LaTeX

Introduction

Comment (bien) rédiger un document ?

PREMIER CHAPITRE

Lorsque j'avais six ans j'ai vu, une fois, une magnifique image, dans un livre sur la Forêt Vierge qui s'appelait « Histoires Vécues ». Ça représentait un serpent boa qui avalait un fauve. Voilà la copie du dessin.

On disait dans le livre : « Les serpents boas avalent leur proie tout entière, sans la mâcher. Ensuite ils ne peuvent plus bouger et ils dorment pendant les six mois de leur digestion. »

J'ai alors beaucoup réfléchi sur les aventures de la jungle et, à mon tour, j'ai réussi, avec un crayon de couleur, à tracer mon premier dessin. Mon dessin numéro 1. Il était comme ça :

J'ai montré mon chef-d'œuvre aux grandes personnes et je leur ai demandé si mon dessin leur faisait peur.

– 5 –

Figure 1: Document d'origine

Comment (bien) rédiger un document ?

PREMIER CHAPITRE

Lorsque j'avais six ans j'ai vu, une fois, une magnifique image, dans un livre sur la Forêt Vierge qui s'appelait « Histoires Vécues ». Ça représentait un serpent boa qui avalait un fauve. Voilà la copie du dessin.

On disait dans le livre : « Les serpents boas avalent leur proie tout entière, sans la mâcher. Ensuite ils ne peuvent plus bouger et ils dorment pendant les six mois de leur digestion. »

J'ai alors beaucoup réfléchi sur les aventures de la jungle et, à mon tour, j'ai réussi, avec un crayon de couleur, à tracer mon premier dessin. Mon dessin numéro 1. Il était comme ça :

J'ai montré mon chef-d'œuvre aux grandes personnes et je leur ai demandé si mon dessin leur faisait peur.

– 5 –

Figure 1: Document d'origine

PREMIER CHAPITRE

Lorsque j'avais six ans, j'ai vu, une fois, une magnifique image, dans un livre sur la Forêt Vierge qui s'appelait « Histoire Vécues ». Ca représentait un serpent boa qui avalait un fauve. Voilà la copie du dessin.

On disait dans le livre : « Les serpents boa avalent leur proie tout entière, sans la mâcher. Ensuite, ils ne peuvent plus bouger et ils dorment pendant les six mois de leur digestion. »
J'ai alors beaucoup réfléchi sur les aventures de la jungle et, à mon tour, j'ai réussi, avec un crayon de couleur, à tracer mon premier dessin. Mon dessin numéro 1. Il était comme ça :

J'ai montré mon chef-d'oeuvre aux grandes personnes et je leur ai

demandé si mon dessin leur faisait peur.

Figure 2: Tentative de reproduction

- Premier essai : en rédigeant simplement, la mise en page n'est pas au rendez-vous (il faut déplacer le texte à droite de la figure manuellement)

Comment (bien) rédiger un document ?

- Et si on voulait changer de style (par exemple en double colonne)
- Et si on voulait ajouter un chapitre entre deux (et donc mettre à jour tous les numéros de chapitre qui suivent, les références, la table des matière, ...)
- Et si on voulait ajouter de la bibliographie

Comment (bien) rédiger un document ?

- Et si on voulait changer de style (par exemple en double colonne)
- Et si on voulait ajouter un chapitre entre deux (et donc mettre à jour tous les numéros de chapitre qui suivent, les références, la table des matière, ...)
- Et si on voulait ajouter de la bibliographie
- Utilisation de style (disponible sous Word par exemple)
- Utilisation d'ouils grâce auxquels on ne se soucie pas de la mise en page (LATEX)

Historique

- TEX (Donald E. Knuth) : programme pour des documents (scientifiques)
- TEX grande stabilité, portabilité, absence quasi totale de bogues, respect des usages des typographes et composeurs de textes
- TEX se prononce "tech" (comme dans tech-nologie), X $\rightarrow \chi$ (chi)
- LATEX (Leslie Lamport) sur-couche basée sur TEX comme système de mise en page avec de nombreux modules

Introduction

LaTeX vs. WYSIWYG

- Publication d'un manuscrit :
  auteur → éditeur (police, taille des colonnes, ...) → technicien typographe
- Dans LaTeX, LaTeX : éditeur et TeX : le typographe
- L'auteur → structure logique du document (section, sous-section, etc)
  LaTeX → calcul la mise en page optimale avec les bonnes règles de typographie
- Information de structure insérée dans le texte avec des **commandes** : distinction entre le fond du document et sa forme
- Démarche opposée aux traitements de texte WYSIWIG (What You See Is What You Get) : mélange de structure du document et de mise en forme. L'auteur voit à l'écran le résultat final du document
- LaTeX → pas de résultat final immédiat, nécessite une phase de compilation
  compilateur : programme qui permet de transformer un langage vers un autre langage
  'code source latex' → LaTeX→ (pdf, html...)

