

Formation L^AT_EX — Niveau avancé

Vincent Feuvrier

10 avril 2009

<http://www.math.u-psud.fr/~feuvrier/enseignement/2008/CIES>

1 Types de documents supplémentaires en L^AT_EX

1.1 Manipulation de fichiers PostScript et PDF

Changer la taille du papier d'un fichier PostScript

Le programme `psresize` permet de changer la taille de papier utilisée pour l'impression.

Usage

`psresize [-wWidth] [-hHeight] [-pPaperSize] Input.ps Output.ps`

- `Width` et `Height` permettent de spécifier la taille du papier de sortie (par exemple `-w10in`);
- `PaperSize` définit une taille de papier standard (par exemple `a4` ou encore `letter`);
- les mêmes options mais en majuscule (`-W`, `-H` ou encore `-P`) permettent de définir la taille du papier d'entrée (si différente de `a4`).

Plusieurs pages par feuille avec `psnup`

Le programme `psnup` permet de rassembler plusieurs pages d'un document PostScript par feuille.

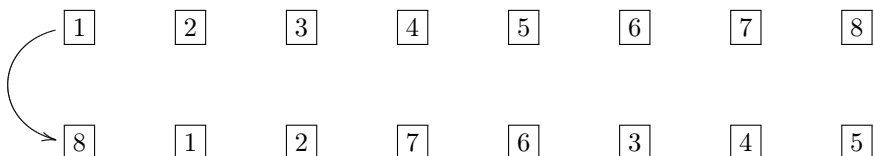
Usage

`psnup -p[PaperSize] -[NumberPerPage] [-bMargin] Input.ps Output.ps`

- `PaperSize` définit la taille de papier qui sera utilisée (par exemple `a4` ou encore `letter`);
- `NumberPerPage` est le nombre de pages du document original qu'on souhaite disposer sur chaque page du document final (par exemple 2, 4 ou 8);
- `Margin` permet d'ajouter une marge supplémentaire autour des pages du document original (éventuellement négative).

Changer l'ordre des pages pour une reliure centrale avec [psbook](#)

Avant `psnup` on peut utiliser `psbook` qui va changer l'ordre des pages :



Usage

```
psbook Input.ps Output.ps
```

Concaténer des fichiers PostScript

S'il est installé, on peut utiliser `psmerge` :

Usage

```
psmerge [-oOutput.ps] Input1.ps Input2.ps ...
```

Sinon avec GhostScript :

Usage

```
gs -dBATCH -dNOPAUSE -q -sDEVICE=pswrite -sOutputFile=Output.ps Input1.ps  
Input2.ps ...
```

▷ Sous Windows, remplacer `gs` par `gswin32c` (dans le répertoire `/bin` de l'installation de GhostScript).

Concaténer des fichiers PDF

Avec GhostScript :

Usage

```
gs -dBATCH -dNOPAUSE -q -sDEVICE=pdfwrite -sOutputFile=Output.pdf Input1.pdf  
Input2.pdf ...
```

Extraire un intervalle de pages d'un document

Exemple avec `psselect`, pour extraire les pages de 5 à 10 et de 15 à 20 d'un document PostScript :

Usage

```
psselect -p5-10,15-20 Input.pdf Output.pdf
```

Exemple avec GhostScript, pour extraire les pages de 5 à 10 d'un document PDF :

Usage

```
gs -dBATCH -dNOPAUSE -q -sDEVICE=pdfwrite -sOutputFile=Output.pdf -dFirstPage=5  
-dLastPage=10 Input.pdf
```

Vectoriser les fontes d'un fichier PostScript ... pour importer des formules dans un logiciel de dessin par exemple

Avec `pstoedit` :

Usage

```
pstoedit -dt -f ps Input.ps Output.ps
```

Exemple d'utilisation pour importer une formule en PDF dans Inkscape :

Commandes

```
latex formule.tex  
dvips formule.dvi  
pstoedit -dt -f ps formule.ps out.ps  
ps2pdf out.ps
```

La formule est alors prête à être incluse dans un logiciel de dessin, sous forme de courbes (et avec l'aspect des fontes originales de L^AT_EX).

Exercice

1.2 Listings et programmes informatiques

L'extension `listings`

Trois manières d'écrire du code, d'usage similaire aux environnements `verbatim` :

```
- \lstinline@let rec factoriel n = if n<2 then 1 else n*factoriel (n-1);;@ af-  
fichera let rec factoriel n = if n<2 then 1 else n*factoriel (n-1);; au mi-  
lieu du texte;
```

- `\begin{lstlisting}...\end{lstlisting}` en mode « display » donnera par exemple :

```
function Factoriel(const n: Integer): Integer;
var
  i: Integer;
begin
  Result:=1;
  for i:=2 to n do
    Result:=Result*i;
  end;
```

- `\lstinputlisting{File}` fera la même chose en affichant le contenu d'un fichier source externe.

Listings flottants

Code

```
\begin{lstlisting}[
  language=C,
  frame=Tb,
  float,
  caption=Un exemple flottant
]
/* Fonction factorielle */
int factoriel(int n){
  int s=1;
  for (;n;--n)
    s*=n;
  return s;
}
\end{lstlisting}
```

Sortie

Listing 1 – Un exemple flottant

```
/* Fonction factorielle */
int factoriel(int n){
  int s=1;
  for (;n;--n)
    s*=n;
  return s;
}
```

Contrôle de l'apparence

Deux manières de contrôler la façon dont le code s'affiche :

- au niveau des commandes elles-mêmes, en argument optionnel

Code

```
\begin{lstlisting}[language=html,basicstyle=\itshape,
  keywordstyle=\color{magenta}]
<HTML><HEAD><TITLE>Ma page web</TITLE></HEAD><BODY><H1>Ma page
  web</H1></BODY></HTML>
\end{lstlisting}
```

Sortie
`<HTML><HEAD><TITLE>Ma page web</TITLE></HEAD><BODY><H1>Ma page
web</H1></BODY></HTML>`

- au niveau global

Code

```
\lstset{language=Matlab,numbers=left,numberstyle=\tiny\white}
\begin{lstlisting}
function result=factoriel(n)
if (n<=1) result=1;
else result=n*factoriel(n-1);
end;
\end{lstlisting}
```

Sortie

```
function result=factoriel(n)
if (n<=1) result=1;
else result=n*factoriel(n-1);
end;
```

Contrôle de l'apparence

Quelques options, parmi d'autres :

- `language=[dialect]language` pour spécifier le langage utilisé (par exemple [LaTeX]TeX ou [ANSI]C);
- `basicstyle=, identifierstyle=, commentstyle=, stringstyle=, keywordstyle=, directivestyle=` pour le style de texte des différents éléments syntaxiques (couleur, fontes);
- `showspaces = <true|false>` (ou encore `showstringspaces`) pour imprimer ou non les espaces avec le symbole `_`;
- `numbers = <none|left|right>` pour la numérotation des lignes;
- `frame = <trblTRBL>` pour tracer sélectivement certains côtés du rectangle englobant le listing.

Contrôle de l'apparence

On peut définir des styles globaux :

Code

```
\lstdefinestyle{mylatex}{
  language=[LaTeX]TeX,
  breaklines,showspaces,
  basicstyle=\footnotesize\color{black}\ttfamily,
  keywordstyle=\color{red},
  commentstyle=\color{green},
  identifierstyle=\color{blue},
  frame=tblr,
  numbers=left,
  numberstyle=\color{white}
}
\begin{lstlisting}[style=mylatex,title=Style_mylatex]
%Un commentaire
Comment_croyez-vous_qu'a_été_défini_l'environnement_LaTeX_qui_
affiche_ceci?
```

```
\end{lstlisting}
```

Sortie

Style mylatex

```
%Un commentaire
Comment_croyez-vous_qu'a_été_défini_l'environnement_\LaTeX_\qui_
affiche_ceci?
```

Contrôle de l'apparence

On peut définir des environnements raccourcis, sur le modèle de `\newenvironment` :

Code

```
\lstnewenvironment{mylatexenv}{
  \lstset{style=mylatex}
  \begin{center}---Début_du_code---\end{center}
}{
  \begin{center}---Fin_du_code---\end{center}
}
\begin{mylatexenv}
\lstnewenvironment{mylatexenv}{
  \lstset{style=mylatex}
  \begin{center}---Début_du_code---\end{center}
}{
  \begin{center}---Fin_du_code---\end{center}
}
\end{mylatexenv}
```

Sortie

— Début du code —

```
\lstnewenvironment{mylatexenv}{
  \lstset{style=mylatex}
  \begin{center}---Début_du_code---\end{center}
}{
  \begin{center}---Fin_du_code---\end{center}
}
```

— Fin du code —

Caractère d'échappement

L'option `escapechar=` définit un caractère permettant de revenir au mode \LaTeX , par exemple pour écrire des formules dans les commentaires.

Code

```
\begin{lstlisting}[language=Java,escapechar=@]
class Factoriel{
/*_Cette_fonction_utilise_la_relation_@_begin{equation}n!=\prod_{i=1}^ni\end{equation}_@_*/
static int factoriel(int n){
  int res=1;
  for(int i=1;i<=n;i++)
    res=res*i;
  return res;
}
```

```

    }
}
\end{lstlisting}

```

Sortie

```

class Factoriel{
/* Cette fonction utilise la relation

```

$$n! = \prod_{i=1}^n i \quad (1)$$

```

*/
static int factoriel(int n){
    int res=1;
    for(int i=1;i<=n;i++){
        res=res*i;
    }
    return res;
}
}

```

On peut aussi utiliser `mathescape=true`, dans ce cas le symbole `$` permet de passer en mode mathématique. L'option `texcl=true` permet de passer automatiquement en mode L^AT_EX dans les commentaires.

Les extensions `algorithm` et `algorithmic`

Pour écrire du pseudo-code.

Code

```

\begin{algorithm}
\caption{$\operatorname{factorial}(n)$}\label{alg:factorial}
\begin{algorithmic}[1]
\REQUIRE An integer $n \geq 0$.
\ENSURE The value of $n!$.
\medskip
\IF{$n=0$}
\RETURN 1
\ELSE
\RETURN $n \cdot \operatorname{factorial}(n-1)$
\ENDIF
\end{algorithmic}
\end{algorithm}

```

Sortie

Algorithm 1 `factorial(n)`

Require: An integer $n \geq 0$.

Ensure: The value of $n!$.

```

1: if  $n = 0$  then
2:   return 1
3: else
4:   return  $n \cdot \text{factorial}(n - 1)$ 
5: end if

```

Exercice

Solution Écrire un « quine » en **L^AT_EX** (c'est à dire un programme qui affiche son propre code-source).

Code

```
\documentclass{article}

\usepackage{listings}

\lstset{language=[LaTeX]TeX}

\pagestyle{empty}

\begin{document}
\input{listing.tex}
\end{document}
```

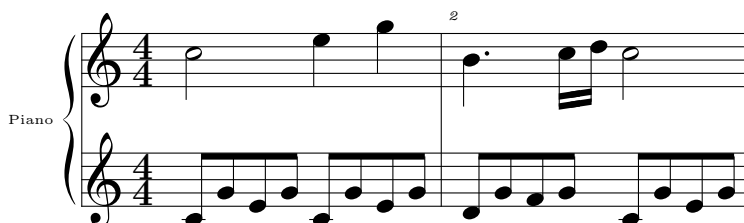
1.3 Partitions de musique

Écrire des partitions de musique avec l'extension [musixtex](#)

Code

```
\begin{music}
\tiny
\parindent1cm
\instrumentnumber{1}\%_a_single_instrument
\setname1{Piano}\%_whose_name_is_Piano
\setstaves1{2}\%_with_two_staves
\generalmeter{\meterfrac44}\%_4/4_meter_chosen
\startextract\%_starting_real_score
\notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl\j\en
\notes\ibu0f0\qb0{cge}\tbu0\qb0g|\ql\l\sk\ql\l\en
\bar
\notes\ibu0f0\qb0{dgf}|\qlp\l\en
\notes\tbu0\qb0g|\ibbl1j3\qb1j\tbl1\qb1k\en
\notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl\j\en
\endextract\%_terminate_excerpt
\end{music}
```

Sortie



1.4 Lettres et courrier

La classe [letter](#)

Exemple minimaliste

```
\documentclass[francais]{letter}

\usepackage[francais]{babel}

\address{...\\...\\...}
\signature{...}

\begin{document}
  \begin{letter}{M...\\...\\...}
    \opening{Cher...}
    ...
    \closing{Bien cordialement}
  \end{letter}
  \begin{letter}{M...\\...\\...}
    \opening{Cher...}
    ...
    \closing{Bien cordialement}
  \end{letter}
\end{document}
```

La classe `letter`

Commandes supplémentaires à utiliser juste après `\closing` :

- `\cc{...}` s'il y a des destinataires en copie ;
- `\encl{...}` s'il y a des pièces jointes ;
- `\ps{PS: ...}` s'il y a un post-scriptum (ou autre).

Château de Moulinsart

12 avril 2009

Jean Dupond
Château de Moulinsart
Belgique

Cher homonyme,

Ça ne peut plus durer, il faut que l'un de nous change de nom. Sans mentionner les quiproquos ridicules que cela engendre lorsque quelqu'un s'adresse à nous, nous sommes déjà la risée de toute la Belgique.

Étant l'aîné selon l'ordre alphabétique, je suggère que ce soit vous qui changiez.

Bien cordialement

Jean Dupont

Copie à : Tintin et Milou

P. J. : Formulaire administratif de changement de nom de famille

PS : Inutile de discuter, la décision est prise

La classe **lettre**

Elle fonctionne quasiment comme **letter**, avec quelques commandes supplémentaires, et respecte mieux les conventions françaises.

On peut définir des paramètres par défaut pour toutes ses lettres.

À mettre dans un fichier `default.ins` ou juste avant `\opening` dans chaque lettre

```
%Nom et signature de l'expéditeur
\name{...}
\signature{...}
%Coordonnées de l'expéditeur
\address{...}
\location{...}
\telephone{...}%ou\notelephone
\fax{...}%ou\nofax
\email{...}%ou\nomail
%lieu d'expédition des lettres
\lieu{...}
```

La classe `lettre` Commandes propres à chaque lettre

Commandes à utiliser juste avant `\opening` :

– `\conc{...}` pour définir l'objet de la lettre.

Commandes à utiliser juste après `\closing` :

– `\cc{...}` s'il y a des destinataires en copie ;

– `\encl{...}` s'il y a des pièces jointes ;

– `\ps{PS:}{...}` s'il y a un post-scriptum.

Pour faire un téléfax il suffit de remplacer

```
\begin{letter}{destinataire}
```

par

```
\begin{telefax}{numéro}{destinataire}
```

Château de Moulinsart
Moulinsart
Belgique
Tél. 123-456-789
Fax : 123-456-789
E-Mail : dupont@milou.com

Château de Moulinsart, le 12 avril 2009

Jean Dupont

Objet : Homonymie

Cher homonyme,

Ça ne peut plus durer, il faut que l'un de nous change de nom. Sans mentionner les quiproquos ridicules que cela engendre lorsque quelqu'un s'adresse à nous, nous sommes déjà la risée de toute la Belgique.

Étant l'ainé selon l'ordre alphabétique, je suggère que ce soit vous qui changiez.

Bien cordialement

Jean Dupont

PS : Inutile de discuter

P.j. Formulaire de demande de changement de nom

C.c. Tintin et Milou

Château de Moulinsart

T É L É F A X

TÉLÉPHONE : 123-456-789

TÉLÉFAX : 123-456-789

E-Mail : dupont@milou.com

À : Jean Dupond

Télécopie : 987-654-321

De : Moulinsart
Belgique

Nombre de pages : 1

En cas de mauvaise transmission, appelez s.v.p. l'opérateur téléfax

Château de Moulinsart, le 12 avril 2009

Objet : Homonymie

Cher homonyme,

Ça ne peut plus durer, il faut que l'un de nous change de nom. Sans mentionner les quiproquos ridicules que cela engendre lorsque quelqu'un s'adresse à nous, nous sommes déjà la risée de toute la Belgique.

Étant l'ainé selon l'ordre alphabétique, je suggère que ce soit vous qui changiez.

Bien cordialement

Jean Dupont

PS : Inutile de discuter

P.j. Formulaire de demande de changement de nom

C.c. Tintin et Milou

Exercice

- écrire une (courte!) lettre adressée à la personne de votre choix ;
- la transformer en un téléfax.

1.5 Affiches et posters

La classe `a0poster`

Pour écrire sur de grandes tailles de papier avec des fontes adaptées.

Les options sont les suivantes :

- `a0b`, `a0`, `a1`, `a2` ou `a3` pour la taille de papier ;
- `landscape` ou `portrait` pour l'orientation.

La classe introduit trois tailles de fontes supplémentaires : `\veryHuge`, `\VeryHuge` et `\VERYHuge`.

L'extension `textpos`

Deux environnements supplémentaires pour disposer des éléments à un endroit précis :

- `\begin{textblock}{hsize}[x0,y0](hpos,vpos)` permet de placer du texte dans une boîte de largeur `hsize` dont l'origine est décalée de `(hpos, vpos)` par rapport à la position courante ;

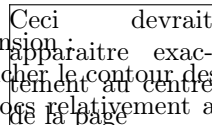
Le décalage et la largeur sont exprimés en une unité correspondant par défaut au 1/16 de la page.

Par défaut l'origine `(x0,y0)` est le coin supérieur gauche de la boîte, elle peut être déplacée en donnant une fraction des dimensions de la boîte (par exemple `(0.5,1)` pour le milieu du bas de la boîte).

- L'environnement `textblock*` fait la même chose, mais avec des longueurs en unités standard ;
- `\TPGrid[marginx,marginy]{nx}{ny}` modifie les unités de façon à découper la page en `nx × ny` (éventuellement avec des marges).

L'extension `textpos`

- Quelques options de l'extension :
- `showboxes` permet d'afficher le contour des boîtes ;
 - `absolute` dispose les blocs relativement au coin supérieur gauche de la page (au lieu de la position courante) ;
 - `overlay` dispose les blocs par-dessus le texte normal du document.



Code

```
\begin{textblock}{2}[0.5,0.5](8,8)
  On a utilisé les options absolute, showboxes et overlay
  Ceci devrait apparaître exactement au centre de la page
\end{textblock}
```

Sortie

1.6 Les extensions `pstricks`

Les extensions `pstricks`

Il s'agit d'un ensemble d'extensions pour réaliser des dessins dans des domaines très variés :

- `pstricks` pour les figures de base ;

- `pstricks-add` corrige certains bugs et ajoute des fonctionnalités supplémentaires (notation infix pour les formules entre autres) ;
- `pst-grad` pour colorier avec des gradients de couleur ;
- `pst-text` pour la manipulation du texte ;
- `pst-plot`, `pst-2dplot` et `pst-3dplot` pour des graphiques en 2D ou 3D, avec support Matlab ;
- `pst-all` charge la plupart de ces extensions (mais ralentit aussi la compilation) ;
- `pst-tree` pour dessiner des arbres (syntaxiques ou autres) ;
- `pst-optic` pour dessiner des schémas optiques ;
- ...

