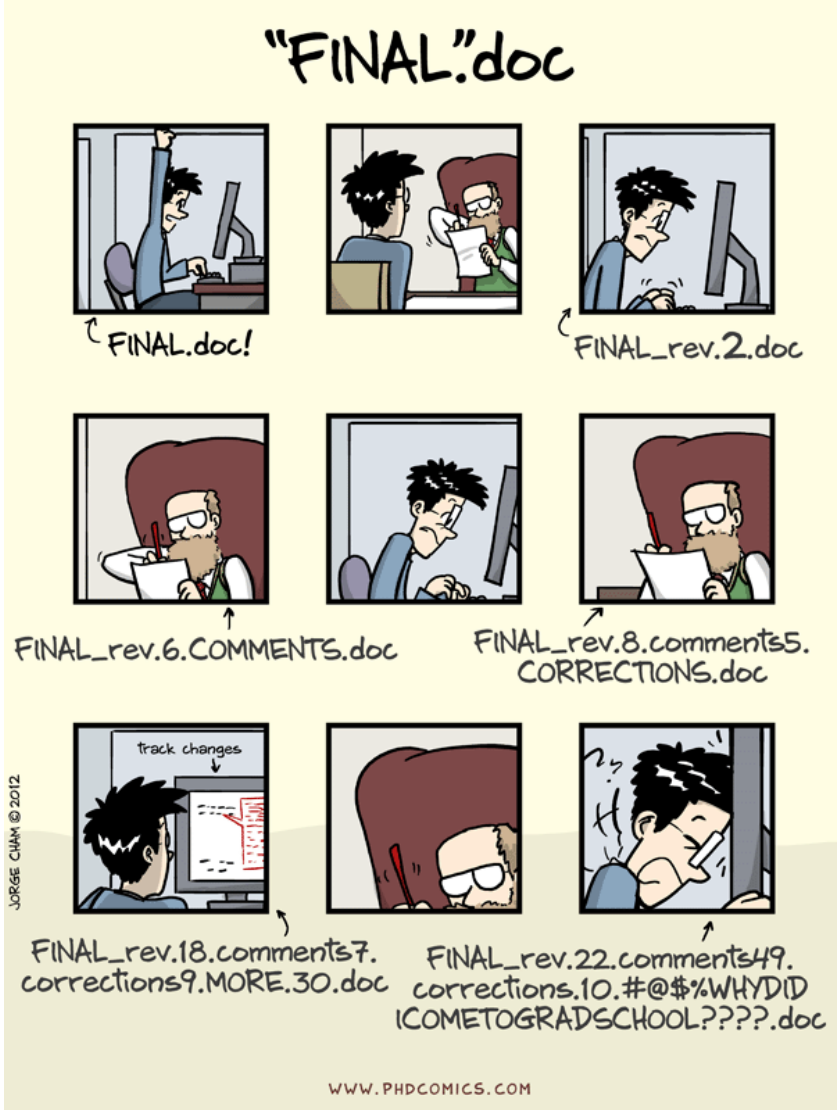
Source / Version Control
Version control systems (`cvs`, `svn`, `git`)
Version control hosts (bitbucket, github)

Thomas Erben, Argelander-Institut für Astronomie

Python lecture on 24/07/2017

The context - Why Version Control?



What is Version Control?

Program development cycle:
first revision $\rightarrow$ bug fixes (second revision) $\rightarrow$ extensions (third revision) $\rightarrow$ change of numerical methods (fourth revision) $\rightarrow$ bug fixes .....

Typical sitouations to handle:

- I have created some data/made some simulations. With what revision of my program did I create this?
- Last year my program gave some result. With the current version I get something different. What exactly did I change?
- I need to regenerate a product from last year!
- I want to change my program but I would like to do this with an old revision.

When you are taking measures to meet these sitouations you are doing Version Control. `cvs`, `svn` or `git` can help you with that.

Some terminology

- Version control
- Version control system: `cvs`, `svn`, `git`. For one-person projects it does not matter too much which one you use. For collaboration efforts `git` is definitely the best!
- repository: `cvs` and `svn` store the codes in special directories on your computers. You can tar these and ship them around.
- Version control host: `git` is optimised for Web based usage and cloud services (`github` and `bitbucket`). Cloud services are typically free for public projects!

Poor mans version control

```
user$ ls
calc.c    main.c    README
user$ mkdir safe_29082013
user$ cp *.c README safe_29082013
```

Idea:

Create a snapshot from a stable, well-tested program version (before an important production run; for release purposes; .....)

This is an *important* aspect of Version Vontrol and its most basic idea!

The poor mans approach becomes unfeasible for larger projects, to keep a *detailed* history of your development, to easily find code differences between 01/02/2005 and 29/10/2012, ...!

Work-cycle under Version Control

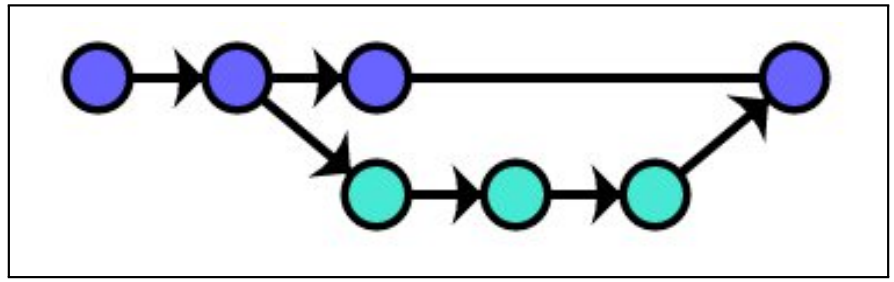
Examples for `cvs` with single-user development.

1. You put existing source code under a VCS (`cvs` or `svn` repository, `github` upload)
2. check out the code from the VCS
3. modify code, check differences, create branches, tag versions, ...
4. check a new revision into the VCS (*commit*; develop a habit on *when* to do this!)
5. Go back to 3 or check out code on another system ....

More steps are necessary for collaboration work under a VCS (e.g. verify whether a collaborator checked in something new etc.).
There is an additional layer for web-based VCS hosting systens (`github`; local and remote checkins!).

What are branches?

You often would like to *branch off* from your main development trunk:



- You want to branch off production code from new developments.
- You want to create a *sandbox* for new developments
- You want to provide bug fixes for old releases.
- Branching is great to allow code verification *before* merging new developments to the (stable) main trunk.
- Branching is the way to contribute to open source projects!

Branches are often *merged* to the main development trunk after some time. Also ideas that do not work out just can be deleted without harming the main branch!

Further Characteristics of Version Control Outlook

You should put ALL your codes and code related documents under VCS!

- You can get version/revision information into your source codes
- It enforces a certain degree of documentation (commit messages)
- It can give free secure backup of source codes in addition (cloud services such as `github` and `bitbucket`; `github` gives free private repositories to academia!)
- It allows great code-writing in collaborations (private projects, Open Source projects; `bitbucket`, `github`); No need to work in a lock-file mode; Version Control does not substitute communication!
- It is useful for *all* kind of source code, e.g. also for LaTeX files (thesis, science-papers etc.)
- YOU DETERMINE THE DEGREE OF CONTROL. You need to develop habits for it!