Literate programming with Emacs and Org-mode

Table of Contents

1. README

- [ ] This file introduces literate programming using the GNU Emacs editor and Org-mode as IDE (Integrated Development Environment)[1].
- [ ] There is a much longer, more detailed version with more information on installation, customization and background

2. What you will learn:

- What is literate programming
- How to tangle code and weave documentation
- Understanding Org-mode code blocks
- What will you have to know and do in this course

3. Setup

To make sure that you can run the Org-mode C code blocks, type `CTRL-c` twice (written `C-c C-c`) with the cursor anywhere on the code block "1":

```
puts("Yes, it works.");
```

```
Yes, it works.
```

Anything else but the output `Yes, it works` spells trouble:

- Check that you have a `.emacs` file in `$HOME`.
- Check that the `.emacs` file contains these lines:

  ```
  (org-babel-do-load-languages
  'org-babel-load-languages '((C . t)))
  ```

- Check that you have GCC or another C compiler installed by entering in a terminal: `gcc --version`. This is what I get:

```
pi@raspberrypi:~$ gcc --version
gcc (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
```

- Check the `*Messages*` buffer of this Emacs session for error messages

4. What is literate programming

- A programming *paradigm* that produces programs both for humans and for machines, invented by Donald Knuth since 1984.

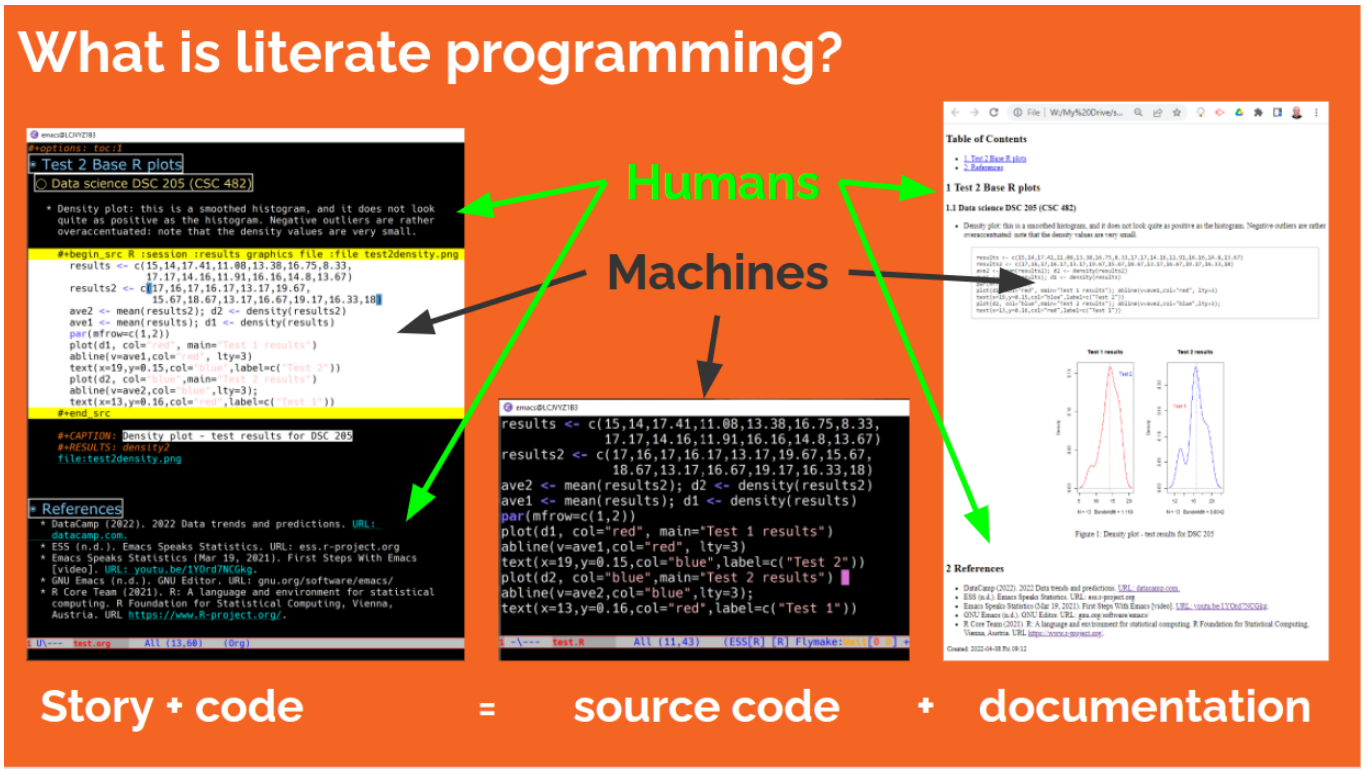

Figure 1: What is literate programming? (2022)

- In my field, data science, literate programs in the form of interactive notebooks are the dominant development medium[2].
- Tangle code or weave documentation
- This GNU Emacs Org-mode file is an example. It contains both code (see 1) and documentation.
- You can infer the fact that it's an Org-mode file from the file ending `.org`
- The code in this file can be *tangled* into a C source code file, and it can be *woven* into documentation (in different formats).
- GNU Emacs commands can usually either be entered in a long version (as a function), or in a short version (as a keyboard sequence)[3].
- To tangle code blocks, type `C-c C-v t` or `M-x org-babel-tangle` where `C-` stands for `CTRL`, and `M` (meta key) stands for `ALT`.