Caractéristiques de LaTeX

Arguments en faveur de LaTeX par rapport aux logiciels WISIWYG

- mise en page professionnelle / respect des règles typographiques
- facilité de la composition des formules mathématiques
- on ne se préoccupe pas de la mise en page, on peut donc se concentrer sur le fond
- gestion automatique des structures complexes telles que : tables des matières, notes de bas de pages, liste des figures, références croisées, bibliographies, index, et bien d'autres
- possibilité de programmer/très extensible
- multi-plateforme / libre
- extensions conventionnelles : `.tex` (LaTeX), `.bib` (fichier de bibliographie), `.aux` (fichiers auxiliaires), `.sty` (fichiers de style), etc
- Possible gestion des versions avec `git` par exemple

Remarque

Création de bibliographie : voir le système `bibtex` (fichier `.bib`)

Structure d'un fichier source

LaTeX : Langage balisé à base de commande

- **Squelette d'un fichier**

```
\documentclass[options]{parameters}
\usepackage[package-options]{package-name} %optionnel
% >>preambule<<

\begin{document}
   le document commence ici
\end{document}
```

- `\documentclass[options]{paramaters}`
  `parameters` : article, report, book, etc ; `options` : 10pt, twocolumn, etc
- `\usepackage[package-options]{package-name}`
  `package-name` : graphicx, amsmath, etc ; `package-options` (personnalisation)
- ligne commençant par % = commentaires
- **Gros documents, plusieurs fichiers**
  `\input{filename}` ou `\include{filename}` (idem avec saut de page)

Compilation / visualisation

LaTeX : Langage compilé qui nécessite un compilateur (programme informatique)
Code source LaTeX→ compilateur (`pdflatex`) → fichier visualisable, imprimable

- **Compilation standard avec le programme** : `pdflatex`
  - comprend les commandes et instructions du fichier source
  - phase de compilation crée un ensemble de fichiers : `.pdf` (mise en forme), `.aux` (numéro de pages, figures, références, etc), `.log` (messages d'informations ou d'erreur), `.toc` (table des matières), ...
    Remarque : Pensez à compiler souvent (n'attendez pas d'avoir écrit 1000 lignes)
  - visualisation : `evince` (sous linux)
- Edition partagée sur `http://www.sharelatex.com`
  Apprendre à compiler et faire du latex proprement AVANT ! ! !
- Solution alternative à préférer :
  - `emacs + synctex` pour la compilation à la volée
  - `git` pour l'édition partagée
  - `Visual Studio Code` avec extension LaTeX `Workshop`

Exercices

1. Récupérer le fichier `https://ramet.gitlab.io/aop3-doc/latex/first-doc.tex` (avec `wget`)
2. Ajouter le à votre dépôt git et commiter la version d'origine
3. Compiler le fichier, il y a des erreurs de compilation, interpréter, corriger les erreurs jusqu'à compilation **totale**
   Remarque : Afin de quitter une compilation échouée, taper sur la touche x
4. Vérifier dans le répertoire courant la présence des fichiers auxiliaires. De quels types sont-ils ?
5. Visualiser le document produit avec `evince`. Modifier le fichier source et vérifier la modification dans le document final
6. Regarder le status de votre dépôt git. Faut il tout ajouter dans le dépôt ?
7. Créer un fichier `.gitignore` à la racine de votre dépôt et ajouter la ligne `*.aux`. Qu'est ce qui change dans le status de votre dépôt ?
8. Compléter le fichier pour ignorer tous les fichiers générés. ne JAMAIS ajouter des fichiers temporaires ou générés dans un dépôt git

