CHAPITRE 1

Principes de base de la rédaction à l'aide de LaTeX

1.1 Valeurs ajoutées de LaTeX

LaTeX se charge de structurer les textes rédigés et des éléments dits *flottants* (image, dessins...), qu'on désignera comme étant le *fond* du document. Tous les processus permettant de modifier le style du texte (gras, italique, police...), sa disposition (doubles colonnes, alignement à droite...) ou encore les informations récurrentes sont exécutés sous forme d'instructions incluses dans le langage de programmation TeX.

La première valeur ajoutée de LaTeX est sa capacité à gérer la *forme* du document selon les normes en vigueur et sa classe, et de pouvoir le laisser gérer cet aspect-là, souvent chronophage, durant la rédaction.

Une autre valeur ajoutée de LaTeX est d'automatiser certains processus de rédaction par la création de nouvelles instructions (ou l'utilisation d'instructions existantes). Le package AOCDTF comporte ainsi une multitude de *macro-commandes* aux buts aussi variés qu'utiles, je vous conseille d'aller consulter le code du package AOCDTF pour les décrypter. Toutes les nouvelles macro-commandes (ainsi que les instructions existantes nécessaires à la bonne rédaction des documents) seront détaillées dans ce recueil.
Sachant que chaque caractéristique de tout ce qui constitue le *fond* du document est *à priori* encadrée par une instruction précise, chacune d'entre elles peut être modifiée sur tous les documents rédigés en une seule mise à jour. Il s'agit dès lors d'encadrer un maximum de caractéristiques du document qui peuvent l'être par des instructions. Et le faire avant la production de cours, afin d'éviter de devoir rajouter de nouvelles instructions après coup, mais aussi dans un but d'uniformisation des documents produits.

Cela peut être illustré par deux exemples (un *exemple* étant déjà caractérisé par un lot d'instructions, cela est détaillé dans l'**??**) :

Exemple 1.1: Notation des siècles

Cela sera abordé dans la **??**, mais afin d'automatiser la rédaction de cas particuliers de la langue française comme la notation des siècles (et afin que chacun n'en fasse pas qu'à sa manière), le package AOCDTF contient une nouvelle macro-commande définie comme telle :

Code

```
\newcommand*{\siecle}[1]{
\ifnum#1=1
\bsc{\romannumeral #1}\textsuperscript{er}~siècle
\else
\bsc{\romannumeral #1}\textsuperscript{e}~siècle
\fi}
```

Lorsqu'on appelle l'instruction `\siecle {1}` , cela produira Ier siècle. Si l'on appelle l'instruction `\siecle {2}` , cela produira IIe siècle. Je vous laisse décrypter la définition de cette instruction ci-dessus pour essayer de comprendre son fonctionnement.

Code

```
Lorsqu'on appelle l'instruction
\mintinline{latex}{\siecle{1}}, cela produira
\siecle{1}. Si l'on appelle l'instruction
\mintinline{latex}{\siecle{2}}, cela produira
\siecle{2}. Je vous laisse décrypter la
définition de cette instruction ci-dessus
pour essayer de comprendre son
fonctionnement.
```

Exemple 1.2: Style du chapitre

La personnalisation du style de chapitre a demandé une quinzaine de jour. Son bloc de code se situe en fin du package AOCDTF, sous le titre `%Mise en page du document` et le sous-titre `%Chapitre`, les plus téméraires peuvent donc essayer de le décrypter.
Résultat, l'instruction redéfinie `\chapter {Visualition du style d'un chapitre}` donnera donc :

CHAPITRE 1

Visualition du style d'un chapitre

Le style du titre de chapitre pourra être modifié ultérieurement d'un simple mise à jour pour *tous* les documents rédigés avec le package AOCDTF car tout ce qui caractérise ce style est défini dans ce package, qui constitue donc le tronc commun de tous ces documents.

