Éthique des Logiciels Libres

Richard M. Stallman & Eben Moglen

Table des Matières

Cette brochure a été réalisée par Florent Gallaire (http://fgallaire.flext.net).

Crédits photographiques :

Montage photographique : © F&F Gallaire.
Richard Stallman à Bruxelles en février 2005 : Creative Commons BY, Chrys (http://exposur3.com).
Eben Moglen à New York en avril 2003 : © Declan McCullagh (http://www.mccullagh.org).

Qu'est ce qu'un « Logiciel Libre » ?

Nous maintenons cette définition du logiciel libre pour décrire clairement les conditions à remplir pour qu'un logiciel soit considéré comme libre.

L'expression «Logiciel libre» fait référence à la liberté et non pas au prix. Pour comprendre le concept, vous devez penser à la «liberté d'expression», pas à «l'entrée libre».

L'expression «Logiciel Libre» fait référence à la liberté pour les utilisateurs d'exécuter, de copier, de distribuer, d'étudier, de modifier et d'améliorer le logiciel. Plus précisément, elle fait référence à quatre types de liberté pour l'utilisateur du logiciel :

- La liberté d'exécuter le programme, pour tous les usages (liberté 0).
- La liberté d'étudier le fonctionnement du programme, et de l'adapter à vos besoins (liberté 1). Pour ceci l'accès au code source est une conditon requise.
- La liberté de redistribuer des copies, donc d'aider votre voisin, (liberté 2).
- La liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté (liberté 3). Pour ceci l'accès au code source est une condition requise.

Un programme est un logiciel libre si les utilisateurs ont toutes ces libertés. Ainsi, vous êtes libres de redistribuer des copies, avec ou sans modifications, gratuitement ou non, à tout le monde, partout. Être libre de faire ceci signifie (entre autre) que vous n'avez pas à demander ou à payer pour en avoir la permission.

Vous devez aussi avoir la liberté de faire des modifications et de les utiliser à titre personnel dans votre travail ou vos loisirs, sans en mentionner l'existence. Si vous publiez des modifications, vous n'êtes pas obligé de prévenir quelqu'un de particulier ou de le faire d'une manière particulière.

La liberté d'utiliser un programme est la liberté pour tout type de personne ou d'organisation de l'utiliser pour tout type de système informatique, pour tout type de tâche et sans être obligé de communiquer à ce sujet avec le développeur ou tout autre entité spécifique. Dans cette liberté, il est question de *l'utilisateur*, pas du développeur ; en tant qu'utilisateur, vous êtes libre d'exécuter un programme comme bon vous semble et si vous le redistribuer à quelqu'un d'autre, cette personne est libre de l'exécuter comme bon lui semble, mais vous n'êtes pas autorisé à lui imposer vos conditions.

La liberté de redistribuer des copies doit inclure les formes binaires ou exécutables du programme (tout comme le code source) à la fois pour les versions modifiées ou non modifiées du programme. (Diffuser des programmes sous une forme exécutable est nécessaire pour une installation commode des systémes d'exploitation libres). Il y a une exception s'il n'y a pas moyen de produire une version binaire ou exécutable (puisque certains langages ne supportent pas cette caractéristique), mais le public doit avoir la liberté de distribuer de telles formes s'ils ont un moyen d'en produire.

Pour avoir la liberté d'effectuer des modifications et de publier des versions améliorées, vous devez avoir l'accés au code source du programme. Par conséquent, l'accéssibilté du code source est une condition requise pour un logiciel libre.

Un moyen important de modifier un programme est de le fusionner avec des modules ou des sous-

programmes disponibles libres. Si la licence du programme indique que vous ne pouvez pas le fusionner dans un module existant, telle que la nécessité que vous soyez le détenteur du copyright de tout code que vous ajoutez, alors la licence est trop restrictive pour être qualifié de libre.

Pour que ces libertés soient réelles, elles doivent être irrévocables tant que vous n'avez rien fait de mal; si le développeur du logiciel a le droit de révoquer la licence sans que vous n'ayez fait quoi que ce soit pour le justifier, le logiciel n'est pas libre.

Cependant, certains types de règles sur la maniére de distribuer le logiciel libre sont acceptables tant que ces règles ne rentrent pas en conflit avec les libertés fondamentales. Par exemple, le copyleft (pour résumer très simplement) est une règle qui établit que lorsque vous redistribuez les programmes, vous ne pouvez ajouter de restrictions pour retirer les libertés fondamentales au public. Cette règle ne rentre pas en conflit avec les libertés fondamentales, en fait elle les protège.

Vous pouvez avoir payé pour obtenir une copie d'un logiciel libre ou vous pouvez l'avoir obtenu gratuitement. Mais indifféremment de la manière dont vous vous l'êtes procuré, vous avez toujours la liberté de copier et de modifier un logiciel et même d'en vendre des copies.

«Logiciel libre» ne signifie pas «non commercial». Un logiciel libre doit être disponible pour un un usage commerciale, pour le développement commercial et la distribution commerciale. Le développement commerciale de logiciel libre n'est plus l'exception; de tels logiciels libres commerciaux sont très importants.

Les règles sur la manière d'empaqueter une version modifiée sont acceptables si elles n'entravent pas votre liberté de la publier, ou votre liberté de faire et d'utiliser pour votre usage personnel des version modifiées. Les règles disant «si vous publiez le programme par ce moyen, vous devez le faire par ce moyen aussi» sont acceptables aux même conditions (notez que de telles règles doivent vous laisser le choix de publier ou non le programme). Les règles qui nécessitent que le code source soit publié pour les utilisateurs pour les versions que vous rendez publique sont aussi acceptables. Il est également acceptable que la licence l'exige, si vous avez ditribué une version modifiée et qu'un développeur précédent vous en demande une copie, vous devez lui envoyer, ou que vous indiquiez vos modifications.

Dans le projet GNU, nous utilisons le «copyleft»[1] pour protéger ces libertés. Mais des logiciels libres non-copyleftés existent aussi. Nous croyons qu'il y a de bonnes raisons qui font qu'il est mieux d'utiliser le copyleft[2], mais si votre programme est libre non-copylefté, nous pouvons tout de même l'utiliser.

Lisez Les catégories de Logiciel Libre[3], où sont décrites les relations entre «logiciel libre», «logiciel copylefté» et les autres catégories de logiciel.

Parfois le contrôle gouvernementale des exportations ou des sanctions économiques peuvent vous priver de la liberté de distribuer des copies de programmes à l'étranger. Les développeurs de logiciel n'ont pas le pouvoir d'éliminer ou de passer outre ces restrictions, mais ce qu'ils peuvent et doivent faire, est de refuser d'imposer eux-mêmes des conditions à l'utilsation des programmes. De cette manière, les restrictions n'affecteront pas les activités et les personnes se trouvant hors de la juridiction de leurs gouvernements.

La plupart des licences de logiciels sont basées sur le droit d'auteur, or les types d'exigences que le droit d'auteur peut imposer ont des limites. Si une licence basée sur le droit d'auteur respect la liberté de la manière décrite plus haut, il est improbable que nous ayons une autre sorte de problème que nous n'ayons pas anticipé (bien que cela arrive parfois). Cependant, certaines licences de logiciels sont

basées sur le droit du contrat et les contrats impliquent un champ bien plus large de restrictions. Cela signifie qu'il y a bien plus de possibilités pour qu'une licence de ce type puisse restreindre de manière inacceptable la liberté des utilisateurs et ainsi devenir non libre.

