Sébastien **Blondeel**
Daniel **Cartron**
Hermantino **Singodiwirjo**

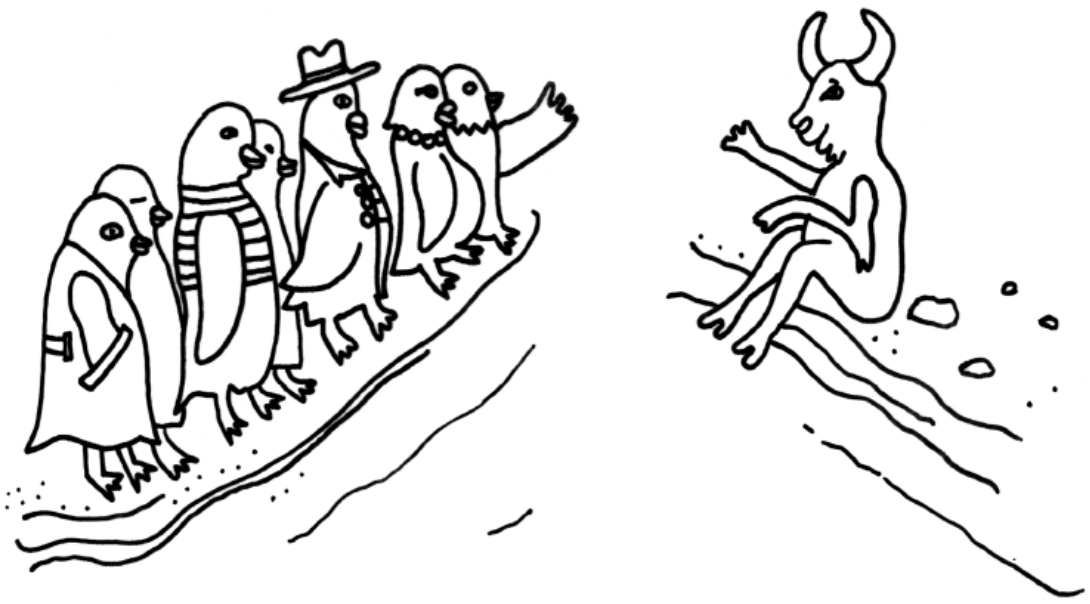
Débuter sous **Linux**

Avec la contribution de
Juliette **Risi**,
Laurent **Rathle**
et Gaël **Thomas**

ISBN : 2-212-11559-8

EYROLLES

chapitre 18



Ce qu'est vraiment le logiciel libre

Linux, GNU, les logiciels Open Source, le logiciel libre… Ces termes ont une exposition médiatique croissante mais sont rarement clairement définis et expliqués.
Ce chapitre se propose donc de familiariser le lecteur avec les notions et les points importants de ces domaines. Il s'inspire d'une conférence donnée par un auteur au printemps 2003.

SOMMAIRE

MOTS-CLÉS

- Logiciel libre
- Logiciel propriétaire

Aspects techniques

Le logiciel libre, comme beaucoup de notions juridiques, se définit avant tout par des critères organiques et techniques. Toutes ses applications aux domaines humains, sociaux, économiques, psychologiques, etc. n'en sont que des conséquences.

Code source et compilation

Les logiciels sont écrits par des hommes. Au début, c'était difficile et les contraintes des premières machines étaient lourdes. Mais le temps a passé et les programmeurs disposent maintenant d'environnements et de langages de programmation conviviaux, plus intuitifs, et éloignés de la froideur mécanique de la machine (voyez aussi l'annexe traitant de l'émulation pour avoir des informations complémentaires sur ces sujets).

L'aspect de la programmation destiné à être manipulé et partagé par des hommes s'appelle le code source. Les machines utilisent, pour des raisons historiques ou d'économie de place ou de vitesse, des codes binaires.

