Study Guide: Directory and file structure

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Apr 25, 2015

Directory structure for a book project

Overview of the directory structure

- `doc`: documentation
- `doc/src`: (DocOnce) source for documentation
- `doc/src/chapters`: source for individual chapters
- `doc/src/book`: source for the book
- `doc/pub`: published (compiled) documents in specific formats
- `doc/web`: entry point (`index.html`) for GitHub web pages

Can have lots of chapters and more than one `book` directory if multiple books are relevant

Directory structure within a chapter

Each chapter has a short nickname used in file and directory names.

- Sample nickname: `ch2`
- `ch2`: DocOnce source files `.do.txt`
- `ch2/src-ch2`: source files for program programs, especially those to be copied into `.do.txt` files
- `ch2/fig-ch2`: figures
- `ch2/mov-ch2`: movies
- `ch2/exer-ch2`: answers to exercises

The directories `src-ch2`, `fig-ch2`, `mov-ch2`, and similar may have any subdirectory structure, but the names should as indicated here since the setup for DocOnce books has many tools relying on the naming convention.

The total directory structure

Here is a big project:

```
doc
  src
    chapters
      ch2
        fig-ch2
        src-ch2
        mov-ch2
        exer-ch2
      ch3
        fig-ch3
        src-ch3
        mov-ch3
        exer-ch3
    book1
    book2
  pub
    chapters
      ch2
        html
        pdf
        ipynb
      ch3
        html
        pdf
        ipynb
    book
```

The `book` directory

- One main file for the book: `book.do.txt`
- `book.do.txt` includes chapters from `../chapters/ch2/ch2.do.txt`, etc
- Scripts for compiling, spell checking, ...

Newcommands

- Files `newcommands*.tex` are by DocOnce treated as files with definition of newcommands for LaTeX *mathematics*
- Must be in the directory where `doconce format` is run
- Use a common `newcommands.p.tex` for all chapters
- `.p.tex` indicates that it can be run by preprocess and contain different newcommand definition for LaTeX and MathJax

Example on `doc/chapters/newcommands.p.tex`:

```
%% #if FORMAT in ("latex", "pdflatex")
%% Use footnotesize in subscripts
\newcommand{\subsc}[2]{#1_{\mbox{\footnotesize #2}}}
%% #else
%% In MathJax, a different construction is used
\newcommand{\subsc}[2]{#1_{\small\mbox{#2}}}
%% #endif
```

`make.sh` runs typically

```
preprocess -DFORMAT=pdflatex ../newcommands.p.tex > newcommands.tex
# make latex versions

preprocess -DFORMAT=html ../newcommands.p.tex > newcommands.tex
# make html versions
```

DocOnce newcommands are for mathematics only! Do not use newcommands for general typesetting commands, use Mako functions instead - they work for all output formats.

Assembling different pieces to a book

Organization of DocOnce chapter files

- Chapter nickname: `ch2`
- DocOnce file: `ch2.do.txt`
- `ch2.do.txt` may contain all the text or just include several files: `part1.do.txt`, `part2.do.txt`, `part3.do.txt`
- No title, author, date, table of contents, or bibliography in `ch2.do.txt` - otherwise that file cannot be included in a book

`ch2.do.txt`:

```
# #include "part1.do.txt"

# #include "part2.do.txt"

# #include "part3.do.txt"
```

Compiling a chapter requires a wrapper file with title, author, ...

To compile a stand-alone document for the chapter, create `main_ch2.do.txt`:

```
TITLE: Some chapter title
AUTHOR: A. Name Email:somename@someplace.net at Institute One
AUTHOR: A. Two at Institute One & Institute Two
DATE: today

TOC: on

# Externaldocuments: ../ch3/main_ch3, ../ch4/main_ch4

# #include "ch2.do.txt"

======= References =======

BIBFILE: ../papers.pub
```

Figures and source code

- `fig-ch2`: figures
- `src-ch2`: source code for program files

Important that figure and source code files have chapter-unique names when combining all files to a book. Use `fig-nickname` and `src-nickname`.

Optional directories:

- `mov-ch2`: movies in various formats
- `exer-ch2`: answers to exercises, project work, etc.

Assembly of chapters to a book

Recall: DocOnce = *Document Once, Include Anywhere*

- Make `book.do.txt` for including chapters in a book
- The entire book is in `book.tex` (!)
- Compile individual chapters first - it is easier to track down a latex error in a chapter than going from `book.tex` to the relevant `.do.txt` file

A `book.do.txt` file

```
TITLE: This is a book title
AUTHOR: A. Name Email:somename@someplace.net at Institute One
AUTHOR: A. Two at Institute One & Institute Two
DATE: today
```

```
TOC: on

========= Preface =========
label{ch:preface}

# #include "../chapters/preface/preface.do.txt"

========= Heading of a chapter =========
label{ch:ch2}

# #include "../chapters/ch2/ch2.do.txt"

# Similar inclusion of other chapters

========= Appendix: Heading of an appendix =========
label{ch:somename}

# #include "../chapters/nickname/nickname.do.txt"

======= References =======

BIBFILE: ../papers.pub
```

