Unix source code editors

Zdeněk Žabokrtský

October 14, 2020




Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

unless otherwise stated

Motivation

- As an NLP developer, sooner or later you'll need to be able to work in a UNIX system.
- As a developer in UNIX, you simply need some text editor for coding.
- Actually your coding editor is likely to be one of your most frequently used tool, with an enormous effect on your overall productivity.
- Thus it's crucial for you to master (at least) one text editor in UNIX.
- In a long term view, you'll have to learn a new programming languages every now and then...
- ...but (if you are lucky) you may stick to a single editor for quite a long time.

Command line vs. GUI

- What's specific for UNIX-like style of work is that often you have to work in a text-console regime, and there's no GUI available to you.
- It's a feature! Among other advantages, it allows you controlling remote machines efficiently.
- Implication: you need to learn at least one editor that supports the text-console regime.

Command line vs. GUI, cont.

- So you must be able to work in a text-console regime...
- ...but maybe you find working in a GUI editor more comfortable and more efficient for your main code production.
- Options:
  - either you master an editor that supports both text-console and GUI regime,
  - or you choose a GUI editor for your main production, and you learn at least some basics of some other text-console editor.
  - Working in a text-regime editor all the time is also an option, though perhaps not that attractive.

Minimum requirements on a modern source-code editor

- What we find really essential for an editor to support:
  - modes for programming and markup languages (xml, html...)
  - syntax highlighting
  - completion
  - indentation
  - support for character encodings (utf-8)
- other potentially useful features:
  - integration with a compiler
  - integration with a versioning system
  - ...

Overview of text editors for coding in UNIX

- There are perhaps hunreds of source-code editors existing in UNIX systems, of of which a dozen or two is used by quite big communities.
- Informally, they can be grouped as follows:
  - "good old" editors such as `vi` and `emacs`
    - support both text-console and GUI regimes
    - especially `vi` available virtually everywhere
    - extremely universal and configurable
    - a bit harder to digest for newcommers (non typical keyboard shortcuts)
  - GUI-only Integrated Development Environments such as `sublime` or `atom`
  - GUI-only simple editors such as `kate` or `gedit`
  - text-regime-only simple editors such as `nano`, `ed` or `joe`

So which editor should one choose?

Our (highly subjective) recommendation:

- IF your unix working environment is already stabilized AND your favourite editor fulfills all the minimum requirements above AND your are able to edit a file in a text console regime too (possibly using some other editor) THEN no need for a change.
- ELSIF you are going to work in a unix environment on a regular basis, THEN: master a full-fledged modern GUI editor such as `atom` and use a simple text-console editor such as `nano` as your secondary editor.
- ELSE: combine using some simple GUI editor (e.g. `kate`) and some simple text-console editor (e.g. `nano`)

Last resorts if everything goes wrong

What if I am ssh-ed to a remote computer on which none of the editors I can work with is installed:

- last resort 1, if all you need is just a quick tiny change of a single file (e.g. a config file): you can "edit" the file in-place using text-processing bash commands such as `sed`
- last resort 2, if more changes are needed: edit the file in your computer and use `scp` to move to the remove computer.

Final remarks

- The main take-home message: it is really important to master at least one source-code editor (to master $\equiv$ to learn to use its modes, to memorize keyboard shortcuts for all frequently needed operations...).
- Disappointed because of getting no ultimate answers as for the editor choice? It's because there is no ultimate winner indeed. For instance, the "war of editors" between `vi` and `emacs` users started more than 3 decades ago and new jokes are still being invented.