CE QUE TOUT DÉBUTANT (LA)TEX DEVRAIT SAVOIR

Maxime Chupin a traduit l'article de Barbara Beeton What every (LA)TEX newbie should know, *paru dans le* TUGboat *volume 44, numéro 2, 2023 [1].*

Résumé

LATEX a la réputation de produire d'excellents résultats, mais au prix d'un apprentissage assez long. C'est vrai, mais en comprenant quelques principes de base, et en apprenant comment éviter d'adopter quelques réflexes menant souvent à l'échec, il est possible de s'épargner quelques mauvais moments. Notre but est ici d'encourager les bonnes habitudes pour éviter que les mauvaises puissent proliférer.

Introduction

Les exemples présentés ici proviennent de deux sources principales :

- l'expérience de l'autrice pendant les années où elle a fait partie du support TEXnique d'un grand éditeur de mathématiques et ses responsabilités consistaient notamment à répondre aux questions des utilisateurs et à rédiger des documentations d'utilisation ;
- le forum web StackExchange [33] qui collecte un nombre gigantesque de questions, tant élémentaires qu'avancées. Ce site propose même, grâce à un effort collectif, une liste de *questions souvent référencées* classée par sujet et lisible à l'adresse suivante : `https://tex.meta.stackexchange.com/q/2419`.

Exhortation : lisez les documentations (ceci sera répété).

Vocabulaire

Plusieurs concepts semblent soit faire défaut au bagage élémentaire du débutant, soit être mal compris. Clarifions-les tout de suite.

Template

Beaucoup de nouveaux utilisateurs de (LA)TEX pensent qu'une *classe* de document est un modèle (*template*) pour un style ou une publication particulière. Ce n'est pas le cas, même si l'idée n'est pas si mauvaise. Un modèle est un fichier source `.tex` qui est une ébauche. Il commence par `\documentclass` et contient un minimum de commandes structurelles de base permettant d'y ajouter simplement le texte et les définitions additionnelles souhaités. Idéalement, le modèle lui-même peut être compilé sans erreurs même si la sortie n'aura que peu d'intérêt.

Ligne de commande

De nos jours, la plupart des nouveaux utilisateurs entre dans le monde de (LA)TEX avec un éditeur de texte dédié ou autre GUI et lancent des compilations non-interactives qui vont compiler le fichier source jusqu'à ce que le processus s'achève (avec ou sans erreur) ou se bloque dans une boucle. Lancer une compilation en ligne de commande permet d'interagir avec la session et, dans certains cas, d'effectuer des corrections *à la volée*, ou, si cela n'est pas possible, d'arrêter la compilation avant que le nombre d'erreurs signalées ne devienne trop important pour être réellement utile. Une erreur « corrigeable » proviendra par exemple d'une commande mal rédigée :

```
%
  ! Undefined control sequence.  l.37 \scetion {Section} ?
```

33. NdT : dont l'équivalent francophone est TeXnique.fr.

Répondez à cela avec l'orthographe correcte :

```
%
  i\section
```

appuyez sur « entrée », et poursuivez. N'omettez pas de corriger le fichier lorsque que vous rencontrerez une erreur facile à résoudre.

Un environnement mal orthographié ne peut pas être corrigé de cette manière, ainsi, si cela arrive, stoppez la compilation en tapant `x`, corrigez le fichier, et recommencez la compilation. Continuer la compilation avec une erreur ne pouvant être corrigée de manière interactive ne produira que de plus nombreux messages d'erreurs, la plupart ne voulant alors plus rien dire ou semant la confusion. Il vaut donc mieux se passer de tels messages.

Fichier de *log*

Chaque fois qu'une compilation TEX est lancée, un fichier de journalisation (autrement appelé fichier de *log*) est créé. Il vous faut savoir où se trouve ce fichier ! En plus des erreurs et des alertes (*warnings*), il y est reporté tous les fichiers lus en entrée, incluant les versions des classes et packages utilisés, les numéros de pages produites, et, à la fin, les ressources utilisées pour la compilation. Nous ne mentionnons ici que quelques-uns des éléments intéressants qui s'y trouvent. En revanche, pour entreprendre un débogage sérieux, nous renvoyons à l'article [34] issu d'un exposé précédent [2].

Conventions

De nombreux messages d'erreur produits par LATEX sont composés de plusieurs lignes : la première affiche le message d'erreur ; la suivante montre le numéro de ligne où l'erreur est identifiée, accompagné du contenu de la ligne jusqu'à l'endroit produisant l'erreur ; et celle d'après, indentée de telle sorte qu'avec le numéro de ligne précédent, elle complète la ligne pour qu'elle apparaisse comme dans le fichier source.

