Pourquoi les chercheurs devraient n’écrire que des logiciels libres

Sébastien Paumier

Le logiciel libre

- en anglais:
  - *free software* (ambigu avec *gratuit*)
  - *open source*
- mêmes principes, mais pour des raisons idéologiques différentes:
  - free software: générosité envers l'humanité, car tout le monde a le droit d'accéder à la connaissance
  - open source: efficacité du travail collectif où celui qui sait est celui qui fait

Les 4 libertés fondamentales

- selon Richard Stallman, père de la Free Software Foundation:
  - liberté d'utiliser le programme, pour n'importe quel usage
  - liberté d'étudier et d'adapter le programme
  - liberté de redistribuer le programme
  - liberté d'améliorer le programme et de diffuser vos modifications

Des chercheurs, mais lesquels ?

- nous parlons ici des chercheurs financés par le public: académiques, instituts nationaux de recherche, ...
- on s'intéresse à tous les logiciels qu'ils produisent:
  - preuves de concept
  - développements industriels
  - outils internes
  - etc.

Compatibilité avec l'industrie

- pas d'incompatibilité:
  - certaines licences comme la LGPL autorisent l'intégration de code libre dans des projets propriétaires
- le développement industriel est différent de la production de connaissance scientifique:
  - design, maintenance, gestion de divers formats de données, possibilité de plugins, interaction avec d'autres outils, etc.

Un LL n'a pas de maître unique

- avec un LL, vous ne dépendez pas d'un groupe ni d'une personne particulière
- si la politique ne vous plaît pas, vous pouvez toujours garder l'ancienne version ou créer une branche dissidente

- exemple: un logiciel de calcul formel qui change le caractère symbolisant la disjonction de + en U

Gérer les bugs et les ajouts

- en cas de bug ou de fonctionnalité manquante, les non informaticiens, face à un LL, peuvent demander à quelqu'un de patcher le code pour eux:
  - on évite les affreuses bidouilles en Perl/Python
  - on évite de devoir affronter des choses terribles pour le profane comme UTF8 ou les dépendances de bibliothèques

Tout le monde n'est pas codeur

- beaucoup de logiciels ne sont pas écrits par des experts en développement:
  - codes difficiles à maintenir et à intégrer dans d'autres applications
- un LL peut être étudié et amélioré par des gens compétents:
  - optimisation du code par un industriel
  - correction de l'anglais des commentaires par un locuteur natif
  - etc.

La crainte de l'anonymat

- certains auteurs craignent que le LL ne les rende anonymes, et donc non cités, à cause de son aspect collectif
- le LL protège les auteurs:
  - les contributions peuvent être signées par des commentaires dans le code
  - il est interdit de modifier anonymement le code d'autrui

Le besoin de contrôle

- certains auteurs veulent contrôler l'évolution de leur création, mais tout le monde peut modifier et redistribuer un LL à sa guise
- solution: désigner la version "officielle" par un nom qui, lui, peut être protégé
- exemple: Ubuntu et Debian sont des marques de distribution Linux clairement identifiées par leurs noms

Compatibilité

- la majorité des logiciels non libres sont distribués sous une forme binaire qui dépend d'un système précis, voire d'une machine précise
- dans combien de laboratoires y a-t-il une machine qui ne sert qu'à faire tourner la version 2.7.1 de BiniouCalculator ?

- un LL peut être adapté et recompilé partout

Peer-reviewing

- "*Empirism is not a matter of faith*"

  Ted Pedersen

- comment contrôler un article utilisant un logiciel si celui-ci n'est accessible ?
- viol du principe de reproductibilité des expériences

Marge d'amélioration

- si le logiciel n'est pas ouvert, comment savoir quelles sont les possibilités d'améliorations ?
  - améliorations théoriques: algorithme avec une meilleure complexité
  - améliorations pratiques: peut-être qu'un changement de constante pourrait accélérer grandement le programme, mais comment savoir ?

Marge d'amélioration

- sans LL, pour tester une idée, on doit:
  - tout recoder, avec sa modification
  - comparer
  - prier pourqu'il y ait une différence expérimentale statistiquement significative:
    - oui: c'est gagné, mais peut-être que le logiciel original était juste mal optimisé...
    - non: vous avez perdu *X* mois, car personne n'aime publier des résultats négatifs

La force de l'inertie

- les gens n'aiment pas changer leurs habitudes:
  - ils ne changeront pas de logiciel, à moins d'une *killer feature*
- plutôt que de tout reconstruire, peut-être en vain, vos modifications gagneront à être intégrées à un logiciel déjà utilisé:
  - le LL remplace la concurrence stérile par la coopération constructive

Vie et mort des logiciels

- beaucoup de logiciels sont écrits par des doctorants, et meurent quand ils sortent du monde de la recherche:
  - combien de logiciels sur lesquels on a écrit des articles les 10 dernières années sont encore accessibles et utilisables aujourd'hui ?
- un LL peut survivre à son auteur
- on peut éviter le syndrome du programme que seul son auteur peut faire tourner

Le code secret est une chimère

- il n'est aucun programme qui ne puisse être réécrit
  - le secret n'est qu'une protection temporaire
- si un programme protégé est vraiment intéressant, il sera réécrit tôt ou tard
- le protectionnisme encourage la naissance de concurrents, mais:
  - quand il y a aura une concurrence libre, serez-vous sûr de gagner contre toute une communauté motivée ?

Prestige académique

- en ouvrant votre logiciel, vous permettez aux autres de contribuer à votre projet
- ils complètent votre travail, ce qui le valorise

- d'un point de vue bibliométrique, l'héritage est bien plus intéressant que la compétition

Vitesse de la science

- en évitant aux autres de devoir réinventer la roue, vous leur permettez de se focaliser sur les vraies nouveautés

- la boîte à outils de la science croît plus vite

Qualité de la science

- si chacun fournit des briques libres dans son propre domaine d'expertise, on évite de mal résoudre des problèmes maîtrisés depuis longtemps par les gens qui savent

- les améliorations peuvent venir de tous les horizons:
  - optimisation de code, ajout de fonctionnalités, documentation, etc

Beaucoup de bonnes raisons

- la science partage beaucoup d'idéaux avec le LL:
  - partage du savoir, travail coopératif, transparence, ...
- le LL est une pratique constructive qui aide la science à progresser mieux et plus vite

Conclusion

- la recherche publique ne devrait produire que des logiciels libres
- on ne devrait jamais publier ou laisser publier d'article sur un logiciel qui n'est pas libre

Références (articles disponibles en ligne):

Why academic software should be open source, Sébastien Paumier

Empirism is not a matter of faith, Ted Pedersen