

# Style et Structure des Documents

STEP, Mines Paristech\*

3 septembre 2019 (#2bf3c07)

## Table des matières

|  |          |
|--|----------|
| <b>Markdown</b>                                    | <b>1</b> |
| <b>Sections</b>                                    | <b>2</b> |
| Hiérarchie . . . . .                               | 2        |
| Type/Propriétés des sections de niveau 3 . . . . . | 2        |
| <b>(Hyper-)Liens</b>                               | <b>2</b> |
| <b>Lignes</b>                                      | <b>3</b> |
| <b>Mathématiques</b>                               | <b>3</b> |
| <b>Italique &amp; Gras</b>                         | <b>4</b> |
| <b>Images</b>                                      | <b>4</b> |


## Markdown

Les documents sources exploitent une variante du format Markdown.

Pour une introduction à Markdown, voir par exemple <https://commonmark.org/>.

La variante de Markdown utilisée est celle supportée par l’outil Pandoc; elle supporte à la fois des fonctionnalités indispensables à un cours de Mathématiques

---

\*Ce document est un des produits du projet  **boisgera**/CDIS, initié par la collaboration de (S)ébastien Boisgérault (CAOR), (T)homas Romary et (E)milie Chautru (GEOSCIENCES), (P)auline Bernard (CAS), avec la contribution de Gabriel Stoltz (Ecole des Ponts ParisTech, CERMICS). Il est mis à disposition selon les termes de la licence Creative Commons “attribution – pas d’utilisation commerciale – partage dans les mêmes conditions” 4.0 internationale.

(formules, bibliographie, etc.) et la production de documents dans une large gamme de formats. Se reporter à son guide utilisateur pour les détails du format.

## Sections

### Hierarchie

TODO:

- 3 niveaux normalement assez (4 le cas échéant pour segmenter les longues preuves ?).
- “Atomes”, l’essentiel du contenu est au niveau 3
- Structure des sections d’exercices ?
- Exercices. Section (de top niveau) à part, mais en plus exercices dans le corps du texte ou bien uniquement remarques avec éventuels renvois à la section des exercices (l’info intéressante étant le “point d’insertion” d’un exercice dans le narratif du cours) ? A priori je dirais à ce stade plutôt remarque + renvoi.

### Type/Propriétés des sections de niveau 3

- theorem, proposition, corollary, lemma.
- proof
- exercise (?), questions, answers (pluriel ?)
- remark, example, meta, anonymous

## (Hyper-)Liens

Il existe plusieurs mécanismes standards pour introduire un lien vers une section interne au document ou un lien externe (page Web), qui sont documentés dans le guide utilisateur de pandoc: <https://pandoc.org/MANUAL.html#links-1>

Le plus simple en interne consiste sans doute à faire la chose suivante:

```
### Théorème intégral de Cauchy {.theorem}  
bla bla bla
```

(...)

... par le [théorème de Cauchy][Théorème intégral de Cauchy], ...

Alternativement, on pourra donner un identifiant à la section, ce qui permettra des références plus “compactes” (mais moins explicites ...):

```
### Théorème intégral de Cauchy {.theorem #TIC}  
bla bla bla
```

(...)

... par le [théorème de Cauchy](#TIC), ...

Un besoin qui n’a pas de solution 100% standard est la référence à une section **dans un autre document du projet**. On fera l’hypothèse pour spécifier ces liens que les ressources prennent la forme d’une collection de documents pdf dans le même répertoire<sup>1</sup>. Alors pour faire référence à une section dans un document `a.pdf`, il faudra lui donner un identifiant explicite (ici: `towel`)

```
### Théorème de la serviette {.theorem #towel}  
bla bla bla
```

et dans le document `b.pdf` y faire référence de la façon suivante:

Comme le prouve le [théorème de la serviette](a.pdf#towel), ...

## Lignes

Parce que je suis bigleux, j’utilise des fontes assez grosses dans mon éditeur, ce qui me pousse à limiter – quand c’est possible – la largeur des lignes à 80 caractères, ce qui est un des standards “de fait” assez commun (à défaut d’être universel), cf. [https://en.wikipedia.org/wiki/Characters\\_per\\_line](https://en.wikipedia.org/wiki/Characters_per_line).