Bien que nous allions ici principalement parler d'aspects assez précis de l'utilisation de LATEX, gardez bien en tête qu'un des concepts fondamentaux de LATEX est de séparer le fond de la forme.

Structures fondamentales : commandes, modes et groupes

Ici nous allons aborder quelques concepts fondamentaux de LATEX.

Commandes

Les instructions sont données à (LA)TEX sous forme de commandes, ou « des séquences de contrôle », qui, par défaut, commencent par un *backslash* (\). Il y en a deux catégories : les commandes constituées d'un backslash et d'un seul caractère qui n'est pas une lettre (les « symboles de contrôle »), et les commandes constituées d'un backslash suivi d'une ou plusieurs lettres (les « mots de contrôle ») pour lesquelles seules les lettres sont acceptées (bas de casse ou capitales) et qui ne contiennent pas de chiffre ni de caractères spéciaux. Un mot de contrôle peut avoir un ou plusieurs arguments (`\title{...}`) ou non (`\alpha`). Un mot de contrôle sans argument (*standalone*) se terminera par une espace ou n'importe quel caractère qui n'est pas une lettre. Cependant, une espace après un symbole de contrôle apparaîtra comme une espace dans le document produit. Plusieurs symboles de contrôle sont prédéfinis pour produire leur propre caractère dans le document final : `\#`, `\%`, `\$`, `\&`. Par exemple, `\$` produira « $ ».

Un utilisateur peut définir de nouvelles commandes, ou peut changer la définition de commandes existantes. LATEX fournit la commande `\newcommand` pour créer une nouvelle définition. `\newcommand` vérifie si le

34. Nous nous proposons de publier une traduction de cet utile article, intitulé *Déboguer des fichiers LATEX — Illegitimi non carborundum*, dans un prochain numéro de la *Lettre*. Toute aide sera la bienvenue !

nom de commande n'a pas été déjà utilisé, et alerte si cela est le cas (la primitive TEX `\def`, elle, ne le fait pas). S'il est nécessaire de redéfinir une commande qui existe déjà, alors il faut utiliser `\renewcommand` …mais assurez-vous de savoir ce que vous faites le cas échéant. Par exemple, redéfinir `\par` est très risqué puisque LATEX l'utilise « en interne » pour de nombreux ajustements lors de la composition, et il est très facile de tout gâcher.

Les commandes constituées d'une lettre unique sont aussi de mauvais candidats pour des (re)définition par l'utilisateur puisque beaucoup d'entre elles sont prédéfinies pour être des accents ou des formes de lettre rares dans un texte en anglais ; `\i` a de fortes chances d'être utilisé avec (ou sans) accent dans une liste de références. Comme (mauvais) exemple, considérons l'auteur Haïm Brezis [35] :

Exemple 18

code

```
\renewcommand{\i}{\ensuremath{\sqrt{-1}}}
Brezis, Ha\"\i m
```

résultat

Brezis, Ha¨$\sqrt{-1}$m

Les commandes avec un seul chiffre (`\0`, `\1`, , etc.) ne sont pas prédéfinies dans le noyau de LATEX, et donc sont définissables pour un usage particulier.

Environnements

Un *environnement* est un bloc de contenu entre `\begin{`⟨nomenv⟩`}...\end{`⟨nomenv⟩`}`. Le nom d'environnement doit correspondre au début et à la fin ; et si ce n'est pas le cas, l'erreur suivante est rapportée dans le fichier de log ainsi que dans le terminal.

```
%
! LaTeX Error: \begin{xxx} on input line nn
    ended by \end{yyy}
```

La plupart des environnements peuvent être imbriqués, mais l'ordre doit être respecté.

D'autres commandes sont disponibles pour créer des nouvelles définitions comme `\NewDocumentCommand`, `\NewEnvironment`, `\NewDocumentEnvironment` et leur pendant pour les redéfinitions de commandes. Pour plus de détails sur leur utilisation, allez consulter une référence sur le sujet [36].

Modes

D'une manière générale, le « mode courant » indique où l'on se trouve sur la page (en sortie), mais ici, nous allons adopter un point de vue basé sur le fichier source.

Il y a trois modes : vertical, horizontal et mathématique.

À la suite de `\documentclass` ou après une ligne vide ou après un appel explicite à `\par`, LATEX est en mode vertical. Certaines opérations sont plus adaptées au mode vertical ; mais nous y reviendrons.

Commencer à taper du texte est une façon de passer en mode horizontal. Les commandes `\indent`, `\noindent` et `\leavevmode` permettent, elles aussi, de passer du mode vertical au monde horizontal.

35. NdT : avec LuaLATEX, on observe pas ce résultat. C'est pour cela que le résultat montré n'est pas composé en *Arsenal*, fonte inaccessible avec pdfLATEX.