Nous ne pouvons pas lister tout ce qui pourrait se passer. Si une licence basée sur le contrat restreint l'utilisateur d'une manière inhabituelle que les licences basées sur le copyright ne peuvent pas faire et qui n'est pas mentionée comme légitime, nous devrons y réfléchir et nous conclurons problablement qu'elle n'est pas libre.

Quand vous parlez des logiciels libres, il est préférable de ne pas utiliser de termes comme «donner» ou «gratuit», car ils laissent supposer que la finalité des logiciels libres est le prix et non la liberté. Certains termes répandus comme «piratage» comportent des idées auxquelles nous espérons que vous n'adhérerez pas. Lisez Termes prêtant à confusion, que vous devriez éviter[4] pour un essai sur l'utilisation de ces termes. Nous avons aussi une liste de traductions de «free software»[5] dans de nombreuses langues.

Enfin, notez bien que les critères tels que ceux développés dans cette définition du logiciel libre demandent une réflexion sérieuse quant à leur interprétation. Pour décider si une licence de logiciel particulière est définie comme libre, nous la jugeons sur ces critères pour déterminer si elle convient à leur esprit tout comme à leur formulaton précise. Si une licence inclut des restricitions inacceptables, nous la rejetons même si nous n'avons pas anticipé le problème dans ces critères. Quelquefois les conditions d'une licence soulève un problème qui nécessite des réflexions intenses, y compris des dicussions avec un juriste, avant que nous pussions décider si la condition est acceptable. Quand nous arrivons à une conclusion concernant un nouveau problème, nous mettons souvent à jour ces critères pour rendre plus facile le fait de savoir si une licence est une licence logicielle libre ou non.

Si vous voulez savoir si une licence spécifique est définie comme «libre», reportez-vous à notre listes de licences[6]. Si la licence qui vous intéresse n'y est pas listée, vous pouvez nous demander des précisions en nous envoyant un mail <licensing@gnu.org>.

Si vous envisagez d'écrire une nouvelle licence, veuillez contacter la FSF en écrivant à cette adresse. La prolifération de différentes licences de logiciels libres implique un travail grandissant pour les utilitsateurs dans la compréhension des licences; nous pouvons vous aider à trouver une licence existante de logiciel libre et éviter divers problèmes pratiques.

Au-delà du logiciel

Les manuels de logiciels doivent être libres, pour les mêmes raisons que les logiciels doivent être libres, et parce que les manuels font en effet partie des logiciels.

Les mêmes arguments peuvent aussi s'appliquer à d'autres types de travaux à usage pratique — c'est-à-dire, des travaux qui intègrent de la connaissance utile, tels que les ouvrages éducatifs et les ouvrages de référence. Wikipedia est le meilleur exemple connu.

Tout type d'œuvre peut être libre, et la définition du logiciel libre a été étendu à la définition des œuvres culturelles libres applicable à tout type d'œuvre.

Open Source ?

Un autre groupe a commencé à utiliser le terme «open source» pour exprimer quelque chose de proche (mais pas d'identique) au «logiciel libre». Nous préférons le terme «logiciel libre» parce que, une fois que vous ayez entendu que ce terme réfère à la liberté plutôt que le prix, il appelle à prêter attention à la liberté[7]. Le mot «open» ne rend pas cela.

[1] Qu'est-ce que le copyleft ? (page 7)
[2] Copyleft : Idéalisme pragmatique (page 11)
[3] http://www.gnu.org/philosophy/categories.fr.html
[4] http://www.gnu.org/philosophy/words-to-avoid.fr.html
[5] http://www.gnu.org/philosophy/fs-translations.fr.html
[6] http://www.gnu.org/licenses/license-list.fr.html
[7] http://www.gnu.org/philosophy/free-software-for-freedom.fr.html

Traduction : Karl Pradène <Karl.Pradene@dpci.fr>
Révision : trad-gnu@april.org

Qu'est-ce que le copyleft ?

Le copyleft est une façon générale de rendre un programme (ou tout autre œuvre) libre, et qui requiert que toutes les versions modifiées et étendues du programme soient libres également.

La manière la plus simple de faire d'un programme un logiciel libre est de le distribuer dans le domaine public, sans copyright. Cela autorise les gens à partager le programme et leurs améliorations si le cœur leur en dit. Mais cela autorise aussi des personnes indélicates à faire du programme un logiciel propriétaire. Ils peuvent très bien y effectuer des changements, juste quelques-uns ou plusieurs, et distribuer le résultat comme un logiciel propriétaire. Ceux qui recevront le programme dans sa forme modifiée n'auront pas la liberté que l'auteur original leur aura donné; l'intermédiaire l'aura fait disparaître.

Dans le projet GNU, notre but est de donner à tous les utilisateurs la liberté de redistribuer et de modifier les logiciels GNU. Si des intermédiaires pouvaient enlever cette liberté, nous aurions beaucoup d'utilisateurs, mais ils n'auraient aucune liberté. Alors, au lieu de mettre les logiciels GNU dans le domaine public, nous les mettons sous «copyleft» ou «gauche d'auteur». Le copyleft indique que quiconque les redistribue, avec ou sans modifications, doit aussi transmettre la liberté de les copier et de les modifier. Le copyleft garantit cette liberté pour tous les utilisateurs.

Le copyleft fournit aussi un encouragement aux autres programmeurs qui veulent ajouter des logiciels libres. Des programmes importants comme le compilateur C++ de GNU n'existent que grâce à lui.

Le copyleft aide aussi les programmeurs qui veulent contribuer à des améliorations sur des logiciels libres à obtenir la permission de le faire. Ces programmeurs travaillent souvent pour des entreprises ou des universités qui feraient n'importe quoi pour plus d'argent. Un programmeur pourrait vouloir faire profiter la communauté de ses modifications, mais son employeur pourrait vouloir transformer le travail en un produit propriétaire.

Quand nous expliquons à l'employeur qu'il est illégal de distribuer la version améliorée autrement que comme logiciel libre, celui-ci décide souvent de le distribuer librement plutôt que de le laisser tomber.

Pour mettre un logiciel sous copyleft, nous déclarons d'abord qu'il est sous copyright, ensuite nous ajoutons les conditions de distribution, qui sont un outil légal donnant à chacun le droit d'utiliser, de modifier, et de redistribuer le code du programme, ou *tous les programmes qui en sont dérivés*, mais seulement si les conditions de distribution demeurent inchangées. Ainsi, le code et ses libertés sont légalement indissociables.

Les développeurs de logiciels propriétaires utilisent le copyright pour restreindre la liberté des utilisateurs; nous utilisons le copyleft pour la garantir. C'est pourquoi nous avons inversé le nom, en changeant «copyright» en «copyleft».

Le copyleft est un moyen d'utilisation du copyright du programme. Cela ne signifie pas d'abandonner le copyright ; en fait, faire cela rendrait le copyleft impossible. Le mot « left » (NdT : en anglais, left signifie à la fois laissé, abandonné ou gauche) dans « copyleft » n'est pas une référence au verbe « to leave » (NdT : laisser, abandonner) c'est seulement une référence à la direction inverse de « right » (NdT : droite).

Le copyleft est un concept général, et vous ne pouvez pas utiliser un concept général directement ; vous pouvez seulement utiliser une mise en œuvre spécifique du concept. Dans le projet GNU, les conditions de distribution spécifiques que nous utilisons sont contenues dans la GNU General Public License[1]. La GNU General Public License est appelée la GNU GPL. Il y a également une page Foire aux questions[2] à propos de la GPL GNU. Vous pouvez lire également Pourquoi la FSF obtient l'attribution des droits d'auteurs des contributeurs[3].