Quoi qu’il en soit, cela ne veut pas dire nécessairement aller jusqu’au bout de la ligne: s’il y a une rupture “sémantique” naturelle, ne pas hésiter à changer de ligne de façon précoce; c’est plus lisible, plus facile à éditer, les “diffs” de Git sont plus facile à interpréter, etc. A ce propos, voir par exemple: <https://sembr.org/>

## Mathématiques

- Au moins à court terme, inclure les macros LaTeX nécessaires au début de chaque document (disons l’entête composée du titre, des auteurs et de la date); comme Pandoc se charge lui-même de faire la substitution, a priori pas de risque de collision entre des conventions différentes d’un document à l’autre en cas de regroupement.

Cf la section Pandoc – LaTeX macros pour lus de détails.  
usage des macros

---

1. Si *in fine* on génère des documents contenant plusieurs chapitres, où si l’on génère des documents HTML, il faudra (automatiquement) détecter et réinterpréter ces liens pour les adapter au nouveau contexte.

- usage libéral du “display style” si nécessaire plutôt que d’utiliser la commande `\displaystyle` dans des formules “inlines”, ce qui casse le rythme vertical du document.

TODO:

- ponctuation dans les formules (à terme), par exemple  
 Dans le cas où  $a=1$ , la situation est identique.  
 Sinon dans l’export HTML, Mathjax va faire des césures à des endroits inappropriés. Bonus de cette convention: c’est de toute façon ce qu’il faut faire en mode “display style”, ça à le mérite de la cohérence.

## Italique & Gras

L’italique sera utilisé pour désigner les nouveaux termes dans les définitions, et éventuellement des termes en langue étrangère dans le corps du texte.

Le gras sera utilisé pour mettre une emphase<sup>2</sup>.

Par exemple:

**\*\*Attention\*\***, dans ce qui suit, le terme *\*intervalle\** désigne est un sous-ensemble connexe de la droite réelle; un intervalle peut donc être non borné.

## Images

Les fichier d’images “statiques” – comme des photos au format PNG, JPG, etc. – doivent être stockées dans le répertoire `images`. Dans le document Markdown, on les insère de la façon suivante:

```
![Ceci est une image de chat](images/lolcatz.png)
```

Voir aussi: <https://pandoc.org/MANUAL.html#images> (mais ne pas utiliser les attributs de taille ...)

Un identifiant peut être rajouté pour faire référence à l’image (via un hyperlien) dans le corps du texte:

```
![Ceci est une image de chat](images/lolcatz.png){#lolcatz}
```

Comme le montre `[la photo de mon chat](#lolcatz)`, etc.

---

2. voir en particulier la remarque de Matthew Butterick concernant le manque de visibilité de l’italique dans les fontes sans sérif, qui fait pencher la balance en faveur du gras, d’autant que l’italique a déjà un rôle qui lui est propre.

Pour les images **générées par un programme**, c'est le fichier source de l'image – une description de l'image dans un format plus ou moins exotique (SVG, LaTeX, etc.), ou bien le programme qui la génère (python, etc.) – qui devra être stocké dans **images**, et c'est à lui qu'on devra faire référence dans le document markdown.

Par exemple, si `images/image.tex` contient

```
\documentclass[tikz]{standalone}
\begin{document}
\begin{tikzpicture}
  \draw (0,0) -- (1,1) -- (0,1); % ...
\end{tikzpicture}
\end{document}
```

On peut insérer l'image associée dans un document Markdown par:

```
![Image TikZ] (images/image.tex)
```

**Sous le capot.** Le script de build se charge de la conversion du `.tex` dans des formats images plus comestibles (tels que le format PDF pour la génération de documents PDF) et de la modification du document pour qu'il fasse référence à ces images générées. Ces fichiers générés – qui ne doivent pas être versionnés – sont reconnaissable à leur double extension: `images/image.tex` génère un fichier `images/image.tex.pdf` par exemple.

**Note.** A ce stade seul les fichiers `.tex` et `python` sont supportés. La conversion des fichiers `.tex` en fichiers `.pdf` est prise en charge automatiquement par le script de build. Dans le cas de Python, c'est au programme de tout prendre en charge et de générer le fichier `.pdf` avec le nom adéquat lorsqu'il est appelé. Un exemple de programme de ce type se trouve dans le répertoire `utils/images`.