36. NdT : par exemple l'article *Passer à la définition de commandes de LATEX3* dans la *Lettre* 49 : https://publications.gutenberg-asso.fr/lettre/article/view/113/106.

Lors que l'on se trouve en mode horizontal, plusieurs espaces consécutives sont traitées comme une espace unique et le fait qu'ils soient *consécutifs* est capital ici. Une *fin de ligne* (un caractère EOL) est traitée comme une espace, même si cela n'est pas explicitement visible dans le fichier source ; un GUI qui coupe automatiquement les lignes insère ou n'insère pas (habituellement ne le fait pas) un caractère EOL en fin de ligne, et les systèmes d'exploitations définissent différemment un caractère EOL, mais toutes ces différences sont gérées par le moteur TEX. Les espaces en début de ligne sont ignorées. Nous reviendrons plus tard sur la gestion des espaces.

Le troisième mode, le mode mathématique, peut-être inclus *en ligne* dans du texte ou afficher en tant que contenu hors-texte en mode vertical. Les mathématiques en ligne sont encadrées par des `$` ou par le couple `\(...\)`. Des mathématiques hors-texte non numérotées peuvent être écrites en utilisant le couple `\[...\]`. Les mathématiques hors-texte sur plusieurs lignes sont plus faciles à saisir en utilisant les environnements fournis par les packages amsmath et mathtools (reportez-vous à leurs documentations). Le package mathtools charge amsmath, il est donc inutile de les charger tous les deux. Des mathématiques hors-texte sont habituellement la continuité du paragraphe précédent : ainsi, il ne faut pas laisser de ligne vide entre des mathématiques hors-texte et le paragraphe précédent ; faire cela peut créer des changements de page non voulus.

En mode mathématique, les lignes vides ne sont pas autorisées ; cette décision fut prise par Knuth pour éviter les erreurs de saisie involontaires, car les mathématiques ne se poursuivent jamais au-delà d'un saut de paragraphe.

Groupe

Outre les modes, il existe le concept de groupe permettant de localiser les définitions et les opérations.

Le mode mathématique est un exemple de groupe ; certains caractères et certaines opérations ne sont autorisés qu'en mode mathématique, et d'autres, au contraire n'y sont pas autorisés. Dans du texte, le mode mathématique commence et termine par `$`. Les mathématiques hors-texte rompent la continuité du texte ; clôturer un contenu mathématique hors-texte remet LATEX en mode texte sauf si le contenu mathématique hors-texte est suivi par une ligne vide ou un `\par`. Nous y reviendrons plus tard.

Une autre façon de définir des groupes est de délimiter le contenu par des accolades : `{...}` À l'intérieur d'un groupe, la définition d'une commande peut-être modifiée pour obtenir des effets temporaires ; la définition en œuvre avant l'ouverture du groupe par `{` sera restaurée dès lors que l'accolade fermante sera traitée. À la place d'une paire d'accolage, on peut utiliser les commandes `\begingroup...\endgroup` qui produiront le même effet.

Une autre manière d'obtenir un groupe est de placer le contenu désiré dans une boîte (une *box*). Par exemple, on peut utiliser `minipage`, `\mbox` ou `\parbox`. D'autres boîtes sont définies dans des packages comme tcolorbox.

Des environnements (pas tous) sont définis pour être un groupe. C'est le cas de l'environnement `theorem`, à l'intérieur duquel le texte est composé en italic ; lorsque le théorème se termine, le style de texte reviendra automatiquement dans le style du document par défaut.

Espacement dans le texte

L'un des objectifs d'une composition de haute qualité est d'obtenir un espacement régulier dans le texte. Ceci n'est possible que si l'on compose le texte *au fer à gauche*[37], où les espaces ont simplement leur *largeur naturelle*. Cependant, il est généralement préférable d'avoir des marges elles aussi régulière, et c'est pourquoi TEX est conçu pour optimiser l'espacement dans ce contexte.