Exercices

1.
   - Effacer tous les fichiers sauf le fichier source
   - Modifier le fichier source
   - Compiler avec `pdflatex`
   - Vérifier la génération des fichiers auxiliaires et du `pdf`
   - Visualiser le `pdf`
   - Modifier le fichier source
   - Recompiler et visualiser la modification
2.
   - Créer le fichier `first-input.tex` contenant la chaîne `An $input`
   - Créer le fichier `first-include.tex` contenant la chaîne An _include
   - En utilisant les bonnes commandes LaTeX inclure les deux nouveaux fichiers
   - Compiler, corriger, visualiser
3. Ajouter et commiter les nouveaux fichiers sources à votre dépôt

Environnements

Permet de composer du texte dans un contexte particulier
Syntaxe générale : `\begin{env} ...contenu... \end{env}`

- Listes : `enumerate, itemize`, utilisation avec `\item`
- Justification : `flushleft, flushright`
- Impression verbatim : `verbatim`
- Tableau : `\begin{tabular}{description}`

a	b	c	d
e	fg	h	i

$\Rightarrow$

```
\begin{tabular}{r|c l c}
 a & b & c & d\\
 e & fg & h & i\\\hline
\end{tabular}
```

- droite, gauche, centré, trait vertical `r, l, c, |`
- séparateur de colonne, saut de ligne : `&`, `\\`
- trait horizontal, trait horizontal de la colonne `i` à `j` : `\hline`, `\cline{i-j}`
- fusion de colonnes : `\multicolumn{n}{description}{texte}`

Objets flottants

Objets ne pouvant être coupés par un saut de page (figure, tableau, algorithme, etc)
LaTeX place automatiquement les objets flottants, là où il y a de la place dans la page

- **Squelette standard**

```
\begin{...} %figure ou table
  \centering
  % commandes pour une figure, le tableau ou autre chose
  \caption[court]{legende}
  \label{unlabel} % \label se trouve apres \caption
\end{...} % figure ou table
```

- Par ex : figure, tableau
  - `\begin{figure}[pref-placement]`
  - `\begin{table}[pref-placement]`
- Ordre de préférence de placement : `h, t, b, p` (here, top, bottom, page)

Inclusion de figures

- dans le préambule : `\usepackage{graphicx}`, attention au `x`, `graphics` existe aussi !
- commande : `\includegraphics[options]{filename}`
  - `options` : permet de régler la taille de la figure
  - préférer les tailles relatives, préservation de l'aspect/ratio
  - jouer sur la largeur par rapport à la largeur du texte :

```
\includegraphics[width=0.5\textwidth]{filename}
```

  - `filename` : sans l'extension
- si les images sont dans un répertoire à part : `\graphicspath{{./img/}}` dans le préambule
- préférer les images vectorielles (pdf ou svg) au bitmap (.bmp, .jpeg, .png, etc)

Inclusion des flottants : ce qu'il ne faut pas faire

NE PAS faire

PREMIER CHAPITRE

Lorsque j'avais six ans j'ai vu, une fois, une magnifique image, dans un livre sur la Forêt Vierge qui s'appelait « Histoires Vécues ». Ça représentait un serpent boa qui avalait un fauve. Voilà la copie du dessin.

On disait dans le livre : « Les serpents boas avalent leur proie tout entière, sans la mâcher. Ensuite ils ne peuvent plus bouger et ils dorment pendant les six mois de leur digestion. »

J'ai alors beaucoup réfléchi sur les aventures de la jungle et, à mon tour, j'ai réussi, avec un crayon de couleur, à tracer mon premier dessin. Mon dessin numéro 1. Il était comme ça :

J'ai montré mon chef-d'œuvre aux grandes personnes et je leur ai demandé si mon dessin leur faisait peur.

– 5 –

- NE JAMAIS mettre une figure sans titre, sans référence dans le texte
- Vous écrivez des rapports et non des livres illustrés