Les extensions `pstricks`

Elles utilisent le langage bas niveau des fichiers PostScript, on ne peut donc pas utiliser Pdf \LaTeX et la plupart des visionneuses DVI ne les affichent pas.

Généralement on compile avec \LaTeX , puis on convertit avec `dvips` (et éventuellement `ps2pdf`).

Certains logiciels de dessin vectoriel permettent d'exporter du code PSTricks. Il est préférable d'utiliser `\input` ou `\include` depuis un fichier annexe à l'endroit désiré.

Si on souhaite malgré tout utiliser Pdf \LaTeX , il est possible de compiler séparément les dessins PSTricks, les convertir en PDF puis les inclure avec `\includegraphics` : c'est ce que fait l'extension `pdftricks`.

L'extension `pdftricks`

Définit les environnements `psinputs` pour les parties du préambule spécifiques à PSTricks et `pdfpic` pour les dessins dans le document.

Structure générale

```
\documentclass{article}

\usepackage{pdftricks}

\begin{psinputs}
  \usepackage{pstricks,pst-3d}
\end{psinputs}

\begin{document}

\begin{pdfpic}
  \begin{pspicture}(0,0)(10,10)
    ...
  \end{pspicture}
\end{pdfpic}

\end{document}
```

Il faudra rajouter l'option `-shell-escape` (ou `--enable-write18` sous Windows/MikTeX) sur la ligne de commande pour autoriser L^AT_EX à lancer les compilations externes nécessaires.

Primitives

Par défaut, les primitives de dessin PSTricks n'occupent aucune place, et sont affichées sur la ligne de base à l'emplacement du prochain caractère.

Code

```
\ttfamily
%Le mot barré va être affiché par-dessus le segment
Un mot \psline(0,0)(2.5em,1ex)barréen diagonale.
```

Sortie

Un mot **barré** en diagonale.

Dessins PostScript

L'environnement `pspicture` permet de créer des assemblages de primitives dans une boîte de taille fixée.

`\begin{pspicture}(x0,y0)(x1,y1)` pour une boîte de taille $(x1 - x0) \times (y1 - y0)$.

La boîte sera disposée à la place du prochain caractère, de façon à ce que l'origine $(x0,y0)$ soit sur la ligne de base du texte.

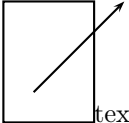
Si l'on ne les précise pas, les unités par défaut sont le centimètre pour les longueurs et le degré pour les angles.

Si l'on ne précise pas le premier argument $(x0,y0)$ alors ce sera $(0,0)$.

Code

```
Au milieu du
\begin{pspicture}(-1,-1)(2,3)
\psframe(-1,-1)(2,3)
%Cette flèche va dépasser à droite de 1 cm
\psline{->}(0,0)(3,3)
\end{pspicture}%
texte.
```

Sortie

Au milieu du  texte.

Options globales

À tout endroit du document on peut changer certains paramètres avec `\psset{param1=value1,param2=value2,...}`, en particulier :

- `unit = \langle longueur \rangle` pour changer l'unité de mesure par défaut (qui vaut 1cm par défaut) ;
- `origin = \{ \langle x \rangle , \langle y \rangle \}` pour déplacer l'origine ($\{0,0\}$ par défaut) ;
- `algebraic = \langle true, false \rangle` pour activer ou non la notation infixe des formules (par défaut en notation polonaise). Nécessite l'extension `pstricks-add`.

La portée des changements est limitée au groupe en cours.

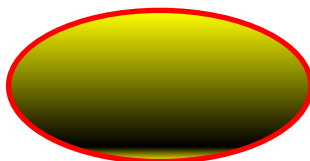
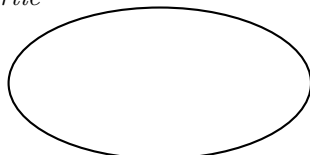
`\newpsstyle{nom}{paramètres}` permet de définir des ensembles de paramètres qu'on peut activer facilement avec `\psset{style = \langle nom \rangle}`.

Commandes personnalisées

Code

```
\newpsobject{superpsellipse}{psellipse}{linecolor=red,linewidth=2pt
,fillstyle=gradient,gradbegin=yellow,gradend=black}
\begin{pspicture}(0,0)(5,5)
  \uu%Ellipse normale
  \uu\psellipse(2.5,4)(2,1)
  \uu%Ellipse personnalisée
  \uu\superpsellipse(2.5,1)(2,1)
\end{pspicture}
```

Sortie



Couleurs

- `\newgray{nom}{valeur}` permet de définir un nouveau niveau de gris (qui peut ensuite être activé n'importe où avec `\nom`). Les niveaux suivants sont prédéfinis :

`black` `darkgray` `gray` `lightgray` `white`

- `\newrgbcolor{nom}{R G B}` permet de définir une nouvelle couleur RGB.


Les couleurs suivantes sont prédéfinies :

`red` `green` `blue` `cyan` `magenta` `yellow` `brown` `lime` `olive`
`orange` `pink` `purple` `teal` `violet`

Code

```
\newrgbcolor{CHAMEAU}{0.9\0.4\0.1}
{\CHAMEAU\couleur}\désactivée
\psframe*[linecolor=CHAMEAU](0,0)(1,1ex)
```

Sortie

couleur désactivée 

Systèmes de coordonnées

La commande `\SpecialCoor` permet d'utiliser des systèmes coordonnées supplémentaires :

- les coordonnées cartésiennes usuelles (x,y) disponibles par défaut ;
- les coordonnées polaires (rayon;angle) ;
- etc...

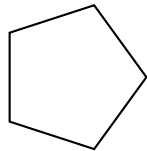
La commande `\degrees[valeur]` permet de travailler avec une unité d'angle correspondant à $\frac{1}{valeur}$ du cercle trigonométrique.

Code

```
\SpecialCoor
\degrees[5]

\begin{pspicture}(-2.5,-1)(2.5,1)
\ps%Dessin facile d'un pentagone régulier
\pspolygon(1;0)(1;1)(1;2)(1;3)(1;4)
\end{pspicture}
```

Sortie



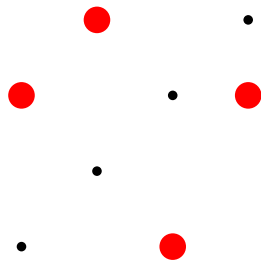
Points

`\psdots[options](x1,y1)(x2,y2) ... (xn,yn).`

Code

```
\begin{pspicture}(0,0)(5,5)
\psdots(1,1)(2,2)(3,3)(4,4)
\psdots[linecolor=red,dotsize=10pt](2,4)(3,1)(4,3)(1,3)
\end{pspicture}
```

Sortie



Options spécifiques des points

- `dotstyle = \langle o, *, +, x, triangle, triangle*, square, square*, diamond, diamond*, pentagon, pentagon*, ... \rangle`

pour obtenir respectivement



- `dotscale = \langle valeur \rangle` pour augmenter la taille par défaut des points.

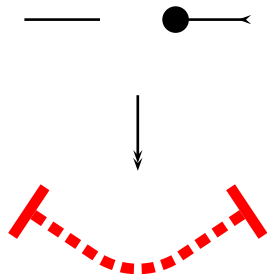
Segments et lignes brisées

`\psline[options]{extrémités}(x1,y1)(x2,y2) ... (xn,yn).`


Code

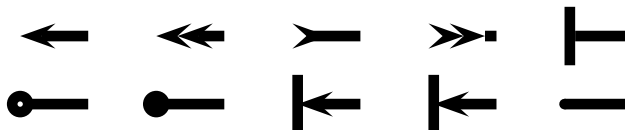
```
\begin{pspicture}(0,0)(5,5)
  \psline(1,4)(2,4)
  \psline{*-<}(3,4)(4,4)
  \psline{->>}(2.5,3)(2.5,2)
  \psline[linearc=1,linestyle=dashed,linecolor=red,linewidth=4pt]
    {|-|}(1,1.5)(2.5,0.5)(4,1.5)
\end{pspicture}
```

Sortie



Options spécifiques des lignes

- `linewidth` = $\langle \text{longueur} \rangle$ pour la largeur de trait ;
- `linecolor` = $\langle \text{couleur} \rangle$ pour la couleur des traits (ou du remplissage pour les commandes étoilées) ;
- `linestyle` = $\langle \text{none, solid, dashed, dotted} \rangle$ pour le type de trait :

- `doubleline` = $\langle \text{true, false} \rangle$ pour doubler le trait ;
- `showpoints` = $\langle \text{true, false} \rangle$ pour afficher ou non les extrémités des éléments ;
- `linearc` = $\langle \text{longueur} \rangle$ pour le rayon de courbure utilisé pour arrondir les angles ;
- les extrémités sont de la forme $\{ \dots - \dots \}$ avec les possibilités $<$, $<<$, $>$, $>>$, $|$, o , $*$, $|<$, $>|$ ou c :



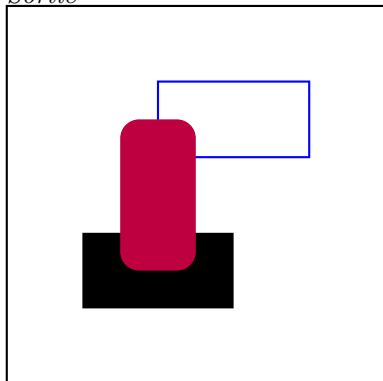
Rectangles

`\psframe` [\[options\]](#) (x1,y1) (x2,y2), ou la variante étoilée `\psframe*` pour un rectangle plein.

Code

```
\begin{pspicture}(0,0)(5,5)
  \psframe(0,0)(5,5)
  \psframe*(1,1)(3,2)
  \psframe[fillcolor=red,linecolor=blue](2,3)(4,4)
  \psframe*[framearc=0.5,linecolor=purple,fillstyle=solid](1.5,1.5)
    (2.5,3.5)
\end{pspicture}
```

Sortie



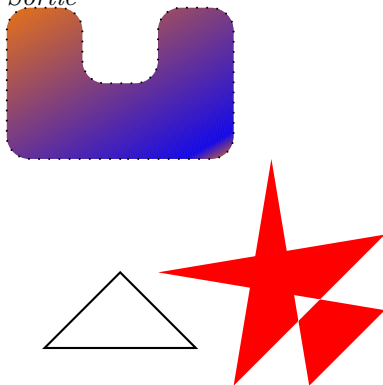
Polygones

On utilise `\pspolygon[options](x1,y1)(x2,y2) \dots (xn,yn)`, ou la variante étoilée `\pspolygon*` pour un polygone plein.

Code

```
\begin{pspicture}(0,0)(5,5)
\pspolygon(0.5,0.5)(1.5,1.5)(2.5,0.5)
\pspolygon[linearc=0.3,linestyle=dotted,fillstyle=gradient,
gradbegin=orange,gradend=blue,gradangle=30](0,3)(0,5)(1,5)(1,4)
(2,4)(2,5)(3,5)(3,3)
\pspolygon*[linecolor=red](3,0)(3.5,3)(4,0)(5,1)(2,1.5)(5,2)
\end{pspicture}
```

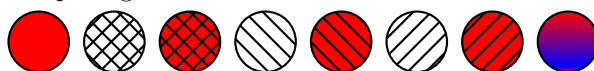
Sortie



Options spécifiques de remplissage

- `fillcolor` = `<couleur>` pour la couleur de remplissage ;
- `fillstyle` = `<solid,crosshatch,crosshatch*,vlines,vlines*, pour hlines,hlines*,gradient>`

le motif de remplissage :



- `hatchcolor` = `<couleur>` et `hatchwidth` = `<longueur>` pour la couleur et la largeur de trait des hachures ;
- `hatchsep` = `<longueur>` et `hatchangle` = `<angle>` pour l'espace et l'inclinaison des hachures ;
- `gradbegin` = `<couleur>` et `gradend` = `<couleur>` pour les couleurs de dégradé (nécessite `pst-grad`) ;
- `gradmidpoint` = `<valeur>` pour décaler le dégradé ;
- `framearc` = `<longueur>` pour le rayon de courbure utilisé pour arrondir les angles.

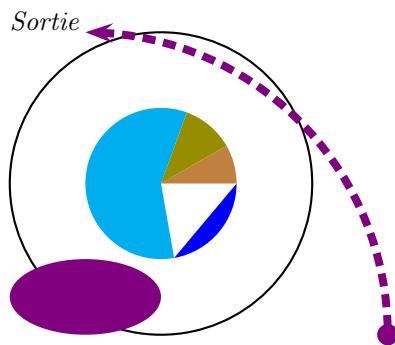
Cercles, ellipses et arcs

- `\pscircle[options](x,y){r}` pour un cercle (ou `\pscircle*` pour un disque) ;
- `\psellipse[options](x,y)(rayon horizontal,rayon vertical)` pour une ellipse (ou `\psellipse*` pour un disque elliptique) ;

- `\psarc[options]{extrémités}(x,y){r}{angle1}{angle2}` pour l'arc de cercle limité par les angles (ou `\psarc*` pour remplir la région délimitée par la corde) ;
- `\pswedge[options](x,y){r}{angle1}{angle2}` pour une part de camembert (`\pswedge*` pour la remplir).

Code

```
\begin{pspicture}(0,0)(5,4)
  \u\pscircle(2,2){2}
  \u\psellipse*[linecolor=violet](1,0.5)(1,0.5)
  \u\psarc[linecolor=violet,linewidth=3pt,linestyle=dashed]{*->}(1,0)
    {4}{0}{90}
  \u\pswedge*[linecolor=brown](2,2){1}{0}{30}
  \u\pswedge*[linecolor=olive](2,2){1}{30}{70}
  \u\pswedge*[linecolor=cyan](2,2){1}{70}{280}
  \u\psarc*[linecolor=blue](2,2){1}{280}{360}
\end{pspicture}
```



Courbes

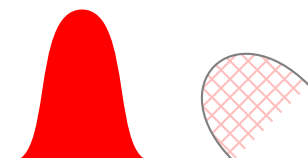
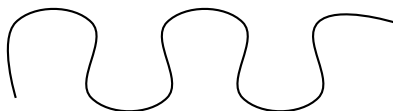
Les variantes étoilées ferment la courbe avec une corde et remplissent la région.

- `\pscurve[options](x1,y1)(x2,y2) ... (xn,yn)` pour tracer une courbe d'interpolation passant par tous les points (`\psccurve` ferme la courbe) ;
- `\psecurve` passe par tous les points sauf ceux des extrémités (qui servent à contrôler les tangentes) ;
- `\psbezier` trace une courbe de Bézier.

Code

```
\begin{pspicture}(0,0)(5,4)
  \u\pscurve(0,3)(0,4)(1,4)(1,3)(2,3)(2,4)(3,4)(3,3)(4,3)(4,4)(5,4)
  \u\psecurve*[linecolor=red](1,1)(1,0)(2,2)(3,0)(3,1)
  \u\psbezier[linecolor=gray,fillstyle=crosshatch,hatchcolor=pink]
    (4,0)(3,1)(4,2)(5,1)
\end{pspicture}
```

Sortie



Translations et rotations

- `\rput[référence]{rotation}(x,y){texte}` pour placer du **texte** relativement au point (x,y) . La **référence** (par défaut centrée) peut être une combinaison de B (ligne de base), b (bas) ou t (haut) avec l (gauche) ou r (droite);
- `\uput{distance}[direction]{rotation}(x,y){texte}` pour placer du **texte** à distance donnée dans une direction.

Code

```
\begin{pspicture}(0,0)(5,4)
  \psdots[linecolor=red](1,4)
  \rput[t](1,4){\ttfamily\textbackslash\rput(1,4)}
  \rput{45}(2.5,2){Au centre}
  \uput{5}[30]{-90}(0,0){\huge\ttfamily\textbackslash\uput}
  \uput{4}[-70](0,4){\color{orange}\huge\ttfamily\textbackslash\uput}
\end{pspicture}
```

Sortie
`\rput(1,4)`

Au centre




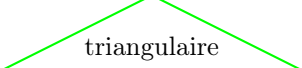
`\uput`

`\uput`

Cadres

Ces commandes tracent un cadre autour de leur argument et les variantes étoilées remplissent le cadre :

- `\psframebox{texte}` pour un cadre rectangulaire (équivalent de `\fbox`);
- `\psdblframebox{texte}` pour un cadre rectangulaire à filet double;
- `\psshadowbox{\white texte}` pour un cadre rectangulaire ombré;

- `\pscirclebox{texte}` pour un cadre  circulaire ;
- `\psovalbox{texte}` pour un cadre  oval ;
- `\psdiabox{texte}` pour un cadre  losange ;
- `\pstribox{texte}` pour un cadre  triangulaire .

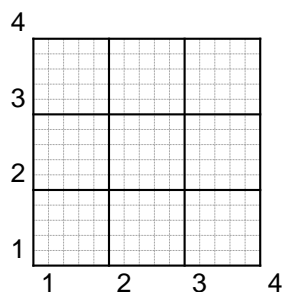
Quadrillage

`\psgrid[options](x0,y0)(xmin,ymin)(xmax,ymax)` affiche un quadrillage sur le pavé $[xmin, ymin] \times [xmax, ymax]$. L'unité des graduations est $x0-xmin$ en abscisses et $y0-ymin$ en ordonnées (par défaut 1).

Code

```
\begin{pspicture}(0,0)(5,5)
  \psgrid[subgriddots=10,subgridcolor=gray](1,1)(4,4)
\end{pspicture}
```

Sortie



Options spécifiques des quadrillages

- `subgriddiv = <n>` pour afficher n sous-graduations entre chaque graduation principale ;
- `griddots = <n>` et `subgriddots = <n>` pour remplacer les traits pleins par n points entre chaque graduation ;
- `gridcolor = <couleur>` et `subgridcolor = <couleur>` pour la couleur des graduations ;
- `gridwidth = <longueur>` et `subgridwidth = <longueur>` pour l'épaisseur des traits de graduation ;
- `gridlabels = <longueur>` pour la taille de fontes des étiquettes.

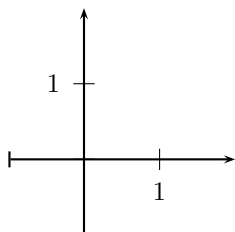
Repères et axes

`\psaxes[options]{extrémités}(x0,y0)(xmin,ymin)(xmax,ymax)` affiche deux axes perpendiculaires dans le pavé $[xmin, ymin] \times [xmax, ymax]$ et se coupant en $(x0, y0)$.

