Présentation de l'utilisation de LaTeX*

Nathaniel Carré

15 mars 2024

1 Introduction

1.1 Présentation

Contrairement à un logiciel comme Microsoft Word, LaTeX ne permet pas de faire de l'édition WYSIWYG (*what you see is what you get*), c'est-à-dire que dans un document d'édition, on ne voit pas le résultat qu'on va obtenir, mais un simple document texte avec différentes commandes, qu'on doit compiler pour obtenir le document final.

1.2 Installation

De manière similaire à l'utilisation des langages de programmation, il faut installer deux choses pour pouvoir travailler avec LaTeX : une distribution LaTeX et un éditeur de texte.

Pour la distribution, je recommande Texlive (`https://www.tug.org/texlive/`), qui est la plus répandue et disponible sur tous les systèmes d'exploitation.

Quelque soit le système d'exploitation, si vous tapez la commande `texdoc lshort-fr` après avoir installé Texlive, cela lancera un pdf qui est un manuel d'introduction (184 pages) à LaTeX. On y trouve des exemples bien plus détaillés que dans ce document.

Pour l'éditeur, vous avez un vaste choix. Le mien est Texmaker, mais contrairement aux distributions, le choix de l'éditeur fait moins l'unanimité.

*À modifier avec le bon titre

1.3 Compilation

Une distribution TEX contient généralement plusieurs compilateurs. Le plus répandu (que j'utilise) est `pdflatex`, mais il commence à devenir obsolète sur certains aspects. Le compilateur `lualatex` est se répand de plus en plus et est une très bonne alternative. Attention, le résultat du pdf obtenu change selon le compilateur utilisé, voire peut échouer si des paquets ou polices sont manquants pour l'un des compilateurs.

La plupart des éditeurs standards de LATEX proposent différentes options de compilation.

1.4 D'autres manières de travailler avec LATEX

Il est possible de se passer de l'utilisation d'un éditeur et d'une distribution. On peut mentionner le site Overleaf, qui permet de faire de l'édition LATEX en ligne, ou la solution LyX qui permet de faire du LATEX en WYSIWYM (*what you see is what you mean*, autrement dit de contrôler de manière un peu plus simple la structure du document, sans faire apparaître les commandes sous-jacentes, mais sans que l'éditeur ne représente non plus le résultat final.

2 Structure d'un document LATEX

Un fichier LATEX est composé d'une entête, dans laquelle on importe généralement les paquets nécessaires à la compilation, et on définit les caractéristiques du document. Le corps du document est encadré par :

```
\begin{document}

\end{document}
```

2.1 Commandes et environnements

LATEX contient un grand nombre de commandes permettant de faire une mise en forme particulière, ou d'intégrer un symbole. Par exemple, l'affichage de LATEX se fait avec la commande `\LaTeX{}`. L'affichage de `\LaTeX{}` en police `tt` (*typewritter*, c'est-à-dire machine à écrire) se fait avec `\verb"\LaTeX{}"`.

On remarque que les commandes commencent par une contre-oblique (backslash) \, qui est le caractère usuel d'échappement. La plupart des commandes sont suivies d'une paire d'accolades, entre lesquelles on peut mettre des arguments. Par exemple, `\textit{typewritter}` écrira en italique *typewritter*. Ce n'est pas systématique, comme par exemple l'environnement verbatim

précédent, qui peut utiliser différents délimiteurs (c'est ce qui m'a permis d'écrire des accolades dans l'environnement), ou des commandes sans délimiteurs, comme par exemple # qui s'écrit avec `\#`.

Une autre manière de gérer la mise en forme est la notion d'environnement. On peut voir cela comme une commande avec un très long argument. Un environnement est généralement encadré par :

```
\begin{environnement}

\end{environnement}
```

comme c'était le cas pour l'environnement `document`. Il existe un très grand nombre d'environnements très pratiques, comme les tableaux, les figures, les dessins, etc.

2.2 Entête

Outre quelques caractéristiques de forme, l'entête peut contenir l'importation des paquets (les bibliothèques nécessaires à la compilation, qui contiennent des commandes toutes faites), ainsi que des commandes ou macro personnalisées.