La transformation du code source en code binaire s'appelle la compilation. Elle est menée par un compilateur et elle détruit des méta-informations ; on ne peut pas revenir en arrière. On dispose de peu d'éléments sur ce qui s'est passé dans l'esprit du programmeur si l'on ne dispose que du code binaire ; on en est presque réduit à un rôle de consommateur passif. Même informaticiens, les développeurs n'en sont pas moins des hommes, et leur intelligence a besoin de socles intuitifs pour fonctionner, sans pouvoir se contenter de séries de chiffres sans aucune sémantique ni aucune explication.

Le code source est un peu comme la recette de cuisine : un gâteau une fois sorti du four s'adressera directement aux papilles gustatives, qui en apprécie-

CONCEPT Méta-information

Une méta-information est un élément informatif accompagnant un ensemble de données mais s'appliquant à un niveau supérieur (méta). Si l'on considère par exemple le texte d'un roman comme de l'information, les méta-informations associées pourront être le titre, le nom de l'auteur, l'année de publication, etc.

Les méta-informations ont parfois plus d'importance que les informations sur lesquelles elles portent : elles pourront par exemple se prononcer sur la qualité ou la validité de la source d'une information, ou sur la durée de péremption d'une information.

Dans le cas d'un code source, les méta-informations sont les éléments indiqués par le programmeur qui ne sont d'aucune utilité à la machine : cela va d'une mise en page aérée et claire aux noms des variables et des sous-programmes (un lecteur qui trouve une procédure intitulée `multiplication_de_matrices` saura déjà de quoi il retourne) en passant par des commentaires, complètement ignorés par la machine, qui expliqueront les écueils du calcul et justifieront l'emploi de tel algorithme ou tel choix d'implémentation.

MAUVAISE ANALOGIE

Le code source n'est pas comparable à une partition de musique

Cette comparaison peut être tentante, car on démarre d'un support écrit obscur au profane (la partition) pour aboutir, après interprétation par un artiste, à une piste sonore qui s'adresse directement aux tympans et aux sensations.

Cette analogie est cependant incorrecte : il existe de nombreuses personnes dotées de l'oreille absolue, qui après quelques années de solfège sauront sans grande difficulté retrouver la partition correspondant à une interprétation (surtout en musique classique !). Dans le cas du logiciel, personne au monde n'est capable sans gros effort et gros investissements de retrouver un code source lisible à partir d'un code binaire : même les méta-informations d'une partition se retrouvent dans une exécution (allegro ma non troppo, etc.).

De plus, une exécution apporte des éléments à une partition : le timbre de la voix, de l'instrument, l'attaque des notes, etc. Dans le cas contraire, on ne parlerait pas de solistes ou concertistes de génie, et l'on se contenterait de faire jouer des machines. À l'inverse, une compilation détruit des éléments d'information et de compréhension sans rien apporter.

ront quasi primairement les qualités. Il sera très difficile d'en retrouver la recette et d'en fabriquer une version un peu plus sucrée ou en remplaçant telle saveur par telle autre (un remplacement d'ingrédients). Au contraire, le rédacteur de la recette aura émaillé cette dernière de commentaires et de conseils pour la préparation, qui disparaîtront lors de la réalisation de la recette.

Il n'est pas nécessaire d'avoir étudié l'informatique pour comprendre un code source. C'est ce qui explique l'intérêt d'en disposer, même pour des non-informaticiens ; ils pourront effectuer eux-mêmes des modifications simples, ou chercher un prestataire de services au juste prix (souvent, dans les cas les moins complexes, si le fils de la concierge en pince un peu pour l'informatique, une friandise le comblera). Voici un exemple d'un programme écrit en langage C, socle de nombreux systèmes modernes :

```
#include <stdio.h>
int main () {
  int compteur;
  for (compteur = 1; compteur <= 10; compteur = compteur + 1) {
    int carre, cube;
    carre = compteur * compteur;
    cube = carre * compteur;
    printf("%d\t%d\t%d\n", compteur, carre, cube);
  }
}
```

À l'exécution, on obtient :

```
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000
```

On devine aisément la fonction de ce petit programme : afficher les nombres de 1 à 10, leur carré, puis leur cube. Le code source donne d'ailleurs des noms explicites aux différentes variables impliquées, comme c'est recommandé dans le génie logiciel pour assurer une bonne maintenance du code.

