Instructions pour rédiger et compiler le livre Exo7

Kroum Tzanev

29 décembre 2015

1 Fichiers et répertoires

1.1 Structure générale

Les fichiers `book.tex` et `main.tex` sont les fichiers principaux de deux versions du livre. Ils sont petits et contiennent essentiellement des instructions d'inclusion des chapitres.

Chaque chapitre est dans un répertoire séparé : `zeros`, `algo`, `arithmetique`, `autres`, `complexes`, `courbes`, `crypto`, `derivee`, `determinants`, `dimension`, `dl`, `ensembles`, `equadiff`, `erene`, `ev`, `fonctions`, `groupes`, `int`, `lecons`, `logique`, `matlin`, `matrices`, `polynomes`, `reels`, `suites`, `syslin`, `usuelles`.

La plupart de ces répertoires contiennent un sous-répertoire `figure` qui contient les figures. La majorité des figures sont écrites dans le langage Ti*k*Z/PGF et portent une extension `.tikz`.

1.2 Fichier de styles

Le fichier `exo7book.sty` est le fichier de style principal et peut être évoqué avec l'une des options `[screen]`, `[print]`, `[minimal]` ou `[test]`. En fonction du choix de l'option, l'un des fichiers `exo7book-screen.sty`, `exo7book-print.sty`, `exo7book-minimal.sty` ou `exo7book-test.sty` est évoqué. Et la partie commune du style, qui ne dépend pas du choix de l'option, se trouve dans le fichier `exo7book-common.sty`. Tous ces fichiers sont commentés.

2 Compilation

Les options `[screen]` et `[minimal]` sont compilables avec PDF/Lua/XeLaTeX. Par contre, il faut compiler l'option `[print]` de préférence avec PDFLaTeX.

Vu que le livre contient des centaines d'images, pour accélérer la compilation, les images TikZ sont pré-compilées. Le résultat de cette pré-compilation est stocké dans un sous-répertoire, nommé `tikzcach`, du répertoire de chaque chapitre. Ce sous-répertoire contient pour chaque image deux versions pré-compilées en PDF, une en couleur (les fichiers se terminant par `_col.pdf`) et une en nuances de gris (les fichiers se terminant par `_bw.pdf`). Si pendant la compilation, LaTeX constate que la version pré-compilée existe, il inclut directement le pdf en question et saute le code de la figure.
Ainsi si vous faites un changement dans une figure, pour que ce changement devienne visible il faut supprimer la version pré-compilée de l'image en question avant la compilation.

Pour pouvoir pré-compiler les images il faut autoriser LaTeX à exécuter des commandes extérieures. Pour cela il faut appeler latex avec l'option `-shell-escape` de TeXLive ou `-enable-write18` pour MiKTeX. *Attention : ne jamais compiler une source non vérifiée avec cette option, car elle ouvre la porte aux hackers.*

Vous pouvez compiler les chapitres un par un également. Pour cela il suffit de compiler le fichier `ch_xxxxx.tex` qui se trouve dans le répertoire en question. Ceci est rendu possible grâce à l'utilisation de package `standalone`.

3 Git

`Git` est un logiciel de gestion de versions décentralisé. C'est une version plus moderne de `SVN`, qui est lui-même une version plus moderne de `CSV`. Dans ce chapitre je ne vais pas faire un cours sur `Git`, je vais simplement décrire le «workflow» que j'utilise.

1. Pour rapatrier les sources du livre à partir du serveur git du CNRS :

   ```
   git clone git@git.math.cnrs.fr:plm/bodin/exo7
   ```

   Ceci va créer un sous-répertoire `exo7` qui va contenir toutes les sources, ainsi qu'un sous-répertoire caché `.git`. C'est dans ce répertoire caché que se trouvent les versions du livre.

2. Avant de débuter l'édition sur un des fichiers, il faut récupérer la dernière version des fichiers avec la commande (exécutée dans le répertoire où se trouvent les sources)

   ```
   git pull
   ```

3. Quand vous avez fini une modification d'un fichier existant vous pouvez le mettre «à l'abri» ([ang.] staged area) avec la commande

   ```
   git add -u
   ```

Et si vous avez créé un nouveau fichier (par exemple une image tikz ou un chapitre) vous pouvez le rajouter avec la commande :

`git add nom-du-fichier`

Attention ces deux commandes ne créent pas une nouvelle version dans le dépôt git, elles «préparent» les fichiers qui vont être mis à jour lors de la prochaine version.

4. Quand vous avez terminé avec vos rédactions (souvent sur un seul fichier, mais ça peut être aussi sur plusieurs fichiers) et quand vous avez rajouté ces fichiers avec la commande `git add`, il est temps de créer une nouvelle version dans le dépôt `git` en rajoutant un message décrivant cette version :

   `git commit -m"La description de la version bla, bla"`

   Attention : cette commande ne met pas à jour les fichiers sur le serveur git du CNRS, mais seulement en local dans le répertoire `.git`.

5. A la fin de la journée, ou de la semaine, quand vous estimez que vos modifications doivent être mises à disposition de vos collègues, il faut synchroniser votre dépôt local (votre répertoire .git) avec le serveur cnrs en utilisant la commande

   `git push`

En cas de conflit entre votre version et la version de vos collègues, les choses se compliquent, mais vous pouvez toujours envoyer votre répertoire complet (y compris le sous-répertoire caché .git) à un collègue qui s'y connaît mieux que vous en git pour qu'il résolve le conflit à votre place ;)

3.1 Quelques liens pour Git

— `http://rogerdudler.github.io/git-guide/index.fr.html`
— `http://www.miximum.fr/enfin-comprendre-git.html`
— `http://openclassrooms.com/courses/gerez-vos-codes-source-avec-git`
— `https://git-scm.com/book/fr`

3.2 Quelques commandes avancées bien utiles (Arnaud)

— `git status` permet de voir ce qui est à jour ou pas, indexé ou pas...
— `git ls-files` liste les fichiers indexés
— `git mv`, `git rm` pour manipuler les fichiers en conservant l'indexation git.
— Pour nettoyez son répertoire de tous les fichiers auxilliaires LaTeX (tout .gitignore sauf les pdf) :
  — `git clean -x -e "*.pdf" -n` voir les fichiers que l'on souhaite effacer
  — `git clean -x -e "*.pdf" -f` efface vraiment !