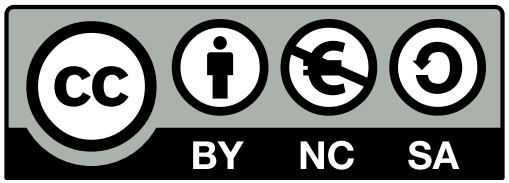
Herramientas de Programación en Bioquímica y Biología Molecular: Diff, backups, version control

Ramón Díaz-Uriarte

October-2016

(Rev: 84fa67a)

License and copyright



This work is Copyright, ©, 2015, Ramón Díaz-Uriarte, and is licensed under the **Creative Commons** Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit

http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

****

Please, **respect the copyright and license**. This material is provided freely. If you use it, I only ask that you use it according to the (very permissive) terms of the license: acknowledging the author, not making money from copies or derivatives, and redistributing copies and derivatives under the same license. If you have any doubts, ask me.

Outline

(Why?)

- We need to show this.
- Not part of the exercises, but remember when you are out there alone in the world.

Diffs on moderately sized files

- A possible scenario: I found a bug in a package I am writing. Where is it?

- We will see different ways of finding **diff**s.
- (In a sense, all versions of `diff`)
- (We will show an example with kdiff3)

Diffs on moderately sized directories

- A possible scenario: I found a bug in a package I am writing. Where is it?
- I want to recursively look over **all** files in **all** directories.
- (We will show an example with kdiff3)

Other ways?

- There are other programs.
- I'd use a combination of kdiff3 and Emacs (which has ediff).
- Your editor might include facilities for diffs.
- So ...

Summary of comparing differences between files

- **kdiff3**
- **Emacs** with ediff mode
- jEdit with JDiffPlugin (only shows lines that differ)
- Sublime Text, Kate, etc, also have plugins or ways for using diffs.
- Similar things for Macs.
- Other editors (that allow window splitting): Eclipse
- (There are a variety of other scripts and programs, but few highlight differences within the lines that differ)
- You can compare three files.
- You can **merge** changes.
- You might even be able to edit **at the same time**.

A trivial example of merging

(As it says, show a trivial example of merging)

When is diff useful?

- Figuring out why two or more files differ.
- So not just limited to programs: any text file can be processed.
- (What about non-text?)
- Comparing programs: v1 works and v2 doesn't. Why?
- Comparing output: you write two or more similar programs, and you want to know where/when the programs start differing. (I end up doing this a lot from time to time)
- What are the differences between two or more directories.
- A standard way to provide patches to software.
- . . . which brings us to versions and version control.

Backups

Always keep backups of things that matter to you.

Loosing large amounts of data/work is not acceptable.

Backups

- Regular (better if automatic)
- Do not keep backup in the same hard drive as the original!!!!
- Keep backup in a different, separate, physical location (theft, fire, etc).
- Always test your backups work and you'll know what to do when disaster strikes (it is **when**, not **if**).

Backups: miscell ideas

- Pen drives, CDs, external hard drives
- Cloud et al.
- Most of those are a “case-by-case” solution.
- You want something automatic if possible.

Backups: terminology and procedures

- **Full backup**
- **Differential backup**: anything changed since last full backup.
- **Incremental backup**: anything changed since last backup.
- Trade-offs with each: speed and what you need to keep around to restore.
- **Links to decrease storage**: specially useful if repeated files and multiple machines.
- Beware of confidentiality issues and encrypt files as required.

Backups: some free software

- **backuppc**: http://backuppc.sourceforge.net/ Industrial strength solution, slightly hard to configure, excellent performance, easy to backup to remote locations. All OSs (Windows with some difficulty?)
- **Areca**: http://www.areca-backup.org/ Java-based (all OSs)
- **Back In Time**: http://backintime.le-web.org/. Linux only.
- ...
- Macs have their own solution (but local HD?)

Motivating version control

- In the beginning: program.py, program-v1.py, program-v2.py
- . . .
- Do I need to keep all that around?
- What if several files?
- What if several people?
- Version control, revision control, source code control.
- (The paper “So you want to be a computational biologist”, Loman and Watson, 2013, mentions this: p. 997).

Version control

- Essential for medium to large programming projects
- Absolutely essential if several people in same project
- Absolutely essential if projects lasts a decent amount of time
- Easily go back to previous versions
- Tailored for text files: **code**.
- Many things you can download you download from repositories under version control

Version control vs. backup

- Fine control
- Diffs
- Tags and comments
- Explicit commits
- Explicit what is under version control (e.g., binaries are rarely under version control, because you generate them from the sources)
- Really simple to provide unlimited read access (anybody can **check out**) but limited write access (only some can **commit** changes).

Version control: a couple of examples

- Local
  - From Emacs, with magit
  - With `gitk`
- On the web
  - github as an example (there is also bitbucket)

Version control: terms in a typical day

- There is a **repository** where files are stored.
- **Check out** files from the repository into your machine
- You explicitly **add new files** (or rename or move, etc)
- You make changes to the files. Other collaborators make changes.
- When you are satisfied, **commit** your changes. This creates a new **revision**.
- Modifications are **merged**.
- Sometimes **conflicts** arise. These need to be **resolved**.
- What did you do exactly? See the **diffs**
- A textual description of the major changes? Check the **logs**
- By the way, there can be several **branches** (e.g., stable, where only bug fixes are committed, and development).

What to put under version control

- Software projects
- Anything you keep as plain text
  - Agendas/TODOs
  - Phone books
  - Papers you write
  - Recipes
  - ...

Version control: some systems

- Centralized (or client-server)
  - cvs
  - **svn**
- Distributed
  - **git**
  - **Mercurial**
  - bazaar
  - Darcs

(command line, but there are GUIs for most)

Version control vs. dropbox, wuala, etc

- Much finer control
- Can combine both?
  - Short answer: don't.
  - Long answer: yes, but you have to be **very, very** careful. Do you really need to do this?

A suggestion? Use git

- What most new projects tend to use
- Very good documentation:
  - *Pro Git* (freely available)
  - Basically any question with answers in stackoverflow, etc
  - *Git pocket guide*
- github, bitbucket

Version control

- If your programs are more than 10 lines
- if your projects include several files
- if your projects expand over a long time (*gt* a week)
- if multiple people contribute
- **Definitely consider version control**
- An initial investment (half a day?) that pays off

Encryption

- Surprinsingly simple to recover information from deleted files.
- Encrypt specific files/directories (e.g., `eCryptfs, EncFS`)
- Full-disk encryption (e.g., `dm-crypt + LUKS`)
- The **TrueCrypt** issue.
- Beware of swap partitions, temporary files, etc.
- Beware of backups.
- Good intro: https://wiki.archlinux.org/index.php/Disk_encryption
- And of course, use decent passwords. (Google for "password check" and use common sense).
- What are you trying to protect against?