Voyons maintenant ce que l'éditeur de logiciels fournit au client qui achète une version propriétaire de ce programme, qui ne propose que le code binaire :

```
0000000: 7f45 4c46 0101 0100 0000 0000 0000 0000 .ELF............
0000010: 0200 0300 0100 0000 1083 0408 3400 0000 ................
0000020: 1408 0000 0000 0000 3400 2000 0600 2800 ........4. ...(.
```

```
0000030: 1900 1800 0600 0000 3400 0000 3480 0408 ........4...4...
0000040: 3480 0408 c000 0000 c000 0000 0500 0000 4...............
[...]
00005b0: f682 0408 0683 0408 0000 0000 0047 4343 .............GCC
00005c0: 3a20 2847 4e55 2920 322e 3935 2e34 2032 : (GNU) 2.95.4 2
00005d0: 3030 3131 3030 3220 2844 6562 6961 6e20 0011002 (Debian
00005e0: 7072 6572 656c 6561 7365 2900 0047 4343 prerelease)..GCC
00005f0: 3a20 2847 4e55 2920 322e 3935 2e34 2032 : (GNU) 2.95.4 2
0000600: 3030 3131 3030 3220 2844 6562 6961 6e20 0011002 (Debian
0000610: 7072 6572 656c 6561 7365 2900 0047 4343 prerelease)..GCC
[...]
0000bb0: 0700 0000 0000 0000 0000 0000 dc06 0000 ................
0000bc0: 7800 0000 0000 0000 0000 0000 0100 0000 x...............
0000bd0: 0000 0000 0100 0000 0300 0000 0000 0000 ................
0000be0: 0000 0000 5407 0000 bf00 0000 0000 0000 ....T...........
0000bf0: 0000 0000 0100 0000 0000 0000           ............
```

Le code source pouvait fonctionner sur des machines différentes, mais ce code binaire ne sera utilisable que sur des machines aux caractéristiques proches de celle pour laquelle il a été compilé.

Tout ceci est évidemment un peu simplifié. Après tout, les professeurs de cours préparatoire enseignent à leurs élèves qu'il est impossible de « soustraire à un petit nombre, un nombre plus grand ». Pourtant, tout le monde sait bien ce qui se passe quand la température chute de 10 degrés en partant d'une valeur de 5 degrés au-dessus de zéro, ou ce que devient un compte en banque de 1 000 euros quand on tire un chèque de 3 000 euros.