Une forme alternative de copyleft, la GNU Lesser General Public License (LGPL)[4], s'applique à quelques (mais pas à toutes) bibliothèques GNU. Cette licence était initialement appelée la Library GPL (GPL pour les bibliothèques), mais nous avons changé le nom car l'ancien nom encourageait l'utilisation de cette licence plus souvent qu'elle aurait dû être utilisée. Pour une explication sur les motivations qui nous ont convaincu que ce changement était nécessaire, lire l'article pourquoi vous ne devriez pas utiliser la LGPL pour votre prochaine bibliothèque[5].

La GNU Free Documentation License (FDL)[6] est une forme de copyleft conçue pour être utilisée pour un manuel, un livre ou un autre document de manière à assurer à chacun la liberté effective de le copier et de le redistribuer, avec ou sans modifications, de façon commerciale ou non.

La licence appropriée est incluse dans beaucoup de manuels et dans chaque distribution de code source GNU.

Toutes ces licences sont conçues de façon à pouvoir être appliquées à votre programme si vous en détenez le copyright. Vous n'aurez pas à modifier la licence pour le faire, mais seulement à ajouter une copie de la licence à votre programme et des références appropriées dans les fichiers sources qui se réfèrent correctement à la licence.

L'utilisation des mêmes conditions de distribution pour plusieurs programmes différents facilite la copie de code entre divers programmes. quand ils ont tous les mêmes conditions de distribution, il n'y a pas de problème. La LGPL version 2, contient une clause qui vous autorise à modifier les conditions de distribution de la GPL ordinaire, ainsi vous pouvez copier du code dans un autre programme couvert par la GPL. La version 3 de la LGPL est construite comme une exception ajoutée à la GPL version3, rendant la compatibilité automatique.

Si vous désirez mettre votre programme sous copyleft avec la GNU GPL ou la GNU LGPL, veuillez lire la page d'instructions de la GPL[7] pour des conseils. Veuillez noter que vous devez reproduire le texte intégral de nos licences, si vous en utilisez une. C'est un tout, et les copies partielles ne sont pas autorisées.

Si vous désirez mettre votre manuel sous copyleft avec la GNU FDL, veuillez lire les instructions à la fin du texte de la FDFL, ainsi que la page d'instructions de la GFDL. De même, les copies partielles ne sont pas autorisées.(Note de Florent Gallaire : Préférer maintenant les licences Creative Commons).

[1] http://www.gnu.org/licenses/gpl.html
[2] http://www.gnu.org/licenses/gpl-faq.fr.html
[3] http://www.gnu.org/licenses/why-assign.fr.html
[4] http://www.gnu.org/licenses/lgpl.html
[5] http://www.gnu.org/philosophy/why-not-lgpl.fr.html
[6] http://www.gnu.org/licenses/fdl.html
[7] http://www.gnu.org/licenses/gpl-howto.fr.html

Traduction : Benjamin Drieu.

Révision : trad-gnu@april.org

Pourquoi le Copyleft ?

« Quand il est temps de défendre la liberté d'autrui, ne pas résister et laisser faire est un acte de faiblesse, non d'humilité ».

Au sein du projet GNU, nous recommandons généralement l'utilisation de licences de type « copyleft » comme la GNU GPL plutôt que des licences de logiciels libres plus permissives. Nous ne polémiquons pas contre les licences « non-copyleft » (en fait, nous les recommandons ponctuellement dans certaines circonstances) mais les partisans de ces licences montrent, eux, une tendance marquée à la polémique contre la licence GPL.

D'une certaine manière, quelqu'un a décidé que l'utilisation d'une des licences BSD était un « acte d'humilité » : « je ne demande rien à la personne qui utilise mon code, à l'exception de la reconnaissance de mon travail ». C'est quelque part une exagération de considérer une demande légitime de reconnaissance comme de « l'humilité », mais il y a ici quelque chose de plus fondamental à examiner.

L'humilité consiste à faire abstraction de votre propre intérêt, mais vous et celui qui utilise votre code n'êtes pas les seuls à être affectés par le choix de la licence de logiciel libre pour votre code. Quiconque utilise votre code dans un programme non libre essaie d'interdire cette possibilité aux autres, et si vous laissez faire, vous manquez à votre devoir de défendre cette liberté pour les autres. Quand vient le moment de défendre la liberté d'autrui, ne pas résister et laisser faire est un acte de faiblesse et non d'humilité.

Diffuser votre code sous une des licences BSD ou une des licences « non-copyleft » permissives n'est pas mal faire; le programme reste un logiciel libre et une contribution pour notre communauté. Mais il est néanmoins faiblement protégé et ce n'est pas, dans la plupart des cas, la meilleure manière de promouvoir la liberté des utilisateurs de modifier les logiciels et de les partager.

Traduction : Laurent Bertaud.
Révision : trad-gnu@april.org

Copyleft : Idéalisme pragmatique

Toute décision prise par une personne découle de ses valeurs et buts. Les gens peuvent avoir beaucoup de buts et valeurs différents ; la gloire, le profit, l'amour, la survivance, l'amusement, ou la liberté, ne sont qu'une partie des buts qu'une personne normale peut avoir. Quand ce but est d'aider les autres aussi bien que soi-même, cela s'appelle de l'idéalisme.

C'est un but idéaliste qui motive mon travail pour le logiciel libre : propager la liberté et la coopération. Je veux encourager la diffusion des logiciels libres[1] et le remplacement des logiciels propriétaires qui empêchent la coopération, et rendre ainsi notre société meilleure.

C'est pour cette raison fondamentale que la GNU General Public License est écrite de cette manière — comme un copyleft[2]. Tout code ajouté à un programme couvert par la GPL doit être un programme libre, même s'il est placé dans un fichier séparé. Je mets mon code à disposition pour une utilisation dans des logiciels libres, et pas pour un usage avec des logiciels propriétaires, afin d'encourager ceux qui programment des logiciels à les rendre libres également. Je suppose que, comme les développeurs de logiciels propriétaires utilisent le copyright pour nous empêcher de partager, nous, coopérateurs, pouvons utiliser le copyright pour donner aux autres coopérateurs un avantage bien à eux : ils peuvent utiliser notre code.

Ceux qui utilisent la GNU GPL n'ont pas tous ce but. Il y a plusieurs années, on a demandé à un de mes amis de rééditer un programme copylefté sous des termes non copyleftés, et il a plus ou moins répondu ainsi :

> « Parfois je travaille sur des logiciels libres et parfois sur des logiciels propriétaires ; mais quand je travaille sur des logiciels propriétaires, j'exige d'être *payé* ».

Il voulait bien partager son travail avec une communauté qui partageait les logiciels mais ne voyait aucune raison d'aider un entreprise commerciale dont les produits seraient interdits à notre communauté. Son but était différent du mien mais il a décidé que la GNU GPL le satisfaisait également.

Si vous voulez accomplir quelque chose dans le monde, l'idéalisme ne suffit pas ; le choix d'une méthode qui mène à l'accomplissement de ce but est nécessaire. En d'autres termes, vous devez être « pragmatique ». La GPL est-elle pragmatique ? Regardons ses accomplissements.