37. Où toutes les lignes sont alignées à gauche, sans se soucier de leur alignement à droite

Dans les documents en langue anglaise, les espaces qui terminent une phrase sont plus larges que les espaces entre les mots. Ceci n'est pas vrai pour des documents dans d'autres langues, et cette fonctionnalité peut être désactivée grâce à la commande `\frenchspacing`. Cependant, dans les documents académiques, les abréviations fréquentes peuvent rendre difficile le repérage des fins de phrase. Pour éviter des espaces trop larges après une abréviation, il suffit de faire suivre l'abréviation de la commande `\␣` [38] :

Exemple 19

code

```
\foreignlanguage{english}{%
(abc vs. xyz) vs.\\
(abc vs.\ xyz)\\
}
```

résultat

(abc vs. xyz) vs.
(abc vs. xyz)

Si la ligne ne doit pas être coupée après l'abréviation, il suffit de la faire suivre du ~ :

Exemple 20

code

```
\foreignlanguage{english}{%
seen on page.~23
}
```

résultat

seen on page. 23

Une situation similaire, mais inverse, peut se produire lorsqu'une majuscule est suivie d'un point. Supposons que c'est l'initiale d'un nom ; c'est généralement le cas, et une espace inter-mots ordinaire est insérée à la suite. Cependant, quelques fois, la lettre majuscule se trouvant à la fin d'un acronyme est aussi la fin d'une phrase. Dans ce cas [39], il suffit d'ajouter `\@` avant le point pour permettre de rétablir l'espace de fin de phrase plus large.

Tout cela se résume à une simple règle : sauf en fin de phrase (et dans une moindre mesure après d'autres symboles de ponctuation ou dans des mathématiques), toutes les espaces d'une même ligne doivent avoir la même largeur. Si ce n'est pas le cas, c'est qu'il y a un problème.

Espaces parasites

Des espaces multiples peuvent s'infiltrer dans un fichier source de nombreuses manières, mais dans la grande majorité des cas, ils sont le résultat d'un effort excessif pour définir des commandes qui sont visuellement agréables (et facilement lisibles). En voici un exemple :

38. NdT : Ici, nous avons du mettre l'exemple en anglais pour observer la différence entre les espaces.
39. NdT : et toujours pour la langue anglaise.

Exemple 21

code

```
\newcommand{\abc}{
  \emph{abc def}
}
word \abc\ word vs.\\
word \emph{abc def} word
```

résultat

word *abc def* word vs.
word *abc def* word

Avec cette définition de macro, l'entrée `word \abc\ word` produit des espaces supplémentaires introduits par notre commande `\abc`. Les espaces parasites peuvent être évitées en insérant des `%` là où, de façon cachée, des caractères EOL (fin de ligne) sont insérés :

Exemple 22

```
\newcommand{\abc}{%
  \emph{abc def}%
}
```

Ceci permet de produire ce que l'on souhaitait « word *abc def* word ». Le caractère `%` commence un commentaire, c'est-à-dire qu'il fait en sorte que le reste de la ligne est ignoré, en particulier le caractère d'EOL.

Une autre source d'espaces parasites en sortie est la présence de plusieurs éléments consécutifs qui ne font pas partie du texte principal, comme les notes de pied de page ou des labels [40] :

Exemple 23

code

```
An important topic\label{abc}
\label{def}
\label{xyz}
is labeled several ways. vs.\\
An important topic is labeled several ways.
```

résultat

An important topic is labeled several ways. vs.
An important topic is labeled several ways.

Ici, l'effet du caractère EOL est encore à l'œuvre (après « topic »), et les espaces ne sont plus alors consécutives. Une fois de plus, le caractère `%` vient à la rescousse :

Exemple 24

```
An important topic\label{abc}%
\label{def}%
\label{xyz}
```

40. NdT : dans l'article original, l'exemple était produit avec la commande d'index. Cependant, cette commande ne produit pas l'effet escompté. Nous avons donc utilisé la commande de label.

```
4  is labeled several ways.
```

N'oubliez de laisser une espace malgré tout.

Quelques fois, l'utilisation d'un % est une mauvaise idée

Souvenez-vous qu'une espace indique la fin d'un mot de contrôle et qu'elle est alors ignorée en tant que caractère de texte ; à cet endroit-là, il n'est alors pas nécessaire d'ajouter un `%`. Cependant, il y a des endroits où l'ajout de `%` peut vraiment être problématique.

Après avoir défini n'importe quelle valeur numérique, TEX va continuer à interpréter tout ce qui suit et qui peut considéré comme une valeur numérique, ainsi, si une ligne se termine avec `\xyz=123`, aucun `%` ne doit être ajouté. De même, si l'on fixe une longueur de ressort (*glue*), disons avec `\parskip=2pc`, TEX va continuer de chercher un `plus` ou un `minus`. Utiliser une unité lexicale (*token*) vide, `{}`, est un meilleur arrêt pour une telle commande (si `plus` ou `minus` suit la commande et se trouve être du texte, un message d'erreur déroutant sera alors affiché, mais cela est rare et dépasse le cadre de cette discussion).