Inclusion des flottants : ce qu'il faut faire

Faire

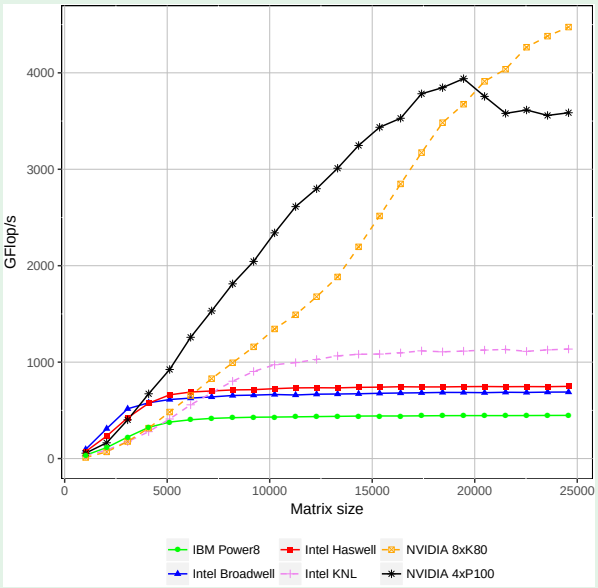


Figure 3: Etude de la portabilité des performances de l'application X sur l'ensemble des architectures testées.

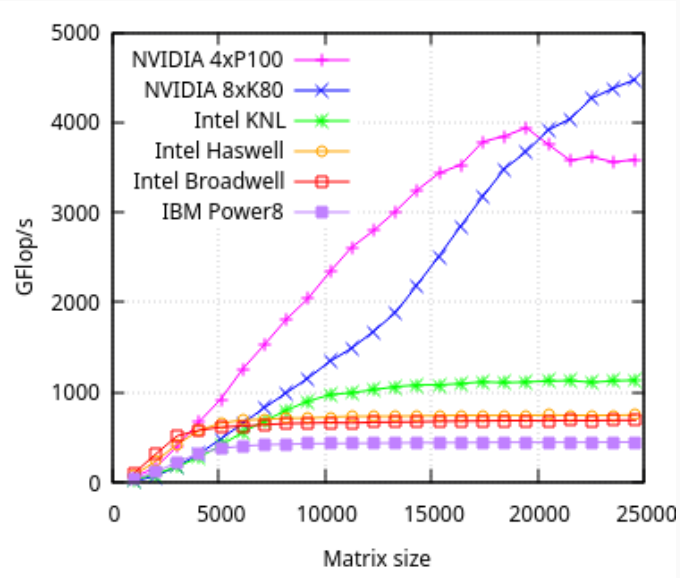
Le figure 3 montre que l'application X permet d'exploiter un large spectre d'architectures différentes tout en obtenant des performances proches du maximum théorique de chacun d'entre elles.

- Important : un flottant (une figure, un tableau...), doit **toujours**
  - comporter une légende explicative
  - être cité, référencé dans le texte à l'aide de `\label{fig:xxx}`, `\ref{fix:xxx}` et éventuellement `\pageref{fig:xxx}`
- Le flottant fait partie de la phrase, on ne le cite pas entre parenthèse.

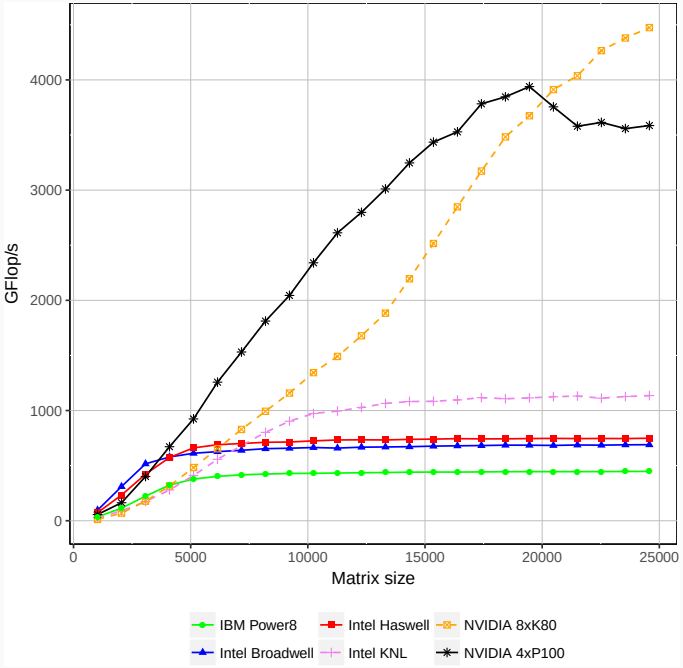
Inclusion des figures

- **Prise en charge des figures/images**
  - `pdflatex` : `pdf`, `png`, `jpeg`
  - Privilégier les figures/images vectorielles par rapport aux images bitmap