- To weave file documentation, type `C-c C-e` to get to the `*Org Export Dispatcher*` where you can choose your format.

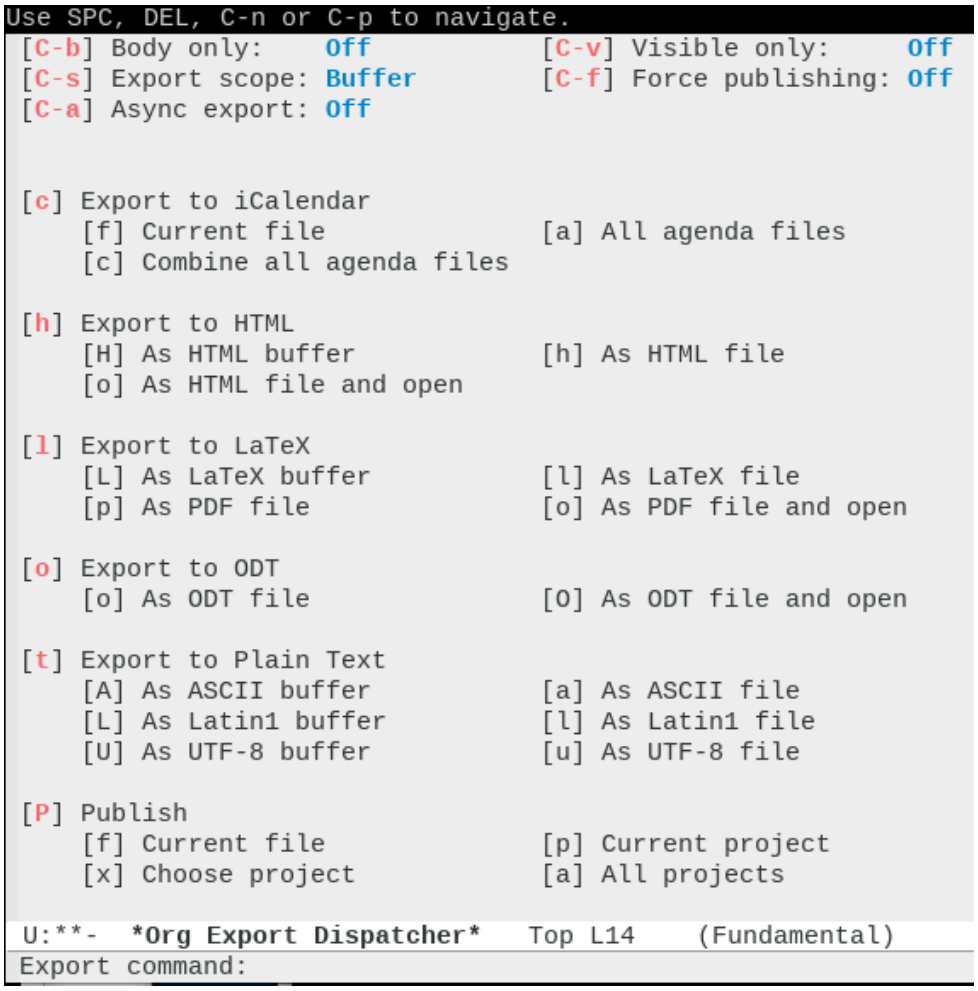

Figure 2: The **Org Export Dispatcher** menu

5. Code blocks explained

- A code block can be executed in a given language, for example C[4]. "Execution" in the case of a compiled language like C includes several steps:
    1. parse the code ("read it")
    2. put the code into a C file
    3. compile and link the C file into a binary executable
    4. execute the binary file
- So instead of going back and forth between file and command line, you only need one command (`C-c C-c`) to do it all in one go.
- Anything between the meta characters `#+begin_src` and `#+end_src` is executed by Emacs. The file must be an Org-mode file, i.e. it must end in `.org`.
- You can pass arguments to Emacs that are used during the execution, e.g. `:main yes` to wrap the code in a `int main ()` C function, or `:includes <stdio.h>` to include the `stdio.h` header file[5].
- If you want to tangle the code, you need to add `:tangle filename.c` to the header - this leads to a C source code file `filename.c`.
- Code blocks should be named (so that you can link to them) using the meta characters `#+name:` . E.g. `#+name: blck` can be linked to from anywhere using double square brackets: `[[blck]]`.