Code

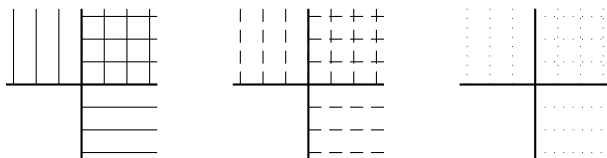
```
\begin{pspicture}(0,0)(5,5)
\psaxes{|->}(2,2)(1,1)(4,4)
\end{pspicture}
```

Sortie



Options spécifiques des axes

- `Dx = <longueur>` et `Dy = <longueur>` pour l'espacement entre les graduations ;
- `comma` pour utiliser la virgule comme séparateur décimal ;
- `labels = <xy, x, y, none>` pour contrôler l'affichage des étiquettes ;
- `ticks = <xy, x, y, none>` pour contrôler l'affichage des graduations ;
- `xAxis = <true, false>` et `yAxis = <true, false>` pour activer ou désactiver un axe ;
- `ticksize = <longueur>` pour la taille des graduations ;
- `ticklinestyle = <solid, dashed, dotted>` pour le style des graduations :



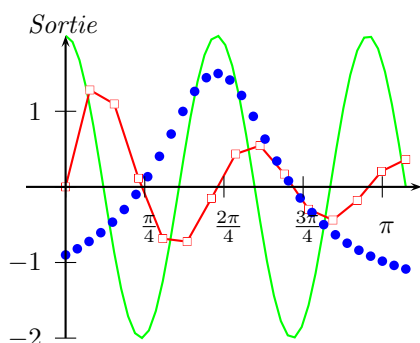
Tracé de fonctions

On utilise `\psplot[options]{xmin}{xmax}{fonction}`.

Par défaut les fonctions sont données en notation polonaise, l'option `algebraic` permet de passer en notation infixe.

Code

```
\begin{pspicture}(-0.5,-2)(4.5,2)
\psplot[linecolor=green]{0}{4.5}{x_180_mul_cos_2_mul}
\psplot[algebraic,linecolor=red,plotpoints=15,showpoints=true,
dotstyle=square]{0}{4.5}{2*sin(3.14159*x)/(x+1)}
\psplot[algebraic,plotpoints=30,plotstyle=dots,dotstyle=*,
linecolor=blue]{0}{4.5}{3/((x-2)^(2)+1)-1.5}
\psaxes[trigLabels=true,trigLabelBase=4,xunit=\pstRadUnit
]{->}(0,0)(-0.5,-2)(4.5,2)
\end{pspicture}
```



Plots

On utilise `\savedata\commande[data]` pour stocker des coordonnées dans une commande.

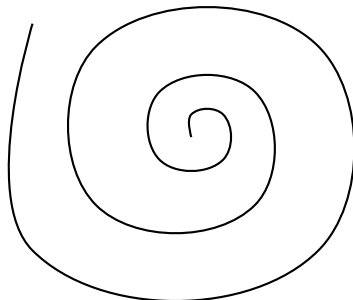
On peut ensuite utiliser `\dataplot[options]{\commande}` pour l’affichage.

L’option `plotstyle` peut prendre les valeurs `dots`, `line`, `polygon`, `curve`, `ecurve` ou `ccurve`.

Code

```
\savedata\MYDATA[0_0_0_1_1_1_1_1_1_1_1_1_1_1_1_2_2_2_2_2_3_3_3_3_3_4_4_4_4_4_5_5_5_5_5]
\begin{pspicture}(-6,-7)(6,6)
\psaxes[trigLabels=true]{->}(0,0)(-6,-7)(6,6)
\dataplot[plotstyle=curve]{\MYDATA}
\end{pspicture}
```

Sortie



Importation de données externes

Les données peuvent être générées par Matlab ou Mathematica par exemple.

Elles doivent être constituées de couples de coordonnées séparés par d'autres caractères (espaces, virgules, etc...).

Les autres caractères (parenthèses, crochets, etc...) sont ignorés.

- `\readdata\commande{fichier}` définit une commande pour stocker les données du fichier (pour utilisation avec `\dataplot`);
- `\fileplot[options]{fichier}` affiche directement les données d'un fichier.

Code

```
\readdata\COEUR{Coeur.txt}

\begin{pspicture}(-2.5,-1)(2.5,2)
uu\fileplot*[linecolor=red]{Coeur.txt}
uu\dataplot[linecolor=pink,unit=1.5cm]{\COEUR}
\end{pspicture}
```

Sortie



Régions

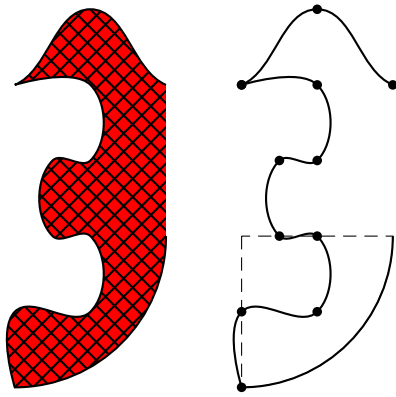
`\pscustom[options]{frontière}`.

Si la frontière spécifiée n'est pas connexe, les extrémités sont reliées par des cordes pour définir la zone de remplissage.

Code

```
\begin{pspicture}(0,0)(5,5)
uu\newcommand\perimeter{
uuuu\psecurve(3,4)(2,4)(1,5)(0,4)(-1,4)
uuuu\pscurve(0,4)(1,4)(1,3)(0.5,3)(0.5,2)(1,2)(1,1)(0,1)(0,0)
uuuu\psarc(0,2){2}{-90}{0}
uu}
uu\pscustom[fillstyle=crosshatch*,fillcolor=red]{\perimeter}
uu\psset{showpoints=true}
uu\rput(3,0){\perimeter}
\end{pspicture}
```

Sortie



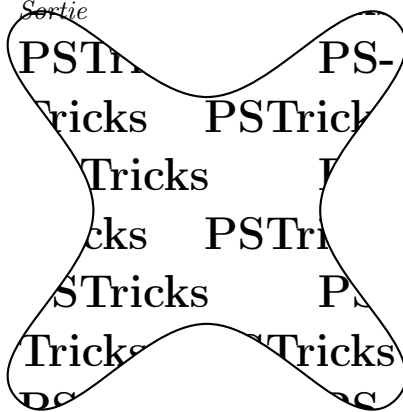
Clipping

`\begin{psclip}{frontière}...\end{psclip}` pour dessiner uniquement ce qui est inclus dans la frontière.

Code

```
\begin{pspicture}(0,-2.5)(5,2.5)
\begin{psclip}{
\psccurve(1,0)(0,-2.5)(2.5,-1.5)(5,-2.5)(4,0)(5,2.5)(2.5,1.5)
(0,2.5)
}
\parbox{5cm}{%
\LARGE\bfseries\PSTricks\PSTricks\PSTricks\PSTricks\PSTricks\PSTricks
\PSTricks\PSTricks\PSTricks\PSTricks\PSTricks\PSTricks\PSTricks\PSTricks
\PSTricks\PSTricks
}
\end{psclip}
\end{pspicture}
```

Sortie



Effets de texte

- `\pstextpath[pos](x,y){chemin}{texte}` trace le `texte` en suivant le chemin, aligné à la position `pos` (l, c ou r) par rapport à la longueur totale du chemin, et décalé de (x,y) par rapport à celui-ci;

- `\pscharpath[options]{texte}` affiche la région délimitée par `texte`.

Code

```
\begin{pspicture}(0,0)(5,4)
\pstextpath{
\psellipse[linestyle=none](2.5,2)(2.3,1.8)
}{
\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_
\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_\LaTeX\_
}
\rput(2.5,2){
\pscharchapath[fillstyle=gradient,gradbegin=red,gradend=blue]{
\fontsize{1.3cm}{1.3cm}\selectfont\bfseries\LaTeX
}
}
\end{pspicture}
```

Sortie

X $\overset{\text{A}}{\text{L}}\text{T}_{\text{E}}\text{X}$ $\overset{\text{A}}{\text{L}}\text{T}_{\text{E}}\text{X}$ $\overset{\text{A}}{\text{L}}\text{T}_{\text{E}}\text{X}$

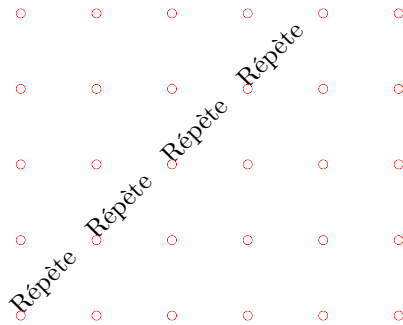
Pavages

- `\multirput[référence]{rotation}(x0,y0)(dx,dy){n}{texte}` va placer `n` fois `texte` en translatant à chaque fois de `(dx,dy)` par rapport au point `(x0,y0)` ;
- `\multips{rotation}(x0,y0)(dx,dy){n}{texte}` fera la même chose pour des objets de taille nulle.

Code

```
\begin{pspicture}(0,0)(5,4)
uu\multirput[1B]{45}(1,1){4}{R      }
uu\multirput(1,0){6}{
uuuu\multirput(0,1){5}{
uuuuuu\psdots[linecolor=red,dotstyle=o](0,0)
uuuu}
uu}
\end{pspicture}
```

Sortie



Répétitions

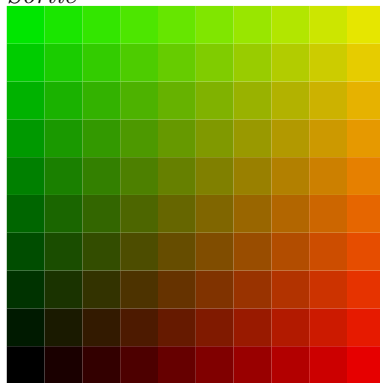
`\multido{\variable=debut+delta}{n}{...}` va répéter n fois son dernier argument pour `\variable` variant de `debut` à `debut + (n - 1) · delta` (nécessite l'extension `multido`).

Code

```
\psset{fillstyle=solid,linestyle=none}

\begin{pspicture}(0,0)(1,1)
  \multido{\nx=0.0+0.1}{10}{
    \multido{\ny=0.0+0.1}{10}{
      \newrgbcolor{c}{\nx}{\ny}{0}
      \rput{\nx,\ny}{
        \psframe[fillcolor=c](0,0)(0.1,0.1)
      }
    }
  }
\end{pspicture}
```

Sortie



Graphiques 3D avec [pst-3dplot](#)

...et encore beaucoup d'autres possibilités, consulter par exemple :

<http://tug.org/PSTricks/main.cgi?file=examples>

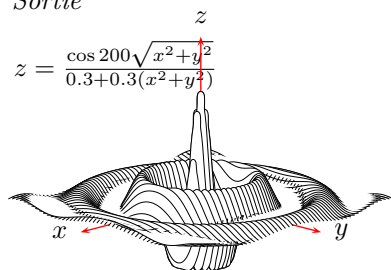
Code

```

\begin{pspicture}(-2.5,-1)(2.5,2.5)
\psset{Beta=15,unit=0.45cm}
\rput[1](-5.5,4){$z=\frac{\cos 200\sqrt{x^2+y^2}}{0.3+0.3(x^2+y^2)}$}
\pstThreeDCoor[xMin=-1,xMax=5,yMin=-1,yMax=5,zMin=-1,zMax=5]
\psplotThreeD[yPlotpoints=50,xPlotpoints=50,linewidth=0.1pt,
hiddenLine=true,drawStyle=xyLines,plotstyle=curve,fillstyle=
solid](-4,4)(-4,4){x_2\exp y_2\exp add 0.5\exp 200\mul\cos 0.3\exp
0.3\exp x_2\exp y_2\exp add\mul add\div}
\end{pspicture}

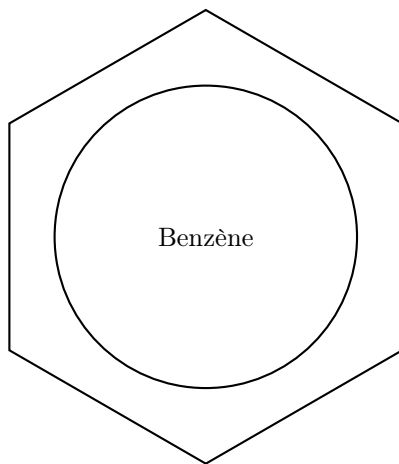
```

Sortie



Exercice

Sortie



Solution

Code

```

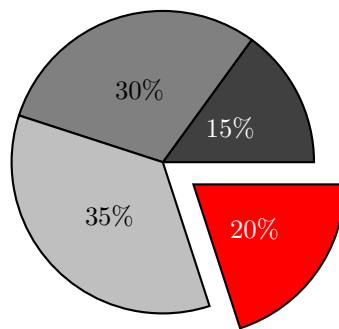
\SpecialCoor
\degrees[12]

\begin{pspicture}(-5,-3)(5,3)
\pspolygon(3;1)(3;3)(3;5)(3;7)(3;9)(3;11)
\pscicle{2}
\rput(0,0){Benzène}
\end{pspicture}

```

Exercice

Sortie



Solution

Code

```
\SpecialCoor
\degrees[100]

\begin{pspicture}(-5,-3)(5,3)
\psset{fillstyle=solid}

\pswedge[fillcolor=darkgray](0,0){2}{0}{15}
\rput(1;7.5){\white 15\%}

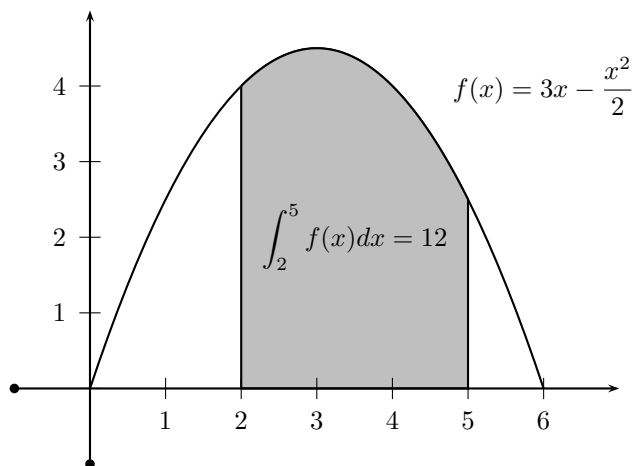
\pswedge[fillcolor=gray](0,0){2}{15}{45}
\rput(1;30){30\%}

\pswedge[fillcolor=lightgray](0,0){2}{45}{80}
\rput(1;62.5){35\%}

\rput(0.5;90){
\pswedge[fillcolor=red](0,0){2}{80}{100}
\rput(1;90){\white 20\%}
}
\end{pspicture}
```

Exercice

Sortie



Solution

Code

```
\begin{pspicture}(-2,-1.2)(8,5.2)
  \psplot[algebraic]{0}{6}{-0.5*x^2+3*x}
  \pscustom[fillstyle=solid,fillcolor=lightgray]{
    \psplot[algebraic]{2}{5}{-0.5*x^2+3*x}
    \psline(5,0)(2,0)(2,4)
  }
  \psaxes{* ->}(0,0)(-1,-1)(7,5)
  \rput(6,4){$\displaystyle f(x)=3x-\frac{x^2}{2}$}
  \rput(3.5,2){$\displaystyle \int_2^5 f(x)dx=12$}
\end{pspicture}
```

Exercice [*Difficile*]

Sortie



Solution

Code

```
\newrgbcolor{ORSAYBLUE}{0.15\0.32\0.57}
\begin{pspicture}(-10,-4)(10,4)
```

```

uu\begin{psclip}{
uuuu\pspolygon[linestyle=none](-3.5,0)(0,3.5)(3.5,0)(0,-3.5)
uu}
uuuu\multips(0,3.4)(0,-0.4){18}{
uuuuuu\psframe*[linecolor=lightgray](-3.5,-0.1)(3.5,0.1)
uuuu}
uu\end{psclip}
uu\psset{linecolor=ORSAYBLUE}
uu\pswedge*(0,0.1){1.8}{0}{158}
uu\pswedge*(0,0.1){1.8}{180}{338}
\end{pspicture}

```

1.7 Transparents avec Seminar

La classe `seminar`

Permet de faire des transparents rapidement.

La compilation se fait avec L^AT_EX (et non PdfL^AT_EX) si l'on souhaite utiliser `semcolor`, qui repose sur `pstricks`.

Chargement

```

\documentclass[a4]{seminar}

\usepackage{semcolor,slidesec,fancybox}
%Correction de bugs
\input{seminar.bug}
\input{seminar.bg2}

%pour gérer correctement les tailles et changements d'orientation de page
\def\printlandscape{\special{papersize=297mm,210mm}}

```

La classe `seminar`

Nouveaux environnements :

- `\begin{slide}[largeur,hauteur]... \end{slide}` pour ajouter un transparent supplémentaire en mode « paysage » ;
- `\begin{slide*}[largeur,hauteur]... \end{slide*}` idem mais en mode « portrait » ;
- `\begin{note}... \end{note}` pour ajouter une note.

Options de la classe :

- `notes` pour imprimer notes et transparents ;
- `slidesonly` pour n'imprimer que les transparents ;
- `notesonly` pour n'imprimer que les notes ;
- `article` pour imprimer un article dont le corps est formé du contenu des notes avec les transparents en figures flottantes.

Extensions à utiliser avec `seminar`

- L'extension `slidesec` :

- `\slidechapter`, `\slideheading` et `\slidesubheading` pour définir des titres et sous-titres ;
- `\listofslides` pour afficher une table des transparents ;
- `\slidecontents` pour afficher le contenu des transparents ;
- `\Slidecontents` pour afficher en plus l'endroit où l'on se trouve.
- L'extension `semcolor` permet d'utiliser de la couleur.
- L'extension `semhelv` permet d'utiliser une fonte sans sérif.
- L'extension `semlayer` permet des effets spéciaux (superposition).

Examples

[illegible]

Personnalisation

- `\slideframe{style}` pour la décoration du bord, avec `style = \langle none, plain, shadow, double, oval \rangle` (les 3 dernières nécessitent `fancybox`).
- `\slideframe*` superpose les styles.
- On peut colorier le fond avec les styles `scshadow`, `scdouble` ou `scoval` et en donnant des paramètres de remplissage avec `psset`.
- `\newslideframe{nom}{apparence}` pour définir un nouveau style, le contenu du transparent étant accessible dans `apparence` via `#1`.