Un minimum d'honnêteté intellectuelle oblige donc à signaler que si l'on a la chance de disposer d'un code binaire non « strippé », on peut le faire parler de manière un peu plus convaincante en utilisant un débogueur, déplombeur, désassembleur ; on appelle cela l'ingénierie à revers (*reverse engineering*). On obtient alors les informations - ô combien plus parlantes - suivantes :

```
(gdb) disassemble main
Dump of assembler code for function main:
0x80483f0 <main>:       push   %ebp
0x80483f1 <main+1>:     mov    %esp,%ebp
0x80483f3 <main+3>:     sub    $0x18,%esp
0x80483f6 <main+6>:     nop
0x80483f7 <main+7>:     movl   $0x1,0xfffffffc(%ebp)
0x80483fe <main+14>:    mov    %esi,%esi
0x8048400 <main+16>:    cmpl   $0xa,0xfffffffc(%ebp)
0x8048404 <main+20>:    jle    0x8048408 <main+24>
0x8048406 <main+22>:    jmp    0x8048440 <main+80>
0x8048408 <main+24>:    mov    0xfffffffc(%ebp),%eax
0x804840b <main+27>:    imul   0xfffffffc(%ebp),%eax
0x804840f <main+31>:    mov    %eax,0xfffffff8(%ebp)
0x8048412 <main+34>:    mov    0xfffffff8(%ebp),%eax
0x8048415 <main+37>:    imul   0xfffffffc(%ebp),%eax
0x8048419 <main+41>:    mov    %eax,0xfffffff4(%ebp)
0x804841c <main+44>:    mov    0xfffffff4(%ebp),%eax
```

```
0x804841f <main+47>:    push   %eax
0x8048420 <main+48>:    mov    0xfffffff8(%ebp),%eax
0x8048423 <main+51>:    push   %eax
0x8048424 <main+52>:    mov    0xfffffffc(%ebp),%eax
0x8048427 <main+55>:    push   %eax
0x8048428 <main+56>:    push   $0x80484a4
0x804842d <main+61>:    call   0x8048300 <printf>
0x8048432 <main+66>:    add    $0x10,%esp
0x8048435 <main+69>:    incl   0xfffffffc(%ebp)
0x8048438 <main+72>:    jmp    0x8048400 <main+16>
0x804843a <main+74>:    lea    0x0(%esi),%esi
0x8048440 <main+80>:    leave
0x8048441 <main+81>:    ret
0x8048442 <main+82>:    lea    0x0(%esi,1),%esi
0x8048449 <main+89>:    lea    0x0(%edi,1),%edi
End of assembler dump.
```

C'est un exemple de programme écrit en langage d'assemblage. Produit automatiquement, il ne contient pratiquement aucune méta-information et il faut suivre pas à pas son fonctionnement pour espérer pouvoir reconstituer sa sémantique.

Ce résultat n'est donc pas complètement inexploitable, mais le travail nécessaire pour comprendre un petit programme simple comme celui-ci n'est déjà pas négligeable.

Travaillant quasi-exclusivement avec du logiciel libre, l'auteur connaît mal les techniques de déplombage et de décompilation. Vous aurez cependant compris qu'il est bien plus difficile de comprendre ce que le programme fait et de le modifier pour le corriger, le faire évoluer, ou l'adapter aux besoins, si l'on ne dispose pas de son code source.

Pour vous convaincre de la différence d'échelle dans les difficultés comparées, tâchez de répondre aux questions suivantes : comment procédez-vous pour obtenir les carrés et les cubes jusqu'à 12 et non 10, en disposant :

- du code source ?
- du code binaire ?
- d'un débogueur ?

Comment procéderez-vous pour calculer les puissances quatrième, travailler par incréments de deux, etc. ?

Cet exemple peut sembler simpliste et taillé sur mesure. Il en est exactement de même avec la plupart des programmes. Nul besoin d'assimiler toute l'architecture de l'arborescence de fichiers de code pour intervenir sur un point de détail précis. L'auteur intervient parfois sur de gros codes sources pour y effectuer une simple modification. Ce type d'intervention serait inimaginable sur un code binaire.

Ne diffuser que la version binaire, compilée, d'un programme revient donc à ériger un obstacle gratuit et artificiel pour restreindre la souplesse d'utilisation de ce produit par autrui.

Système d'exploitation, applications

Le système d'exploitation est assez distinctement séparé des applications sur les systèmes Unix. Il s'agit de la couche de virtualisation logicielle qui s'interface entre les programmes et les ressources de la machine : elle a pour tâche de reconnaître le matériel et le gérer.

Ainsi, on accédera à n'importe quel modèle de moniteur, de carte vidéo ou de disque dur de la même manière, et le même programme fonctionnera sur des systèmes différents.

Les applications sont les programmes destinés à l'utilisateur.

Le logiciel libre est traditionnellement bon et riche en logiciels plutôt proches du système, mais il se dote peu à peu d'une bibliothèque de programmes applicatifs de plus en plus fournie et convaincante.