The book file relies much on running preprocessors

- Sometimes debugging requires you to see the effect of running preprocessors
- The effect of `# #include` and `# #if` tests are seen in `tmp_preprocess__book.do.txt` (input to `mako`)
- The effect of Mako variables and functions are seen in `tmp_mako__book.do.txt` (input to `doconce` translation)

The `book` directory must be coupled to source and figure directories

Running

Terminal

```
Terminal> doconce format pdflatex book [options]
```

will most likely involve constructs like

```
@@@CODE src-ch2/myprog.py  fromto: def test1@def test2
```

but `pdflatex` sees no `book/src-ch2` directory! A local link resolves the problem:

Terminal

```
Terminal> ln -s ../chapters/ch2/src-ch2 src-ch2
```

Similar problems for figures!

Terminal

```
Terminal> ln -s ../chapters/ch2/fig-ch2 fig-ch2
```

Auto-generation of all links (if `chapters=` is set correctly in `scripts.py`):

```
>>> import scripts
>>> scripts.make_links()
```

About figures when publishing HTML

- HTML versions of chapters/book have `<img src=fig-ch2/fig1.png>` type of tags in the HTML code
- There must be a `fig-ch2` subdirectory
- *Copy* `chapters/ch2/fig-ch2` to the directory where the HTML files are published

Tools

Making a new chapter

- Clone https://github.com/hplgit/setup4book-doconce
- Go to `doc/chapters`
- Decide on a chapter nickname
- Create a brand new chapter: `bash mkdir.sh nickname`
- Look at an existing chapter like `rules` to see syntax/details
- Start writing in `nickname/*.do.txt` files, programs in `src-nickname`, place figures in `fig-nickname`
- Edit `make.sh` if necessary
- Compile chapter: `bash make.sh`
- Include `../chapters/nickname/nickname.do.txt` with chapter heading in `doc/src/book/book.doc.txt`

Compiling a chapter

To LaTeX/PDF:

Terminal

```
Terminal> bash make.sh
```

To HTML:

Terminal

```
Terminal> bash ../make_html.sh main_nickname.do.txt
```

The `doc/src/chapters/make.sh` script is quite general and can be edited according to your layout preferences of the LaTeX documents.

There is also a script `doc/src/chapters/make_html.sh` for making HTML versions of the chapter. Just call this as

Terminal

```
Terminal> bash ../make_html.sh main_mychap
```

Three HTML versions with an `index.html` table of contents are generated

Compiling the book

Go to `book` directory and produce LaTeX/PDF book by

Terminal

```
Terminal> bash make.sh
Terminal> bash make.sh nospell   # turn off spell checking
```

About book styles and tools:

- Current example employs the Springer T2 book layout
- DocOnce supports some other styles and, in general, a user-specific template for the preamble
- `scripts.py` has function `pack_src` for packing all the `src-*` directories in a tarfile for book readers
- `pack_Springer.sh` packs all needed LaTeX book files for publishing with Springer

Study guides and slides

- Study guide: presentation of a chapter in a very condensed, effective, summarizing form for overview, use in lectures, and repetition
- Slides: a good way of writing study guides

Why is it so challenging to convert a chapter to slides?

Balance:

- enough information for self-study by *reading* the study guide
- minimized information for *viewing* and *listening* to an oral presentations of the slides

Many iterations and use of slides in teaching over and over again are needed!

Slide directory

For a chapter `ch2`,

- let slides be in `slides-ch2`
- a chapter file `ch2/part1.do.txt` has its slide counterpart in `ch2/slides-ch2/part1.do.txt`
- `ch2/lectures_ch2.do.txt` includes all relevant `slides-ch2/*` files and is the main file for the slide collection

Compile slides:

Terminal

```
Terminal> bash ../make_lectures.sh lectures_ch2.do.txt
```

Note the possibility to turn the `TOC` on and off: Beamer has its own table of contents, while HTML5 slides may benefit from having one

The `lectures_ch2.do.txt` file

```
TITLE: Study Guide: Some title
AUTHOR: Author Name Email:somename@someplace.net at Institute One
DATE: today

# #ifdef WITH_TOC
!split
TOC: on
# #endif

# #include "lec-ch2/part1.do.txt"

# #include "lec-ch2/part2.do.txt"

# #include "lec-ch2/part3.do.txt"
```

The requirements to a slide collection

Tree purposes:

1. Read as a study guide to get overview before reading the full text of chapter
2. Watch as slides during an oral presentation
3. Read as a study guide to repeat and enforce overview of the material

Rules:

- Make slides self-contained
- Limit the information on the slides!
- Make slides as visual as feasible
- Use references to the underlying chapter text for details