Examples

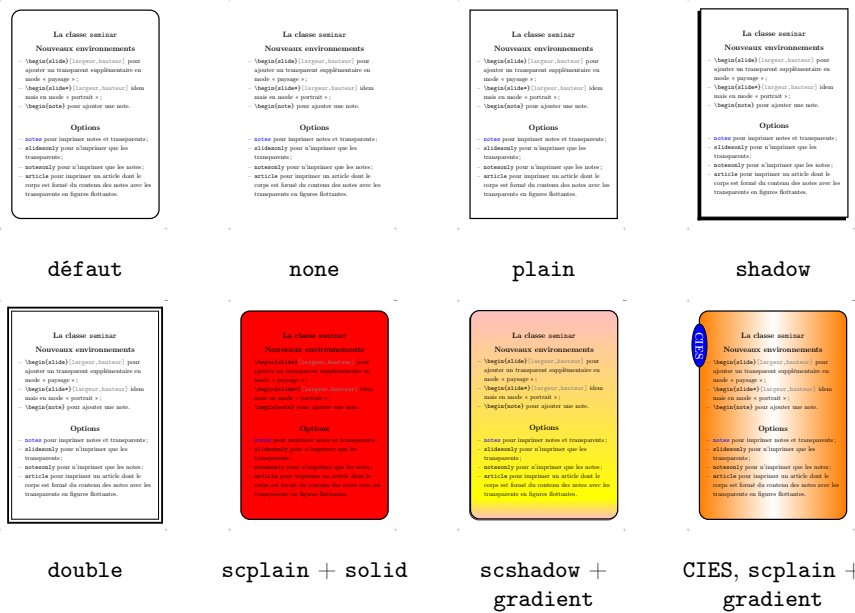
```
\slideframe[\psset{fillstyle=solid,fillcolor=red}]{scplain}

\slideframe[\psset{fillstyle=gradient,gradbegin=pink,gradend=yellow}]{scshadow}

\newslideframe{CIES}{%
\rrput(1.5,-0.4){\psovalbox[fillstyle=solid,fillcolor=blue]{\white
\LARGE\bfseries\text{CIES}}}%
\rr#1%
```

}

Exemples



1.8 Présentations avec Beamer

La classe beamer

Exemple

```
\documentclass[french]{beamer}

\begin{document}
  \begin{frame}
    \frametitle{Ma première page beamer}
    \framesubtitle{C'est très simple à écrire}
    Le contenu se centre verticalement dans la page, la fonte est sans serif et le texte est aligné à gauche. \bigskip

    \begin{itemize}
      \item Les listes
      \item à puces
      \item sont jolies
    \end{itemize} \bigskip

    \begin{enumerate}
      \item Les listes
      \item numérotées
      \item aussi
    \end{enumerate}
  \end{frame}
\end{document}
```

Ma première page beamer C'est très simple à écrire

Le contenu se centre verticalement dans la page, la fonte est sans serif et le texte est aligné à gauche.

- Les listes
- à puces
- sont jolies

1. Les listes
2. numérotées
3. aussi

La page de titre

Code

```
\title{Mon_premier_document_Beamer}  
\author{Jean_Dupont}  
\institute{CIES}  
\date{\today}  
  
\begin{frame}  
  \titlepage  
  \begin{center}\url{http://jean-dupont.com}\end{center}  
\end{frame}
```

Sortie

Mon premier document Beamer Jean Dupont

Colonnes

- `\begin{columns}[options]` permet de commencer un groupe de colonnes. Les options peuvent être :
 - `c` ou `T` pour aligner le milieu ou le haut des colonnes entre elles ;
 - `b` ou `t` pour aligner les lignes du bas ou du haut entre elles ;
 - `onlytextwidth` pour remplir toute la largeur et `totalwidth=largeur` pour remplir une largeur donnée (par défaut `totalwidth=\pagewidth`, c'est à dire plus que `\textwidth`).
- `\begin{column}{largeur}` à l'intérieur d'un environnement `columns` crée une nouvelle colonne.

Code

```
\begin{columns}[t,onlytextwidth]
  \begin{column}{0.2\textwidth}La première colonne\end{column}
  \begin{column}{0.3\textwidth}La deuxième colonne\end{column}
  \begin{column}{0.4\textwidth}La troisième colonne\end{column}
\end{columns}
```

Sortie

La première colonne La deuxième colonne La troisième colonne

Blocs

Code

```
\begin{block}{Un titre normal}
  Pour un bloc normal
\end{block}
\begin{block}{Un bloc de blocs}
  On peut imbriquer sans problème blocs et colonnes, les théorèmes
  et preuves ont aussi leurs propres blocs.
  \begin{columns}
    \begin{column}{0.45\textwidth}
      \begin{theorem}Ceci n'est pas une négation en bloc.\end{theorem}
      \begin{proof}C'est évident!\end{proof}
    \end{column}
    \begin{column}{0.45\textwidth}
      \begin{exampleblock}{Un bloc d'exemple}Pour les illustrations
        \end{exampleblock}
      \begin{alertblock}{Un bloc d'alerte!}Pour les choses
        importantes\end{alertblock}
    \end{column}
  \end{columns}
\end{block}
```

Sortie

Un titre normal

Pour un bloc normal

Un bloc de blocs

On peut imbriquer sans problème blocs et colonnes, les théorèmes et preuves ont aussi leurs propres blocs. Ceci n'est pas une négation en bloc.

Démonstration. C'est évident!

□

Un bloc d'exemple

Pour les illustrations

Un bloc d'alerte !

Pour les choses importantes

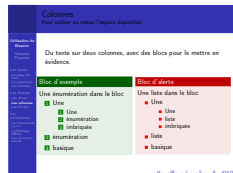
Thèmes de Beamer

On active un thème général avec `\usetheme[options]{theme}` dans le pré-ambule.

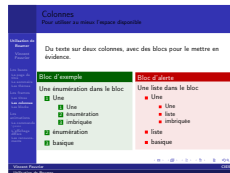
Ce sera une combinaison de quatre sous-thèmes qui peuvent aussi être activés individuellement :

- `\usecolortheme[options]{theme}` pour le thème de couleur (qui peut être global, interne ou externe) ;
- `\useinnertheme[options]{theme}` pour le thème des éléments internes (blocs, énumérations, etc...) ;
- `\useoutertheme[options]{theme}` pour le thème des éléments externes (barre de navigation, foliation, etc...) ;
- `\usefonttheme[options]{theme}` pour le thème de fontes.

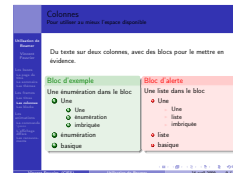
Thèmes généraux, avec `sidebar` en thème externe



Antibes



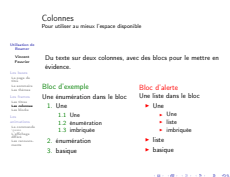
Berlin



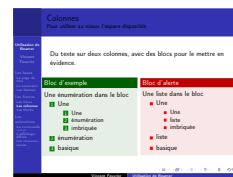
Boadilla



CambridgeUS



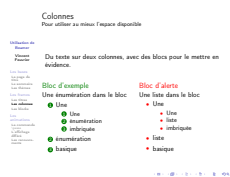
default



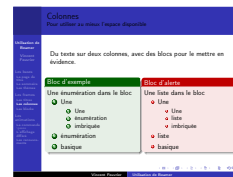
Luebeck



Madrid

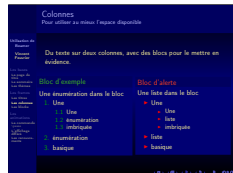


Pittsburgh

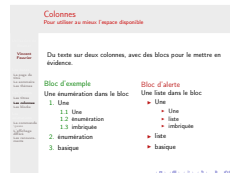


Warsaw

Thèmes de couleur complets, avec sidebar en thème externe



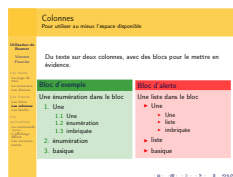
albatross



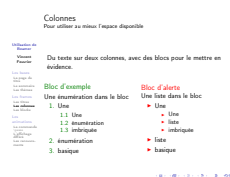
beaver



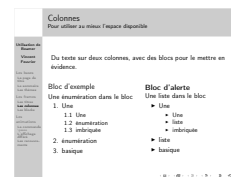
beetle



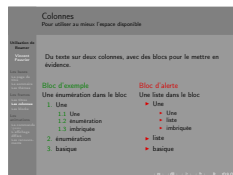
crane



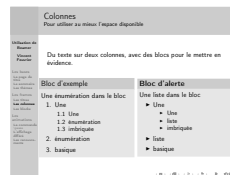
default



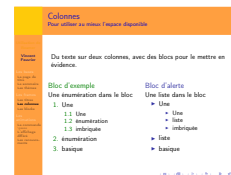
dove



fly



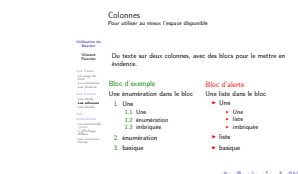
seagull



wolverine

Thèmes de couleur partiels, avec sidebar en thème externe

– Thèmes de couleur internes



lily

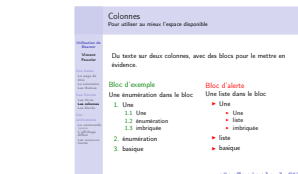


orchid

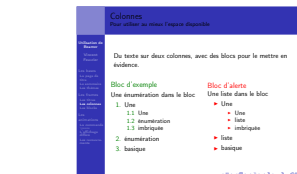


rose

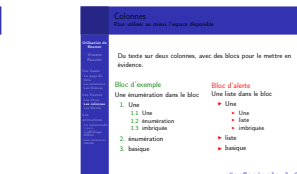
– Thèmes de couleur externes



whale

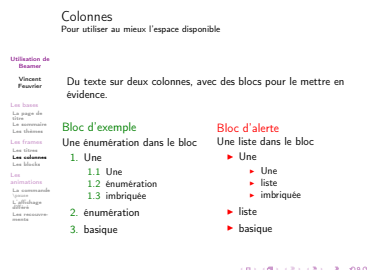


seahorse

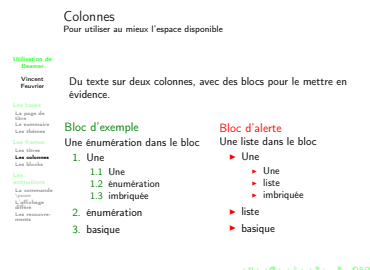


dolphin

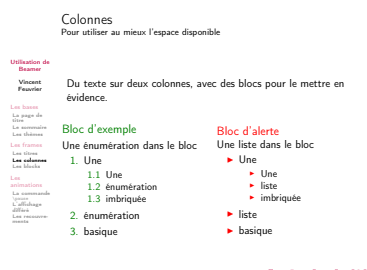
Thème de couleur « structure » avec une option pour la couleur de base



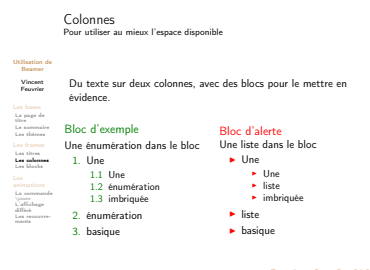
[named=violet]



[named=green]

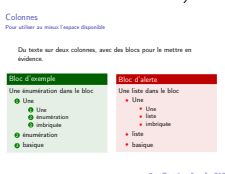


[named=purple]

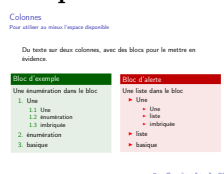


[named=brown]

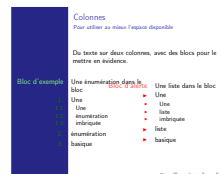
Thèmes internes, avec orchid pour les couleurs



circles



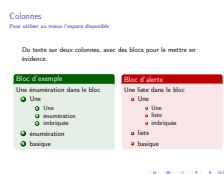
default



inmargin

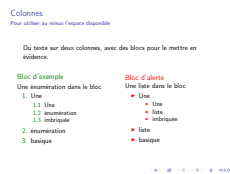


rectangles



rounded

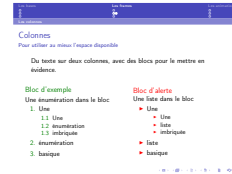
Thèmes externes, avec dolphin pour les couleurs



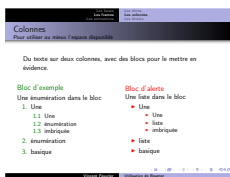
default



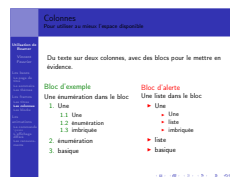
infolines



miniframes



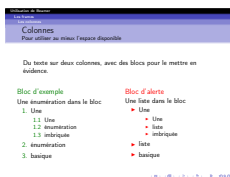
shadow



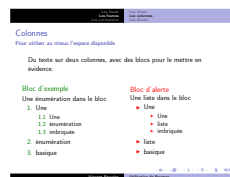
sidebar



smoothbars



smoothtree

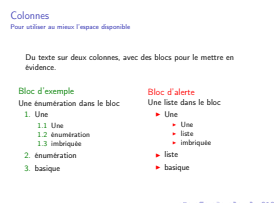


split

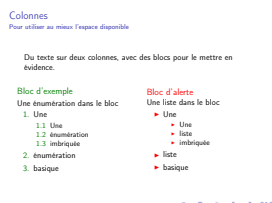


tree

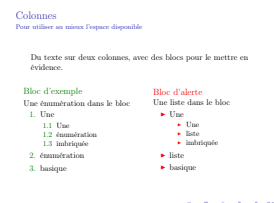
Thèmes de fontes



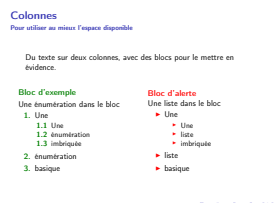
default



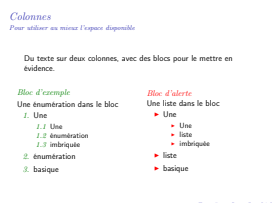
professional



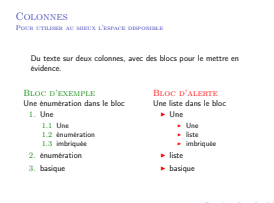
serif



structurebold



structureitalicserif



structuresmallcapsserif

Personnalisation du thème

- `\setbeamercolor{élément}{attribut=couleur}` permet de changer la couleur d'un attribut (fg, bg...) d'un élément (background canvas, normal text, alerted text...):

- on peut donner une couleur par son nom, préexistant ou défini par `\definecolor{couleur}{rgb}{R,G,B}`,
- par une combinaison de deux couleurs : `red!20!black` pour un mélange 20% rouge/80% noir;
- `\setbeamertemplate{élément}modèle` permet d'utiliser un modèle pré-défini (`[vertical shading]` `[options]`, `[square]`, `[circle]{rayon}...`);
- `\setbeamerfont{élément}{attribut=valeur}` permet de changer un attribut (`family`, `series`, `size`) de fonte.

Exemple

```
%Retour aux valeurs par défaut
\setbeamertemplate{background canvas}[default]
%Changement de la couleur de fond
\setbeamercolor{normal text}{bg=red!30!black}
```

Exemple

```
%Retour aux valeurs par défaut
\setbeamertemplate{background canvas}[default]
%Changement du modèle de fond
\setbeamertemplate{background canvas}[vertical shading][bottom=
orange!40!black,middle=pink!60!black,top=blue!50!black]
```

Affichage différé

- `\pause` affiche uniquement ce qui précède sur la page en cours, puis l'affiche de nouveau suivi du reste sur les pages suivantes;
- `\item<pages>` affiche un élément de liste uniquement sur les pages spécifiées;
- `\only<pages>{texte}` affiche le texte uniquement sur les pages spécifiées.

Code

```
Voici\only<1>{\ldots}
\pause
une liste\only<-5>{incomplète}\only<6>{complète}:
\pause
\begin{itemize}
  \itemPremier élément
  \item<4->Deuxième élément
  \item<5,6>Troisième élément
  \item<6>Quatrième élément
  \itemDernier élément (toujours visible)
\end{itemize}
\pause
Et voilà du texte.
```

Sortie

Voici... une liste incomplètecomplète :

- Premier élément
 - Deuxième élément
 - Troisième élément
 - Quatrième élément
 - Dernier élément (toujours visible)
- Et voilà du texte.

Affichage différé Conventions de désignation des pages d'un frame

- si on ne met rien (par exemple `\only{texte}`) alors implicitement ce sera toutes les pages ;
- un nombre tout seul désigne un numéro de page ;
- + désigne la page en cours la première fois, et la page qui suit à chaque nouvelle utilisation ;
- x-y désigne toutes les pages dont les numéros sont compris entre x et y ;
- -x désigne toutes les pages de 1 à x ;
- x- désigne toutes les pages à partir de x jusqu'à la fin du frame ;
- on sépare les pages ou groupes de pages par des virgules ;
- le nombre total de pages du frame sera augmenté pour être au moins aussi grand que la dernière page désignée.

Code

```
\begin{itemize}  
  \item<+>Premier élément  
  \item<+>Deuxième élément  
  \item<+>Troisième élément  
  \item<+>Quatrième élément  
\end{itemize}
```

Sortie

- Premier élément
- Deuxième élément
- Troisième élément
- Quatrième élément

Affichage différé Commandes et environnements acceptant des paramètres d'affichage différé

- `\emph`, `\textit`, `\textbf`, ...
- `\alert` pour souligner du texte, `\structure` pour le mode menu ;
- `\color`, `\includegraphics` ;
- `\label` pour étiqueter une page particulière d'un frame ;
- les colonnes et les environnements de blocs (`block`, `alertblock`, ...).

Code

```
Ce\alert<2>{texte}\color<2>{blue}est\textbf<2>{torturé}àla\textit<2>{page}\pageref{torture}.\label<2>{torture}  
\begin{columns}  
  \begin{column}{0.45\textwidth}  
    \begin{alertblock}<2>{Bloc différé}\end{alertblock}  
  \end{column}  
  \begin{column}<2>{0.45\textwidth}  
    \begin{exampleblock}{Colonne différée}\end{exampleblock}  
  \end{column}  
\end{columns}
```

Sortie

Ce *texte* est **torturé** à la page 46.

Bloc différé

Colonne différée

Animation et superposition

- `\uncover<pages>{...}` s'utilise comme `\only`, mais conserve l'espace occupé par les éléments cachés ;
- `\onslide<pages>` fait la même chose mais s'applique à tout ce qui suit (pas besoin d'accolades).

Code

```
Ce\texte\only<2>{n'}est\only<2>{pas}caché.
```

```
Ce\texte\uncover<2>{n'}est\onslide<2>pas\onslide\caché.
```

Sortie

Ce texte n'est pas caché.

Ce texte n'est pas caché.

Animation et superposition Commandes et environnements supplémentaires

- `\visible<pages>{texte}` fait la même chose que `\uncover` mais le texte caché n'est jamais visible, même en transparence.
- `\invisible<pages>{texte}` fait le contraire ;
- `\begin{overlayarea}{largeur}{hauteur}` permet de faire une boîte de taille fixée dont le contenu peut varier ;

Code

```
\begin{column}{0.5\textwidth}
\only<1>{Un\paragraphe.}\only<2>{Un\paragraphe\plus\long.}\hrule
\end{column}
\begin{column}{0.5\textwidth}
\begin{overlayarea}{\textwidth}{6ex}
\only<1>{Un\paragraphe.}\only<2>{Un\paragraphe\plus\long.}
\end{overlayarea}\hrule
\end{column}
```

Sortie

Un paragraphe.Un paragraphe plus long.

Texte qui bouge Un paragraphe.Un paragraphe plus long.

Texte qui ne bouge pas

- l'environnement `onlyenv` fonctionne comme `\only` (et permet par exemple d'utiliser du *verbatim*) ;

Code

```

\begin{onlyenv}<2>
\begin{verbatim*}
Saviez-vous que l'environnement
verbatim* fonctionne comme verbatim,
mais affiche les espaces du texte?
\end{verbatim*}
\end{onlyenv}

```

Sortie
Saviez-vous que l'environnement
verbatim* fonctionne comme verbatim,
mais affiche les espaces du texte?