Dans les principes de base, une dernière valeur ajoutée de LaTeX – il y en aura d'autres sur une multitude de sujets – se concrétise dans la *relativité* et l'*exactitude* des unités de mesure et de la disposition des éléments sur la page. Les unités sont multiples :

millimètre : `mm` ;

centimètre : `cm` ;

point anglo-saxon `pt` ;

point Didot `dd` ;

hauteur de la lettre x : `ex` ;

cadratin (largeur de la lettre M) : `em`.

Les unités de mesures énumérées ci-dessus sont soit absolues, soit relatives. Cela permet aux éléments définis par ces mesures d'être dimensionnés très précisément et de se restructurer en cas de changement de police ou de taille d'écriture pour conserver une même échelle.
L'*exactitude* de ces mesures permet au rédacteur de disposer d'une grande précision dans l'agencement des pages ou encore lors du codage d'un schéma en instructions LaTeX (cela sera abordé dans le **??**).
L'autre atout est de pouvoir indiquer des valeurs *relatives* aux espaces de rédaction paramétrées en préambules ou en rédaction. Cela est explicité dans l'exemple 1.3.
Il convient toutefois de prêter attention à l'écriture des nombres, qui se fait au format américain. Le séparateur décimal sera donc `.` et non `,` , sous peine d'erreur de compilation.

1.2 Paramètre du code `master.tex`

Tous les paramètres nécessaires pour bien démarrer la rédaction d'un document sont détaillés à cette page du wiki.

1.3 Division du document

Avant de rédiger du texte, il convient de structurer le document en plusieurs divisions. Chaque style de chaque titre de division est défini dans le package AOCDTF, et pourra donc être modifié – après consultation – d'une seule intervention de mise à jour du code.
Selon les instructions suivantes, elles seront automatiquement numérotées dans l'ordre ou elles sont appelées dans le code :

1. partie : `\part {<titre de la partie>}` (à appeler dans le code `master.tex`) ;
2. chapitre : `\chapter {<titre du chapitre>}` (à rédiger à partir d'un `copier/coller` du code `MWE.tex`) ;
3. section : `\section {<titre de la section>}` ;
4. sous-section : `\chapter {<titre de la sous-section>}` ;
5. sous-sous-section : `\chapter {<titre de la sous-sous-section>}` ;
6. paragraphe : `\paragraph {<titre du paragraphe>}` .

Les mêmes instructions auxquelles sont rajoutées une astérisque produiront le même résultat à la différence que ces divisions-là ne seront pas numérotées ni référencées dans la table des matières, selon l'instruction-type suivante `\division *{<titre de la division non référencée>}` .
Il convient également d'indiquer si le chapitre que l'on rédige présente des signets et une pagination à la couleur variable selon le chapitre avec la macro-commande `\ChapFrame` .

1.4 Agencement du document

1.4.1 Saut de page

À la compilation du code, il se peut que les flottants et autres éléments occupant l'espace contraignent le contenu textuel à se répartir de façon peu harmonieuse. Pour y remédier – à la toute fin de la rédaction – il existe deux instructions permettant de « jouer » avec cette répartition en forçant les sauts de pages :

— saut de page sans répartition du contenu sur la page précédente : `\newpage` ;

— saut de page avec répartition du contenu sur la page précédente : `\pagebreak` .

1.4.2 En-tête, pied de page

Par souci de légèreté visuelle, les documents à destination de l'Association Ouvrière des Compagnons du Devoir et du Tour de France (AOCDTF) ne comportent pas d'en-tête et le pied de page comprend le logo sans texte de l'AOCDTF ainsi que le numéro de page. Celui-ci voit son format changer selon l'emplacement dans le document (frontmatter, mainmatter et backmatter) et si l'on souhaite indiquer le marqueur de chapitre, la pagination sera incluse dans une boite de la même couleur que celle du marqueur de chapitre (seulement dans le mainmatter).