L'importation de paquets se fait via la commande `\usepackage{nom_du_paquet}`. La plupart des paquets disponibles sont généralement accompagnés d'une documentation détaillant leur utilisation.

Remarque. On peut créer des alias pour de nouvelles commandes, ou pour renommer des commandes existantes. Il faut simplement imaginer que la commande remplace une séquence plus longue. Par exemple, on peut écrire `\newcommand{\lra}{\longrightarrow}` si on veut raccourcir la commande pour obtenir $\longrightarrow$. On peut imaginer des choses plus complexes, voire sur plusieurs lignes.

On peut écrire `\newcommand{\limite}[2]{\underset{#1 \rightarrow #2}{\lra}}` (avec deux arguments), pour pouvoir taper `\limite{n}{+\infty}` et obtenir $\underset{n\rightarrow+\infty}{\longrightarrow}$.

On peut renommer une commande existante, par exemple `\renewcommand{\leq}{\leqslant}` pour que `\leq` affiche $\leqslant$ au lieu de $\leq$.

2.3 Corps du document

2.3.1 Sections

On peut structurer le document en séparant les parties, en utilisant les commandes :

```
\section{Nom de section}
\subsection{Nom de sous-section}
\subsection{Nom de sous-sous-section}

\section*{Nom de section non numérotée}
```

Les versions étoilées existent aussi pour les sous-sections et sous-sous-sections. Il n'existe pas de `\subsubsubsection`. Il existe d'autres commandes pour d'autres classes que `article` (comme `\chapter`).

2.3.2 Espacements

Les espacements répondent aux règles suivantes :

— les espaces consécutives sont ignorées au dela de la première ;

— un retour à la ligne est ignoré ;

— une ligne blanche intermédiaire marque un nouveau paragraphe, les lignes blanches au delà de la première sont ignorées ;

— on peut forcer un retour à la ligne sans marquer de nouveau paragraphe avec `\\`. On peut enchaîner les `\\` pour sauter plusieurs lignes ;

— on peut forcer un retour à la ligne en forçant l'alignement du paragraphe avec `\linebreak` ;

— une espace simple peut être coupée à l'affichage par une fin de ligne. On peut éviter de couper en utilisant `~` qui est le caractère d'espace insécable. Par exemple, si on écrit `les règles suivantes~:`, le deux-points ne sera jamais séparé du mot « suivantes » ;

— on peut forcer une nouvelle page avec `\newpage`.

Dans certains cas (figures, mathématiques, prise de tête sur une suite de commandes particulières), on peut vouloir rajouter des espacements horizontaux ou verticaux à la main. On peut utiliser pour cela `\hspace{1cm}` ou `\vspace{1cm}`. Bien évidemment, la valeur numérique peut être modifiée selon les besoins. On peut également utiliser des valeurs négatives pour « rapprocher » des éléments. On peut rajouter une étoile (`\hspace*(1cm)` par exemple) si on veut que l'espacement rajouté ignore les fins de page (le bloc qui suit peut donc se retrouver dans le pied de page, voire à moitié en dehors de la page).

2.3.3 Commandes particulières

Si votre clavier ne permet pas l'écriture de tous les caractères du français, des commandes peuvent contourner le problème. Par exemple, on peut écrire `\oe` pour obtenir le œ comme dans œil.

En voici une liste non exhaustive (on extrapole facilement aux autres capitales accentuées) :

Commande	Résultat
`\oe`	œ
`\OE`	Œ
`\'E`	É
`\`E`	È
`\^E`	Ê
`\"E`	Ë

2.3.4 Énumération

Pour énumérer des éléments, on peut utiliser les environnements `enumerate` (énumération numérotée) et `itemize` (énumération non numérotée). On indique chaque nouvel élément par un `\item`. On peut choisir ponctuellement de changer le symbole de numérotation. Par exemple :

```
\begin{enumerate}
\item Premier point.
\item Deuxième point.
\item[III.] Troisième point.
\end{enumerate}
```

donnera :

1. Premier point.
2. Deuxième point.

III. Troisième point.

Si on veut gérer plus en précision la numérotation de `enumerate`, on peut :

— reprendre une numérotation interrompue précédemment en commençant par la commande `\begin{enumerate}[resume]` ;

— choisir un numéro courant avec `\setcounter{enumi}{4}` (le prochain numéro sera donc 5).