Une figure bitmap pixellisée



Une figure vectorielle propre et non pixellisée



Quelques commandes

- Structure logique d'un document (toutes ont une version avec `*`)
  - `\section{title}`
  - `\subsection{title}`
  - `\subsubsection{title}`
- Un paragraphe : unité typographique commençant par un alinéa (**obtenu par une ligne vide dans le code source**) = pensée cohérente ou développement d'une idée. Lorsque la réflexion se poursuit, un saut de ligne \\ suffit (pas d'alinéa)
- Références croisées : `\label{nom}, \ref{nom} \pageref{nom} \eqref{nom}`
- Listes automatiques : `\tableofcontents, \listoffigures, \listoftables`

Quelques commandes

- Sauts de ligne, de page et césure : `\\`, `\newline`, `\newpage`, `cé\-su\-re`
- Chaînes toute prêtes : `\today`, `\date`, `\TeX`, `\LaTeX`
- Caractères spéciaux : `\c{c}`, `\oe`, `\dots`, `\slash`, `\^o`, `\"i`, ...
- Espaces : LaTeX calcul l'espacement optimal que les mots doivent avoir.
  - espace insécable, espacement : `~`,`\␣`, `\hspace{m}`, `\vspace{m}`, `\stretch{m}`, `\hfill`, `\dotfill`
  - alinéa, ligne : `\indent`, `\noindent`, `\rule{hm}{wm}`
  - mesure automatique : `\textwidth`, `\columnwidth`, `\textheight`
- Polices : `\emph{}`, `\textbf{}`, `\texttt{}`, `\textsc{}`
- Tailles : `\tiny`, `\scriptsize`, `\small`, `\normalsize`, `\large`, `\huge`
- Utilisation de blocs, par ex : `{\small petit} normal`

Exercices

1.
   - Grâce à la commande `wget`, récupérer le fichier suivant :
     `https://ramet.gitlab.io/aop3-doc/latex/test-latex-skel.tex`
     le mettre dans un répertoire `~/latex/`
   - Récupérer le fichier suivant :
     `https://ramet.gitlab.io/aop3-doc/latex/iut-logo.jpg`
     et le mettre dans un répertoire `~/latex/img`
   - Essayer de reproduire ce document au plus près de sa mise en forme :
     `https://ramet.gitlab.io/aop3-doc/latex/test-latex-cor.pdf`
2. En vous inspirant des thèmes existants, rédiger votre Cv en LaTeX (à faire à la maison)

AMS-LaTeX

- AMS-LaTeX : collection d'extensions et de classes pour saisir des formules mathématiques
- développée par l'*American Mathematical Society*
- `\usepackage{amsmath, amssymb}`
- équation dans le texte `$...$`, par ex. $a+b=c$: `$a+b=c$`
- hors texte : environnement (flottant) `\begin{equation}...\end{equation}`, par ex :

```
\begin{equation}
  a+b=c
 \label{eq:addition}
\end{equation}
```

$\Rightarrow$

$$a+b=c \qquad (1)$$

- la version étoilée enlève la numérotation
- référencer une équation
  1. déposer une étiquette (label) `\label{eq:addition}`
  2. la référencer dans le texte `\eqref{eq:addition}`
  3. `\begin{equation} \label{eq:addition}a+b=c\end{equation}`
  4. `l'équation \eqref{eq:addition} est une addition`
     → *l'équation* (1) *est une addition*
- idem, pour les figures, les tableaux, etc

Commandes en vrac

- `^ _ ^{} _{}` : exposant, indice
- `\sum, \int, \lim, \prod` : somme, intégrale, limite, produit
- `\frac{}{}, \tfrac{}{}` : fraction, fraction en ligne
- les symboles grecs, syntaxe du nom, e.g., `\pi, \epsilon, \Gamma` (`http://detexify.kirelabs.org`)
- opérateurs, syntaxe du nom en anglais, e.g, `\forall, \in, \notin, etc`
- `\, \; \quad \qquad` gestion des espaces
- `\sqrt{}, \sqrt[]{}, \surd{}`, racine carrée, racine n^{ieme}, le symbole racine
- `\cdots, \ldots, \vdots, \ddots`, différents points de suspensions
- `\underline, \overline, \underbrace, \overbrace` soulignés, accolades
- `\text{...}` texte (droit) en mode math
- variables en italique, les fonctions en police droite, e.g., `\sin, \cos, etc`.
- `\partial, \| \| , | |` : dérivée partielle, norme, valeur absolue
- `\newtheorem{com}{text}` théorèmes, définition, lemmes, etc (préambule). `com` est le nom en LaTeX et `text` : le texte affiché