Aspects juridiques

Théorie

I-A-N-A-L. Il s'agit d'une mise en garde classique en anglais : je ne suis pas un avocat. Vous trouverez sans dommage un mathématicien, professeur ou étudiant, pour vous soutenir que deux et deux font quatre, ou qu'en plaçant des poutres de cette manière votre toiture supportera sans problème telle épaisseur de neige.

Je crains qu'il soit autrement plus difficile de trouver un juriste qui soutienne quoi que ce soit. Les lois changent au gré des rapports de force, les juges les interprètent de manières diverses, et on observe même des revirements de jurisprudence. J'en ai conclu - peut-être de manière simplificatrice et abusive - qu'il n'est nulle vérité absolue dans ce domaine.

J'exposerai ici ce que j'ai compris et retenu de diverses lectures et exposés sur le sujet. Il s'agira évidemment de droit américain, puisque c'est historiquement et économiquement le territoire de développement privilégié des questions logicielles, et celui dont j'ai pu suivre des exposés.

Des efforts de traduction et d'adaptation au droit français de diverses licences sont actuellement menés, mais c'est un travail en cours et je n'ai pas vraiment trouvé de synthèse ni d'exemples de poursuites judiciaires ou d'arrangement à l'amiable sur le sujet.

Un logiciel est soumis dès sa création, même s'il ne sort pas du tiroir, au droit du « copyright ». Ce copyright donne par défaut tous les droits classiques à

son auteur, tels que celui d'en interdire la production de copies ou d'exiger une taxe à chaque copie effectuée.

Ce droit se vend ; on n'en conserve alors plus rien. Ni droit de repentir ni droit moral - contrairement au droit d'auteur à la française, qui prévoit un volet moral, dans la pratique presque absent pour les développeurs de logiciels. C'est le « transfert de copyright ».

Si l'on souhaite placer un logiciel sous des conditions d'utilisation et de diffusion différentes, on peut l'accompagner d'une licence, que l'utilisateur final devra ou non, selon les pays ou les époques, accepter sciemment et explicitement.

Les œuvres tombées ou placées dans le domaine public peuvent être utilisées par quiconque de la manière qu'il le souhaite. Il peut les revendre sous son propre copyright, les travestir, les modifier, etc.

Le logiciel libre n'est pas libre de droits. Les auteurs de logiciels libres, même s'ils émettent des réserves quant à l'arsenal juridique actuellement en place sur les questions de copyright et peuvent souhaiter changer certaines de ses lois, exploitent ce dernier pour arriver à leurs fins. Un logiciel libre utilise donc le droit du copyright pour imposer certaines conditions de diffusion à tous. Quiconque ne les accepte pas, n'ayant par défaut aucun droit sur la création intellectuelle d'autrui, on retombe dans le schéma classique, autrement plus restrictif.

Limites

En France et dans certains autres pays européens, on a d'autres droits. Le droit latin notamment prévoit une partie patrimoniale et une partie morale au droit d'auteur.

Il y a également la question des langues (en France, une traduction en français est obligatoire, le contrat de licence devrait donc peut-être être traduit pour espérer avoir une quelconque valeur).

Les grandes familles de licences

Le logiciel privé

Le logiciel privé est le logiciel majoritaire en volume de par le monde. C'est du logiciel que l'on produit et garde par devers soi (ou remet contre rétribution au commanditaire ou client qui a passé commande dans le cas d'une société de services). Il n'est jamais diffusé à des tiers (non détenteurs de droits) ; les questions des conditions de diffusion ne se posent donc pas.

Le logiciel propriétaire

On appelle logiciel propriétaire tout logiciel qui n'est pas libre. Il s'agit probablement du logiciel auquel le lecteur est habitué : il est fréquent qu'il soit payant, et/ou fourni sans code source, et/ou avec interdiction de le recopier et de le diffuser à des tiers, de manière lucrative ou non.