Par exemple :

```
\begin{enumerate}[resume]
\item Troisième point.
\setcounter{enumi}{5}
\item Sixième point.
\end{enumerate}
```

donnera :

3. Troisième point.
6. Sixième point.

On a dû pour cela utiliser le paquet `enumitem`.

2.4 Format de police

On liste ici différents formats d'écriture.

Commande	Résultat
`\text{Défaut}`	Défaut
`\textit{Italique}`	*Italique*
`\textbf{Gras}`	**Gras**
`\underline{Souligné}`	Souligné
`\texttt{Typewritter}`	`Typewritter`
`\textsf{Sans serif}`	Sans serif

2.5 Références

On peut faire des références automatiques à une question, un théorème, une section, une figure, etc. Pour ce faire, on utilise `\label{etiquette}` pour créer une étiquette et `\ref{etiquette}` pour y faire référence. Par exemple, la section suivante est la section 3.

Un numéro de référence sera calculé automatiquement si c'est possible, mais peut nécessiter une double compilation (première compilation pour calculer le numéro, deuxième compilation pour l'afficher).

3 Écrire des mathématiques

On utilise AMS-LaTeX pour tout ce qui est mathématique. Ces paquets, qui ont été chargés en préambule, contiennent de nombreux symboles et commandes supplémentaires à celles par défaut.

3.1 Environnement maths

Le mode mathématiques permet d'écrire des mathématiques dans une police différente de celle du mode texte. On peut introduire un mode mathématique de différentes manières :

— encadrement par `$ $`. Cela permet d'ajouter des mathématiques sur la ligne en cours d'écriture. Par exemple :

```
Ceci est une ligne qui contient la fonction $f(x) = ax^2 + b$.
```

donnera :

Ceci est une ligne qui contient la fonction $f(x) = ax^2 + b$.

— encadrement par `$$ $$`. Cela écrit une nouvelle ligne ne contenant que des mathématiques, de manière centrée. Par exemple :

```
Voici un exemple avec la fonction $$f(x) = ax^2 + b$$ comme précédemment.
```

donnera :

Voici un exemple avec la fonction

$$f(x) = ax^2 + b$$

comme précédemment.

— environnement `align`. Cela permet d'enchaîner des formules. Par défaut, l'environnement `align` ajoute une numérotation des formules à droite. On peut utiliser l'environnement `align*` pour enlever la numérotation. On doit revenir à la ligne (avec \\) pour passer à une nouvelle formule. Par exemple :

```
\begin{align}
f(x) = ax^2 + b\\
g(x) = c\sqrt{x} - d
\end{align}
```

donnera :

$$f(x) = ax^2 + b \tag{1}$$
$$g(x) = c\sqrt{x} - d \tag{2}$$

3.2 Espaces

En mode mathématique, les espaces sont ignorées. On peut forcer des espaces de différentes tailles par :

`a\,b`	$a\,b$
`a\ b`	$a\ b$
`a\quad b`	$a \quad b$
`a\qquad b`	$a \qquad b$

3.3 Formules et symboles usuels

Certains éditeurs ont des raccourcis graphiques pour écrire des commandes. On en liste ici une partie :