Espaces supplémentaires vraiment inattendues

Il y a d'autres possibilités d'apparition d'espaces parasites qui ne sont pas facile à prédire. Ici, nous montrons un exemple qui fut le sujet d'une question sur le web. Considérons un texte avec `\colorbox` (cela peut aussi arriver avec tcolorbox) qui contient une lettre colorée entourée par des espaces au milieu d'un mot. Oo p s! Un petit cadre a été appliqué autour de l'élément coloré :

Exemple 25

code

```
1  \newcommand{\mypink}[1]{%
2  \colorbox{red!20}{#1}}
3  Oo\mypink{p}s!
```

résultat

Oo p s!

La solution fournie par la documentation [41] consiste à supprimer explicitement la marge à l'intérieur du cadre :

Exemple 26

code

```
1  \newcommand{\mypink}[1]{{%
2  \fboxsep=0pt
3  \colorbox{red!20}{#1\strut}}}
4  Oo\mypink{p}s!
```

résultat

Oops!

J'ai ajouté le `\strut` de telle sorte que la couleur s'étende au dessus et au dessous de l'élément mis en valeur, plutôt que de recouvrir seulement le « p » (dans notre cas). Bien qu'il ne s'agisse pas vraiment d'un problème de débutant, il est sage d'être conscient que de telles possibilités existent, et d'être prêt dans ces cas là à demander de l'aide à un expert.

41. NdT : documentation de l'ensemble *graphics* du *The LaTeX project*.

Fin de paragraphes et mode vertical

La fin d'un paragraphe est la transition entre le mode horizontal et le mode vertical. Une ligne vide ou un `\par` permettent d'assurer cette transition. Il est important d'être conscient du mode dans lequel on se trouve, puisque certaines opérations se doivent d'être réalisées dans le mode vertical : la plus importante est l'insertion de flottant (figures, tables, algorithmes).

Un autre élément important est que certains paramètres du texte ne sont « gelés » que lorsque le paragraphe est terminé. Un de ces paramètres important est l'espacement vertical, entre les lignes de base, qui dépendra de la taille de la fonte. Trop de débutants essaient de finir un paragraphe avec un double backslash, ce qui provoque des horreurs comme celle qui suit [42].

Exemple 27

```
\Huge Texts with inconsistent descenders
can result in surprises when the font
size changes without a proper paragraph
ending.\\
```

qui produit le résultat suivant.

Quelques environnements (mais pas *tous*) sont définis avec une fin de paragraphe à la fin. Un problème comme celui que nous venons d'illustrer ne produira pas de message d'erreur ni d'avertissement, et donc la solution consiste à ajouter explicitement une fin de paragraphe.

L'espace vertical entre les paragraphes est déterminé par la valeur de `\parskip` ; cette valeur est définie par la classe du document mais peut être redéfinie lorsque cela est nécessaire. Occasionnellement, il peut être utile d'ajouter explicitement des espaces supplémentaires ; cela est fait avec les commandes `\vspace` ou `\vskip` lorsqu'on se trouve en mode vertical (c'est-à-dire après une ligne vide ou un `\par` qui termine un paragraphe).

Double backslash

Le symbole de contrôle `\\` ne termine *pas* un paragraphe. `\\` termine une ligne. C'est une commande conçue pour terminer les ligne dans les tableaux, les poésies et les environnements mathématiques multilignes, ainsi que dans quelques autres situations. Cependant, il ne termine pas un paragraphe et peut générer quelques messages d'erreur ou d'avertissement.

Si `\\` est seul sur une ligne en mode vertical, l'erreur sera la suivante :

```
%
! LaTeX Error:
    There's no line here to end.
```

De plus, si le `\\` est précédé par une espace, en plus de l'avertissement, il y aura une ligne blanche supplémentaire, non voulue, dans le document de sortie.

42. NdT : ici, l'environnement d'exemple de la *Lettre* empêche d'observer le fonctionnement normal de LaTeX. Ainsi le résultat est affiché en suivant.

Si une ligne qui se termine par `\\` est trop courte, on aura :

```
%
Underfull \hbox (badness 10000)
    in paragraph at lines ...
```

Cela peut être acceptable, mais vérifiez le cas échéant.

Si `\\` est suivi par du texte entre crochets, comme `[du matériel à composer]`, le résultat donnera le message énigmatique suivant :

```
%
! Missing number, treated as zero.
```

La commande `\\` possède en effet un argument optionnel, et `[...]` permet alors d'indiquer la distance verticale à ajouter à la suite de la coupure de ligne ; pour permettre la composition de texte entre crochets, il suffira de rajouter `\relax` avant le crochet ouvrant.

Si l'espace vertical supplémentaire est désiré après la ligne coupée grâce à `\\`, il peut donc être ajouté en insérant, en option, une taille de ressort (couramment juste une dimension) à l'intérieur de crochets : `\\[`⟨valeur⟩`]`.

Notons que `\newline` est souvent une bonne alternative à la coupure de ligne.