Considérons GNU C++. Pourquoi existe-t-il un compilateur C++ libre ? Uniquement parce que la GNU GPL indiquait qu'il devait être libre. MCC, un consortium industriel, a développé GNU C++ à partir du compilateur GNU C. En temps normal, MCC rend sa production aussi propriétaire que possible. Mais ils ont fait une interface C++ libre parce que c'était la seule possibilité de l'éditer que leur laissait la GNU GPL. L'interface C++ comportait beaucoup de nouveaux fichiers, mais comme ils étaient prévus pour être liés à GCC, la GPL s'appliquait à eux. Le bénéfice pour notre communauté est évident.

Considérons GNU Objective C. Au début, NeXT voulait rendre cette interface propriétaire ; ils avaient l'intention de l'éditer sous la forme de fichiers `.o` et de laisser aux utilisateurs le soin de les attacher au reste de GCC, pensant pouvoir ainsi contourner les conditions de la GPL. Mais nos juristes ont dit que cela n'esquivait pas ces conditions et était interdit. Et ainsi l'interface Objective C fut un logiciel libre.

Ces exemples datent de plusieurs années, mais la GNU GPL continue à nous apporter toujours plus de logiciels libres.

Beaucoup de bibliothèques GNU sont couvertes par la GNU Lesser Library General Public License, mais pas toutes. Readline, qui met en œuvre l'édition de ligne de commande, est une des bibliothèques GNU couvertes par la GNU GPL classique. Un jour, j'ai découvert un programme propriétaire conçu pour utiliser Readline, et j'ai dit au développeur que cela était interdit. Il aurait pu sortir du programme l'édition de ligne de commande, mais ce qu'il fit en fait, fut de le rééditer sous la GPL. Maintenant, c'est un logiciel libre.

Les programmeurs qui conçoivent des améliorations pour GCC (ou Emacs, Bash, Linux ou tout programme couvert par la GPL) sont souvent employés par des entreprises ou des universités. Lorsque le programmeur veut remettre son amélioration à la communauté et voir son code dans la prochaine édition, il est probable que son patron lui dise : « Attendez un peu ; votre code nous appartient ! Nous ne voulons pas le partager ; nous avons décidé de placer votre version améliorée dans un logiciel propriétaire ».

C'est à ce moment qu'intervient la GNU GPL. Le programmeur montre à son patron que ce logiciel propriétaire serait une violation de copyright, et ce dernier réalise qu'il ne lui reste que deux possibilités : publier le code en tant que logiciel libre ou pas du tout. Il laisse presque toujours le programmeur faire ce qu'il voulait initialement et le code est inclus dans la prochaine version.

La GNU GPL n'est pas M. Chic Type. Elle dit « non » à certaines choses qu'on aimerait parfois faire. Certains utilisateurs disent qu'il est dommage que la GPL « exclut » certains développeurs de logiciels propriétaires qui « auraient besoin d'être amenés à la communauté du logiciel libre ».

Mais nous ne les excluons pas de notre communauté ; ils ont choisi de ne pas y entrer. Décider de concevoir des logiciels propriétaires, c'est décider de rester en dehors de notre communauté. Appartenir à notre communauté signifie prendre part à notre coopération ; nous ne pouvons pas « les amener à notre communauté » s'ils ne le désirent pas.

Ce que nous *pouvons* faire, c'est les inciter à nous rejoindre. La GNU GPL est faite pour utiliser comme appâts les logiciels que nous possédons : « Si vous faites des logiciels libres, vous pourrez utiliser ces codes ». Bien sûr, cela ne les convaincra pas tous, mais nous en gagnerons un de temps en temps.

La réalisation de logiciels propriétaires ne participe pas à notre communauté, mais ses développeurs souhaitent souvent une aide de notre part. Les utilisateurs de logiciels libres peuvent offrir des encouragements pour l'égo des développeurs de logiciels libres — reconnaissance et gratitude — mais ils peuvent être tentés lorsqu'une entreprise leur dit : « Laissez-nous juste placer votre logiciel dans notre programme propriétaire et il sera utilisé par des milliers de gens ! » La tentation peut être forte, mais à long terme, nous nous sentons tous mieux si nous y avons résisté.

Il est plus difficile de reconnaître les pressions et tentations quand elles viennent indirectement, au travers d'organisations pour les logiciels libres qui ont adopté une politique satisfaisant aux logiciels propriétaires. Le Consortium X (et son successeur l'Open Group) en sont un exemple : fondé par des sociétés qui font des logiciels propriétaires, ils se sont efforcés de persuader les programmeurs de ne pas utiliser de copyleft pendant des décennies. Maintenant que l'Open Group a fait de X11R6.4 un logiciel propriétaire[3], ceux d'entre nous qui ont résisté à cette pression sont heureux de l'avoir fait.

En septembre 1998, plusieurs mois après que X11R6.4 ait été distribué sous une licence non-libre,

l'Open Group a revu sa décision et l'a redistribué sous la même licence de logiciel libre non-copylefté que X11R6.3. Merci, l'Open Group ; mais ce revirement ultérieur n'infirme pas les conclusions que nous avions tirées du fait que l'ajout de restrictions était *possible*.

Pour parler pragmatiquement, avoir une vision à plus long terme affermira votre volonté de résister à cette pression. Si vous concentrez votre attention sur la liberté et la communauté que vous pouvez bâtir en restant ferme, vous trouverez la force de le faire. « Restez debout pour quelque chose ou vous tomberez pour rien ».

Et si des personnes cyniques ridiculisent la liberté, la communauté… si des « réalistes forcenés » disent que le profit est le seul idéal… ignorez-les et utilisez le copyleft tout de même.

[1] Pourquoi le Copyleft ? (page 10)
[2] Qu'est-ce que le copyleft ? (page 7)
[3] http://www.gnu.org/philosophy/x.fr.html

Traduction : ?
Révision : trad-gnu@april.org

L'anarchisme triomphant : Le logiciel libre et la mort du copyright

La propriété logicielle: le paradoxe théorique

LOGICIEL: aucun autre mot ne transporte plus profondément les effets pratiques et sociaux de la révolution numérique. Originalement, le terme était purement technique et servait à définir les parties d'un système d'ordinateur qui, contrairement au « matériel » (hardware) lequel consistait invariablement en un système électronique, pouvait être librement modifié. Les premiers logiciels se réduisaient à la configuration du branchement des câbles ou des interrupteurs sur les panneaux extérieurs d'un appareil électronique, mais à partir du moment où des moyens linguistiques d'altérer le comportement de l'ordinateur ont été développés, le « logiciel » a surtout signifié l'expression d'un langage plus ou moins humainement lisible qui décrivait et contrôlait à la fois le comportement de la machine[1].

C'était alors le cas et ça l'est toujours. La technologie basée sur la manipulation d'informations numériquement encodées est maintenant socialement dominante dans la plupart des aspects de la culture humaine des sociétés « développées »[2]. Le mouvement de la représentation analogique vers la représentation numérique (dans la vidéo, la musique, l'imprimerie, les télécommunications et même la chorégraphie, le culte religieux et le plaisir sexuel) transforme potentiellement toutes les formes de la créativité symbolique humaine en logiciels, qui sont des instructions modifiables décrivant et contrôlant le comportement de machines. La division entre le matériel et le logiciel est maintenant observée dans le monde réel ou social par une forme de dérivation régressive caractéristique de la pensée scientifique occidentale. Cette division est devenue une nouvelle manière d'exprimer le conflit entre les idées du déterminisme et de la libre pensée, de la nature et de l'éducation, des gènes et de la culture. Notre « matériel », câblé génétiquement, est notre nature et nous détermine. Notre éducation est le « logiciel », qui établit notre programmation culturelle et qui constitue notre liberté relative. Et ainsi de suite pour ceux qui n'ont pas peur du verbiage[3]. Ainsi, le « logiciel » devient une métaphore viable pour toute l'activité symbolique, apparemment séparée du contexte technique de l'origine du mot, en dépit du malaise soulevé chez les personnes qualifiées lorsque le terme circule ainsi en supprimant la signification conceptuelle de sa dérivation[4].