Équations trop longues et matrices

```
\begin{align}
  x-1 &= y \nonumber\\
  x &= y+1
\end{align}
```

$$\Rightarrow \quad \begin{aligned} x - 1 &= y \\ x &= y + 1 \end{aligned} \qquad (2)$$

```
x =
$\begin{cases}
   y/2  & y \text{ est pair} \\
   0 & \text{ sinon}
\end{cases}
$
```

$$\Rightarrow \quad x = \begin{cases} y/2 & y \text{ est pair} \\ 0 & \text{ sinon} \end{cases}$$

```
$
\begin{matrix}
 1 & 2 & 3\\
 a & b & c
\end{matrix}
$
```

$$\Rightarrow \quad \begin{matrix} 1 & 2 & 3 \\ a & b & c \end{matrix}$$

Algorithmes : package `algorithm2e`

- environnement flottant (possibilité d'obtenir une liste)
- `\usepackage[french,vlined,lined,linesnumbered,boxed]{algorithm2e}` (`french`) : mots clés en français
- fonctions de base, fonctions avancées voir la documentation

Un algorithme possède

- des données, paramètres `\Donnees{...}`
- des données en entrée `\Entree{...}`
- un résultat `\Res{...}`
- pour formater une boucle `for` : `\PourTous{criteredarret}{corps de la boucle}`
- pour formater une boucle `while` : `\Tq{criteredarret}{corps de la boucle}`

Comme pour `figure` ou `table`, vous **devez** ajouter un `\caption[]{}` et un `\label{alg:...}` et y faire référence dans le texte `l'algorithme~\ref{alg:...} montre...`

Si vous voulez présenter du code et non un algorithme, privilégiez le package `listings`

Algorithmes : package `algorithm2e`

```
\begin{algorithm}[H]
\Donnees{a}
\Res{b}
\eIf{a == 0}{
  b = 0;\
}{
  b = 1;\
}
\caption{Mon premier
    algorithme}
\end{algorithm}
```

$\Rightarrow$

Données : a
Résultat : b
1 **if** $a == 0$ **then**
2 $\quad b = 0;$
3 **else**
4 $\quad b = 1;$

Algorithme 1 : Mon premier algorithme

Exercices

1. Récupérer le fichier suivant :
   `https://ramet.gitlab.io/aop3-doc/latex/test-math.tex`
   et le mettre dans un répertoire du type `~/latex/math/`
2. Essayer de reproduire ce document au plus près de sa mise en forme :
   `https://ramet.gitlab.io/aop3-doc/latex/test-math-cor.pdf`

LaTeX

TikZ / Beamer

Beamer

- une classe de document au même titre que article, report ou book
- permet de concevoir des présentations en LaTeX
- mise en place par Till Tantau (PGF/Ti*k*Z)

Caractéristiques

- accepte les commandes LaTeX, pratique pour la structuration, les équations, etc
- permet de créer de « animations » sobres
- beaucoup de thèmes adaptés pour les présentations
- les thèmes sont conçus pour être très lisibles par l'audience
- les thèmes sont hautement configurables
- le format standard des présentations `Beamer` est le `pdf`

Ce qui suit donne la structure minimale pour une présentation avec Beamer
Beaucoup d'autres options, existent, allez voir la documentation en fonction des besoins

Document Beamer

La manière la plus efficace et la plus rapide pour faire une présentation Beamer est d'utiliser un thème prédéfini

Un fichier source beamer se structure comme un document LaTeX :

```
\documentclass[options]{beamer}
\usetheme{Madrid} % par exemple
\usepackage{...} % tous les packages utiles comme en LaTeX
\begin{document}
\end{document}
```

Frames

L'objet de base est la frame (slide)

```
\begin{frame}
 \frametitle{titre du slide/ou sans titre}
 %contenu de la slide en LaTeX
\end{frame }
```