Le logiciel libre

Le logiciel libre est un terme parfois galvaudé (et l'Open Source Initiative n'ayant pas réussi à déposer la marque Open Source, cette autre dénomination n'apporte pas non plus de solution). Il faut donc veiller à se faire préciser ce que cette expression recouvre exactement dans chaque contexte, personne n'ayant le monopole du mot « libre ».

FSF et OSI

- http://www.fsf.org/
- http://www.opensource.org/

Les entités principales traitant et théorisant sur le sujet sont la Free Software Foundation, ancienne (créée en 1984), solide et réfléchie, et l'Open Source Initiative, plus récente et moins claire.

Toutes deux proposent des définitions permettant de décider si la licence d'un logiciel en fait un logiciel libre (FSF) ou un logiciel Open Source (OSI), définissant ainsi des familles de licences. Leurs sites web respectifs dressent également la liste des licences les plus répandues qu'ils ont analysées et les classent en deux catégories.

L'examen de ces listes montre que ces deux définitions formelles, fort différentes (la définition du logiciel libre compte quatre conditions fort simples, inchangées depuis toujours, alors que la définition de l'Open Source compte une petite dizaine de conditions tarabiscotées et en est à la version 1.9), sont quasiment équivalentes (l'APSL, licence peu utilisée qui nie le droit à la vie privée, est compatible OSI mais pas libre au sens FSF).

On traite donc des mêmes objets sous des noms différents. Cet éclairage a quand même son importance quant aux éléments mis en avant : la FSF traite avant tout de la liberté de l'utilisateur, l'OSI adopte un point de vue de technicien. Leurs slogans pourraient être respectivement « Liberté, Égalité, Fraternité » et « Rentabilité, Efficacité, Fiabilité ».

Il existe plusieurs dizaines de licences de logiciels libres, plus ou moins heureuses, plus ou moins utilisées, répandues et importantes, et elles se divisent en deux grandes classes : les licences « copyleft » et les autres.

Les licences copyleft

Copyleft, ou « gauche d'auteur » est un jeu de mot expliquant que le principe du copyright est utilisé contre l'esprit original du copyright : une œuvre diffusée sous licence copyleft est fournie avec certaines libertés (comme celles

de l'utiliser, la copier, la modifier, la diffuser) et l'obligation de ne jamais priver autrui de ces mêmes libertés.

Cela implique notamment que toute copie remise à un tiers devra être placée sous la même licence et proposer le code source si ce tiers le demande.

L'exemple le plus connu de ces licences est la licence phare de la FSF, la GPL, ou General Public License de GNU, couvrant la majorité des projets de logiciels libres. La FSF propose également une licence LGPL, originellement pour les bibliothèques de programmation et dont l'utilisation est désormais découragée (elle permet de faire une édition de liens avec des programmes propriétaires sous certaines conditions), et la GFDL, qui est le pendant de la GPL pour les documentations techniques.

Il est en particulier possible à un tiers de vendre un programme sous GPL, d'en tirer de substantiels profits, et de n'en rien reverser à l'auteur original. Ceci est voulu : plus le logiciel dispose de médias de diffusion, plus il sera accessible à tous. Le but de la FSF n'est pas de protéger les auteurs ou d'assurer leurs moyens de subsistance. Qui plus est, dans la plupart des cas, cela impliquerait des comptes d'apothicaire impossibles à mener à bien, notamment dans le cas de nombreux auteurs ou des compilations. Une association pourra ainsi sans problème graver des CD et les revendre à ceux qui ont un accès Internet limité sans avoir à traiter aucune paperasserie.

GNU

GNU signifie GNU's Not Unix (GNU N'est pas Unix). C'est un jeu de mots en forme d'acronyme récursif.

Les autres licences

Parmi les autres licences de logiciels libres, les plus importantes historiquement et en volume sont les licences dites BSD ou X, très courtes et simples, utilisées par de nombreux projets logiciels et notamment par les BSD libres (FreeBSD, NetBSD, OpenBSD).

