peu près la hauteur de la lettre x. Un trait qui arriverait au milieu serait bien. Ça donnerait ça comme définition de `\bkvz@top`, si on suppose que la macro `\titre` contient le titre à afficher[146] :

```
\newsavebox{\maboite}
\newlength\essai
\essai=.5ex
\advance\essai by -\fboxrule
\def\bkvz@top{%
  \begin{lrbox}{\maboite}\titre\end{lrbox}
  \hbox to \hsize{%
    \vrule width \fboxrule height 0pt depth \dp\maboite
    \hskip -\fboxrule
    \vrule width \fboxrule height .5ex depth 0pt
    \vrule width 1cm height .5ex depth -\essai
    \hskip \fboxsep
    \usebox{\maboite}%
    \hskip \fboxsep
    \leaders\vrule height .5ex depth -\essai\hfill
    \vrule width \fboxrule height 0pt depth \dp\maboite
    \hskip -\fboxrule
    \vrule width \fboxrule height .5ex depth 0pt
  }
```

[146] Quelques remarques sur la façon dont c'est fait. En particulier le petit trait vertical à gauche du haut du cadre. Ce trait doit faire `.5ex` de haut et avoir la même profondeur que la boîte contenant le titre. Il se trouve que quand on le fait en une seule fois, par exemple avec la commande :

```
\vrule width \fboxrule height 0pt depth \dp\maboite
```

alors, pour une raison que je ne m'explique pas bien, la hauteur est très légèrement fausse (trop haut de 0,2 points, à vue d'œil). Par contre, quand on procède en deux fois comme je le fait là, en mettant un premier trait qui a la profondeur voulue, et superposé un deuxième qui a la hauteur voulue, alors l'alignement est parfait.

Malgré quelques recherches je n'ai pas réussi à trouver d'explication.

Longueur	Rôle(s)
\fboxrule	Épaisseur du trait fin pour \shadowbox. Trait intérieur de \doublebox : 0,75×\fboxrule Trait extérieur de \doublebox : 1,5×\fboxrule Séparation des traits pour \doublebox : 1,5×\fboxrule plus un demi point.
\fboxsep	Distance entre le contenu et le cadre.
\shadowsize	Épaisseur de l'ombre portée pour \shadowbox

TAB. 5.3: Longueurs utilisées par le package fancybox

```
}
```

Pour le reste, les commandes par défaut nous conviennent. Voila le résultat de notre appel si on définit la commande \titre comme contenant le mot Plage (choisi parce qu'il a des lettres montantes et descendantes).

Plage

Ceci est un texte peu utile, mais qui est encadré, comme ça on peut voir que le cadre est bel et bien coupé entre les deux colonnes de mon exemple. Je m'attends à un résultat curieux, voire laid.

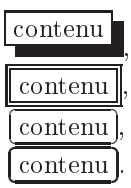
5.9 Le package fancybox

Ce package, que nous devons à Timothy VAN ZANDT [210], offre une pléthore de commandes concernant l'utilisation de boîtes plus ou moins encadrées et l'utilisation de texte en verbatim (comme produit par la commande \verb ou l'environnement verbatim) dans un grand nombre de situations où c'est normalement impossible avec les commandes habituelles de LaTeX.

Je n'aborderai ici que les commandes les plus utiles à l'utilisateur que tu es, en particulier, je ferai l'impasse sur tout ce qui concerne le verbatim, matière en laquelle je préfère des solutions plus légères comme les environnements alltt, ou la commande \Cde (voir section 12 page 537 à ce sujet, au chapitre des trucs et astuces).

5.9.1 Les jolis cadres

fancybox offre quatre nouvelles commandes d'encadrement rigolotes :
\shadowbox{contenu},
\doublebox{contenu},
\ovalbox{contenu}, et
\Ovalbox{contenu},
qui produisent respectivement :



, , , et .

Les longueurs régissant l'épaisseur des différents traits des cadres sont listées au tableau 5.3 page précédente.

La taille des coins pour `\ovalbox` et `\Ovalbox` peut être réglée de deux manières :

```
\cornersize{0.5}
\cornersize*{1cm}
```

la première variante indiquant que les coins doivent occuper 50% du plus petit des deux côtés, alors que la deuxième indique que les coins[147] doivent occuper `1cm`. LaTeX ne disposant dans sa batterie de cuisine que de quarts de cercles de taille fixe, il fera de son mieux pour te satisfaire sans forcément y parvenir.

5.9.2 Boîtes sauvegardées

Le package `fancybox` offre une alternative amusante à l'environnement `lrbox`, l'environnement `Sbox`. Sur le principe, ça se ressemble : ça sauve le contenu dans une boîte qu'on peut réutiliser plus tard. Simplement, il y a quelques différences :

- dans `Sbox` tu ne peux pas choisir la boîte dans laquelle tu sauves ;
- dans `Sbox`, tu sauves de manière globale : si tu sauves un contenu dans une boîte alors que tu es dans un environnement, ce contenu s'y trouve toujour quand tu sors de l'environnement en question (ce n'est pas le cas pour `lrbox`) ;
- avec `Sbox`, tu ne peux te servir qu'une et une seule fois du contenu, le fait de t'en servir l'efface définitivement[148] ; pour utiliser le contenu de l'environnement, il faut appeler `\TheSbox`.

Le seul intérêt de cet environnement, à dire vrai, c'est de pouvoir s'en servir pour en définir d'autres. Par exemple, un environnement comme `minipage`, mais pour faire des cadres, peut s'obtenir comme ça :

```
\newenvironment{Frame}
 {\begin{Sbox}}
 {\end{Sbox}\frame{\TheSbox}}
```

Vérifions tout de suite que cela marche bien sur un exemple :

```
\begin{Frame}
\begin{minipage}{0.5\linewidth}
  Contrairement aux encadrements classiques, celui-ci
  accepte la commande \verb'\verb'.
\end{minipage}
\end{Frame}
```

Qui produit :

Contrairement aux encadrements classiques, celui-ci accepte la commande `\verb`.

[147] Entendons nous bien, l'ensemble des deux coins à chaque fois et pas chaque coin individuellement.

[148] Avec `lrbox` tu n'as pas de moyen simple de vider le contenu sauvegardé, sauf à faire un appel à : `\savebox{\maboite}{}`

5.9.3 Autres environnements

Le package `fancybox` propose quelques autres environnements, en lien avec le fait d'encadrer, dont un bon nombre qu'il vaut mieux ne pas utiliser, parce que d'autres solutions plus puissantes sont apparues depuis.

Ainsi, les environnement `Bcenter`, `Bflushleft` et `Bflushright` te permettent de mettre en forme des boîtes qui ont exactement comme largeur la longueur de la plus longue des lignes. Cette propriété est pratique si on souhaite encadrer, mais elle est plus intéressante à obtenir avec le package `varwidth`.

De même, le package définit des environnements `Bitemize`, `Benumerate` et `Bdescription` qui seront avantageusement remplacés par le package `varwidth`. En particulier, ces environnements ne sont pas les vraies listes de LaTeX, mais des approximations, et du coup tous les paramètres de mise en forme (ou presque) sont négligés, et ils ne sont pas pris en compte par `babel`.