Commande	Résultat
Exposant `x^2` ou `2^{2^n}`	x^2 ou 2^{2^n}
Indice `x_2` ou `(u_n)_{n\in \mathbb{N}}`	x_2 ou $(u_n)_{n\in\mathbb{N}}$
Lettres grecques `\alpha`, `\Gamma`, `\varphi`, `\phi`, `\varepsilon`, `\epsilon`	$\alpha, \Gamma, \varphi, \phi, \varepsilon, \epsilon$
Fonctions usuelles `\cos`, `\sin`, `\log`, ...	cos, sin, log
Racine carrée `\sqrt{x}`	$\sqrt{x}$
Multiplication `\times`	$\times$
Fractions `\frac{1}{n}` (taille par défaut), `\dfrac{1}{n}`, `\tfrac{1}{n}`	$\frac{1}{n}$, $\dfrac{1}{n}$, $\tfrac{1}{n}$
Quantificateurs `\exists`, `\forall`	$\exists$, $\forall$
Infini `\infty`	∞
Ensemble vide `\emptyset` ou `\varnothing`	$\emptyset$ ou $\varnothing$
Relations ensemblistes `\subset`, `\subsetneq`, `\not\subset`, `\in`, `\not\in`	$\subset$, $\subsetneq$, $\not\subset$, $\in$, $\notin$
Opérations ensemblistes `\cup`, `\cap`	$\cup$, $\cap$
Relations `\leqslant`, `\geq`, `\neq`, `\sim`, `\simeq`, `\equiv`, `\prec`	$\leqslant$, $\geq$, $\neq$, $\sim$, $\simeq$, $\equiv$, $\prec$
Flèches `\rightarrow`, `\longleftarrow`, `\Leftrightarrow`, `\mapsto`	$\rightarrow$, $\longleftarrow$, $\Leftrightarrow$, $\mapsto$
Logique `\land`, `\lor`, `\neg`, `\bot`, `\top`, `\vdash`, `\vDash`	$\land$, $\lor$, $\neg$, $\bot$, $\top$, $\vdash$, $\vDash$

Astuce : utiliser `http://detexify.kirelabs.org/classify.html` pour trouver comment écrire un symbole en le traçant à la main.

3.4 Grands opérateurs

On utilise l'opérateur de somme pour illustrer l'utilisation des grands opérateurs. Par défaut, on peut utiliser les indices et exposants pour les bornes, par exemple `\sum_{i=0}^n x_n` pour

$\sum_{i=0}^{n} x_n$. Dans un environnement centré (double dollar ou `align`), le mode *display* est par défaut, donc les indices et exposants sont placés en dessous et au dessus du symbole. On peut, en mode *inline*, soit forcer le mode *display* (avec `\displaystyle`), soit utiliser `\limits` pour avoir le même résultat sans le mode *display*. Par exemple :

`\sum\limits_{i=0}^n x_n = \displaystyle \sum_{i=0}^n x_n` donne $\sum\limits_{i=0}^{n} x_n = \displaystyle\sum_{i=0}^{n} x_n$

Commande	Résultat
Somme, produit `\sum`, `\prod`	$\sum$, $\prod$
Union, intersection `\bigcup`, `\bigcap`	$\bigcup$, $\bigcap$
Opérateurs entourés `\bigoplus`, `\bigotimes`	$\bigoplus$, $\bigotimes$
Intégrale `\int`	$\int$

3.5 Décoration

Outre les exposants et les indices, on peut décorer un morceau de formule mathématique de différentes manières. Les « sous »-commandes ont leur équivalent « sur », en remplaçant `under` par `over`.

Commande	Résultat
Chapeau `\hat{x}`, `\widehat{\sum\limits_{i=0}^nx^n}`	$\hat{x}$, $\widehat{\sum\limits_{i=0}^{n} x^n}$
Tilde `\tilde{x}`, `\widetilde{\sum\limits_{i=0}^nx^n}`	$\tilde{x}$, $\widetilde{\sum\limits_{i=0}^{n} x^n}$
Vecteur `\vec{x}`, `\overrightarrow{AB}`	$\vec{x}$, $\overrightarrow{AB}$
Soulignage `\underline{f(x)}`	$\underline{f(x)}$
Sous-accolade `\underbrace{2x - 3x + x}_{=0}`	$\underbrace{2x - 3x + x}_{=0}$
Superposition `u_n\underset{n\rightarrow +\infty}{\longrightarrow}42`	$u_n \underset{n\rightarrow+\infty}{\longrightarrow} 42$

3.6 Polices particulières

On peut utiliser `\mathcal{A}`, `\mathbb{N}` et `\mathfrak{S}` pour avoir les lettres capitales calligraphiées, doublées ou gothiques : $\mathcal{A}$, $\mathbb{N}$ et $\mathfrak{S}$.