Mais si l'adoption massive de la technologie numérique, pour une utilisation par ceux qui ne comprennent pas les principes de son fonctionnement, permet l'emploi massif de la métaphore du « logiciel », elle ne nous permet pas en fait d'ignorer que les ordinateurs sont maintenant partout dans notre tissu social. Le mouvement de l'analogique vers le numérique est plus important pour la structure de nos relations sociales et légales que les mouvement du statut au contrat plus célèbres bien que moins certains[5]. C'est une mauvaise nouvelle pour des intellectuels en droit qui ne le comprennent pas, c'est pourquoi le nombre de ceux qui font maintenant semblant de comprendre progresse. Potentiellement, cependant, cette grande transition est aussi une bonne nouvelle pour ceux qui peuvent s'approprier ce terrain nouvellement découvert. C'est pourquoi les actuels « propriétaires » de logiciels soutiennent et encouragent si fortement l'ignorance de tous les autres. Malheureusement pour eux, pour des raisons familières aux théoriciens en droit, qui n'ont pas encore compris la manière d'appliquer leur logique traditionnelle sur ce terrain, l'astuce ne marchera pas. Ce texte explique pourquoi[6].

Nous devons commencer par réfléchir à l'essence technique des périphériques familiers qui nous entourent dans l'ère du « logiciel culturel ». Un lecteur de CD est un bon exemple. Son entrée principale est une séquence de bits lue à partir d'un disque optique. La séquence de bits décrit la musique en termes de mesures de fréquence et d'amplitude, prises 44 000 fois par seconde, sur chacun des deux canaux sonores. La sortie primaire du lecteur est un signal sonore analogique[7]. Comme dans tout le reste du monde numérique, la musique est vue par le lecteur de CD comme de la pure information numérique ; un enregistrement particulier de la neuvième symphonie de Beethoven, interprétée par Arturo Toscanini, l'orchestre symphonique et la chorale de la NBC est (en omettant quelques chiffres insignifiants) 1276749873424, alors que le dernier enregistrement particulièrement pervers des variations de Goldberg par Glenn Gould est (de la même manière, particulièrement tronqué) 767459083268.

Assez bizarrement, ces deux nombres sont « copyrightés ». Cela signifie, je suppose, que vous ne pouvez pas posséder une autre copie de ces nombres, une fois figés sur une forme physique, à moins que vous n'ayez un permis pour les utiliser. Et vous ne pouvez pas transformer 767459083268 en 2347895697 pour vos amis (corrigeant ainsi l'opinion ridicule de Gould sur le tempo) sans créer un « travail dérivé », pour lequel une licence est nécessaire.

En même temps, un même support optique contient un autre nombre, appelons le 7537489532. Celui-ci est un algorithme pour la programmation linéaire de grands systèmes avec de multiples contraintes, ce qui est utile, par exemple, si vous désirez une utilisation optimale de votre matériel roulant dans la gestion d'une voie ferrée de marchandises. Ce nombre est « breveté » (aux États-Unis), ce qui signifie que vous ne pouvez pas puiser dans 7537489532 pour vous-même, ou pour « pratiquer l'art » du brevet d'une autre manière en ce qui concerne la résolution de problèmes de programmation linéaire, quelque soit la manière dont vous arriviez à l'idée (ce qui inclut de la trouver vous même), à moins que vous n'ayez une licence du propriétaire du nombre.

Et puis il y a 9892454959483. Celui-ci est le code source de MS-Word. En plus d'être « copyrighté », celui-ci est un secret industriel. Ce qui signifie que si vous obtenez ce nombre de Microsoft, et que vous le donnez à quelqu'un d'autre, vous pouvez être puni.

Enfin, il y a 588832161316. Il ne fait rien, c'est juste 767354 élevé au carré. Autant que je sache, il n'est pas possédé par quelqu'un sous le coup d'une de ces catégories. Pas encore.

À ce point, nous devons nous occuper de la première objection d'érudits. Elle vient d'une créature connue sous le nom de l'IPdroïde. Le droïde a une pensée sophistiquée et une vie culturelle. Il aime beaucoup les dîners élégants et les conférences ministérielles sur les accords TRIPS (ou ADPIC), sans oublier ses apparitions fréquentes sur MSNBC. Il veut que vous sachiez que je commets l'erreur de confondre l'incarnation avec la propriété intellectuelle. Ce n'est pas le nombre qui est breveté, idiot, c'est l'algorithme de Kamarkar. Le nombre *peut* être copyrighté, car le copyright couvre les qualités significatives d'une incarnation tangible particulière d'une idée (dans lequel quelques propriétés fonctionnelles peuvent mystérieusement se mêler, à condition qu'elles ne soient pas emmêlées), mais pas l'algorithme. Même si le nombre n'est pas brevetable, l'« enseignement » même du nombre en rapport à la construction de voies ferrées tombe juste sous le coup du brevet. Et le nombre représentant le code source de MS-Word peut être un secret industriel, mais si vous le trouvez vous-même (en effectuant des manipulations arithmétiques sur d'autres nombres fournis par Microsoft, par exemple, ce qui est connu comme l'« ingénierie inverse » (reverse engineering), vous ne serez pas puni, du moins si

vous vivez dans certaines parties des États-Unis[8].

Ce droïde, comme les autres droïdes, a souvent raison. La condition pour être un droïde est de tout savoir sur quelque chose et rien sur tout le reste. Le droïde a établi, par son intervention opportune et empressée, que le système actuel de la propriété intellectuelle contienne de nombreuses caractéristiques imbriquées et ingénieuses. Ces complexités se combinent pour permettre aux professeurs d'être érudits, aux députés d'obtenir des contributions à leur campagne, aux avocats de porter de beaux costumes et des mocassins à pompon, et à Murdoch[9] de s'enrichir. Ces complexités se sont principalement développées à une époque de distribution industrielle de l'information, quand celle-ci était inscrite dans des formes analogiques sur des objets physiques qui coûtaient significativement cher et à la construction, au déplacement, à la vente. Appliquée à de l'information numérique, qui se déplace sans friction à travers le réseau, et qui a un coût marginal par copie nul, tout cela fonctionne toujours, plus ou moins, tant que vous continuez à loucher.

Mais ce n'était pas ce sur quoi je discourais. Je voulais faire remarquer autre chose: que notre monde consiste de manière grandissante en de grands nombres (ou encore de séquences de bits) et que (pour des raisons qui n'ont rien à voir avec les propriétés émergeantes des nombres eux-mêmes) le système légal s'est actuellement engagé à traiter des nombres similaires de manière radicalement différente. Personne ne peut dire, simplement en regardant un nombre qui est long de 100 millions de chiffres, s'il est sous le coup d'un brevet, du copyright, sous la protection d'un secret industriel ou s'il est vraiment « possédé » par quelqu'un. Alors le système légal que nous avons (bienheureux, comme nous le sommes par ses conséquences, si nous sommes des enseignants sur le copyright, des députés, des lobbyistes de Gucci-gulch [10] ou Grand Ruppert lui-même) est contraint à traiter des choses non distinguables de manière différente.