La grande différence est qu'il est possible de partir d'un logiciel sous ces licences, de le retravailler, et de le diffuser sous forme propriétaire, binaire seulement, en se contentant de le signaler dans un petit coin de la documentation. C'est ce qu'a fait Microsoft dans des MS-Windows récents pour quelques programmes, et c'est ce qu'a fait en particulier Apple avec MacOS X, inspiré des BSD libres.

L'utilisateur peu averti ne sera donc pas protégé malgré lui et pourra faire l'erreur de se laisser séduire par une version propriétaire d'un tel programme, pour lequel seul son éditeur pourra ensuite, éventuellement, lui proposer des correctifs.

La liberté selon la FSF

La FSF propose une définition simple et directe de la liberté du logiciel : un logiciel est libre si et seulement si il propose quatre libertés précises à son utilisateur.

Les types d'œuvres et les conséquences

On trouve différents types d'oeuvre dont nous allons étudier les caractéristiques.

- Les **œuvres fonctionnelles**, telles que les recettes de cuisine, les documentations techniques, les logiciels. Elles sont interchangeables, tant que la nouvelle version remplit le même besoin. Il convient de les diffuser selon les conditions du logiciel libre, et si possible sous copyleft.
- Les **œuvres d'opinion**, telles que les mémoires, les manifestes, les prises de position. Il convient d'en autoriser la copie verbatim, gratuitement ou non, pour en encourager la diffusion, mais toute modification du contenu est évidemment interdite.
- Les **œuvres artistiques**, pour lesquelles la situation est moins claire. De nombreux classiques sont inspirés d'œuvres de leur époque depuis tombées dans l'oubli : Shakespeare, La Fontaine… On peut souhaiter en autoriser la diffusion non commerciale par exemple ; elles ne sont en effet pas interchangeables, et aucune chanson « libre » ne pourra remplacer sur commande l'air qui vous trotte dans la tête !

Aspects humains et psychologiques

La motivation, la concurrence

Dans le logiciel libre, tout le monde a accès aux mêmes informations - évidemment, ceux qui ont plus d'expérience, qui ont écrit le code eux-mêmes, ou qui ont des bases théoriques plus solides ou un esprit plus vif sont avantagés. Tout le monde travaille donc dans la plus parfaite transparence, et seuls les critères techniques sont utilisés pour les jugements de valeur.

Les projets s'affrontent parfois (tels que GNOME et KDE, deux environnements de bureau). Des programmeurs doués font des concours de chevilles enflées. C'est l'émulation généralisée.

De plus, travailler sur un projet copyleft est une motivation essentielle pour certains, qui vivraient mal que leur contribution tombe sous des fourches caudines.

Les passagers clandestins (ceux qui profitent du travail d'autrui sans rien reverser au pot commun) sont nombreux, et on est toujours le passager clandestin d'un autre projet. Qu'importe : aussi incroyable que cela pouvait paraître dans les années 1980, au début de GNU, il est possible que des milliards de dollars d'équivalent industriel en développement logiciel s'écrivent « tous seuls ». Jusqu'à présent, il a toujours existé suffisamment de contributeurs pour que cette dynamique continue ; après tout, les passagers clandestins ne privent personne de rien.

Les communautés

C'est ainsi que se montent et évoluent un certain nombre de petites communautés autour de projets logiciels. Quelques chefs de projet autoritaires et charismatiques, une poignée de développeurs principaux, et une armée de débogueurs, utilisateurs exprimant plus ou moins bien leurs souhaits, les constituent.

Aussi incroyable que cela paraisse dans la vision classique du génie logiciel, tout cela tourne et est expliqué dans les essais d'Eric Raymond sur son site web CatB.org, notamment dans *The Cathedral And The Bazaar*.

La cathédrale et le bazar

- http://www.catb.org/

Aspects industriels

Nous avons vu que le logiciel libre avait une définition technique et éthique, et que les aspects économiques passaient au second plan. Question de priorités.

Il n'en demeure pas moins que ce bouillon de culture psychologique a des effets secondaires très intéressants dans un contexte industriel.