Changement de fonte

Le changement de police permet traditionnellement d'exprimer des nuances de sens ou de mettre en évidence des concepts particuliers ou importants. Ces changements de police sont implémentés dans les classes ou les packages ; par exemple, les théorèmes sont présentés dans une police *italique*, les titres de sections sont en **gras**, et certaines revues définissent les légendes de figure en sans sérif pour les distinguer du texte principal.

LaTeX fournit deux méthodes différentes pour changer de polices. Le premier type de commandes prend un argument et limite la persistance du changement au contenu de l'argument ; ces commandes sont de la forme `\textbf{...}` pour le **gras**, `\textit{...}` pour l'*italique*, etc. L'autre type de commandes change le style de fonte de telle sorte que 1/ cela ne change plus jusqu'à ce qu'un prochain changement explicite ne soit fait, ou 2/ le changement soit limité par le groupe d'un environnement. Les commandes suivantes en sont des exemples : `{\itshape...}`, `{\bfseries ...}`, `{\sffamily...}`. Il est préférable de consulter un bon guide d'utilisation de LaTeX pour en connaître plus sur ces commandes.

D'autres commandes de changement de police vont produire des effets différents suivant le contexte d'utilisation. `\emph{...}` va faire passer en italique si le texte courant est en roman, ou en roman si le texte courant est en italique. En mode mathématique, `\text{...}` permettra de composer du texte dans le même style que le style de texte dans lequel le contenu mathématique est inséré ; ainsi, dans un théorème, `\text{...}` permettra de composer le texte en italique. Si le texte à composer en mode mathématique doit être dans tout contexte en roman, comme les noms de fonctions, il faudra alors utiliser `\textup{...}`.

En TeX classique, les changements de style de police sont faits avec des commandes dont les noms ne sont produits qu'avec deux lettres. Toutes ces commandes sont du type persistant. On ne doit pas les utiliser avec le format LaTeX puisque celui-ci fournit des commandes améliorées, par exemple pour mieux gérer les transitions entre italique et roman.

Mathématiques

Les mathématiques sont toujours composées dans un groupe. Si le mode mathématique est ouvert, alors il doit être fermé explicitement et sans ambiguïté. Dans du texte courant, les mathématiques sont délimitées en début et en fin par `$`. LaTeX fournit aussi le couple `\(...\)` pour les mathématique en ligne, mais la plupart des utilisateurs utilisent toujours `$`. De nombreux environnements pour les mathématiques hors-texte sont définis par les packages amsmath et mathtools, et cela vaut le coût d'aller consulter leurs documentations pour les découvrir.

En mode mathématique, tous les espaces en entrée sont ignorés par (La)TeX ; ils peuvent être utiles dans le source pour en améliorer la lecture du code par un humain. Cependant, les lignes vides, elles, provoquent des erreurs. Dans les deux modes, en ligne et hors-texte, le message d'erreur sera :

```
%
! Missing $ inserted.
```

Ce message d'erreur arrivera aussi si un environnement de mathématiques en ligne n'est pas fermé avant la fin de paragraphe, ou si un symbole mathématique ou une commande propre au mode mathématique se trouvent hors mode mathématique.

Si une ligne vide se trouve dans un environnement hors-texte multiligne du package amsmath, le *premier* message d'erreur sera

```
%
! Paragraph ended before <env-name>
    was complete.
<to be read again>
```

Cela sera suivi de *nombreux* autres messages d'erreur, tous provoqués par ce premier. Tout cela prêtera à confusion et induira en erreur. Corrigez toujours le problème identifié par le premier message d'erreur et ignorez le reste ; les autres messages disparaîtrons souvent une fois la première erreur corrigée, ici en enlevant la ligne vide.

Si, pour rendre le source plus lisible, une ligne vide est désirée, utilisez une ligne contenant uniquement un `%`.

Comme avec tous les environnements, le nom dans le `\end` doit correspondre avec celui du `\begin` correspondant. Une abréviation pour composer des mathématiques hors texte sur une seule ligne est le couple `\[...\]`. Les environnements conçus pour composer des mathématiques sur plusieurs lignes ne doivent pas être utilisés pour des mathématiques sur une seule ligne.

Bien que LaTeX fournisse l'environnement `eqnarray` comme un environnement hors texte, ne l'utilisez pas. Si l'environnement est numéroté, et que l'équation est trop longue, l'équation pourra être surimprimée par le numéro de l'équation.

Tables, figures et autre flottants

Le nombre de flottants autorisé par page, leurs positions sur la page, et les espaces autour et entre eux sont définis par la classe de document. Ainsi, si quelque chose ne se passe pas comme vous l'espérez, toute personne susceptible de vous aider insistera pour savoir quelle classe de document est utilisée.