3.7 Parenthésage

Outre les parenthèses `(`, `)`, on peut délimiter les expressions de différentes manières : `\[`, `\]` (crochets), `\{`, `\}` (accolades), `\lVert`, `\rVert` (norme : $\|x\|$), `\langle`, `rangle` (produit scalaire : $\langle x, y \rangle$).

Pour une expression de grande taille, on peut rajouter `\left` et `\right` avant les délimiteurs. Attention, il faut dans cas toujours fermer (même de manière invisible) une parenthèse ouverte, par exemple avec `\right.` (le point étant invisible).

Par exemple, `\left\{\left(\dfrac{1}{n}\right)\times n=1\right.` donnera la formule : $\left\{\left(\dfrac{1}{n}\right)\times n=1\right.$

4 Autres environnements

4.1 Tableaux

Il existe deux environnements principaux de tableaux : `tabular` (texte normal) et `array` (mode maths). J'en utilise un troisième (`tblr`, du paquet `tabularray`), dont les options sont généralement compatible avec les précédents, mais permet plus de choses.

Le schéma d'un tableau est le suivant :

```
\begin{tabular}{|cl||r|}
\hline
Colonne 1 & Colonne 2 & Nom de colonne 3\\
\hline
$x_1$ & Valeur du vecteur & non nulle\\
\cline{2-3}
\multicolumn{2}{|c||}{FU-SI-ON AH !} & \\
\hline
\multirow{2}{*}{Fusion (again)} & $x_2$ & $v_2$\\
\cline{2-3}
& Il faut & une nouvelle ligne\\
\hline
\end{tabular}
```

affichera :

Colonne 1	Colonne 2	Nom de colonne 3
x_1	Valeur du vecteur	non nulle
FU-SI-ON AH !		
Fusion (again)	x_2	v_2
	Il faut	une nouvelle ligne

On distingue :

— les options `{|cl||r}` permettent de choisir les types de colonnes (ici centrée, alignée à gauche, alignée à droite), et leur séparation par des lignes verticales ;

— la commande `hline` permet de tracer une ligne horizontale sur toute la largeur du tableau ; `cline` permet de tracer une ligne partielle en indiquant les colonnes de début et de fin ;

— l'esperluette `&` permet de séparer les colonnes ;

— le retour à la ligne `\\` permet de séparer les lignes ;

— `\multicolumn` permet de fusionner des cases consécutives sur une même ligne : on indique en premier argument le nombre de colonnes, en deuxième argument les séparateurs et le type de la colonne résultante, et en troisième argument le contenu de la case ;

— `\multirow` (du paquet éponyme) permet de fusionner des cases consécutives sur une même colonne : on indique en premier argument le nombre de lignes, en deuxième argument la largeur (le `*` garde la valeur par défaut), et en troisième argument le contenu. Il faut penser à laisser un champ vide pour les lignes qui suivent.

Dans un mode maths, les matrices sont des tableaux particuliers. On n'a pas à préciser les colonnes, qui sont déterminées automatiquement. Par exemple :

```
$\begin{pmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{pmatrix}$
```

affichera $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$.

On distingue différents types de matrices : `matrix` (sans délimiteur), `pmatrix` (parenthèses), `bmatrix` (crochets), `vmatrix` (module/déterminant).

4.2 Programme informatique

Même si ce n'est pas mon choix usuel, la manière la plus simple d'intégrer un programme informatique est avec le paquet `listings`.