- \alt<pages>{alt1}{alt2} affichera alt1 sur les pages spécifiées, sinon alt2 ;;

Code

```
Nous sommes sur la \alt<1>{première}{seconde} page.
```

Sortie
Nous sommes sur la première page.

- \begin{altenv}<pages>{avant}{après}{sinon avant}{sinon après} permet d'entourer son contenu sélectivement sur les pages spécifiées :

Code

```

\begin{altenv}<1>{\begin{center}}{\end{center}}{\begin{
flushright}}{\end{flushright}}
\verb|TEXTE| à déplacer
\end{altenv}

```

Sortie

TEXTE à déplacer

Animation et superposition Affichage dynamique de tableaux

- Il faut d'abord utiliser l'extension colortbl.
- affichage différé des lignes ;;

Code

```

\begin{tabular}{c|c|c}
A&B&C\\ \hline
\pause
D&E&F\\ \hline
\pause
G&H&I
\end{tabular}

```

Sortie

A	B	C
D	E	F
G	H	I

- affichage différé des colonnes ;;

Code

```
\begin{tabular}{>{\onslide}c|>{\onslide<2->}c|>{\onslide<3>}c
<{\onslide}}
\A&B&C\\ \hline
\D&E&F\\ \hline
\G&H&I
\end{tabular}
```

Sortie

A	B	C
D	E	F
G	H	I

- affichage différé des cellules :

Code

```
\begin{tabular}{|c|c|}
\hline
\A&\uncover<2->{B}\\ \hline
\uncover<3->{C}&\uncover<4>{D}\\ \hline
\end{tabular}
```

Sortie

A	B
C	D

Options de l'environnement `frame`

- ```
\begin{frame}<pages>[<mode>][options]{title}{subtitle}
```
- `<pages>` pour n'afficher que certaines pages du frame ;
  - `[<mode>]` pour un mode par défaut des sous-éléments (par exemple `[<+>]`) ;
  - les `[options]` peuvent être :
    - `allowframebreaks` pour que le frame soit découpé si le contenu dépasse (*à réserver pour la bibliographie !*),
    - `fragile` pour inclure du verbatim ou des listings,
    - `containsverbatim` lorsque `fragile` ne suffit pas,
    - `label=étiquette` pour définir une étiquette (permet de réafficher le frame avec `\againframe<pages>{étiquette}`),
    - `plain` pour enlever les décorations,
    - `shrink` pour diminuer la taille de la fonte s'il y a trop de texte,
    - `squeeze` pour resserrer les espacements verticaux et rendre le frame plus compact.

## Sectionnement

Il est possible (et même recommandé!) de structurer le document à l'extérieur des frames avec `\part`, `\section`, `\subsection`.

- `\partpage` pour faire une page de titre de partie (sur le modèle de `\titlepage`) ;
- `\tableofcontents[options]` pour afficher la table des matières.

### Code

```
\begin{block}{Table des matières}
\tableofcontents[pause sections,hide all subsections]
\end{block}
```



*Sortie*

## Table des matières

## Table des matières

|          |                                                                  |           |
|----------|------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Types de documents supplémentaires en <math>\LaTeX</math></b> | <b>1</b>  |
| 1.1      | Manipulation de fichiers PostScript et PDF . . . . .             | 1         |
| 1.2      | Listings et programmes informatiques . . . . .                   | 3         |
| 1.3      | Partitions de musique . . . . .                                  | 8         |
| 1.4      | Lettres et courrier . . . . .                                    | 8         |
| 1.5      | Affiches et posters . . . . .                                    | 13        |
| 1.6      | Les extensions <code>pstricks</code> . . . . .                   | 14        |
| 1.7      | Transparents avec <code>Seminar</code> . . . . .                 | 34        |
| 1.8      | Présentations avec <code>Beamer</code> . . . . .                 | 36        |
| <b>2</b> | <b>Utiliser <math>\LaTeX</math> pour le web</b>                  | <b>53</b> |
| 2.1      | Prérequis . . . . .                                              | 53        |
| 2.2      | Conversions . . . . .                                            | 53        |
| 2.3      | HeVeA et HachA . . . . .                                         | 53        |
| <b>3</b> | <b>Trucs et astuces avancés</b>                                  | <b>54</b> |
| 3.1      | Les fontes . . . . .                                             | 54        |
| 3.2      | Calculer en $\LaTeX$ . . . . .                                   | 63        |
| 3.3      | En finir avec les <b>bad boxes</b> . . . . .                     | 64        |
| 3.4      | Enlever les espaces indésirables . . . . .                       | 66        |
| 3.5      | Écrire dans des fichiers . . . . .                               | 68        |
| 3.6      | Divers . . . . .                                                 | 70        |
| <b>4</b> | <b>Programmation <math>\LaTeX</math> avancée</b>                 | <b>73</b> |
| 4.1      | Tests et branchements conditionnels . . . . .                    | 73        |
| 4.2      | Écrire une nouvelle extension . . . . .                          | 81        |
| 4.3      | Écrire une nouvelle classe . . . . .                             | 82        |
| 4.4      | Définition avancée des commandes . . . . .                       | 84        |
| 4.5      | Boucles . . . . .                                                | 88        |
| 4.6      | Codes de catégorie . . . . .                                     | 90        |

### Sectionnement Options de la table des matières

- `pausesections` ou `pausesubsections` pour faire une pause à chaque élément ;
- `sectionstyle=courante/autres` pour changer le style d’affichage de la

- section courante et des autres : `show`, `shaded` ou `hide`;
- `subsectionstyle=courante/soeurs/cousines` pour changer le style d’affichage de la sous-section courante, de ses `soeurs` (filles de la même section) et de ses `cousines` (filles des autres sections);
- `currentsection` est équivalent à `sectionstyle=show/shaded`, `subsectionstyle=show/show/shaded`;
- `currentsubsection` est équivalent à `subsectionstyle=show/shaded`;
- `hideallsubsections` est équivalent à `subsectionstyle=hide`;
- `hideothersubsections` est équivalent à `subsectionstyle=show/show/hide`.

## Sectionnement

### *Sommaires automatiques (à mettre dans le préambule)*

```
\AtBeginPart{
 \begin{frame}
 \partpage
 \end{frame}
}

\AtBeginSection{
 \begin{frame}
 \frametitle{Table des matières}
 \tableofcontents[sectionstyle=show/shaded,subsectionstyle=show/
 shaded/hide]
 \end{frame}
}

\AtBeginSubsection{
 \begin{frame}
 \frametitle{Table des matières}
 \tableofcontents[sectionstyle=show/shaded,subsectionstyle=show/
 shaded/hide]
 \end{frame}
}
```

## Modes de Beamer

Le mode est spécifié en option de la classe.

- `beamer` pour la version vidéoprojecteur;
- `handout` pour la version papier à distribuer pendant la présentation;
- `trans` pour faire des transparents pour rétroprojecteur;
- `article` pour une version sous forme d’article papier.

### Modes de Beamer Le mode `handout`

Il est recommandé d’utiliser un thème clair.

On peut utiliser l’extension `pgfpages` pour disposer plusieurs pages par feuille.

### *Exemple de configuration*

```
\documentclass[french,handout]{beamer}

\usepackage{pgfpages}
%au_choix:
```

```

\pgfpagesuselayout{2_\on_1}[a4paper]
\pgfpagesuselayout{4_\on_1}[a4paper,landscape]

\begin{document}
...

```

### Modes de Beamer Le mode `trans`

Il est recommandé d'utiliser un thème « blanc ».

#### *Exemple de configuration*

```

\documentclass[french,trans]{beamer}

\begin{document}
...

```

### Modes de Beamer Le mode `article`

Il est préférable d'utiliser la classe `article` et l'extension `beamerarticle`.

Les thèmes, les blocs et les effets de superposition seront désactivés.

#### *Exemple de configuration*

```

\documentclass[french]{article}

\RequirePackage{beamerarticle}

\begin{document}
...

```

### Modes de Beamer Adapter le contenu au mode

- `\mode<liste>{texte}` lira le `texte` uniquement si le mode courant est dans la liste;
- ça marche aussi avec `\item`, `\only...`
- `\mode <liste>` lira ce qui suit uniquement si le mode courant est dans la liste. *Ne pas oublier de le désactiver avec* `\mode <all>`

Le mode "all" représente n'importe quel mode, et "presentation" n'importe quel mode à part `article`.

#### *Code*

```

Les modes suivants sont activés :
\begin{itemize}
 \item<presentation>{presentation}
 \item<beamer>{beamer}
 \item<handout>{handout}
 \item<trans>{trans}
 \item<article>{article}
\end{itemize}

```

#### *Sortie*

Les modes suivants sont activés :

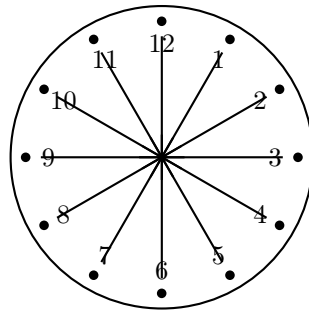
- presentation
- beamer
- handout
- trans
- article

### Exercice

1. Écrire un document Beamer avec quelques frames.
2. Y ajouter des colonnes et des blocs.
3. Faire quelques effets d’affichage différé ou de superposition.
4. Sectionner le document et afficher des tables des matières.
5. Faire une version pour rétroprojecteur, ou à distribuer au public.
6. Convertir le document en un article.

### Exercice [*Difficile*]

*Sortie*



### Solution

#### *Code*

```
\transduration<1-11>{1}
\SpecialCoor
\degrees[12]
\begin{pspicture}(-5,-3)(5,3)
uu\pscircle{2}
uu\multido{\nn=1+1}{12}{
uuuu\rput{3}{
uuuuuu\rput{-3}(1.5;-\nn){\nn}
uuuuuu\only<\nn>{
uuuuuuuu\psline(-0.3;-\nn)(1.6;-\nn)
uuuuuu}
uuuuu}
```

```

\psdots(1.8;\nn)
\end{pspicture}

```

## 2 Utiliser L<sup>A</sup>T<sub>E</sub>X pour le web

### 2.1 Prérequis

#### Comment disposer d'un site web ?

La plupart des facultés proposent un service d'hébergement de pages personnelles.

Une fois activé, le compte dispose d'un répertoire (par exemple `wwwdoc`) dont l'arborescence est visible depuis l'extérieur à une adresse racine (par exemple `www.mondépartement.mafac.fr/username/`).

La page web par défaut d'un répertoire doit s'appeler (en général) `index.html`.

On peut envoyer des fichiers à distance par FTP ou SCP.

### 2.2 Conversions

#### Convertir du HTML vers L<sup>A</sup>T<sub>E</sub>X

On peut utiliser `html2latex` :

`http://htmltolatex.sourceforge.net/`

Son avantage est d'être écrit en Java (donc multiplateforme).

#### Convertir du L<sup>A</sup>T<sub>E</sub>X en HTML

Il existe plusieurs solutions :

- `latex2html` (formules rendues en images, nécessite Perl) ;
- `TeX4ht` (formules rendues en images, nécessite ImageMagick) ;
- `TtH` (formules rendues en HTML) ;
- `HeVeA` (formules rendues au choix en images ou en HTML) ;

### 2.3 HeVeA et HachA

#### Fonctionnement

Dans un premier temps, on compile avec `HeVeA` :

#### *Usage*

```
hevea monfichier.tex
```

Puis, `HachA` permet de découper le fichier HTML obtenu en petits morceaux avec des liens de navigation en suivant la hiérarchie du document.

#### *Usage*

```
hacha monfichier.html -o index.html
```

## Formules en images

On peut utiliser l'environnement `toimage` :

### Exemple

```
\begin{toimage}
□□$$\sum_{k=1}^+\infty\frac{1}{k^2}=\frac{\pi^2}{6}$$$
\end{toimage}
```

Un fichier externe contenant les formules est généré, puis compilé et converti en une série d'images GIF à l'aide d'un script fourni avec HeVeA : `imagen`.

## Commandes HeVeA propres au HTML

- L'environnement `rawhtml` permet de taper du code HTML directement (un peu comme `verbatim`);
- `\newstyle{élément}{styles}` permet d'exporter le formatage d'un élément dans le fichier CSS;
- `\begin{divstyle}{classe CSS}` permet de créer un élément HTML `<DIV>` de la classe spécifiée;
- `\@open{TAG}{attributs}...\@close{TAG}` pour ouvrir et fermer un élément HTML;
- `\ahref{lien}{texte}` pour faire un lien.

## La classe `site`

Elle permet de faire un site avec un menu et un style de couleur configurable. Chaque commande de sectionnement ajoute une nouvelle page au site.

### Exemple

```
\documentclass{site}

\usecolortheme{blue}

\title[Accueil de mon site]{Accueil}

\begin{document}
...
\end{document}
```

## Exercice

# 3 Trucs et astuces avancés

## 3.1 Les fontes

### Les attributs de fontes

Il y en a cinq en tout :

1. l'encodage (encoding) qui peut être :
  - `OT1` pour l'ancien format de texte;
  - `T1` pour le nouveau format de texte;

- OML pour les maths en italique ;
  - OMS pour les symboles mathématiques ;
  - OMX pour les gros symboles ;
  - U pour "inconnu".
2. la famille (family) qui peut être :.
    - cmr pour "Computer Modern Roman" ;
    - cmss pour "Computer Modern Sans" ;
    - cmtt pour "Computer Modern Typewriter" ;
    - cmm pour "Computer Modern Math Italic" ;
    - cmsy pour "Computer Modern Math Symbols" ;
    - cmsy pour "Computer Modern Math Extensions" ;
    - ptm pour "Adobe Times" ;
    - phv pour "Adobe Helvetica" ;
    - pcr pour "Adobe Courier".
    - ...
  3. les séries de graisse (series) qui peuvent être :.
    - m pour "Medium" ;
    - b pour "Bold" ;
    - bx pour "Bold extended" ;
    - sb pour "Semi-bold" ;
    - c pour "Condensed".
  4. la forme (shape) qui peut être :.
    - n pour "Normal" ;
    - it pour "Italic" ;
    - sl pour "Slanted" ;
    - sc pour "Small caps".
  5. la taille (size), généralement exprimée en points.

### Les attributs de fontes Techniques usuelles pour changer les attributs

Les valeurs indiquées sont celles par défaut de la classe `article`.

1. l'encodage (par défaut OT1) se change avec l'extension `fontenc` (préférez T1).
2. la famille se change avec `\text{...}{...}` ou `\...family` :.
  - `\textrm` ou `\rmfamily` pour `cmr` ;
  - `\textsf` ou `\sffamily` pour `cmss` ;
  - `\texttt` ou `\ttfamily` pour `cmtt`.
3. les séries se changent avec `\text{...}{...}` ou `\...series` :.
  - `\textmd` ou `\mdseries` pour `m` ;
  - `\textbf` ou `\bfseries` pour `bx`.
4. les formes se changent avec `\text{...}{...}` ou `\...shape` :.
  - `\textup` ou `\upshape` pour `n` ;
  - `\textit` ou `\itshape` pour `it` ;

- `\textsl` ou `\slshape` pour `sl`;
  - `\textsc` ou `\scshape` pour `sc`.
5. les tailles se changent avec :
- `\tiny` pour 5pt;
  - `\scriptsize` pour 7pt;
  - `\footnotesize` pour 8pt;
  - `\small` pour 9pt;
  - `\normalsize` pour 10pt;
  - `\large` pour 12pt;
  - `\Large` pour 14.4pt;
  - `\LARGE` pour 17.28pt;
  - `\huge` pour 20.74pt;
  - `\Huge` pour 24.88pt.

### Les attributs de fontes Modification locale des attributs

- `\fontencoding{encoding}` pour changer l'encodage;
- `\fontfamily{family}` pour changer la famille;
- `\fontseries{series}` pour changer les séries;
- `\fontshape{shape}` pour changer la forme;
- `\fontsize{size}{baselineskip}` pour changer la taille ainsi que la hauteur de ligne;
- `\selectfont` active les changements précédemment sélectionnés.

*Ne pas écrire de texte entre un changement d'attribut et `\selectfont` !*

- `\selectfont` ne s'applique qu'au groupe en cours;
- `\usefont{encoding}{family}{series}{shape}` appelle toutes ces commandes en une fois (sauf `\fontsize`).

### Nouvelles commandes de sélection de fonte

À utiliser dans le préambule :

- `\DeclareFixedFont\cmd{enc}{family}{series}{shape}{size}` déclare `\cmd` qui active la police sélectionnée, sur le modèle de `\itshape`, `\bfseries`...
- `\DeclareTextFontCommand\cmd{...}` déclare `\cmd` sur le modèle de `\textit`, `\textbf`...

Par exemple, `\textrm` a été implémentée en utilisant `\DeclareTextFontCommand{\textrm}{\rmfamily}`.

### Code

*% Dans le préambule :*

```
\DeclareFixedFont\funnynormal{T1}{cmfr}{m}{n}{10pt}
\DeclareTextFontCommand\textfunny{\funnynormal}
```

```
Du texte en \textfunny{Computer Modern funny roman}.
```

### Sortie

Du texte en Computer Modern funny roman.



**Computer Modern roman**, bold, normal (T1/cm<sub>r</sub>/b/n):

**Computer Modern roman**, bold extended, normal (T1/cmr/bx/n):

**Helvetica**, bold, normal (T1/phv/b/n):

**Helvetica**, bold condensed, normal (T1/phv/bc/n):

**Times**, bold, small capitals (T1/ptm/b/sc):

**New Century**, medium, normal (T1/pnc/m/n):

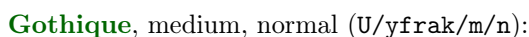
ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789

**Computer Modern funny roman**, medium, normal (T1/cmfr/m/n):

**Computer Modern dunhil**, medium, normal (T1/cmdh/m/n):

**Zapf Chancery**, medium, normal (T1/pzc/m/n):

**Zapf Dingbats**, medium, normal (U/pzd/m/n):



**Ralph Smith's Formal Script**, medium, normal (U/rsfs/m/n):

*A B C D E F G H I J K L*

- La fonte à utiliser par défaut (pour `\normalfont`) peut être modifiée en redéfinissant les commandes suivantes :
  - `\encodingdefault` (par défaut `OT1`);

- `\familydefault` (par défaut `\rmdefault`);
- `\seriesdefault` (par défaut `\mddefault`);
- `\shapedefault` (par défaut `\updefault`).
- Les familles à utiliser (pour `\rmfamily`, `\sffamily` ou `\ttfamily`) peuvent être modifiées en redéfinissant les commandes suivantes :.
  - `\rmdefault` (par défaut `cmr`);
  - `\sfdefault` (par défaut `cms`);
  - `\ttdefault` (par défaut `cmtt`).
- Les séries à utiliser (pour `\bfseries` ou `\mdseries`) peuvent être modifiées en redéfinissant les commandes suivantes :.
  - `\bfdefault` (par défaut `bx`);
  - `\mddefault` (par défaut `m`).
- Les formes à utiliser (pour `\itshape`, `\slshape`, `\scshape` ou `\upshape`) peuvent être modifiées en redéfinissant les commandes suivantes :.
  - `\itdefault` (par défaut `it`);
  - `\sldefault` (par défaut `sl`);
  - `\scdefault` (par défaut `sc`);
  - `\updefault` (par défaut `n`).