Frames

Frame particulière : la page de titre. À mettre dans le préambule

```
\title[court]{long}
\subtitle[court]{long}
\author[court]{long}
\date[court]{long}
\institute[court]{long}
```

Commande `\titlepage` dans un environnement `frame` pour générer la page de titre

Blocs et listes

Déclarer des blocs dans Beamer, la syntaxe est

```
\begin{block}{blocktitle}
contenu du bloc
\end{block}
```

$\Rightarrow$

blocktitle

contenu du bloc

Les listes se font comme en LaTeX(`itemize, enumerate, etc`). Possibilité de faire apparaître les items au fur et à mesure, par ex.

```
\begin{itemize}
  \item<1-> d'abord lui
  \item<2-> ensuite lui
\end{itemize}
```

$\Rightarrow$

- d'abord lui

Blocs et listes

Déclarer des blocs dans Beamer, la syntaxe est

```
\begin{block}{blocktitle}
contenu du bloc
\end{block}
```

⇒

blocktitle

contenu du bloc

Les listes se font comme en LaTeX(`itemize, enumerate, etc`). Possibilité de faire apparaître les items au fur et à mesure, par ex.

```
\begin{itemize}
  \item<1-> d'abord lui
  \item<2-> ensuite lui
\end{itemize}
```

⇒

- d'abord lui
- ensuite lui

Présentation orale

Quelques règles concernant les présentations :

- une présentation comporte toujours : une page de titre, un plan, une introduction, le pourquoi de la présentation, une conclusion
- chaque slide/diapo ne doit pas être trop chargée (ne pas prendre exemple sur ces slides ! ! !), pour que le public puisse lire rapidement l'idée contenue dans la slide et ensuite écouter
- une présentation doit être structurée et claire afin que l'audience puissent facilement suivre le propos
- éviter de faire des animations de transitions
- privilégier la sobriété et la clarté devant les artifices décoratifs qui masquent le propos
- en terme de longueur : compter un discours de 1 à 2 min par slide
- privilégier un schéma, un diagramme à de longues phrases de discours

Exercices

1. Récupérer la présentation `Beamer` à l'adresse `https://ramet.gitlab.io/aop3-doc/latex/test-beamer-cor.pdf` et son squelette : `https://ramet.gitlab.io/aop3-doc/latex/test-beamer-skel.tex`
2. Mettre dans une arborescence du type `~/latex/beamer`
3. Essayer à partir des éléments vu en cours de reproduire la même présentation
4. Tester différents thèmes

Les figures avec PGF/TikZ

- Approche *orientée structure* (à la LaTeX), i.e., à base de description textuelle de la figure à mettre en oeuvre
- Tout système LaTeX est capable de fabriquer du contenu vectoriel mais tous ne le font pas de la même manière
- Pas la même interface (`dvips`, `pdflatex`, etc)
- Till Tantau (chercheur à l'institut d'informatique théorique, Allemagne)
- Unification / abstraction des différentes interfaces $\rightarrow$ PGF (*Portable Graphic Format*)
- PGF est accompagné du module Ti*k*Z
- Ti*k*Z$\rightarrow$ langage de programmation de haut niveau
- permet de décrire et de réaliser des graphiques de très haute qualité directement dans le code source LaTeX
- PGF/Ti*k*Z est accompagné d'une documentation : `pgfmanual.pdf` (1321 pages à la version 3.1.10)
- Nous n'aborderons que la base ...
- Multitude d'exemples sur `http://www.texample.net/tikz`

Autre module orienté texte : `mermaid`
Alternative Wysiwyg : logiciels de dessin `inkscape`

Le minimum à connaître

- Pour charger Ti*k*Z : `\usepackage{tikz}` dans le préambule.
- Ti*k*Z permet de créer des graphiques *en ligne*, *dans le texte*.
- Pour commencer un graphique, utiliser l'environnement `\begin{tikzpicture}...\end{tikzpicture}` ;
- Chaque instruction de dessin doit se terminer par `;`
- Si le graphique ne comporte qu'une seule instruction vous pouvez utiliser uniquement `\tikz`

Exemples d'utilisation de TikZ

```
\tikz \draw(0,0) -- (1,2);
```

⇒

```
\begin{tikzpicture}[nodes={
    draw,circle},->]
 \node{racine}
     child{ node{a}}
     child{ node{b}};
\end{tikzpicture}
```

⇒