Maintenant, en tant qu'historien du droit concerné par le développement séculaire (c'est à dire, sur le très long terme) de la pensée légale, j'affirme que les régimes légaux basés sur des distinctions fines mais non déterministes entre des sujets similaires, sont radicalement instables. Ils tombent à l'eau avec le temps, car chaque occasion d'application de leurs règles est une invitation pour au moins une des parties, à revendiquer qu'au lieu de faire partie de la catégorie A, le sujet particulier du litige devrait être jugé comme faisant partie de la catégorie B, là où les règles leur seront plus favorables. Ce jeu (celui où une machine à écrire devrait être jugée comme étant un instrument de musique pour des besoins de régulation de vitesse sur voie ferrée et où une pelleteuse à vapeur est un véhicule à moteur) est le B.A.-ba de l'ingéniosité légale. Mais quand les catégories légales, approuvées conventionnellement, ont besoin de juges pour distinguer celles qui sont identiques, le jeu est infiniment long, infiniment coûteux et infiniment déplaisant pour le spectateur impartial[11].

Ces parties peuvent dépenser tout l'argent qu'elles veulent en autant d'avocats et de juges qu'elles peuvent se permettre (ce qui, pour les nouveaux « propriétaires » du monde digital, est assez peu), mais les lois qu'elles achètent ne fonctionneront pas en fin de compte. Tôt ou tard, les paradigmes s'effondreront. Bien sûr, si tard signifie d'ici deux générations, la distribution de la richesse et du pouvoir, sanctifiée entre temps, pourrait ne pas être réversible par un procédé moins radical qu'une *bellum servile* des esclaves de la télévision contre les magnats des médias. Alors savoir que l'histoire n'est pas du côté de Bill Gates n'est pas suffisant. Nous prédisons le futur d'une perception très limitée: nous savons que les règles existantes, qui ont jusqu'à présent la ferveur de la pensée conventionnelle solidement engagée derrière elle, ne signifient plus rien. Les parties dont nous parlons les utiliseront et en abuseront librement, jusqu'à ce que le courant dominant de l'opinion conservatrice « respectable »

reconnaisse leur mort, avec des résultats incertains. Mais les universitaires réalistes devraient déjà être en train de porter leur attention vers le besoin évident de nouvelles réflexions.

Lorsque nous atteignons ce point de la discussion, nous en arrivons à nous battre avec un autre protagoniste principal de l'imbécilité éduquée: l'écononain. Comme l'IPdroïde, l'écononain est une espèce de hérisson[12]. Mais là où le droïde soutient la logique par dessus l'expérience, l'écononain se spécialise dans une vision, énergétique et très orientée, mais complètement fausse, de la nature humaine. D'après la vision de l'écononain, chaque être humain est un individu possédant des « motivations », qui peuvent être rétrospectivement exhumées, en imaginant l'état de son compte en banque à des moments différents. Ainsi, de cette manière, l'écononain est obligé d'objecter que sans les règles que je tourne en dérision, il n'y aurait pas de motivation pour créer ce que ces règles traitent comme de la propriété: sans la possibilité de priver les gens de la musique, il n'y aurait pas de musique, car personne ne serait sûr d'être payé pour la créer.

La musique n'est pas vraiment notre sujet ; le logiciel dont je parle en ce moment est de la vieille école: les programmes d'ordinateurs. Mais comme l'écononain est déterminé à s'occuper du sujet à la hâte, et parce que, comme nous l'avons vu, il n'est plus vraiment possible de distinguer les programmes d'ordinateurs des morceaux de musique, nous devons en dire un mot ou deux. Au moins nous pouvons avoir la satisfaction de nous livrer à un argument *ad pygmeam*. Quand l'écononain devient riche, d'après mon expérience, il va à l'opéra. Mais peu importe le nombre de fois où il écoute *Don Giovanni*, il ne lui vient jamais à l'esprit que le destin de Mozart aurait dû, dans cette logique, avoir entièrement découragé Beethoven ou que nous pouvons écouter la *Flûte Enchantée*, même si Mozart savait très bien qu'il ne serait pas payé. En fait, la *Flûte Enchantée*, la *Passion selon Saint Mathieu* et les motets de Carlo Gesualdo, dont la femme a été assassinée, sont tous des parts de la tradition, vieille de plusieurs siècles, du logiciel libre, au sens le plus large. Ce que l'écononain n'accepte jamais vraiment.

Le problème basique du nain est que les « motivations » sont purement et simplement une métaphore. Et, en tant que métaphore pour décrire l'activité créatrice humaine, elle est plutôt minable. Je l'ai déjà dit auparavant[13] mais la meilleure métaphore s'est présentée le jour où Michael Faraday a découvert le premier ce qui arrivait quand on enroule un rouleau de câble autour d'un aimant et qu'on fait tourner l'aimant. Du courant circule dans un tel câble, mais nous ne nous demandons pas quelle est la motivation des électrons à quitter leur position initiale. Nous affirmons que le courant résulte d'une propriété émergente du système, que nous appelons l'induction. La question que nous nous posons est: « quelle est la résistance du câble ? ». Alors, la métaphore de Moglen, corollaire à la loi de Faraday, dit que si vous enroulez l'Internet autour de chaque personne de la planète, et que vous faites tourner la planète, le logiciel parcourt le réseau. C'est une propriété émergente des esprits humains connectés. Ils créent des choses pour le plaisir de l'autre, et pour surmonter la sensation désagréable d'être trop seul. La seule question à poser est: quelle est la résistance du réseau ? La métaphore de Moglen, corollaire à la loi de Ohm, dit que la résistance du réseau est directement proportionnelle à la force du champ du système de la « propriété intellectuelle ». Alors la réponse correcte à l'écononain est: résistez à la résistance.

Bien sûr, c'est très joli dans la théorie. « Résistez à la résistance » sonne bien, mais nous aurions un problème sérieux, en dépit de la théorie, si le nain avait raison, et nous nous retrouverions sous-produisant de bons logiciels car nous ne permettrions pas de les posséder. Mais les nains et les droïdes sont des formalistes de différentes sortes, et l'avantage du réalisme est que si vous commencez par les faits, les faits sont toujours de votre côté. Il s'avère que traiter le logiciel comme une propriété produit

de mauvais logiciels.

La propriété logicielle: le problème pratique

Pour comprendre pourquoi la transformation des logiciels en propriétés produit de mauvais logiciels, nous avons besoin d'une introduction à l'histoire de l'art. En fait, nous ferions mieux de commencer par le mot « art » lui-même. En effet, la programmation des ordinateurs combine le raisonnement déterministe et l'invention littéraire.

Au premier coup d'oeil, bien sûr, le code source apparaît comment une forme non littéraire de composition[14]. Le principal but d'un programme d'ordinateur est qu'il fonctionne, c'est-à-dire qu'il s'accomplisse en suivant des spécifications décrivant formellement ses résultats, en fonction de ses entrées. À ce niveau de généralité, le contenu fonctionnel du programme est tout ce qu'il y a d'apparent.

Mais les programmes d'ordinateurs existent en tant que parties de systèmes informatiques, qui sont des ensembles intéragissants de matériels, de logiciels et d'êtres humains. Les composants humains d'un système informatique incluent non seulement les utilisateurs, mais aussi (ce qui est potentiellement différent) les personnes qui maintiennent et améliorent le système. Le code source communique non seulement avec l'ordinateur qui exécute le programme, à travers l'intermédiaire du compilateur qui produit le code objet (en langage machine), mais aussi avec les autres programmeurs.