Dans le préambule, j'ai chargé le paquet et défini les couleurs de la coloration syntaxique, avec :

```
\usepackage{listings}
```

```
\usepackage{xcolor}

\lstset{basicstyle=\ttfamily,
        keywordstyle=\color{teal},
        commentstyle=\color{gray},
        stringstyle=\color{violet},
        identifierstyle=\color{blue}}
```

Dès lors, pour créer un environnement de code, on peut par exemple écrire :

```
\begin{lstlisting}[language=c, frame=single, numbers=left]
int main(void){
    int x = 5; // Je pose x qui vaut 5.
    printf("La valeur de x est %d\n", x);
    return EXIT_SUCCESS;
}
\end{lstlisting}
```

ce qui donne :

```
1 int main(void){
2     int x = 5; // Je pose x qui vaut 5.
3     printf("La␣valeur␣de␣x␣est␣%d\n", x);
4     return EXIT_SUCCESS;
5 }
```

On peut choisir d'enlever des options, par exemple enlever le cadre ou les numéros.

4.3 Figures

On inclut une figure avec l'environnement `figure`. On peut importer une image (qui se trouve dans le même répertoire) avec `includegraphics` (avec le paquet `graphicx`).

Par exemple, le code :

```
\begin{figure}[!h]
\centering
\includegraphics[scale=0.2]{martin_pecheur.png}
\caption{Ceci est un martin pêcheur.}
\label{martin_pecheur}
\end{figure}
```

donnera :

FIGURE 1 – Ceci est un martin pêcheur.

On trouve différentes commandes :

— l'option `!h` permet de placer l'image « ici » (*here*). Sans cette option, l'image sera placée par défaut en haut d'une page (l'actuelle ou la suivante, selon là ou il y a de la place), ce qui peut couper un morceau de texte ou d'environnement ;

— l'option `\centering` permet de centrer la figure ;

— `\caption` permet de donner un titre à la figure ;

— `\label` permet d'ajouter une étiquette pour une référence future (ou passée). Il faut mettre l'étiquette après le titre, pour faire référence au numéro de figure.

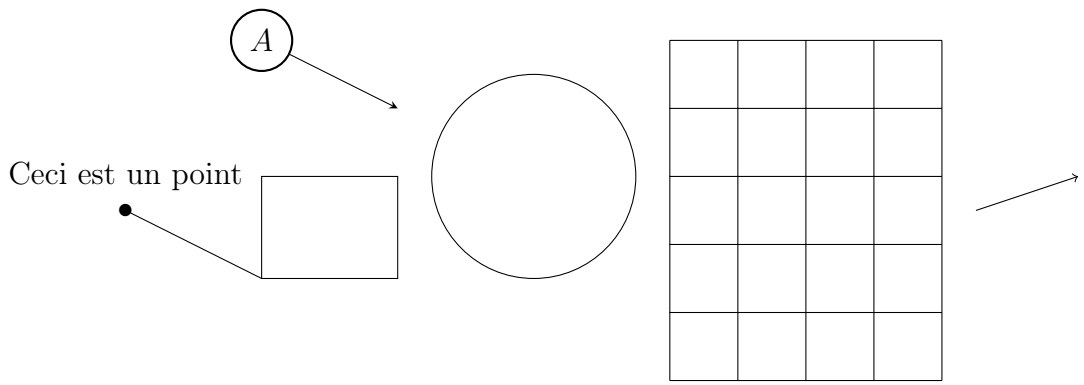
4.4 Tracés graphiques

On présente ici une très brève introduction au tracé graphique, via le paquet `tikz`. Très brève, puisque le manuel de Tikz fait 1400 pages, et un manuel d'introduction fait plus de 300 pages. Impossible, donc, de tout présenter en un temps raisonnable. Le plus simple est de procéder par mimétisme en partant d'un graphique existant.

```
\begin{tikzpicture}[scale=.9] % On peut changer l'échelle de l'image
\draw (1, 2.5) node {$\bullet$}; % Création d'un point
\draw (1, 3) node {Ceci est un point}; % On peut écrire du texte

% On peut tracer des traits
\draw (1, 2.5) -- (3, 1.5);
% Des rectangles (on indique les coins)
\draw (3, 1.5) rectangle (5, 3);
% Des cercles (coordonnées du centre avant, rayon après)
\draw (7, 3) circle (1.5);
% Des grilles (on indique les coins)
\draw (9, 0) grid (13, 5);
% Des flèches
\draw[->] (13.5, 2.5) -- (15, 3);

% On peut nommer les points
\node[draw, circle, thick] (A) at (3, 5) {$A$};
% On peut utiliser le nom d'un point au lieu des coordonnées
\draw[->, >=stealth] (A) to (5, 4);
\end{tikzpicture}
```