1.4.3 Minipage

Une page peut contenir des « pages » additionnelles à l'intérieur de la page initiale nommées « minipage », qui sont des espaces qui se comporteront comme une « page dans la page ». Les « minipages » peuvent se montrer utiles pour aligner deux éléments sur un même axe horizontal, comme une figure avec une liste descriptive ou encore deux tableaux...

Exemple 1.3: Minipage

Une minipage est appelée avec l'environnement `\begin {minipage}[<position>]{<largeur>}` . Les instructions suivantes produiront donc deux « minipages », qui s'aligneront selon leurs paramètres d'alignement. Celle de gauche fera 8cm et va aligner le bas de son environnement – variable `[b]` – avec le haut de l'environnement – variable `[t]` – de celle de droite, qui fera 30% de la largeur de l'espace de rédaction. L'instruction `\hfill` remplira l'espace horizontal disponible entre les deux « minipages » :

Code

```
\begin{minipage}[b]{8cm}
\lipsum[66]
\end{minipage}
\hfill
\begin{minipage}[t]{0.30\linewidth}
\lipsum[75]
\end{minipage}
```

Cela produira :

Nunc sed pede. Praesent vitae lectus. Praesent neque justo, vehicula eget, interdum id, facilisis et, nibh. Phasellus at purus et libero lacinia dictum. Fusce aliquet. Nulla eu ante placerat leo semper dictum. Mauris metus. Curabitur lobortis. Curabitur sollicitudin hendrerit nunc. Donec ultrices lacus id ipsum.

Pellentesque interdum sapien sed nulla. Proin tincidunt. Aliquam volutpat est vel massa. Sed dolor lacus, imperdiet non, ornare non, commodo eu, neque. Integer pretium semper justo. Proin risus. Nullam id quam. Nam neque. Duis vitae wisi ullamcorper diam congue ultricies. Quisque ligula. Mauris vehicula.

L'alignement vertical de cet environnement sur la page est paramétré selon la variable définissant l'argument facultatif `[<position>]` :

haut de la page (top) : `t` ;

milieu de la page (middle) : `m` ;

bas de la page (bottom) : `b` ;

à l'emplacement défini par le code (here) : `h` .

Ces variables sont facultatives pour une minipage, qui s'intégrera par défaut dans le texte à l'emplacement ou l'environnement sera appelé dans le code correspondant.

Il se paramètre également sur la largeur de la minipage selon la variable définissant l'argument obligatoire `{<largueur>}` :

largeur relative à la largeur du texte : `<x\linewidth >` ;

largeur relative à la largeur de rédaction : `<x\textwidth >` ;

largeur absolue : `<xcm>` (ou autre unité de mesure propre à LaTeX).

1.4.4 Espace de rédaction

Dans le package AOCDTF sont paramétrés les différents espaces de rédaction. Les marges horizontales et verticales sont toutes de 20mm à l'exception de celle qui n'est pas du côté de la reliure, mesurant quant à elle 25mm. L'espace de rédaction mesure donc 165mm. Selon les instructions entrées dans le code `master.tex` sur le type de média, les marges verticales peuvent soit :

— être décalées entre les pages paires et impaires si le paramétrage est prévu pour un document format papier, afin de pouvoir imprimer et relier correctement le document rédigé.

— ne pas être décalées entre les pages paires et impaires si le paramétrage est prévu pour un document format écran.

1.4.5 Flottants

Les tableaux et les figures peuvent être traités comme des élément *flottants*, c'est-a-dire qu'il sont considérés hors du corps de texte et voient leurs emplacements non précisément défini par le rédacteur. Celui-ci va définir les paramètres d'emplacement généraux et c'est LaTeX qui se chargera du placement automatique des éléments flottants à la compilation. Cela permet d'éluder les problèmes d'insertions d'éléments qui mettent en désordre l'intégralité d'un document. C'est également aux éléments flottants que l'on associe des légendes référencées et des listes les compilant dans le *frontmatter*.

