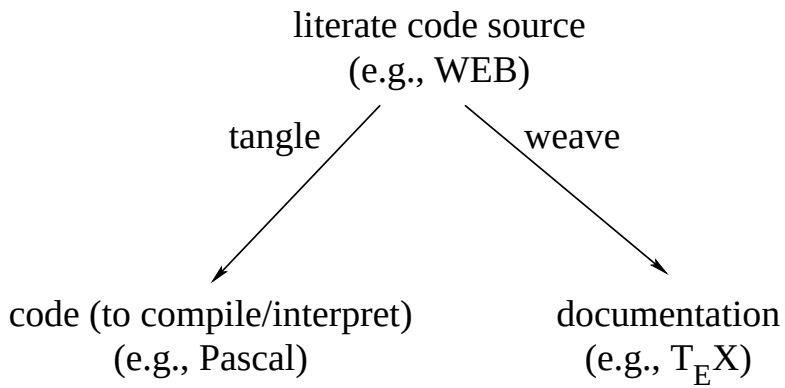
Literate programming, Matlab & finite element method

David Stewart

September 10, 2012

Literate programming

- Originally developed by Donald Knuth while creating TEX
- Combines code (e.g., Pascal) and documentation (e.g., TEX/LATEX)
- Knuth's system was called WEB



NoWeb

- NoWeb is a simple version of this designed for any kind of source code
- Unlike WEB, there is no prettyprinting of the source code
- Documentation is in TEX/LATEX

What NoWeb looks like

Source (NoWeb):

```
For the PDE $-\Delta u=f(\mathbf{x})$ with
$f(\mathbf{x})=x_{1}^{2}\,\exp(x_{2})$, we use:
<<pde-struct-eg>>=
pde = struct('coeffs',@(x)diag([0,1,1]), ...
        'rhs',@(x)[x(1)^2*exp(x(2));0;0],'order',1)
@
```

What is looks like in the documentation:

For the PDE $-\Delta u = f(\mathbf{x})$ with $f(\mathbf{x}) = x_1^2 \exp(x_2)$, we use:

5a ⟨*pde-struct-eg* 5a⟩≡

```
pde = struct('coeffs',@(x)diag([0,1,1]), ...
             'rhs',@(x)[x(1)^2*exp(x(2));0;0],'order',1)
```

In the code:

```
pde = struct('coeffs',@(x)diag([0,1,1]), ...
             'rhs',@(x)[x(1)^2*exp(x(2));0;0],'order',1)
```

More Noweb...

The basic syntax for the chunks of code is

```
<<name-of-code>>=
code chunk goes here
it can go for many lines
but it ends with ‘‘@’’
@
```

Actually “@” indicates the start of normal LaTeX. You can have several different chunks one after the other:

```
<<chunk1>>=
This is chunk 1.
<<chunk2>>=
This is chunk 2.
@
```

Chunks can refer to other chunks.

```
<<my-code.txt>>=
This is a collection of chunks, strung together.
1st: <<chunk1>>
2nd: <<chunk2>>
@
```

If a chunk is “defined” multiple times, just concatenate the chunks to form one large chunk:

```
<<chunk3>>=
This is yet another chunk.
<<my-code.txt>>=
3rd: <<chunk3>>
@
```

To generate code "`chunks`" from NoWeb file:

```
notangle -Rmy-code.txt source-file.nw > my-code.txt
```

Note that "`my-code.txt`" is the name of the root code chunk that is generated. A NoWeb file can have many root chunks.

Using with LyX

LyX is a WYSIWYM LaTeX-based editor.

LyX has a large number of different styles (all from LaTeX) with convenient ways of handling cross-references, indexes, bibliographies, structured documents, graphics, and can provide output in multiple formats (DVI, PS, PDF, text, LaTeX, OpenDocument) and NoWeb.

The NoWeb export option produces a "`.nw`" file.

Pros & Cons of literate programming

Pros:

- Documentation and code are together.
- Recursive use of code chunks encourages a top-down ("stepwise refinement") method of software development.
- Not only can you have the formulas with the code, but also the *derivation* of the formulas can be kept with the code.
- Graphics can be included in the documentation easily (unlike documentation generated from source code).
- There is a single source file (for everything!)

Cons:

- The run-debug-edit cycle is more complex.
- Recursive use of code chunks encourages a top-down ("stepwise refinement") method of software development.
- The documents tend to be rather large.
- There is more work involved.
- Not suitable for very large scale software development.

Pros and Cons of NoWeb

Pros:

- Language independent — can include scripts, Makefiles, and text files for data or output
- Simple syntax
- Integrated into LyX
- Multi-platform (Unix/Linux, Mac, Windows under Cygwin)

Cons:

- Not maintained
- Main implementation is based on the Icon programming language
- Installation could be simpler
- Does not allow underscore "_" in chunk/file names

Tricks to make things easier

Scripts and Makefiles can reduce the effort needed to use NoWeb: Makefiles can be generated in NoWeb that will create all files desired from the NoWeb file:

```
notangle -t8 -RMakefile my-noweb-file.nw > Makefile
make all
```

You need to add some items to your NoWeb file to do this. For each root chunk for a file, add a `filelist` entry (only needed once for each file):

```
<<filelist>>=
new-file.txt \
<<new-file.txt>>=
....
```

A `Makefile` should be included:

```
<<Makefile>>=
files1 = <<filelist>>
files2 = my-noweb-file.tex filelist
files = $(files1) $(files2)
source = my-noweb-file.nw
all: $(files)
$(files): $(source)
    notangle -R$@ $(source) > $@
my-noweb-file.tex: $(source)
    noweave -delay -index $(source) > $@
@
```

Matlab likes underscores in file names for "`.m`" files, but can't use dashes "–".

So all "`.m`" filenames in NoWeb are written with dashes, which are turned into underscores later by a script.

There is a further script that combines all these to give a one-line command to create the code and documentation files.

In spite of these complications, the system works reasonably well.