Logiciel Libre
TP 1 — Project Presentation

Stefano Zacchiroli
zack@pps.univ-paris-diderot.fr

Laboratoire PPS, Université Paris Diderot

2014–2015

Evaluation — reminder

1. exam
2. TD
   - exercises
   - short essays on selected topics, including external speaker interventions
3. project (mandatory, not CC)

Final note

- 1e session: 50% exam + 50% project
- 2e session: 50% exam + 50% project

Project — principle

Relevant code contribution
to a relevant, existing, Free Software project.

Project — principle

Relevant code contribution
to a relevant, existing, Free Software project.

- ... code contribution

Project — principle

> Relevant code contribution
> to a relevant, existing, Free Software project.

- relevant. . .
- . . . code contribution

Project — principle

> Relevant code contribution
> to a relevant, existing, Free Software project.

- relevant. . .
- . . . code contribution

- . . . existing. . .

Project — principle

> Relevant code contribution
> to a relevant, existing, Free Software project.

- relevant. . .
- . . . code contribution

- to a relevant. . .
- . . . existing. . .

Project — principle

> Relevant code contribution
> to a relevant, existing, Free Software project.

- relevant. . .
- . . . code contribution

- to a relevant. . .
- . . . existing. . .
- . . . Free Software project

Project — meta

- work individually or in pairs (*binômes*)
- no exceptions granted for groups > 2

Step 1 — find a project you're excited about

- *any* Free Software project you like
  - (obviously: must be licensed under a FOSS license)

- try to find a match between your skills and the technologies used by the project (prog. language, framework, tools, etc.)

Step 1 — find a project you're excited about

- *any* Free Software project you like
  - (obviously: must be licensed under a FOSS license)

- try to find a match between your skills and the technologies used by the project (prog. language, framework, tools, etc.)

- tip (*non* mandatory): choose a project that maintains a list of "easy hacks", "bugs for newcomers", etc.

- I've collected a list at
  https://openhatch.org/wiki/Easy_bugs_for_newcomers
  (link on the course page)

Example

Debian, Django, Fedora, GNOME, KDE, LibreOffice, Linux Kernel, Mediawiki, Mozilla, Mozilla, OpenMRS, OpenOffice, OpenStack, Python, Snowdrift, Ubuntu, VLC, . . .

Step 2 — learn the basics of the project

- how to obtain the code (e.g., VCS) of the current development version of the project
- how to build (dependencies, build instruction, README, etc.)
- how to install on your machine
- how to run on your machine

Step 2 — learn the basics of the project

- how to obtain the code (e.g., VCS) of the current development version of the project
- how to build (dependencies, build instruction, README, etc.)
- how to install on your machine
- how to run on your machine

- look for introductory development documentation

Step 2 — learn the basics of the project

- how to obtain the code (e.g., VCS) of the current development version of the project
- how to build (dependencies, build instruction, README, etc.)
- how to install on your machine
- how to run on your machine

- look for introductory development documentation

- try to make a tiny teeny modification
  - ▸ e.g., change a label in a UI element, print a debug message at some point, ...

Step 3 — look at pending issues

- Bug Tracking System of the project
- list of "easy hacks" maintained by the project (see before)
- feature requests by users (tip: only consider those that have been acknowledged by developers)
- "roadmap" documents (more difficult)

- try to reproduce bugs: do they affect your build of the development version too?

- pick a task

Step 4 — agree with me on a task

- without preliminary agreement with me, your task will not constitute a valid project

Step 5 — get the job done locally

- fix the bug / implement the missing feature / etc.

that's the easy part :-)

Step 6 — get your changes accepted upstream

- produce a patch / pull request / whatever is customary for the project you've chosen
- inform the developers of your work, attaching your changes
- convince them that your code is good and worth being integrated ("upstream") as part of the project code
- this might require several iterations during which you improve your code
- remember: upstream developers have the last word, it's your responsibility to convince them

Step 6 — get your changes accepted upstream

- produce a patch / pull request / whatever is customary for the project you've chosen
- inform the developers of your work, attaching your changes
- convince them that your code is good and worth being integrated ("upstream") as part of the project code
- this might require several iterations during which you improve your code
- remember: upstream developers have the last word, it's your responsibility to convince them

- if you hit a wall, go back to step 1 and pick another project/task
- (hence, better start waaaay in advance)

Interlude — exclusion criteria

- do not use the fact this a university assignment as argument to convince upstream developers
- it will not work (in fact: it will likely work *against* you)
- and if I find evidence of it, e.g.:

    ~~"pour le cours de Logiciel Libre de M. Zacchiroli je dois contribuer un patch à votre projet, pourriez vous m'aider s'ils vous plaît?"~~

    your project note will be 0

- your *code* should be convincing, not your student status

Step 7 — write a report

Length: 10 pages max.

The structure of the report is up to you, but it must contain at least the following information:

- background: chosen project, reasons for that choice
- technicalities: short technical description of the chosen project (programming lang., frameworks, tool, etc.)
- background: chosen task, reasons for that choice
- technical description of your work: what have you done? how?
- references to *public* evidence of your work: commits, patches sent to Bug Tracking Systems, mailing list threads, etc.
- evidence of the fact your work has been integrated upstream
- discussion: difficulties encountered, feedback, etc.

Supervision

- there will be dedicated TP sessions to work on the project
- I'm available throughout the full process (both during dedicated TPs and outside of them) to give you feedback, hints, and advice for your specific project

?