Pour un tableau ou une figure que l'on souhaite définir en tant qu'élément *flottant*, on fait appel aux environnements `\begin {table}[<emplacement>]` et `\begin {figure}[<emplacement>]` . L'argument `[<emplacement>]` donne des indications concernant le placement sur page selon les variables suivantes :

haut de la page (top) : `t` ;

milieu de la page (middle) : `m` ;

bas de la page (bottom) : `b` ;

sur une page dédiée (page) : `p` ;

à l'emplacement défini par le code (here) : `h` ;

force l'emplacement défini par le code (HERE) : `H`.

Horizontalement, les éléments flottants peuvent être alignés avec les instructions suivantes insérées en début des environnements `table` ou `figure` :

aligné à gauche : `\raggedright` ;

centré : `\centering` ;

aligné à droite : `\raggedleft` .

La légende de l'élément flottant est insérée dans ces environnements avec l'instruction `\caption {<légende de l'élément flottant>}` , qui sera toujours située en dessous de l'élément flottant. La version étoilée `\caption *{<légende de l'élément flottant non référencé>}` supprime la numérotation et le référencement de cet élément flottant.
Cette légende sera référencée dans une liste après la table des matière si le rédacteur le souhaite en activant les instructions situées dans le code `master.tex` :

liste des tableaux : `{\hypersetup {linkcolor=black}\listoftables *}` ;

liste des figures : `{\hypersetup {linkcolor=black}\listoffigures *}` .

Il s'agit donc d'éléments *indépendants* qui ne sont donc pas prévus être insérés dans d'autres environnements (exemple, minipage...) sous peine probable d'erreur de compilation. Cela dit, s'il est nécessaire d'insérer un tableau ou figure avec une légende dans un environnement ou une minipage, il existe deux solutions selon les erreurs de compilation :

sans environnement flottant : utilisation l'instruction `\captionof {<type d'élément flottant>}{<légende de l'élément flottant>}` ;

avec environnement flottant : mention la variable `H` dans l'argument `[<emplacement>]` .

1.5 Compilation du document

La compilation du code `master.tex` ou des sous-programmations peuvent nécessiter plusieurs essais pour que les différents éléments se positionnent correctement sur leur emplacements spécifiés, ou encore pour que les référencements (table des matières, bibliographie...) soient correctement exécutés.
Pour la compilation finale, il est nécessaire d'effectuer une `Compilation rapide` plutôt que de compiler à l'aide du moteur `PDFLaTeX`. Sur le logiciel Texmaker, la démarche pour paramétrer cette `Compilation rapide` est détaillée sur cette page du wiki.

1.5.1 Sous-programmation

Lorsqu'on rédige un document conséquent, comme abordé sur cette page du wiki, on fait appel à une architecture de code de type *master/slave*. Cette architecture s'articule autour de la *subdivision* du code et l'insertion de *sous-programmations* au format `.tex` dans des programmations parentes. Pour alléger le code `master.tex`, les chapitres sont donc des sous-programmations que l'on insère dans le code `master.tex` à l'aide de l'instruction `\include {<chemin d'accès/objet_mot-clé_mot-clé_plus_précis>}` dans le but de aérer ce dernier.
Les sous-programmations peuvent également être des tableaux ou des figures aux nombres de lignes de code trop importants pour être insérées directement dans le corps de texte. Elles sont injectées dans la programmation parente – généralement un chapitre – avec l'instruction `\input {<chemin d'accès/objet_mot-clé_mot-clé_plus_précis>}` , toujours dans le but d'aérer les différents codes et de faciliter la navigation entre ceux-ci durant la programmation.

