Introduction aux Logiciels Libres – TD n° 3

Processus de contribution

Exercice 1 – Motivation des contributeurs

1. Citer au moins cinq motivations possibles de contributeurs à des projets open source (l'article de Gerosa et al. les regroupe en dix catégories).
2. Quelle est la motivation la moins commune, d'après cet article datant de 2021 ?

Exercice 2 – Évaluer un projet open source

1. Quels signes peuvent laisser entendre qu'un projet n'est pas ou est mal maintenu ?
2. Est-ce un mauvais signe pour un projet open source d'avoir beaucoup de tickets ouverts ? Pourquoi ?

Exercice 3 – Contribuer un rapport de bug

1. Que faut-il faire de préférence avant de créer un rapport de bug ?
2. Quels éléments doit contenir un rapport de bug ?

Exercice 4 – Contribuer une modification

1. Qu'est-ce qu'un patch ?
2. Depuis quand le concept de patch existe-t-il ?
3. Citer un projet recevant (encore aujourd'hui) les contributions sous forme de patch.
4. Qu'est-ce qu'une pull request ?
5. Quels peuvent être les usages d'un fork sur GitHub ?

Exercice 5 – Revue de code Le livre "Producing Open Source Software" de Karl Fogel contient le passage suivant :

> In the Subversion project, we did not at first make a regular practice of code review. There was no guarantee that every commit would be reviewed, though one might sometimes look over a change if one were particularly interested in that area of the code. Bugs slipped in that really could and should have been caught. A developer named Greg Stein, who knew the value of code review from past work, decided that he was going to set an example by reviewing every line of every single commit that went into the code repository. Each commit anyone made was soon followed by an email to the developer's list from Greg, dissecting the commit, analyzing possible problems, and occasionally praising a clever bit of code. Right away, he was catching bugs and non-optimal coding practices that would otherwise have slipped by without ever being noticed. Pointedly, he never complained about being the only person reviewing every commit, even though it took a fair amount of his time, but he did sing the praises of code review whenever he had the chance. Pretty soon, other people, myself included, started reviewing commits regularly too.

1. Qu'à démontré Greg Stein en pratiquant systématiquement la revue de code ?
2. Ce passage décrit-il la revue de code de contributions internes ou externes ? La revue de code était-elle effectuée de la même manière pour les contributions internes et externes dans le projet Subversion ?
3. Comment les mainteneurs de projets open source sur GitHub peuvent-ils rendre systématique la pratique de la revue de code ? S'effectue-t-elle de la même manière pour les contributions internes et externes ?
4. Y a-t-il besoin d'être un mainteneur pour contribuer à la revue de code ?

Exercice 6 – Le Copyright Transfer Agreement de la FSF Un article de la Free Software Foundation (`https://www.fsf.org/blogs/licensing/FSF-copyright-handling`) datant d'août 2021 explique la manière dont la FSF gère le copyright de certains projets GNU :

> For some GNU packages, the ones that are FSF-copyrighted, we ask contributors for two kinds of legal papers : copyright assignments, and employer copyright disclaimers. We drew up these policies working with lawyers in the 1980s, and they make possible our steady and continuing enforcement of the GNU General Public License (GPL).
>
> These papers serve four different but related legal purposes, all of which help ensure that the GNU Project's goals of freedom for the community are met. [...]
>
> One purpose is to give explicit permission to include the material in that GNU package. That is the most basic need.
>
> The second purpose is to empower the FSF to go to court and say, "That company is infringing our copyright when it tramples the freedom of users, denying them the freedom that our license gives them." [...]
>
> A third purpose is to make it possible to add additional permission to specific pieces of code. [...]
>
> The fourth purpose is to protect the community from the danger of employers' surprise claims. [...]
>
> The maintainers of some GNU packages would like to use a simple mechanism instead of these legal papers. It is called a "Developer Certificate of Origin" (DCO), and it makes a rough attempt at serving the first purpose and the fourth purpose. [...] Sadly, using DCOs would make relicensing code to move it into certain libraries a precarious and unwieldy proposition, relying on getting agreement of the other copyright holders or removing their code.

1. Reformuler les quatre objectifs présentés par la FSF.
2. Quel intérêt la FSF pourrait avoir à ajouter des permissions à certaines portions de code ?
3. En quoi un DCO ne permet-il pas de réaliser le second et le troisième objectif ?
4. Un projet dont le copyright est géré par la FSF peut-il intégrer du code provenant d'un fork quelconque ? Quid d'un projet qui utiliserait un DCO ?