Introduction aux Logiciels Libres – TD n° 4

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 ?
3. Que font certains projets pour éviter d'avoir trop de tickets ouverts ? Est-ce nécessairement un meilleur signe pour un contributeur potentiel ?

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.

Exercice 5 – Distributed version control Voici un extrait d'un article sur l'histoire de Git, disponible à l'adresse `https://www.welcometothejungle.com/en/articles/btc-history-git`.

> Traditionally, version control was client server, so the code sits in a single repository —or repo—on a central server. Concurrent Versions System (CVS), Subversion [SVN] and Team Foundation Version Control (TFVC) are all examples of client-server systems.
>
> A client-server VCS works fine in a corporate environment, where development is tightly controlled and is undertaken by an in-house development team with good network connections. It doesn't work so well if you have a collaboration involving hundreds or thousands of developers, working voluntarily, independently, and remotely, all eager to try out new things with the code, which is all typical with open source software (OSS) projects such as Linux.

[...] With distributed VCS, a copy of the most current version of the code resides on each developer's device, making it easier for developers to work independently on changes to the code. [...] "Being truly distributed means forks are non-issues, and anybody can fork a project and do their own development, and then come back a month or a year later and say, 'Look at this great thing I've done.'" [says Torvalds].

Another major drawback with client-server VCS, especially for open-source projects, is whoever hosted the central repository on their server "owned" the source code. With distributed VCS, however, there is no central repository, just lots of clones, so nobody owns or controls the code.

"[This is what makes] sites like GitHub possible. When there is no central 'master' location that contains the source code, you can suddenly host things without the politics that go along with that 'one repo to rule them all' concept," says Torvalds.

1. D'après cet article, pourquoi les systèmes de contrôle de version distribués sont-ils plus adaptés aux projets open source ?
2. Quelle évolution du sens du mot « fork » les systèmes de contrôle de version distribués ont-ils permis ?
3. Pourquoi Torvalds pense-t-il que git rend possible l'existence de sites comme GitHub ?
4. Y a-t-il des limites à la notion de non-centralité mise en avant par Torvalds ? Quel pourrait être un moyen de dépasser ces limites ?