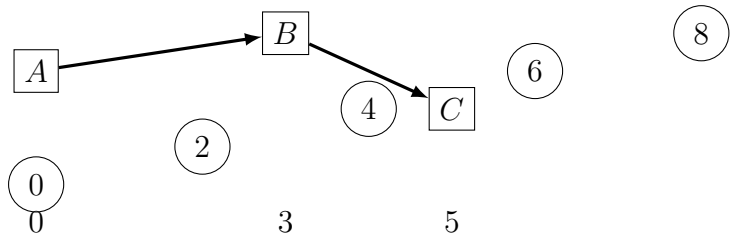



```
% On peut changer les échelles des coordonnées indépendamment
\begin{tikzpicture}[xscale = 1.1, yscale = .5]
% Écriture de boucle
\foreach \x in {0, 3, 5}{
    \draw (\x, 0) node {$\x$};
}

% Le ... permet de compléter automatiquement
\foreach \x in {0, 2, ..., 8}{
    \node[draw, circle] (\x) at (\x, 0.5 * \x + 1) {$\x$};
}
```

```
% On peut itérer sur plusieurs variables (mais dans ce cas, on ne
% peut pas utiliser ...)
\foreach \x/\y/\nom in {0/4/A, 3/5/B, 5/3/C}{
    \node[draw, rectangle] (\nom) at (\x, \y) {$\nom$};
}

\foreach \source/\dest in {A/B, B/C}{
    \draw[->, >=latex, very thick] (\source) to (\dest);
}
\end{tikzpicture}
```

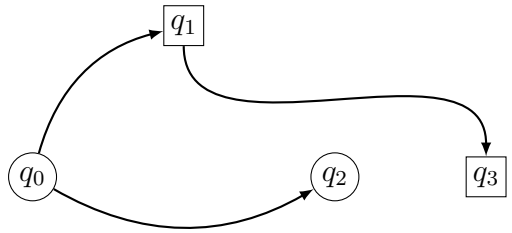



```
% Adapté pour faire des graphes
\begin{tikzpicture}
% On peut créer un type d'objet particulier
\tikzstyle{fleche}=[->,>=latex,thick]
\tikzstyle{sommet}=[draw,circle,minimum size=15pt,inner sep=2pt]
\tikzstyle{carré}=[sommet, rectangle]

% Dès lors, on peut créer les états d'un automate/graphe
\foreach \x/\y/\num/\type in {0/0/0/sommet, 2/2/1/carré,
                              4/0/2/sommet, 6/0/3/carré}{
    \node[\type] (\num) at (\x, \y) {$q_{\num}$};
}

% On peut faire des arêtes courbées
\draw[fleche] (0) to[bend left] (1);
\draw[fleche] (0) to[bend right] (2);

% Voire choisir les angles de départ et d'arrivée
\draw[fleche] (1) to[out = -90, in = 90] (3);
\end{tikzpicture}
```




```
% Et pour les automates
\begin{tikzpicture}[->, >=latex, node distance=2.5cm,
                    initial text=, initial distance=0.5cm, bend angle=35]
\tikzstyle{etat}=[draw,circle,minimum size=15pt,inner sep=2pt]
% initial est un mot clé de la bibliothèque automata
\tikzstyle{init}=[etat, initial]
\tikzstyle{final}=[etat, double, double distance=2pt]
\tikzstyle{finit}=[init, final]

\node[init] (0) {$q_0$};
\node[finit] (1) [above right of=0] {$q_1$};
\node[etat] (2) [below right of=1] {$q_2$};
\node[final] (3) [right of=2] {$q_3$};

\path
    (0) edge[bend right] node[below] {$a$} (2)
    (0) edge node[near start, left] {$b$} (1)
    (1) edge[out=120, in=60, loop] node[above] {$a,b$} (1)
    (2) edge[bend left] node[above left] {$a$} (3);
\end{tikzpicture}
```