- A simple example that upon execution (`C-c C-c`) will print `hello there`:

```
puts("hello there");
```

```
hello there
```

- This code block will only execute if the `#+PROPERTY` is set properly
- An example with more arguments that also asks for input from the file `input` that will print the letter in `input` ('A'):

```
char c;
scanf("%c", &c);
printf("%c", c);
```

```
A
```

- This code block will only execute, if the file `../src/input` exists. Let's check:

```
cat ../src/input
```

6. Further study

1. Reading:
   - Getting Started with Emacs: A Beginner's Guide (Prevos, 2021)
   - Getting started with Emacs (Kenlon, 2020)
2. Viewing:
   - The Absolute Beginner's Guide to Emacs (Wilson, 2021)
   - My notes on the video on GitHub (Birkenkrahe, 2022)
3. Installing:
   - Emacs download for Windows, MacOS or Linux (GNU)
   - Preconfigured Emacs for data science for Windows/MacOS

7. Let's practice

GNU Emacs practice includes two steps:

1. `[ ]`

   Completing the GNU Emacs on-board tutorial - this will enable you to use the editor with ease. To open it type first:

   ```
   $  emacs -nw
   ```

   Emacs should open in the terminal (no graphics). Now type `C-h t` or `M-x help-with-tutorial` and follow the instructions all the way to the end. This will take about 1 hour.

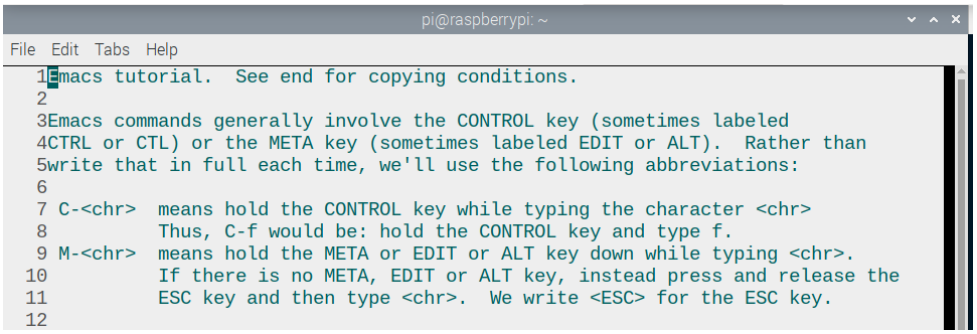

Figure 3: First lines of the Emacs tutorial

2. [ ]

   Completing a few simple practice exercises. Download practice.org using `wget` on the command line:

   ```
   $  wget tinyurl.com/3tjvnws8 -O practice.org -o log
   ```

3. [ ]

   Download the GNU Emacs initialization file .emacs using `wget` on the command line:

   ```
   $  wget tinyurl.com/447rjn2x -O ~/.emacs -o log
   ```

4. Open the practice file with Emacs from the command line (in the same directory where you downloaded it to - probably `Downloads`:

   ```
   $  emacs --name practice --file practice.org &
   ```

   - This command loads the file following the `--file` flag, and pushes the process (Emacs) into the background so that you can keep using the terminal and don't have to open a new one. A new window named `practice` opens. You can also find it in the task bar at the top of the screen.
5. If all goes well, you see the file in an Emacs buffer window. You can open headlines, code blocks and bullet points by typing TAB when the cursor is on the headline. Some examples:
   - headline that goes over three lines
   - named code block:

     ```
     // a C statement
     int i = 1;
     // nothing to see here
     printf("%d\n", i);
     ```

6. [ ] Complete the online exercises, then submit your completed Org-mode file **practice.org** in Schoology .

8. Summary

- Code is often developed using special software (IDE) like Emacs
- Literate programming is a technique to develop programs for both human and machine consumption

- GNU Emacs is a self-extensible text editor
- Org-mode is a major Emacs mode for literate programming
- Literate programming includes tangling and weaving
- Meta information controls layout via macros

cc-glossary

Footnotes:

[1] I introduced literate programming as a teaching and learning technique only in spring 2022, see this presentation given at Lyon College on April 8, 2022 (research paper in preparation). I was inspired to do this by Daniel German's talk at EmacsConf 2021, "Using Org-mode to teach programming".

[2] Examples are: Jupyter notebooks, Google Colaboratory, RStudio Notebooks, or Kaggle.

[3] Emacs is a self-extensible editor - this means that you can completely reprogram it. Imagine you could do that with WORD to create exactly the text editor that you need and like.

[4] Many other languages are supported, too.

[5] The header arguments can also be defined for the entire file with more than one code block using the `#+PROPERTY` meta characters. See the top of this file for an example (for C).

Author: Marcus Birkenkrahe
Created: 2022-05-20 Fri 20:27