### Code

```
\usefont{T1}{phv}{m}{n}
Voici la \rmfamily _fonte d'origine.

\renewcommand\rmdefault{phv}
%On fait un changement de fonte bidon
\rmfamily
Voici la \rmfamily _nouvelle fonte qui reste active.
```

### Sortie

Voici la fonte d'origine.

Voici la nouvelle fonte qui reste active.

Changer les fontes de tout le document avec une extension

`\rmfamily: Computer Modern roman`, medium, normal (T1/cmr/m/n)  
 ABCDEFGHIJKL abcdefghijkl 0123456789

`\sffamily: Computer Modern sans sérif`, medium, normal (T1/cmss/m/n)  
 ABCDEFGHIJKL abcdefghijkl 0123456789

`\ttfamily: Computer Modern typewriter`, medium, normal (T1/cmtt/m/n)  
 ABCDEFGHIJKL abcdefghijkl 0123456789

`\bfseries: Computer Modern roman`, bold, normal (T1/cmr/b/n)  
**ABCDEFGHIJKL abcdefghijkl 0123456789**

`\itshape: Computer Modern roman`, medium, italic (T1/cmr/m/it)  
*ABCDEFGHIJKL abcdefghijkl 0123456789*

`\slanted: Computer Modern roman`, medium, slanted (T1/cmr/m/sl)  
*ABCDEFGHIJKL abcdefghijkl 0123456789*

`\scshape: Computer Modern roman`, medium, small capitals  
 (T1/cmr/m/sc)  
 ABCDEFGHIJKL ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: Concrete`, medium, normal (T1/ccr/m/n)  
 ABCDEFGHIJKL abcdefghijkl 0123456789

`\sffamily: —`  
*Manquante*

`\ttfamily: —`  
*Manquante*

`\bfseries: —`  
*Manquante*

`\itshape: Concrete`, medium, italic (T1/ccr/m/it)  
*ABCDEFGHIJKL abcdefghijkl 0123456789*

`\slanted: Concrete`, medium, slanted (T1/ccr/m/sl)  
*ABCDEFGHIJKL abcdefghijkl 0123456789*

`\scshape: Concrete`, medium, small capitals (T1/ccr/m/sc)  
 ABCDEFGHIJKL ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: Bookman`, medium, normal (T1/pbk/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\sffamily: Avant Garde`, medium, normal (T1/pag/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\ttfamily: Courier`, medium, normal (T1/pcr/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\bfseries: Bookman`, bold, normal (T1/pbk/b/n)

**ABCDEFGHJKLM abcdefghijkl 0123456789**

`\itshape: Bookman`, medium, italic (T1/pbk/m/it)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\slanted: Bookman`, medium, slanted (T1/pbk/m/sl)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\scshape: Bookman`, medium, small capitals (T1/pbk/m/sc)

ABCDEFGHJKLM ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: Charter`, medium, normal (T1/bch/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\sffamily: Charter`, medium, normal (T1/bch/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\ttfamily: Charter`, medium, normal (T1/bch/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\bfseries: Charter`, bold, normal (T1/bch/b/n)

**ABCDEFGHJKLM abcdefghijkl 0123456789**

`\itshape: Charter`, medium, italic (T1/bch/m/it)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\slanted: Charter`, medium, slanted (T1/bch/m/sl)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\scshape: Charter`, medium, small capitals (T1/bch/m/sc)

ABCDEFGHJKLM ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: Latin Modern roman`, medium, normal (T1/lmr/m/n)

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789

`\sffamily: Latin Modern sans sérif`, medium, normal (T1/lmss/m/n)

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789

`\ttfamily: Latin Modern typewriter`, medium, normal (T1/lmtt/m/n)

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789

`\bfseries: Latin Modern roman`, bold, normal (T1/lmr/b/n)

**ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789**

`\itshape: Latin Modern roman`, medium, italic (T1/lmr/m/it)

*ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789*

`\slanted: Latin Modern roman`, medium, slanted (T1/lmr/m/sl)

*ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789*

`\scshape: Latin Modern roman`, medium, small capitals (T1/lmr/m/sc)

ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: New Century`, medium, normal (T1/pnc/m/n)

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789

`\sffamily: Computer Modern sans sérif`, medium, normal (T1/cmss/m/n)

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789

`\ttfamily: Computer Modern typewriter`, medium, normal (T1/cmtt/m/n)

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789

`\bfseries: New Century`, bold, normal (T1/pnc/b/n)

**ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789**

`\itshape: New Century`, medium, italic (T1/pnc/m/it)

*ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789*

`\slanted: New Century`, medium, slanted (T1/pnc/m/sl)

*ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijkl 0123456789*

`\scshape: New Century`, medium, small capitals (T1/pnc/m/sc)

ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: Palatino`, medium, normal (T1/pp1/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\sffamily: Helvetica`, medium, normal (T1/phv/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\ttfamily: Courier`, medium, normal (T1/pcr/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\bfseries: Palatino`, bold, normal (T1/pp1/b/n)

**ABCDEFGHJKLM abcdefghijkl 0123456789**

`\itshape: Palatino`, medium, italic (T1/pp1/m/it)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\slanted: Palatino`, medium, slanted (T1/pp1/m/sl)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\scshape: Palatino`, medium, small capitals (T1/pp1/m/sc)

ABCDEFGHJKLM ABCDEFGHIJKL 0123456789

Changer les fontes de tout le document avec une extension

`\rmfamily: Times`, medium, normal (T1/ptm/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\sffamily: Helvetica`, medium, normal (T1/phv/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\ttfamily: Courier`, medium, normal (T1/pcr/m/n)

ABCDEFGHJKLM abcdefghijkl 0123456789

`\bfseries: Times`, bold, normal (T1/ptm/b/n)

**ABCDEFGHJKLM abcdefghijkl 0123456789**

`\itshape: Times`, medium, italic (T1/ptm/m/it)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\slanted: Times`, medium, slanted (T1/ptm/m/sl)

*ABCDEFGHJKLM abcdefghijkl 0123456789*

`\scshape: Times`, medium, small capitals (T1/ptm/m/sc)

ABCDEFGHJKLM ABCDEFGHIJKL 0123456789

## Exercice

**Solution** Donner trois commandes équivalentes à l'utilisation de l'extension `times`

### Code

```
\renewcommand\rmdefault{ptm}
\renewcommand\sfddefault{phv}
\renewcommand\ttdefault{pcr}
```

## Exercice

**Solution** Définir une commande pour écrire en gothique

### Code

```
\DeclareFixedFont\gotnormal{U}{yfrak}{m}{n}{10pt}
\DeclareTextFontCommand\textgot{\gotnormal}
```

## 3.2 Calculer en L<sup>A</sup>T<sub>E</sub>X

### L'extension `calc`

Redéfinit les commandes `\setcounter`, `\addtocounter`, `\setlength` et `\addtolength`, qui acceptent dès lors des expressions algébriques.

### Code

```
\setcounter{COMPTEUR}{3*6/5+8}
\theCOMPTEUR

\setlength{LONGUEUR}{1cm+1in}
\the{LONGUEUR}

\setlength{LONGUEUR}{\totalheightof{I}}
\the{LONGUEUR}

\setlength{LONGUEUR}{50pt*(3*\real{1.5}+7*\ratio{13cm}{10in}
)}
\the{LONGUEUR}
```

### Sortie

```
11
100.72273pt
6.8872pt
350.0pt
```

### L'extension `fp`

Permet de faire des calculs en virgule flottante.

Limitations :

- incompatible avec `\multido`;
- incompatible avec `[francais]{babel}`.

Solution qui semble fonctionner : charger **fp** en premier.

**Code**

```
\newcommand{\x}{2}
\FPeval{\resultat}{x^2+x+1}
\resultat

\FPeval{\x}{2*pi/3}
\FPeval{\resultat}{cos(x)}
\resultat
```

**Sortie**

```
6.99999999999999968
-0.5000000000000000000
```

### 3.3 En finir avec les bad boxes

**En finir avec les *bad boxes*** Algorithme de mise en page horizontale utilisé par **L<sup>A</sup>T<sub>E</sub>X**

L'algorithme pour découper chaque paragraphe en lignes se fait en trois passes :

1. la première passe tente un ajustement simplifié (espaces) ;
2. si la "badness" est trop importante la deuxième passe tente un ajustement plus complexe (césures) ;
3. si la "badness" est encore trop importante il y a une troisième passe agissant sur les césures ;
4. si la "badness" est toujours trop importante c'est soit une **underfull \hbox** (ligne pas assez remplie) soit une **overfull \hbox** (ligne qui dépasse).

La "badness" est une quantité qui permet d'évaluer la laideur de la pire ligne d'un paragraphe :

- $+\infty$  (10000) si une ligne dépasse de la page ;
- sinon c'est une expression compliquée faisant intervenir le facteur de dilatation de la colle utilisée.

**En finir avec les *bad boxes*** Les règles du jeu

- Les « overfull boxes » sont potentiellement graves ;
- les « underfull boxes » sont laides.

**Paramètres de design**

- Taille du papier ;
- Langage ;
- Jeu de fontes ;
- Taille de fontes ;
- Indentation.

**Paramètres T<sub>E</sub>Xniques**

- **\pretolerance** : borne pour la première passe ;



- `\tolerance` : borne pour la deuxième passe ;
- `\hbadness` : borne des `underfull \hbox` ;
- `\hfuzz` : longueur admissible de dépassement de la ligne ;
- `\emergencystretch` : longueur utilisée dans la troisième passe.

### En finir avec les *bad boxes* Algorithme à utiliser pour supprimer les bad boxes horizontales

1. Initialisation .:

#### *Dans le préambule*

```
\emergencystretch = 0pt
\pretolerance = 150
\tolerance = 250
\hbadness = 150
\hfuzz = 0pt
```

▷ Si pas de overfull hboxes dans le log à ce stade on peut passer à l'étape 3.

2. Re-compiler avec `\tolerance = 9999` et s'il n'y a plus de overfull hboxes passer à l'étape 3, sinon :et supprimer les overfull hboxes.
  - si la plus grande overfull `\hbox` dépasse seulement d'une petite valeur (par exemple `2pt`) alors `\hfuzz = 2pt` devrait régler le problème ;
  - sinon, il peut s'agir d'un problème de césure, qu'on peut régler avec `\-` ou `\hyphenation{hy-phe-na-tion}` ;
  - si à ce stade le problème des overfull hboxes n'est pas réglé, il va falloir changer le texte ou adapter certains paramètres de mise en page (largeur,indentation...)
3. Déterminer la valeur optimale de `\tolerance` :.
  - soit *badmax* la "badness" de la pire `underfull \hbox`, mettre `\tolerance = badmax` et `\hbadness = badmax - 1` ;
  - diminuer la valeur de `\tolerance` tant qu'il n'y a pas de overfull `\hbox`.
4. Si à ce stade certaines `underfull hboxes` sont trop laides (espaces entre les mots trop important) il reste un dernier espoir :Ajuster éventuellement les paramètres pour obtenir le meilleur effet visuel.
  - diminuer la valeur de `\tolerance` et mettre `\hbadness` à cette valeur moins 1 ;
  - utiliser `\emergencystretch = 1em` ;
  - si des overfull bad boxes apparaissent à ce stade il faut soit changer le document, soit augmenter `\tolerance`, soit augmenter `\emergencystretch` (mais pas trop!).
5. Poser `\hbadness=10000` pour supprimer tous les messages "`underfull \hbox`".

### En finir avec les *bad boxes* Supprimer les bad boxes verticales

Elles sont généralement plus rares, mais au cas où :

- utiliser des longueurs élastiques avec `\vspace` (par exemple `\vspace{1cm plus 2mm minus 5mm}`) ;

- `\enlargethispage{longueur}` pour augmenter (ou diminuer) la hauteur de la page courante ;
- les extensions `longtable` ou `supertabular` permettent de faire des tableaux qui peuvent être coupés entre les pages.

### Empêcher les sauts dans certaines zones

- L’extension `needspace` introduit la commande `\needspace{hauteur}` qui passe à la page suivante s’il reste moins que la `hauteur` disponible avant la fin de la page en cours ;
- `\interlinepenalty = 10000` au début d’une liste décourage L<sup>A</sup>T<sub>E</sub>X d’insérer un saut de page dans les items ;
- `\@itempenalty = 10000` pour éviter les sauts de page entre les items ;
- `\widowpenalty=1000` et `\clubpenalty=1000` pour éviter les sauts de page après la première ligne ou avant la dernière ligne d’un paragraphe.

### Suggérer ou empêcher des sauts précis

On peut formuler la demande (`n`) avec plus au moins d’insistance : 1 pour une requête timide et 4 pour un ordre.

Par défaut c’est 4.

- `\linebreak[n]` pour demander de couper la ligne ;
- `\nolinebreak[n]` pour demander de ne pas couper la ligne ;
- `\pagebreak` et `\nepagebreak` fonctionnent de manière identique pour les sauts de page.

#### Code

```
\large
Du\texte\à\couper\lorsque\ n=\linebreak[1]1

Du\texte\à\couper\lorsque\ n=\linebreak[2]2

Du\texte\à\couper\lorsque\ n=\linebreak[3]3

Du\texte\à\couper\lorsque\ n=\linebreak[4]4
```

#### Sortie

```
Du texte à couper lorsque n = 1
Du texte à couper lorsque n = 2
Du texte à couper lorsque n = 3
Du texte à couper lorsque n =
4
```

## 3.4 Enlever les espaces indésirables

### Commentaires de fin de ligne

Dans les définitions de commandes, ils permettent à la fois d’indenter proprement le code et d’éviter les espaces ajoutés par les sauts de ligne.

#### Code

```

\newcommand\PASBIEN[1]{
 \textbf{#1}
}
\newcommand\BIEN[1]{%
 \textbf{#1}%
}

Sans commentaire (\PASBIEN{argument}) avec commentaires (\BIEN{
 argument}).

```

*Sortie*

Sans commentaire ( **argument** ) avec commentaires (**argument**).

### Ignorer les espaces qui suivent

L'utilisateur des commandes n'a pas forcément envie de taper des commentaires à la fin de chaque ligne.

`\ignorespaces` permet d'ignorer les espaces qui suivent un appel de commande (ou le début d'un environnement).

#### Code

```

\newenvironment{EVIDENCE1}{\bfseries{}}
\newenvironment{EVIDENCE2}{\bfseries\ignorespaces{}}

\begin{itemize}
 \item un mot \begin{EVIDENCE1}
 décallé;
 \end{EVIDENCE1}
 \item un mot \begin{EVIDENCE2}
 pas décallé;
 \end{EVIDENCE2}
\end{itemize}

```

*Sortie*

- un mot décallé;
- un mot **pas** décallé;

### Ignorer les espaces après un environnement

`\ignorespacesafterend` permet d'ignorer les espaces qui suivent après l'utilisation de `\end{...}`.

#### Code

```

\newenvironment{EVIDENCE3}{\bfseries\ignorespaces{}}
\newenvironment{EVIDENCE4}{\bfseries\ignorespaces{\ignorespacesafterend}}

\begin{itemize}
 \item un point \begin{EVIDENCE3}
 décallé
 \end{EVIDENCE3}
 ;
 \item un point \begin{EVIDENCE4}
 pas décallé
 \end{EVIDENCE4}

```

```

 \end{itemize}

```

*Sortie*

- un point **décallé** ;
- un point **pas décallé** ;

### Enlever les espaces qui ont déjà été lus

`\unskip` (en mode horizontal) permet d'enlever les espaces qui viennent d'être tapés.

*Code*

```

\newenvironment{EVIDENCE5}{\bfseries\ignorespaces}{.}
\newenvironment{EVIDENCE6}{\bfseries\ignorespaces}{\unskip.}

\begin{itemize}
 \item\unpoint\begin{EVIDENCE5}
 décallé
 \end{EVIDENCE5}
 \item\unpoint\begin{EVIDENCE6}
 pas décallé
 \end{EVIDENCE6}
\end{itemize}

```

*Sortie*

- un point **décallé** .
- un point **pas décallé**.

## 3.5 Écrire dans des fichiers

### L'extension `filecontents`

Elle définit l'environnement `\begin{filecontents}{fichier}` qui écrit son contenu tel quel dans le fichier.

La variante étoilée `filecontents*` n'ajoute pas de commentaires au début.

*Code*

```

\begin{filecontents}{test.tex}
Voici le contenu du fichier.
Voici le contenu du fichier.
Voici le contenu du fichier.
\end{filecontents}
\verbatiminput{test.tex}

```

*Sortie*

```

%% LaTeX2e file 'test.tex'
%% generated by the 'filecontents' environment
%% from source 'Beamer3-article' on 2010/03/18.
%%
Voici le contenu du fichier.
Voici le contenu du fichier.
Voici le contenu du fichier.

```

### Les registres d'écriture

- `\newwrite\registre` définit un `\registre` d'écriture;
- `\openout\registre` fichier ouvre un fichier en écriture;
- `\write\registre{...}` écrit une ligne dans le fichier ouvert;
- `\closeout\registre` ferme le fichier ouvert après usage.

Précautions d'utilisation :

- toujours refermer un fichier après usage;
- faire précéder `\openout`, `\write` et `\closeout` de `\immediate`;
- gare aux caractères spéciaux !

### Les registres d'écriture Exemple d'utilisation

#### Code

```
\newwrite\MONWRITE
\immediate\openout\MONWRITE\test.tex
\immediate\write\MONWRITE{Salut,}
\immediate\write\MONWRITE{}
\immediate\write\MONWRITE{Je suis un fichier généré par \noexpand\
LaTeX}
\immediate\write\MONWRITE{Les \noexpand\commandes non protégées}
\immediate\write\MONWRITE{sont développées: \the\textwidth.}
\immediate\write\MONWRITE{Caractères spéciaux: \&, \#, \{, \}}
\immediate\closeout\MONWRITE
\verbatiminput{test.tex}
```

#### Sortie

Salut,

Je suis un fichier généré par \LaTeX  
Les \commandes non protégées  
sont développées: 345.0pt.  
Caractères spéciaux: &, ##, {, }

### Les registres d'écriture Registres prédéfinis

- `\write-1{...}` pour écrire dans le log;
- `\write17{...}` pour écrire dans le log et dans le terminal;
- `\write18{...}` pour transmettre une commande au système d'exploitation, comme si elle était tapée directement dans le terminal.

La commande `\write18` étant *dangereuse*, elle est désactivée par défaut. On peut l'activer avec `-shell-escape` sous Linux ou `--enable-write18` sous Windows/MikTeX.

#### Code

```
\immediate\write18{ls *r.* > test.txt}
\verbatiminput{test.txt}
```

#### Sortie

```
Coeur.txt
beamer.tex
seminar.aux
seminar.con
seminar.ps
seminar.tex
seminar.tmp
```

### 3.6 Divers

#### Message d'erreur : **no room for a new \dimen/\count**

Certaines classes ou extensions récentes utilisent beaucoup de registres de compteurs et dimensions, or L<sup>A</sup>T<sub>E</sub>X n'en a que 256.