La fonction du code source, lorsqu'on le met en relation avec d'autres êtres humains, n'est pas largement comprise par les non-programmeurs, qui ont tendance à penser que les programmes informatiques sont incompréhensibles. Ils seraient surpris d'apprendre que la majorité de l'information contenue dans la plupart des programmes est, du point de vue du compilateur ou des autres processeurs de langage, du « commentaire », une substance non fonctionnelle. Les commentaires, bien sûr, sont adressés à ceux qui peuvent avoir besoin de résoudre un problème, de modifier ou d'améliorer les fonctions du programme. Dans la plupart des langages informatiques, bien plus d'espace est consacré à expliquer aux autres ce que le programme fait, que de dire à l'ordinateur comment l'exécuter.

La conception des langages informatiques a toujours été effectuée selon le double besoin d'une spécification complète de l'exécution par la machine et d'une description informative pour les lecteurs humains. On pourrait identifier trois stratégies basiques dans la conception de langages informatiques pour atteindre ce double but. La première, suivie initialement pour la conception des langages spécifiques à des produits matériels et collectivement connus sous le nom d'« assembleurs », distinguait essentiellement les portions du programme communicant avec la machine ou avec l'humain. Les instructions de l'assembleur sont très proches des instructions en langage machine: en général, une ligne d'un programme en assembleur correspond à une instruction dans le langage natif de la machine. Le programmeur contrôle l'opération de la machine au niveau le plus spécifique et (s'il est bien discipliné) s'engage à disposer des commentaires tout au long des instructions en langage machine, créant toutes les quelques centaines d'instructions des « blocs de commentaires », qui fournissent un résumé de la stratégie du programme ou qui documentent les structures de données majeures que le programme manipule.

Une seconde approche, illustrée de manière caractéristique par le langage COBOL (qui signifie « *COmmon Business-Oriented Language* » ou « Langage Commun Orienté Industrie »), était de faire ressembler le programme lui-même à un ensemble de directives en langage naturel, d'un style

déplorable mais théoriquement lisible par un humain. Une ligne de code COBOL pourrait dire, par exemple, « MULTIPLY PRICE TIMES QUANTITY GIVING EXPANSION[15]. A l'origine, quand le Pentagone et les experts de l'industrie ont commencé à concevoir COBOL en commun, au début des années 60, cela semblait une approche prometteuse. Les programmes COBOL apparaissaient largement auto-documentés, autorisant à la fois le développement d'équipes de travail capables de collaborer à la création de gros programmes et la formation de programmeurs qui, bien qu'étant des travailleurs spécialisés, n'avaient pas besoin de connaître la machine aussi intimement qu'ils en auraient eu besoin pour des programmes en assembleur. Mais le niveau de généralité auquel de tels programmes s'auto-documentaient était mal choisi. Une expression plus formelle et compressée des détails opérationnels « expansion = prix x quantité », par exemple, était mieux appropriée, même pour les applications industrielles ou financières où les lecteurs et concepteurs de programmes sont habitués aux expressions mathématiques. Et, cependant, la procédure de description des structures de données, tout comme du contexte opérationnel plus large du programme, n'étaient pas rendues obsolètes par la verbosité du langage dans lequel les détails de l'exécution était spécifiés.

En conséquence, les concepteurs de langages de la fin des années 60 ont commencé à expérimenter des formes d'expression dans lesquelles le mélange de détails opérationnels et d'informations non fonctionnelles nécessaires à la modification ou à la correction était plus subtil. Quelques concepteurs choisirent la voie de langages hautement symboliques et compressés, dans lesquels le programmeur manipulait des données abstraites, afin que « A x B » puisse signifier la multiplication de deux entiers, deux nombres complexes, deux gros tableaux ou d'un tout autre type de données capable d'une opération appelée « multiplication », et que cela soit géré par l'ordinateur sur les bases du contexte des variables « A » et « B » au moment de l'exécution[16]. Comme on pensait que cette approche résultait en des programmes extrêmement concis, le problème de rendre le code compréhensible par ceux qui chercheraient plus tard à le modifier ou le corriger était simplifié. En cachant les détails techniques des opérations informatiques, et en mettant l'accent sur l'algorithme, les langages pouvaient être conçus pour être meilleurs que le français ou tout autre langage naturel pour l'expression des procédés séquentiels. Les commentaires ne seraient plus non seulement inutiles mais gênants, tout comme les métaphores utilisées pour faire comprendre les concepts mathématiques en français embrouillent plus qu'elles n'éclaircissent.

Comment nous avons créé la pagaille Microbrain

Ainsi, l'histoire des langages de programmation reflète directement le besoin de trouver des formes de communication de l'humain à la machine aussi efficaces dans la transmission d'idées complexes aux lecteurs humains. « L'expressivité » est devenue une propriété des langages de programmation, non pas parce qu'elle facilite l'intégration, mais parce qu'elle facilite la création collaborative et la maintenance de systèmes logiciels de complexité croissante.

À première vue, cela semble justifier l'application de la pensée traditionnelle du copyright aux travaux résultant. Bien que faisant entrer en jeu des éléments « fonctionnels », les programmes d'ordinateur contenaient des fonctions « expressives » d'une suprême importance. La doctrine du copyright reconnaît la fusion de la fonction et de l'expression comme une caractéristique de bien des formes de travaux copyrightés. Le « code source » contient à la fois les instructions machine nécessaires à l'opération fonctionnelle et les « commentaires » expressifs à l'attention des lecteurs humains. En tant que tel, il était un bon candidat, pour être traité par le copyright.

C'est vrai, à partir du moment où l'on comprend que la composante expressive du logiciel est présente seulement dans le but de faciliter la création de « travaux dérivés ». Si elle n'était pas là dans le but de faciliter la modification, les éléments expressifs des programmes seraient entièrement surérogatifs et le code source ne pourrait pas plus être copyrighté que le code objet (la sortie du processeur du langage), épuré de tout sauf les caractéristiques fonctionnelles du programme.

L'état de l'industrie de l'informatique à travers les années 1960 et 1970, quand les lois implicites de la programmation informatique sophistiquée ont été établies, dissimulait la tension implicite à la situation. Lors de cette période, le matériel coûtait cher. Les ordinateurs étaient des ensembles de plus en plus grands et complexes de machines, et l'industrie de la conception et de la construction d'un tel étalage de machines pour un usage général était dominé (pour ne pas dire monopolisé) par une seule entreprise. Or IBM donnait ses logiciels. Pour sûr, cette entreprise possédait les programmes que ses employés écrivaient et elle plaçait sous copyright ses codes sources. Mais elle distribuait aussi les programmes (ce qui inclut le code source) à ses clients sans coût supplémentaire, et les encourageait à écrire et à partager des améliorations et des adaptations aux programmes ainsi distribués. Pour un constructeur de matériel dominant, cette stratégie était sensée: de meilleurs programmes faisaient vendre plus de matériel et c'était là que la rentabilité du marché demeurait.