Ces commandes impliquent d'identifier les chemin d'accès des sous-programmations, *relatifs* aux programmations parentes. Pour faciliter leurs usages, il faut *appeler ces instructions avec les outils Texmaker* depuis `Latex > \input {file}` . Procéder de la sorte permet au rédacteur de sélectionner le fichier souhaité depuis un explorateur de fichier et cela fera apparaitre le chemin d'accès aux fichiers de code `code.tex`.
Pour information, un chemin d'accès *absolu* indique l'emplacement exact du fichier dans l'ordinateur. Il est possible d'utiliser ce format de chemin d'accès mais il créerai une erreur de compilation en cas de collaboration sur un même fichier de code. Le chemin d'accès *relatif* au code parent, tant que l'on fait appel à des fichiers situés dans le dépôt, ne produira pas d'erreur de compilation car tous les collaborateurs sont censés utiliser la même arborescence concernant les fichiers du dépôt.

Comme explicité à cette page du wiki, ces sous-programmations doivent donc présenter des noms de fichiers de codes qui suivent une nomenclature bien précise, avec leur *objet* à préciser selon la liste suivante :

chapitre : `chap_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` ;

annexe : `ann_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` ;
section (rare) : `sec_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` ;
tableau : `tab_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` ;
figure : `fig_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` ;
équation : `eq_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` ;
graphique : `graph_mot-clé-général_mot-clé-précis(_mot-clé-plus-précis)` .

Pour rédiger une sous-programmation, il suffit de suivre les instructions à cette page du wiki. Il convient donc d'initier une nouvelle sous-programmation en copiant le code `MWE.tex`, commande Texmaker que l'on retrouve dans `Fichier > Nouveau en copiant à partir d'un document existant`. L'optimisation du code contenu dans le package AOCDTF automatise complètement la compilation du code d'une sous-programmation, qu'elle soit compilée seule ou incluse dans une programmation parente.
En cas de rédaction d'un nouveau chapitre, il faut juste ne pas oublier d'indiquer l'instruction `\ChapFrame` juste après l'instruction `\chapter {Titre du chapitre}` si l'on souhaite voir apparaitre les marqueurs sur ce chapitre.

1.5.2 Usage des documents

Comme abordé dans cette page du wiki, il est possible de conditionner la compilation du document selon l'usage qu'il en sera fait avec l'instruction `\media {type_media}` appelée en préambule. Le type de média sera donc `screen` ou `paper`. Ce paramétrage réarrange le document selon qu'il sera diffusé sur papier ou lu à l'écran (ré-agencement des pages, liens en noir...) mais pas que.
Durant toutes la rédaction du code, le rédacteur sera amené à inclure des éléments multimédias (vues 3D, vidéo...), il conviendra donc de doubler chaque élément *interactif* de sa version papier et inversement.
Cela est réalisé à l'aide de la macro-commande conditionnelle `\media {\thetypemedia }{<code pour usage papier>}{<code pour un usage interactif>}` . Les codes rédigés dans cette instruction s'exécuteront donc selon les variables `screen` ou `paper` renseignées dans l'argument `{<type du média>}` en début du code `master.tex`, il s'agit d'un *ou* exclusif, les deux codes ne pourront jamais s'exécuter de concert.

Exemple 1.4: Type de média

L'exemple suivant permettra, selon les paramètres `screen` ou `paper` renseignés à la compilation de ce document, d'exécuter l'un ou l'autre code :

Code destiné à un usage interactif comme par exemple une vidéo qui s'afficherait dans le document si l'instruction `\media {screen}` était appelée dans le code `master.tex` .

Code

```
\media{\thetypemedia}
{Code destiné à un usage papier comme par
exemple une capture d'écran tirée d'une vidéo
qui s'afficherait dans le document si
l'instruction
\mintinline{latex}{\media{screen}} était
appelée dans le code  \texttt{master.tex} .}
{Code destiné à un usage interactif comme par
exemple une vidéo qui s'afficherait dans le
document si l'instruction
\mintinline{latex}{\media{screen}} était
appelée dans le code \texttt{master.tex} .}
```

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.