Solution : l'extension **etex** augmente le nombre de registres disponibles.

10 avril 2009

#### La lettre @

Certaines commandes L<sup>A</sup>T<sub>E</sub>X de bas niveau sont protégées : leur nom contient le caractère @.

Par exemple \@title, \@author et \@date contiennent habituellement les informations de la page de titre du document.

- \makeatletter rend @ semblable aux autres lettres, et permet d'accéder aux commandes protégées ;
- \makeatother restaure le statut habituel de @.

#### Code

```
\makeatletter
Ce document a été écrit le \@date.
\makeatother
```

#### Sortie

Ce document a été écrit le .

#### Longueurs rapides

\longueur=... permet de stocker une valeur dans une longueur, pendant la durée du groupe en cours.

Attention au texte qui suit : il est préférable de mettre un \relax après la valeur à stocker.

#### Code

```
\begin{itemize}
 \itemsep=1cm\relax
 \item Un;
 \item deux;
 \item trois.
\end{itemize}
```

*Sortie*  
– Un ;

– deux ;

– trois.

### Afficher une longueur

`\the\longueur` affiche la valeur (en points) d’une longueur.

#### Code

```
Largeur du texte : \the\textwidth.
```

#### Sortie

Largeur du texte : 345.0pt.

### Correction italique

Dans la mesure du possible, utiliser `\textit` au lieu de `\itshape`.

Sinon, il peut être nécessaire d’ajouter manuellement une correction italique  
`\/` avant le passage au mode normal.

#### Code

```
\Huge
\textit{I}italique

\itshape I \upshape italique

\itshape I \/\upshape italique
```

#### Sortie

*I*talique

*I*talique

*I*talique

### Ajouter du texte à une commande

`\addto\commande{...}` ajoute du contenu à la fin de la `\commande`.

Par exemple, `\addto\maketitle{...}` permet de rajouter une ligne dans la page de titre.

Précautions d’emploi :

– si la `\commande` n’existe pas elle est définie comme vide juste avant l’ajout ;

- si elle existe déjà, elle ne doit pas avoir de paramètres.

### Code

```
\addto\bla{bla}
\addto\bla{bla}
\addto\bla{bla}

\bla
```

### Sortie

blablabla

## Traduction des titres avec babel

À chaque changement de langue, `babel` utilise `\captions...` pour changer les noms de titres.

Il suffit d'ajouter à la commande correspondante les titres désirés.

### Exemples

```
%Pour le français
\addto\captionsfrench{%
 \renewcommand{\refname}{Bibliographie}%
 \renewcommand{\contentsname}{Sommaire}%
}
```

```
%Pour l'allemand
\addto\captionsgerman{%
 \renewcommand{\refname}{...}%
 \renewcommand{\contentsname}{...}%
}
```

## Exercice

### Code

```
Ce document a été écrit le \@date.
```

**Solution** Pourquoi le code suivant ne produit-il pas d'erreur, bien qu'on n'ait pas utilisé `\makeatletter` ?

La commande `\@` existe déjà (elle a pour effet d'agir sur la taille des espaces), donc le code est compris comme :

### Sortie

Ce document a été écrit le date.

## Exercice

**Solution** Donner un code à mettre dans le préambule qui permette d'ajouter l'adresse email de l'auteur avec `\email{...}` dans la page de titre d'un article

### Code



```

\usepackage{url}

\makeatletter

\newcommand\@email{}
\newcommand\email[1]{%
 \renewcommand\@email{\url{#1}}%
}

\addto\maketitle{%
 \begin{center}%
 \@email
 \end{center}%
}

\makeatother

```

## 4 Programmation L<sup>A</sup>T<sub>E</sub>X avancée

### 4.1 Tests et branchements conditionnels

Les `\if...`

On peut créer un nouveau `\if` avec `\newif\ifXXX`, qui va définir trois commandes :

- `\ifXXX`, qu'on doit faire suivre de `\fi` (éventuellement après une clause `\else`);
- `\XXXtrue` pour rendre le test vrai dans le groupe en cours;
- `\XXXfalse` pour le rendre faux (c'est la valeur par défaut après `\newif`).

*Code*

```

\newif\ifmachin
\newcommand\machin[2]{%
 \ifmachin
 #1\%
 \else
 #2\%
 \fi
}%
\machin{Vrai}{Faux}
{\machintrue
\machin{Vrai}{Faux}}
\machin{Vrai}{Faux}

```

*Sortie*

```

Faux
Vrai
Faux

```

Les `\if...` Tests sur les valeurs

L<sup>A</sup>T<sub>E</sub>X contient un certain nombre de `\if...` prédéfinis pour tester des valeurs numériques :

- \ifodd nombre pour tester la parité ::

**Code**

```
\newcounter{XXX}

%Deviner ce qui se passe sans le \relax
\newcommand\parity[1]{%
 \ifodd#1\relax
 \setcounter{XXX}{1}\relax
 \else
 \setcounter{XXX}{0}\relax
 \fi
}

\parity{10}
\parity{\theXXX}
\setcounter{XXX}{5}\parity{\arabic{XXX}}
```

**Sortie**

10 est pair. 0 est pair. 5 est impair.

- \ifnum num1 op num2 pour comparer deux valeurs (op peut être =, < ou >) ::

**Code**

```
\newcommand\compare[2]{%
 \ifnum#1=#2\relax#1estégalà#2\else
 \ifnum#1>#2\relax
 \setcounter{XXX}{1}\relax
 \else
 \setcounter{XXX}{0}\relax
 \fi
}

\compare{-50}{-50}
\compare{3}{+4}
\compare{\theXXX}{100}
```

**Sortie**

-50 est égal à -50. 3 est plus petit que +4. 5 est plus petit que 100.

- \ifdim dim1 # dim2 pour comparer deux longueurs ::

**Code**

```
\newcommand\PROPORTIONS[1]{%
 \settowidth\WIDTH{#1}%
 \settowidth\HEIGHT{#1}%
 \ifdim\WIDTH>\HEIGHT\relax
 \setcounter{XXX}{1}\relax
 \else
 \setcounter{XXX}{0}\relax
 \fi
}

\PROPORTIONS{I},\PROPORTIONS{w}.
```

*Sortie*

"I" est plus haut que large, "w" est plus large que haut.

- `\ifcase num ... \or ... \or ... \else ... \fi` pour effectuer une action différente si `num` vaut 0, 1, 2... *ou une autre valeur* :

*Code*

```
\newcommand\numdujour[1]{%
 \ifcase#1\relax\ERREUR\or_lundi\or_mardi\or_mercredi\or_
 jeudi\or_vendredi\or_samedi\or_dimanche\else\ERREUR\fi
}

\numdujour{0},\numdujour{1},\numdujour{7},\numdujour{-3}.
```

*Sortie*

ERREUR, lundi, dimanche, ERREUR.

#### Les `\if...` Tests sur le mode en cours

- `\ifhmode` pour tester si on est en mode horizontal ;;

*Code*

```
\newcommand\BIDULE[1]{%
 \ifitshape#1%
 \ifhmode
 \unskip
 \\\%
 \fi}%
}
\Large

\BIDULE{\unskip}\fonctionne.

\BIDULE{FONCTIONNE}\plutôt bien.

{\itshape\FONCTIONNE}\pas.
```

*Sortie*

fonctionne.

*FONCTIONNE* plutôt bien.

*FONCTIONNE* pas.

- `\ifvmode` pour tester si on est en mode vertical ;;

*Code*

```
\newcommand\SHOWVMODE{\ifvmode[vertical]\else[horizontal]\fi
}

\SHOWVMODE\en_début_de_paragraphe,\SHOWVMODE\au_milieu.

\begin{itemize}
 \item Dans une liste:
 \item \SHOWVMODE\au_début_d'un_item,\SHOWVMODE\au_milieu.
\end{itemize}\SHOWVMODE\à_la_fin_de_la_liste.
```

*Sortie*  
[vertical] en début de paragraphe, [horizontal] au milieu.

- Dans une liste :
- [vertical] au début d’un item, [horizontal] au milieu.
- [vertical] à la fin de la liste.
- `\ifmmode` pour tester si on est en mode mathématique ;;

**Code**

```
\newcommand\DOLLAR{%
\ifmmode\mathdollar\else\textdollar\fi

\DOLLAR_(en_mode_texte),_\$\DOLLAR+1$_(en_mode_math)_et
\[\sum_{\DOLLAR}\]
(en_mode_display).
```

*Sortie*  
\$ (en mode texte), \$ + 1 (en mode math) et

$$\sum_{\$}$$

(en mode display).

- `\ifinner` pour tester si on est en mode encapsulé (mathématiques en mode texte ou texte sans paragraphe associé comme `\hbox` ou `\vbox`).

**Les `\if...` Tests supplémentaires**

- `\iftrue` est toujours vrai ;;

**Code**

```
\iftrue
\iftrue C'est vrai
\else
\iftrue C'est faux
\fi
```

*Sortie*  
C'est vrai

- `\iffalse` est toujours faux ;;

**Code**

```
\iffalse
\iffalse C'est vrai
\else
\iffalse C'est faux
\fi
```

*Sortie*  
C'est faux

- `\IfFileExists{fichier}{action1}{action2}` effectue `action1` ou `action2` selon que le fichier existe ou non :

*Code*

```
\IfFileExists{\jobname.bidon}{\jobname.bidon_existe}{\jobname.
bidon_n'existe_pas},
\IfFileExists{\jobname.tex}{\jobname.tex_existe}{\jobname.tex_
n'existe_pas}.
```

*Sortie*

Beamer3-article.bidon n'existe pas, Beamer3-article.tex existe.

### Les `\if...` Tests sur les commandes

- `\ifcsname nom\endcsname` pour tester si la commande `\nom` existe ;:

*Code*

```
\ifcsname_only\endcsname
_Ce_document_utilise_beamer.
\else
_Ce_document_n'utilise_pas_beamer.
\fi
```

*Sortie*

Ce document utilise beamer.

- `\ifdefined\commande` pour tester si la commande `\commande` existe ;:

*Code*

```
\ifdefined\letter\relax
_Ce_document_est_une_lettre.
\else
_Ce_document_n'est_pas_une_lettre.
\fi
```

*Sortie*

Ce document n'est pas une lettre.

- `\ifx\commande1\commande2` pour tester si `\commande1` et `\commande2` sont *exactement* identiques :

*Code*

```
\newcommand\AAA{Bonjour}
\newcommand\BBB{Bonjour}
\newcommand\CCC{Bonjour!}
\newcommand\DDD{{Bonjour}}
\let\EEE\AAA
\newcommand\COMPARE[2]{\ifx#1#2\relax_égal\else_pas_égal\fi}

\COMPARE{\AAA}{\BBB},_ \COMPARE{\AAA}{\BBB},_ \COMPARE{\AAA}{\
CCC},_ \COMPARE{\AAA}{\DDD},_ \COMPARE{\AAA}{\EEE}.
```

*Sortie*

égal, égal, pas égal, pas égal, égal.

### Les `\if...` Tests supplémentaires introduits par des extensions

- L’extension `ifpdf` introduit `\ifpdf` pour tester si on est en train de compiler avec  $\text{\LaTeX}$  ou  $\text{Pdf\LaTeX}$  ;

#### *Code*

```
\ifpdf
 Ce document a été compilé en utilisant Pdf\LaTeX.
\else
 Ce document a été compilé en utilisant \LaTeX.
\fi
```

#### *Sortie*

Ce document a été compilé en utilisant  $\text{\LaTeX}$ .

- l’extension `changepage` (récente) introduit `\ifoddpage` pour tester si la page en cours est impaire (à faire précéder de `\checkoddpage` avant chaque utilisation). ;

En effet, dans certains cas, `\ifodd\value{page}\relax` ne fonctionne pas.

L’option `[strict]` de `changepage` permet de régler le problème.

#### *Code*

```
\newcommand\AFFICHE{\checkoddpage\ifoddpage Impaire\else Paire
\fi}

\AFFICHE.\pause
```

#### *Sortie*

Paire.

- l’extension `chnpage` (ancienne) fait la même chose avec `\ifcpoddpage`.

### Les `\if...` L’extension `ifthenelse`

- `\ifthenelse{condition}{vrai}{faux} ;;`

#### *Code*

```
\ifthenelse{1<2}{Vrai}{Faux}

\ifthenelse{1>2}{Vrai}{Faux}
```

#### *Sortie*

Vrai

Faux

- `\boolean{XXX}` permet de tester `\ifXXX ;;`

#### *Code*

```
\ifthenelse{\boolean{true}}{Vrai}{Faux}

\ifthenelse{\boolean{false}}{Vrai}{Faux}

\ifthenelse{\boolean{pdf}}{Vrai}{Faux}
```

*Sortie*  
Vrai  
Faux  
Faux

– \isodd{nombre} pour tester si nombre est pair ;;

**Code**  
\ifthenelse{\isodd{1}}{Impair}{Pair}  
\ifthenelse{\isodd{2}}{Impair}{Pair}

*Sortie*  
Impair  
Pair

– \lengthtest{longueur1 op longueur2} pour comparer des longueurs ;;

**Code**  
\ifthenelse{\lengthtest{1pt<2pt}}{Plus petit}{Plus grand}  
\ifthenelse{\lengthtest{2pt<1pt}}{Plus petit}{Plus grand}

*Sortie*  
Plus petit  
Plus grand

– \isundefined{\commande} pour tester si \commande est indéfinie ;;

**Code**  
\ifthenelse{\isundefined{\BIDON}}{Indéfinie}{Définie}  
\ifthenelse{\isundefined{\verb}}{Indéfinie}{Définie}

*Sortie*  
Indéfinie  
Définie

– \equal{texte1}{texte2} pour tester si texte1 est égal à texte2 (après expansion) ;;

**Code**  
\newcommand\XXX{Bonjour}  
\newcommand\YYY{Bon}  
\newcommand\ZZZ{jour}  
\ifthenelse{\equal{{bonjour}}{{bonjour}}}{\ 'Egal}{Différent}  
\ifthenelse{\equal{\XXX}{\YYY\ZZZ}}{\ 'Egal}{Différent}

*Sortie*  
Différent  
Égal

- \not, \or, \and (ou \NOT, \OR, \AND) pour faire des opérations booléennes ;;

**Code**

```
\ifthenelse{1>3\or\boolean{false}\or\not 3<1}{Vrai}{Faux}

\ifthenelse{1<3\and\boolean{true}\and\not 1<3}{Vrai}{Faux}
```

*Sortie*  
Vrai

Faux

- \ ( et \) pour grouper des expressions dans des calculs :

**Code**

```
\ifthenelse{\not 1<2\or 1<2}{Vrai}{Faux}

\ifthenelse{\not \ (1<2\or 1<2)}{Vrai}{Faux}
```

*Sortie*  
Vrai

Faux

## Exercice

**Solution** Définir un environnement `ignore` qui permette d'ignorer tout ce qui se trouve entre son début et sa fin.

**Code**

```
%Remarque:\l'environment"comment"de\l'extensionverbatimest\
beaucoupplusrobuste!
\newif\ifignore
\ignoretrue

\newenvironment{ignore}{%
\ifignore
\else
}%%
\fi
}
```

## Exercice

**Solution** Définir une commande `\cardinal{nombre}` qui affiche le `nombre` en respectant les conventions typographiques françaises : ..., -2, -1, zéro, un, deux, ..., huit, neuf, 10, 11, ..., 98, 99, cent, 101, ..., 999, mille, 1001, ...

**Code**

```
\newcommand\cardinal[1]{%
\ifcase#1\relax\zéro\or un\or deux\or trois\or quatre\or cinq\or
\six\or sept\or huit\or neuf\else
\ifnum#1=100\relax
```



```

 \cent%
 \else
 \ifnum#1=1000\relax
 mille%
 \else
 #1%
 \fi
 \fi
\fi
}

```

## 4.2 Écrire une nouvelle extension

### Exemple minimaliste

*Fichier `monextension.sty`*

```

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{monextension}[2009/04/17_Ma_première_extension,V
1.0]

%Extensions
\RequirePackage{graphicx,url,fancybox}

%Commandes de l'extension
\newcommand\encadre[1]{\ovalbox{#1}}

\DeclareFixedFont\gotnormal{U}{yfrak}{m}{n}{10pt}
\DeclareTextFontCommand\textgot{\gotnormal}

```

### Les bases

Le nom d'un fichier d'extension se termine par l'extension `.sty` et doit être accessible à  $\text{\LaTeX}$ .

Rien de particulier, à part :

- le fichier doit commencer par

#### *Code*

```

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{nom}[Date_Description_courte,Version]

```

- le caractère `@` n'est pas protégé ;
- pour charger une autre extension on utilise non pas `\usepackage` mais `\RequirePackage[options]{extension}`.

### Les options basiques

- `\DeclareOption{nom}{action}` déclare une option de l'extension, qui effectue une action donnée.

#### *Exemple*

```

\newif\ifpretty
\DeclareOption{pretty}{\prettytrue}

```

- `\ProcessOptions` (à faire suivre de `\relax`) lance effectivement les actions prévues en fonction des options de chargement de l’extension définies par l’utilisateur.

### *Exemple*

```
%Déclaration des actions :
\DeclareOption...
\DeclareOption...

%Exécution des actions :
\ProcessOptions\relax
```

### Les options avancées

- `\DeclareOption*{action}` déclare une action par défaut pour toutes les autres options non déclarées (on peut récupérer le nom de l’option en cours avec `\CurrentOption`);
- `\PassOptionsToPackage{options}{extension}` permet d’ajouter des options supplémentaires lorsqu’on chargera `extension` avec `\RequirePackage`.

### *Exemple pour passer toutes les options à geometry*

```
\DeclareOption*{%
 \PassOptionsToPackage{\CurrentOption}{geometry}%
}

\ProcessOptions\relax

\RequirePackage{geometry}
```

### Commandes différées

- `\AtEndOfPackage{...}` exécute son argument à la fin de l’extension;
- `\AtBeginDocument{...}` exécute son argument juste après `\begin{document}`;
- `\AtEndDocument{...}` exécute son argument juste avant `\end{document}`.

### *Exemple pour changer l’indentation des paragraphes*

```
\DeclareOption{superindent}{%
 \AtBeginDocument{%
 \setlength\parindent{1cm}%
 }%
}
```

### Exercice

## 4.3 Écrire une nouvelle classe

### Exemple minimaliste

#### *Fichier maclasse.cls*

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{maclasse}[2009/04/17 Ma première classe, V1.0]

%Base
\LoadClassWithOptions{article}
```

```
%Extensions
\RequirePackage{geometry}
```

## Les bases

Le nom d'un fichier de classe se termine par l'extension `.cls`.

Rien de particulier, à part :

- le fichier doit commencer par

### *Code*

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{nom}[Date>Description,courte,Version]
```

- le caractère **@** n'est pas protégé ;
- pour charger une autre extension on utilise non pas `\usepackage` mais `\RequirePackage[options]{extension}` ;
- pour modifier une classe existante on utilise `\LoadClass[options]{classe existante}`.