Pour que se produise ce qui est attendu, le code d'un flottant doit être inséré dans le fichier source tant qu'il reste de la place dans la page de sortie. En particulier, pour les pages à deux colonnes, une `figures*` ou une `tables*` doivent être placées dans le source *avant* que quoi que soit composé sur la page. Le noyau de LaTeX qui gère le placement des flottants ne permet pas de placer n'importe où des flottants

occupant une largeur complète mais seulement en haut d'une page. Quelques packages modifient ce comportement, mais ceci ne sera pas discuté ici.

Nous présentons ici les paramètres par défaut de la classe article.

- Le nombre total de flottants par page est au maximum de trois.
- Le nombre de flottants autorisé en haut de page est de deux. Le pourcentage de la page occupé par des flottants en haut de page est de 70 %.
- Le nombre de flottants autorisé en bas d'une page est de un. Le pourcentage de la page occupé par des flottants en bas de page est de 30 %.
- La hauteur minimale de la page requise pour le texte est de 20 %.
- La hauteur minimale d'un flottant nécessitant une page à lui seule est de 50 %

La hauteur de référence est `\textheight`, c'est-à-dire la hauteur de page en excluant l'entête et le pied de page.

Si le matériel que vous voulez ajouter est petit, qu'il doit être placé précisément dans le texte et qu'il y a la place pour le placer, alors n'utilisez pas de flottant. `\includegraphics` ou un de nombreux moyens de produire des tableaux peuvent aussi être utilisés directement, souvent dans un environnement `\begin{center}...\end{center}` (dans un flottant, on utilisera `\centering`).

Le package wrapfig permet d'insérer du matériel sur le côté de la page ou de la colonne autour duquel *coule* le texte. Reportez-vous à la documentation pour plus de détails.

Traditionnellement, les légendes sont composées au dessus d'une table et en dessous d'une figure. Si l'insertion n'est pas un flottant, la commande habituelle `\caption` ne peut pas être utilisée. On peut cependant utilise le package caption et la commande `\captionof`.

Classe de document et préambule

Lorsque vous commencez un nouveau document, commencez par choisir la classe à utiliser. Si la finalité du document est d'être publié dans une revue particulière, prenez connaissance des consignes de d'éditeur pour savoir ce qui est exigé. Bon nombre des revues le plus connues ont des classes disponibles sur le CTAN [43].

Si le projet est une thèse ou un mémoire, renseignez-vous sur ce qui est exigé, et si votre institution fournit une classe adaptée, obtenez-la. Essayez de déterminer si la classe est maintenue et s'il existe un support local. Lisez aussi la documentation.

La classe d'un document se doit de définir la structure générale du document qu'elle sert à produire. Si le document sur lequel vous travaillez diffère de façon essentielle de ce qui est pris en charge par la classe du document, il est nécessaire d'obtenir de l'aide *immédiatement*.

Il y aura des fonctionnalités non prises en charge nativement par la classe du document ; par exemple, le choix de la méthode de production de la bibliographie peut être laissé à l'auteur. C'est pour cela que les packages ont été créés.

Organisation du document

La plupart des packages sont chargés dans le préambule. Il y a une exception : `\RequirePackage`. Cette commande est généralement utilisée avant `\documentclass`, et c'est grâce à cette commande que certaines options particulières de packages doivent être chargées.

Quelques auteurs créent un préambule qui est adapté pour un document particulier puis réutilisent ce même préambule pour les documents suivants, en ajoutant plus de packages au fur et à mesure. Certains débutants « adoptent » ces « modèles » [44] de seconde main sans comprendre comment ils ont été créés.

43. https://ctan.org/search
44. NdT : ces *templates*.

Ne faites pas ça !

Commencez avec une classe convenable et ajoutez les fonctionnalités (packages, options et macros) lorsqu'elles deviennent nécessaires. Organisez le chargement de packages en groupes logiques (toute la gestion de fonte ensemble, par exemple), et faites attention de ne pas charger plusieurs fois le même package ; si des options sont nécessaires, sachez que les chargements produits par un deuxième `\usepackage` seront ignorés. Certains packages chargent automatiquement d'autres packages dont ils dépendent ; par exemple mathtools charge amsmath et amssymb charge amsfonts. De plus, il faut vraiment faire attention à l'ordre le chargement des packages : hyperref doit être chargé (quasiment) en dernier, les quelques packages qui doivent être chargés après hyperref sont bien documentés.

Lisez les documentations.