Les business plans

Il est beaucoup plus difficile, presque impossible, de faire de l'argent en vendant des licences de logiciels.

Et alors ? Environ 5 % des programmeurs dans le monde ont un salaire qui dépend directement de ce type de vente. Les autres font pour la plupart du service, de la maintenance, ou développent des programmes à façon (du logiciel privé).

De plus, les incompatibilités entre les différents systèmes (ex : PC et Mac), impossibles à éviter en l'absence du code source et de toutes manières tolérées par les législateurs, ont abouti à une situation de « Winner Takes All », avec une oligopole confinant au monopole.

TRICHEURS Absence de code source

De sévères soupçons ont pesé sur un important éditeur de logiciels propriétaires : des concurrents l'ont accusé d'avoir sciemment introduit dans ses programmes des tests pour qu'ils fonctionnent moins bien et moins vite si l'on remplaçait sur un système certains de ses composants par ceux de la concurrence.

En l'absence de code source, il est difficile de « prouver » cela et malheureusement, le législateur n'a pas imposé aux nouveaux industriels de la micro-informatique grand public de déposer leurs codes sources avant commercialisation chez un tiers de confiance (comme un notaire), qui se chargerait de leur compilation et de la fabrication des versions binaires.

Cet éditeur a traîné les pieds, fait preuve d'une grande mauvaise foi, puis a fini par déclarer que les codes sources de la version incriminée de son produit avaient été « perdus ». C'est évidemment un mensonge éhonté, un outrage à la justice, mais à ma connaissance, il n'a pas été inquiété outre mesure.

D'un point de vue économique, les ventes de licences constituent des fuites de capitaux pour engraisser des géants rentiers qui empêchent toute concurrence et assomment ou rachètent toute petite entreprise prometteuse. C'est une forme d'impôt nouveau, régulièrement prélevé avec les nouvelles versions des logiciels et leurs incompatibilités de commandes.

Intérêts du logiciel libre

Quoi que certains en pensent, je ne suis pas convaincu que la liberté soit un élément qui ait la moindre importance dans aucune entreprise d'une certaine taille. On peut bien sûr observer ce phénomène dans certaines petites exploitations familiales ou individuelles.

Le logiciel libre confère cependant trois indépendances, toutes trois fort intéressantes au niveau industriel :

- une indépendance technique : si le fournisseur change ses conditions de maintenance ou son service après-vente de manière unilatérale, ou se comporte de manière insupportable, il est envisageable d'en changer si l'on dispose du code source (surtout si ce dernier correspond à un logiciel libre non privé, publié sur Internet et susceptible d'être connu par d'autres). On pourra au pire faire l'effort de rentrer dans le code (coût de sortie).
- une indépendance politique par rapport aux pays des grands éditeurs de logiciels, qui peut avoir son importance pour les États (notamment lorsqu'ils équipent leurs armées) ou les grandes entreprises.
- une indépendance économique enfin, car il est souvent possible et facile d'obtenir du logiciel libre de manière gratuite, sans être l'otage des éditeurs, de leurs mises à jour, de leurs bogues savamment mis en place pour nous forcer la main, et de leurs prétendues assistances.

En résumé...

Ce chapitre a présenté la définition technique du logiciel libre en s'assurant de bien faire comprendre d'abord la notion de code source et l'intérêt d'en disposer.

Il a présenté quelques conséquences et observations empiriques du monde du logiciel libre, et cité des sites web en anglais regorgeant de textes d'analyses sur ce type de sujet. Les lecteurs que l'anglais rebute pourront en savoir plus en consultant les sites web de Linux-France, APRIL (Association pour la Promotion et la Recherche en Informatique Libre) et AFUL (Association Francophone des Utilisateurs de Linux et des Logiciels Libres).

DOCUMENTATION **Sites web en français**

- http://www.linux-france.org/
- http://www.april.org/
- http://www.aful.org/