## Les options

Même principes que pour une extension, avec en plus :

- `\LoadClassWithOptions{classe existante}` au lieu de `\LoadClass` charge la classe existante avec exactement les mêmes options ;
- `\PassOptionsToClass{options}{classe existante}` ajoute une option qui sera activée au moment de `\loadclass`.

### *Exemple pour passer toutes les options à `article`*

```
%Si on ne déclare aucune option particulière, ce qui suit est
équivalent à \LoadClassWithOptions{article}
\DeclareOption*{%
 \PassOptionsToClass{\CurrentOption}{article}%
}

\ProcessOptions\relax

\LoadClass{article}
```

## Exercice

**Solution** Faire une classe simple dérivée de Beamer avec une option `article`

### *Code*

```
\newif\ifarticle
\DeclareOption{article}{\articletrue}

\DeclareOption*{
 \ifarticle
 \PassOptionsToClass{\CurrentOption}{article}
 \else
 \PassOptionsToClass{\CurrentOption}{beamer}
 \fi
}
```

```

}

\ProcessOptions\relax

\ifarticle
\LoadClass{article}
\RequirePackage{beamerarticle}
\else
\LoadClass{beamer}
\fi

```

### Exercice [*Difficile*]

## 4.4 Définition avancée des commandes

### Définition de commandes

Les moyens propres à L<sup>A</sup>T<sub>E</sub>X :

- `\newcommand\commande[n]{...}` définit une nouvelle `\commande` à `n` arguments (une erreur est déclenchée si `\commande` existe déjà, ou si son nom commence par `end`);
- `\providecommand\commande[n]{...}` ne définit `\commande` que si elle n'existe pas déjà;
- `\renewcommand\commande[n]{...}` redéfinit une `\commande` existante (une erreur est déclenchée si `\commande` n'existe pas déjà);
- `\newenvironment{envi}[n]{avant}{après}` définit les deux commandes `\envi` et `\endenvi`, qui ont le même effet que `\begin{envi}` et `\end{envi}` à deux différences près :
  1. pas d'effet de localisation propre aux groupements (retour aux couleurs et à la police précédente, ...),
  2. pas de vérification d'imbrication correcte;
- `\renewenvironment{envi}[n]{avant}{après}` redéfinit `\envi` et `\endenvi`.

Toute (re)définition d'une `\commande` par l'un de ces moyens est globale et non limitée au groupe en cours.

### Plusieurs arguments optionnels

`\newcommand\commande[n][valeur]{...}` rend le premier argument `#1` optionnel, avec une `valeur` par défaut s'il n'est pas donné.

On peut définir des commandes à plusieurs arguments optionnels à l'aide de définitions intermédiaires :

#### Code

```

%Le begingroup/endgroup sert seulement à revenir à la couleur
%normale à la fin de FLASHY
\newcommand\FLASHYAUX[2][yellow]{\colorbox{#1}{#2}\endgroup}
\newcommand\FLASHY[1][blue]{\begingroup\color{#1}\FLASHYAUX}

\FLASHY{Un peu}

```

```
\FLASHY[violet]{Beaucoup}

\FLASHY[red][pink]{À la folie}
```

*Sortie*

Un peu

Beaucoup

À la folie

### Variantes étoilées

`\@ifstar{étoile}{pas étoile}` teste si le prochain caractère est un caractère `*` (et le cas échéant, le fait « disparaître ») et si oui lit "étoile", sinon "pas étoile".

#### Code

```
\makeatletter
\newcommand\FRAMEA[1]{\fbox{#1}}
\newcommand\FRAMEB[1]{\ovalbox{#1}}
\newcommand\FRAME{\@ifstar{\FRAMEA}{\FRAMEB}}
```

Deux `\FRAME{styles}` pour `\FRAME*{encadrer}`.

*Sortie*

Deux styles pour encadrer.

Sur le même modèle :

- `\@ifnextchar{c}{égal}{pas égal}` permet de tester si le prochain jeton (caractère et code de catégorie) est égal à `c` ;
- `\@ifnextcharacter{c}{égal}{pas égal}` fait la même chose mais ne s'occupe pas des codes de catégorie.

### Définition de macros Les moyens propres à T<sub>E</sub>X

`\def\macro#1#2...#n{...}` fait quasiment la même chose que `\newcommand\macro[n]{...}` à quelques différences près :

- la définition est valable uniquement dans le bloc en cours (elle disparaît ensuite) ;
- pas de vérification de l'existence préalable de `\macro` ;
- la partie `#1#2...#n` autorise un « parsing » plus complexe.

#### Code

```
\def\TRUC#1{\textsc{#1}}
\def\TRUCDOUBLE(#1,#2){\textbf{#1}\textit{#2}}
\def\SUPERTRUC#1\par{{\red#1\par}}

\TRUC{Un argument standard}. \TRUCDOUBLE(Deux arguments, spéciaux). \
\SUPERTRUC{Un argument qui va jusqu'à la fin du paragraphe} en
cours.
```

Prochain paragraphe...

*Sortie*

UN ARGUMENT STANDARD. Deux arguments *spéciaux*. Un argument qui va jusqu'à la fin du paragraphe en cours.

Prochain paragraphe...

### Définition de macros Les moyens propres à T<sub>E</sub>X

`\let\macro=\macroa` recopie la signification *exacte* de `\macroa` dans la séquence de contrôle `\macro`.

- L'assignation est valable uniquement dans le bloc en cours (ensuite, tout redevient comme avant) ;
- pas de vérification de l'existence préalable de `\macroa` (auquel cas `\macro` sera elle-même indéfinie) ;
- le signe égal est facultatif.

*Code*

```
\let\bd=\description
\let\ed=\enddescription
\let\im\item
\bd
 \im[Un]deux
 \im[Trois]quatre
\ed
```

*Sortie*

Un deux

Trois quatre

### Définition de macros Les moyens propres à T<sub>E</sub>X

`\edef\macro{texte}` va développer le `texte` (prise en compte des `\if...`, remplacement des macros par leur valeur...) pour en faire une liste de jetons de texte. Cette liste sera la nouvelle définition de la `\macro`.

- La définition est valable uniquement dans le bloc en cours ;
- c'est un moyen de développer des commandes sans les afficher ;
- la liste (après expansion) doit être composée de texte uniquement.

*Code*

```
\def\listevide{}
\let\liste=\listevide
\def\ajouter#1{%
 \ifx\liste\listevide
 \edef\liste{#1}%
 \else
 \edef\liste{\liste,#1}%
 \fi
}

\ajouter{un}\ajouter{deux}\ajouter{trois}\ajouter{quatre}
Voici\la\liste:\liste.
```

*Sortie*

Voici la liste : un, deux, trois, quatre.

### Préfixes de définition

Il y en a trois, à placer avant `\def`, `\let`, `\edef...` :

- `\global` rend la portée de la définition globale;
- `\outer` rend la portée de la définition localisée au groupe de niveau supérieur;
- `\long` avec `\def` autorise des définitions de macros dont les arguments peuvent contenir des sauts de paragraphe.

Les commandes définies avec `\newcommand` sont automatiquement `\long`. Ces préfixes peuvent aussi s'utiliser avec les `\ifXXX` (`\global\XXXtrue`), avec `\addto` ou encore avec des registres de longueurs (`\global\parskip=1cm`).

Il existe aussi des formes abrégées :

- `\gdef` pour `\global\def`;
- `\ldef` pour `\long\def`;
- `\xdef` pour `\global\edef`.

### Appeler une commande par son nom

- `\csname texte\endcsname` désigne (après expansion de `texte`) la macro `\texte`;
- il faut utiliser `\expandafter` avant les définitions dans ce cas.

#### Code

```
\newcounter{moncompteur}
\newcommand\changeitemlabel[2]{%
 \setcounter{moncompteur}{#1}%
 \expandafter\def\csname\labelitem\roman{moncompteur}\endcsname
 {#2}%
}

\changeitemlabel{1}{\rule{1ex}{1ex}}
\begin{itemize}
 \item[\black\rule{1ex}{1ex}] Un
 \item[\black\rule{1ex}{1ex}] Deux
\end{itemize}
```

#### Sortie

- Un
- Deux

### Débogage

- `\show\commande` affiche la définition d'une macro;
- `\showthe\registre` affiche la valeur d'un registre (une longueur par exemple).

### Exercice

**Solution Définir une commande prenant un argument optionnel encadré par `<...>`**

#### Code

```

\makeatletter
\def\machinARG<#1>{\Avec\argument:\#1}
\def\machinNOARG{\Sans\argument}
\def\machin{\@ifnextchar<\machinARG\machinNOARG}
\machin<oui>.\machin.

```

## 4.5 Boucles

### Boucles "tant que" en T<sub>E</sub>X

- \@whilenum test\do{...} répète son argument tant que la comparaison test entre nombres est vraie;
- \@whiledim test\do{...} fait la même chose pour un test sur des longueurs.

#### Code

```

\newlength\LARGEUR
\newlength\LARGEURTOTALE
\makeatletter
\newcommand\REPLIR[2]{%
 \settowidth\LARGEUR{#2}%
 \setlength\LARGEURTOTALE{0pt}%
 \@whiledim\LARGEURTOTALE<#1\do{%
 \addtolength\LARGEURTOTALE{\LARGEUR}%
 }%
}

\REPLIR{3cm}{+}\REPLIR{3cm}{.}\REPLIR{3cm}{\#}

```

#### Sortie

```

+++++++
.....
#####

```

### Boucles "tant que" en L<sup>A</sup>T<sub>E</sub>X Avec l'extension **ifthen**

\whiledo{condition}{...} répète son second argument tant que la condition (décrite avec les mêmes conventions que \ifthenelse) est vraie.

#### Code

```

\newcounter{ca}\newcounter{cb}
\newcommand\pgcd[2]{%
 \setcounter{ca}{#1}%
 \setcounter{cb}{#2}%
 \pgcd(#1,#2)\begin{array}[t]{cl}%
 \whiledo{\not\(\value{ca}=\value{cb}\)}{%
 \ifthenelse{\value{ca}>\value{cb}}{%
 \addtocounter{ca}{-\value{cb}}%
 }{%
 \addtocounter{cb}{-\value{ca}}%
 }%
 }%
 \pgcd(\theca,\thecb)\%
}
\pgcd=
\end{array}

```



```
}
\[\pgcd{512}{388}\]
```

*Sortie*

```
pgcd(512,388)=pgcd(124,388)
 =pgcd(124,264)
 =pgcd(124,140)
 =pgcd(124,16)
 =pgcd(108,16)
 =pgcd(92,16)
 =pgcd(76,16)
 =pgcd(60,16)
 =pgcd(44,16)
 =pgcd(28,16)
 =pgcd(12,16)
 =pgcd(12,4)
 =pgcd(8,4)
 =pgcd(4,4)
 =4.
```

## Boucles "pour" en T<sub>E</sub>X

*Attention* au code de catégorie de ‘:’ qui est changé par `french` ou `[français]{babel}` !

- `\@for\var:=liste\do{...}` répète son argument avec `\var` qui parcourt les éléments de la `liste`. Les éléments de la liste doivent être séparés par des virgules ;
- `\@tfor\var:=texte\do{...}` fait la même chose, en parcourant toutes les lettres du `texte`.

### *Code*

```
\begin{itemize}
\catcode'\:=12
\makeatletter
\@for\x:=1,2,3,soleil!\do{\item\x}
\end{itemize}
```

### *Sortie*

- 1
- 2
- 3
- soleil!

## Exercice

**Solution** Écrire 100 fois « Je n'utiliserai pas les boucles en cours de L<sup>A</sup>T<sub>E</sub>X »

### *Code*

```

\newcounter{puniton}
\makeatletter
\@whilenum\thepuniton<100\do{%
 \Je n'utiliserai pas les boucles en cours de LaTeX\%
 \stepcounter{puniton}%
}

```

## Exercice

**Solution** Écrire une commande `\vertical` qui permette d'afficher verticalement son argument dans un tableau

### Code

```

\makeatletter
\catcode'\:=12

\newcommand\VERTICAL[1]{%
 \begin{tabular}[t]{c}%
 \@tfor\XXX:=#1\do{%
 \XXXX\XXX\%
 }%
 \end{tabular}%
}

```

## 4.6 Codes de catégorie

### Les "tokens" et leurs "catcodes"

En première lecture,  $\text{\LaTeX}$  transforme le texte en une liste de jetons (tokens) :

- soit des "caractères" (table ASCII) ;
- soit des "séquences de contrôle" (noms de commandes).

### Code

```
2\est\textit{premier}.
```

*%Commentaire*

### Jetons

```
2 \ e s t \ textit { p r e m i e r } . \ par par
```

### Les "tokens" et leurs "catcodes"

En plus de sa valeur en tant que caractère, chaque jeton qui n'est pas une séquence de contrôle possède un attribut supplémentaire compris entre 0 et 15 : le code de catégorie (catcode).

### Code

```
2\est\textit{premier}.
```

*%Commentaire*

*Jetons*

$2_{12} \sqcup_{10} e_{11} s_{11} t_{11} \sqcup_{10} \boxed{\text{textit}} \{_1 p_{11} r_{11} e_{11} m_{11} i_{11} e_{11} r_{11} \}_2 \cdot_{12} \sqcup_5 \boxed{\text{par}} \boxed{\text{par}}$

## Codes de catégorie par défaut

|                                                                                   |                                                                                                           |
|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| 0 – caractère d'échappement (début d'une séquence de contrôle) : <code>'\'</code> | 8 – indice : <code>'_'</code>                                                                             |
| 1 – délimiteur gauche : <code>'{'</code>                                          | 9 – caractères ignorés : <code>'^^00'</code>                                                              |
| 2 – délimiteur droit : <code>'}'</code>                                           | 10 – espaces : <code>'_'</code> , <code>'^^I'</code>                                                      |
| 3 – délimiteur mathématique : <code>'\$'</code>                                   | 11 – lettres (peuvent former des caractères) : <code>'a'</code> , <code>'z'</code> , <code>'E'</code> ... |
| 4 – tabulation : <code>'&amp;'</code>                                             | 12 – autres caractères (tous ceux) : <code>'@'</code> , <code>'1'</code> , <code>'('</code> ...           |
| 5 – fin de ligne : <code>'^M'</code>                                              | 13 – caractères actifs (peuvent être contrôlés) : <code>'~'</code>                                        |
| 6 – paramètre de macro : <code>'#'</code>                                         | 14 – début d'un commentaire : <code>'%</code>                                                             |
| 7 – exposant : <code>'^'</code>                                                   | 15 – caractères invalides.                                                                                |

## Changer les codes de catégorie

Pour faire passer le code de catégorie des prochains caractères  $\alpha$  à la valeur  $n$  dans le groupe en cours :

`\catcode'\alpha=n`

Par exemple :

- `\makeatletter` change le code de catégorie de `@` à 11 (`\catcode'\@=11`);
- `\makeatother` change le code de catégorie de `@` à 12 (`\catcode'\@=12`).

### Code

```
\catcode'\-=11
\newcommand\super-commande{Super_ commande!_}

\super-commande\super-commande\super-commande
```

*Sortie*

Super commande! Super commande! Super commande!

## La catégorie 13 (caractères actifs)

Les caractères de cette catégorie peuvent être utilisés comme une séquence de contrôle.

Par exemple, les extensions `french` ou `[francais]{babel}` rendent les caractères de ponctuation doubles (`:` `;` `!` `?`) actifs pour gérer correctement les espaces à la française.

### Code

```
Le_@_par_défaut_est_muche?

\catcode'\@=13
\newcommand@{\MVAt}
Utiliser_le_@_de_l'extension_marvosym!
```

*Sortie*

Le @ par défaut est moche ?

Utiliser le @ de l'extension marvosym !

### Règle d'or des codes de catégories

Une fois qu'un caractère a été lu, son code de catégorie ne peut plus être changé.

*Pourquoi l'utilisation de \letitre va-t-elle provoquer une erreur ?*

```
\newcommand\letitre{%
 \makeatletter
 \let\@title%
 \makeatother
}
```

### Fonctionnement (simplifié) de la commande \verb

1. les codes de catégorie de tous les caractères spéciaux (à part le saut de ligne) sont changés en 12 (autre) ;
2. on commence un groupe ;
3. le premier caractère qui suit ( $\alpha$ ) est lu ;
4. son code de catégorie est changé en 13 (actif) ;
5. la commande associée à  $\alpha$  termine le groupe (et donc restaure les catégories initiales).

*Pourquoi ce code ne produit-il pas d'erreur, et qu'affiche-t-il ?*

```
\newcommand\test[1]{\verb+#1+}
```

```
\test{Oui}Non+
```

*Réponse*

Oui+Non

### Significations spéciales du caractère ‘^’

Lorsqu'il est doublé, il permet d'insérer des caractères sans les taper explicitement :

- ^^xy est lu comme un seul caractère dont le code ASCII est xy en hexadécimal ;
- ^^ $\alpha$  est lu comme un seul caractère  $\beta$  en appliquant la règle suivante :
  - soit  $n$  le code ASCII de  $\alpha$  ;
  - si  $64 \leq n \leq 127$  alors  $\beta$  a pour code ASCII  $n - 64$  ;
  - si  $0 \leq n \leq 63$  alors  $\beta$  a pour code ASCII  $n + 64$ .

Par exemple, `^^M` désigne le caractère de saut de ligne, `^^I` le caractère de tabulation du clavier.

#### Code

```
\itshape_Pas_facile_de_taper_rapidement!

%Pas_de_chance,_la_touche_"e"_(code_ascii_65_en_hexadécimal)_de_mon
 _clavier_vient_de_se_casser...
\itshap^^65_Pas_facil^^65_d^^65_tap^^65_r_rapid^^65_m^^%nt!
```

#### Sortie

*Pas facile de taper rapidement!*

*Pas facile de taper rapidement!*

#### Exercice

**Solution** Donner un code **LaTeX** permettant de taper directement le symbole de l'euro, par exemple en se basant sur le symbole `\texteuro` de l'extension `textcomp`

#### Code

```
\catcode'\e=13
\newcommand{\texteuro}
%On peut aussi utiliser les symboles \EUR ou \EURtm de l'extension
marvosym
```

#### Exercice

**Solution** Comment obliger **LaTeX** à respecter les mêmes sauts de ligne que ceux tapés dans le document, au lieu de les transformer en espaces ?

#### Code

```
\catcode'\^^M=13%
\def^^M{\ifhmode\newline\fi}%
```

#### Exercice [Difficile]

**Solution** Donner un code **LaTeX** permettant d'afficher en rouge les commentaires d'un document

#### Code

```
\def\makeEOLother{\catcode'\^^M=12\relax}
{%
 \makeEOLother%
 \gdef\makeEOLspace{\catcode'\^^M=5\relax}%
 \gdef\eatline#1^^M{%
 \color{red}_#1}%
 \makeEOLspace%
 \makePERCENTactive%
}%
```

```

}

\def\makePERCENTactive{\catcode\%=13\relax}
\makePERCENTactive
\def\makePERCENTother{\catcode'\%=12\relax}
\def%\{\makeEOLother\makePERCENTother\eatline}

```