Compilation

Une fois que le document est rédigé, il est temps de le compiler pour en produire une sortie. Il existe plusieurs moteurs et il faut choisir : pdfLaTeX, XeLaTeX et LuaLaTeX. Ils peuvent être exécutés de manière interactive à partir de la ligne de commande ou lancés à partir d'un éditeur de texte dédié à LaTeX. En supposant qu'il n'y ait pas d'erreurs, le nombre de fois qu'un document doit être compilé pour produire le résultat final dépend des éléments qu'il contient.

En particulier, s'il existe des références croisées ou des citations, ces informations sont écrites dans des fichiers auxiliaires (`.aux`) ; les informations de la table des matières sont inscrites dans un fichier `.toc`, et d'autres listes de contenu sont aussi possibles. La bibliographie doit être traitée avec un programme différent du moteur LaTeX (et son fichier de *log* doit être consulté pour comprendre les erreurs) avec un autre fichier séparé pour les données bibliographiques dans le format `bib`. Ensuite, le document LaTeX doit être compilé au moins deux fois — une fois pour lire le fichier `.aux` et d'autres fichiers auxiliaires, pour inclure la bibliographie et résoudre les références croisées et la deuxième compilation pour obtenir les bons numéros de pages (qui vont changer lorsque la table des matières et autres contenus similaires seront ajoutés).

Tout ceci suppose qu'il n'y a pas d'erreurs. Les erreurs seront enregistrées dans le fichier de *log*. Il vous faut savoir où le fichier de *log* se trouve et prenez l'habitude d'aller le consulter. Les avertissements, tels que ceux qui indiquent que des caractères sont manquants, seront aussi écrits dans ce fichier, mais peuvent aussi être affichés dans la sortie du terminal :

```
%
Missing character: There is no <char>\
                   in font <font>!
```

Dans le fichier de *log*, certaines erreurs peuvent apparaître avec de nombreux numéros de lignes. Si tel est le cas, et que la première est celle qui interrompt le traitement de l'environnement, les erreurs suivantes seront probablement des erreurs parasites. Ainsi, traitez la première erreur et compilez avant d'essayer de comprendre les erreurs suivantes : souvent, elles disparaîtront.

Bonne chance. C'est avec la pratique que l'on comprend.

Oh !…Souvenez-vous : lisez les documentations !

Remerciement

Merci à samcarter, Mikael Sundquist, et (comme toujours) Karl Berry pour les suggestions et pour avoir trouvé et exterminé mes fautes de frappe. J'arrive à les détecter dans les écrits des autres, mais pas souvent dans mes propres productions.

Références

[1] Barbara Beeton. « What every (LA)TEX newbie should know ». In : *TUGboat* 44.2 (2023), p. 164-169. DOI : 10.47397/tb/44-2/tb137beeton-basic. URL : https://doi.org/10.47397/tb/44-2/tb137beeton-basic.

[2] Barbara Beeton. « Debugging LATEX files — Illegitimi non corborundum ». In : *TUGboat* 38.2 (2017), p. 159-164. URL : https://tug.org/TUGboat/tb38-2/tb119beet.pdf.

Barbara Beeton

BRÈVE INTRODUCTION À UNE COMPILATION ASSISTÉE, GRÂCE À ARARA

Introduction

Présentation rapide

Le logiciel arara est un utilitaire, intégré aux distributions LaTeX, qui permet d'automatiser des actions (définies par des règles) de compilation de fichier TeX.

Il existe d'autres assistants similaires, comme latexmk, rubber, ou encore spix. Chacun a ses spécificités propres, ses domaines d'applications, ses avantages et ses inconvénients, et nous ne rentrerons pas dans la comparaison détaillée de chacun.

Une spécificité de arara est d'utiliser `Java` comme moteur d'exécution, donc une machine `Java` est nécessaire au bon fonctionnement de cette méthode.

Pourquoi ?

Ayant parfois (souvent ?) besoin d'utiliser des chaînes de compilations différentes (qui utilisent l'option `shell-escape`, ou les moteurs `pdflatex`, `lualatex`, etc.), j'avais paramétré différentes chaînes de compilation dans mon éditeur `TeXstudio`, mais je n'étais que *moyennement* satisfait de cette méthode !

Donc une petite recherche sur des *outils* de compilation m'a conduit à me pencher sur des solutions de type « *assistants de compilation* ».

Le logiciel arara a retenu mon attention par :

- sa simplicité de configuration, directement dans le document TeX ;
- sa simplicité à s'adapter à mes différents besoins.

Installation(s)

Pour ce qui est du *package* arara, rien de plus simple. Normalement, il est fourni par les distributions LaTeX classiques et il suffit alors de vérifier dans le gestionnaire de paquets de sa distribution LaTeX qu'il est installé, sinon l'installer grâce à ce même gestionnaire (ligne de commandes ou interface graphique).