Les ordinateurs, à cette période, ont eu tendance à s'agglomérer à l'intérieur d'organisations particulières, mais pas à communiquer largement avec d'autres. Les logiciels dont on avait besoin pour fonctionner n'étaient pas distribués à travers un réseau, mais sur des bobines de bande magnétique. Le système de distribution avait tendance à centraliser le développement de logiciels, afin que les clients d'IBM soient libres d'apporter des modifications et des améliorations aux programmes. Ces modifications étant partagées tout d'abord avec IBM, qui décidait alors d'incorporer ces modifications dans les versions distribuées du logiciel sur un modèle de développement centralisé. Ainsi, pour deux raisons importantes, les meilleurs logiciels du monde étaient libres: ils ne coûtaient rien à acquérir et les termes dans lesquels ils étaient fournis autorisaient et encourageaient tout à la fois l'expérimentation, la modification et l'amélioration[17]. Que le logiciel en question soit la propriété d'IBM sous la loi actuelle du copyright a certainement établi quelques limites théoriques sur la possibilité des utilisateurs à distribuer leurs modifications ou adaptations aux autres, mais dans la pratique, les logiciels pour supercalculateurs étaient développés de manière coopérative par le constructeur de matériel dominant et par ses utilisateurs techniquement compétents, employant les ressources de distribution vers la communauté des utilisateurs. Le droit d'exclure les autres, un des plus importants « bâtons dans le paquet » des droits de la propriété (pour reprendre une image que la Cour Suprême des États-Unis affectionne), était pratiquement négligeable ou même indésirable, au coeur de l'industrie du logiciel[18].

Après 1980, tout était différent. Le monde des supercalculateurs a laissé la place, en dix ans, au monde de l'ordinateur personnel. Et, en parallèle au développement de l'industrie, l'élément unique le plus important des logiciels fonctionnant sur cet ordinateur personnel de base, le système d'exploitation, est devenu le seul produit significatif d'une entreprise qui ne faisait pas de matériel. Des logiciels de grande qualité ont cessé de faire partie de la stratégie de différenciation des produits des producteurs de matériel. À la place, une entreprise avec une part prédominante du marché, et avec l'absence d'intérêt ordinaire pour l'adoption de la diversité des quasi-monopolistes, dictait les pratiques de l'industrie du logiciel. Dans un tel contexte, le droit d'exclure les autres de la participation à la formation du produit est devenu d'une importance capitale. La puissance de Microsoft dans le marché est constituée

entièrement par la propriété du code source de Windows.

Pour Microsoft, la création par d'autres de « travaux dérivés », qu'on connaît ailleurs sous le nom de corrections et d'améliorations, a menacé le capital central de leurs affaires. En effet, comme des procédures judiciaires ultérieures ont eu tendance à l'établir, la stratégie commerciale de Microsoft a été de trouver des idées innovantes ailleurs dans le marché du logiciel, de les acheter, et soit de les supprimer, soit de les incorporer dans leur propre produit propriétaire. Le maintien du contrôle sur la manipulation basique des ordinateurs construits, vendus, possédés et utilisés par d'autres représentait un levier essentiel et profitable sur le développement de la culture[19] ; le droit d'exclure est revenu au centre du concept de la propriété logicielle.

Le résultat, d'aussi loin que la qualité du logiciel soit concernée, était désastreux. Le monopole était détenu par une entreprise riche et puissante qui employait un grand nombre de programmeurs, mais qui ne pouvait effectivement pas se permettre le nombre de testeurs, de concepteurs et de développeurs requis pour produire des logiciels flexibles, robustes et techniquement innovants, appropriés au vaste ensemble des conditions sous lesquelles les ordinateurs personnels, de plus en plus omniprésents, fonctionnaient. Sa stratégie marketing fondamentale impliquait la conception de son produit pour les utilisateurs les moins avancés techniquement et l'utilisation de la « peur, de l'incertitude et du doute » (connu au sein de Microsoft comme le « FUD ») pour tenir les utilisateurs compétents à l'écart des concurrents potentiels, dont la survie à long terme en face de la puissance de marché de Microsoft a toujours été en question.

Sans la constante interaction entre les utilisateurs capables de corriger des problèmes et d'améliorer des fonctionnalités et le constructeur de système d'exploitation, l'inévitable détérioration de la qualité est inéluctable. Mais comme la révolution de l'ordinateur personnel a étendu le nombre d'utilisateurs de manière exponentielle, pratiquement tous ceux qui sont entrés en contact avec le système résultant n'ont pu poser d'élément de comparaison face au système qu'ils avaient l'habitude d'utiliser. Ignorants des standards de stabilité, de robustesse, de maintenance et d'efficacité précédemment établis dans le monde des supercalculateurs, on pouvait difficilement attendre de la part des utilisateurs d'ordinateurs personnels qu'ils comprennent à quel point, en termes relatifs, les logiciels du monopole fonctionnaient mal. Comme la puissance et la capacité des ordinateurs personnels se sont étendues rapidement, les défauts des logiciels étaient moins évidents, au sein de l'augmentation générale de la productivité. Les utilisateurs ordinaires, pour plus de la moitié effrayés par une technologie qu'ils ne comprenaient pratiquement pas, ont réellement fait bon accueil à la défectuosité des logiciels. Dans une économie subissant des transformations mystérieuses, avec la déstabilisation concomitante de millions de carrières, il était tranquillisant, d'une manière perverse, qu'aucun ordinateur personnel ne semble capable de fonctionner plus de quelques heures consécutives sans se planter. Bien qu'il soit frustrant de perdre du travail en cours chaque fois qu'une erreur imprévue arrive, l'évidente faillibilité des ordinateurs était intrinsèquement rassurante[20].

Rien de tout cela n'était nécessaire. La faible qualité des logiciels pour ordinateurs personnels aurait pu être évitée en faisant intervenir les utilisateurs dans le processus fondamentalement évolutionnaire de la conception et de l'implémentation de logiciels. Un mode Lamarckien, dans lequel les améliorations peuvent être effectuées partout, par tout le monde et héritées par tous les autres, aurait effacé le déficit, restaurant dans le monde de l'ordinateur personnel la stabilité et la robustesse des logiciels écrits dans un environnement quasi-propriétaire de l'ère des supercalculateurs. Mais le *business model* de Microsoft écartait l'héritage Lamarckien des améliorations de logiciels. La doctrine du copyright, en

général, et comme elle s'applique au logiciel en particulier, dénature le monde vers le créationisme ; dans ce cas, le problème était que BillG le Créateur était loin d'être infaillible et, en fait, il n'avait même pas essayé.

Comble d'ironie, la croissance du réseau a rendu l'alternative non-propriétarienne encore plus censée. Ce que les écrits scolaires et populaires dénomment comme une chose (« l'Internet ») est en fait le nom d'une condition sociale: le fait que tout le monde dans la société du réseau soit connecté directement, sans intermédiaire, à tous les autres[21]. L'interconnexion globale des réseaux a éliminé le goulet d'étranglement qui a requis un éditeur centralisé de logiciels pour rationaliser et distribuer les résultats de l'innovation individuelle dans l'ère des supercalculateurs.

Et ainsi, par une des petites ironies de l'histoire, le triomphe global des mauvais logiciels à l'âge de l'ordinateur personnel a été renversé par une combinaison surprenante de forces: la transformation sociale initiée par le réseau, une théorie européenne de l'économie politique longtemps mise de côté, et une petite bande de programmeurs à travers le monde mobilisés par une simple idée.

Les logiciels veulent être libres, ou comment nous avons appris à ne plus nous stresser et à aimer la bombe

Bien avant que le réseau des réseaux ne soit une réalité pratique, même avant qu'il ne soit une idée, il y avait un certain désir de faire fonctionner les ordinateurs sur la base de logiciels librement disponibles pour tout le monde. Ce désir avait emergé en tant que réaction contre les logiciels propriétariens de l'ère du supercalculateur, et demande une autre brève digression historique.

Bien qu'IBM soit le plus grand vendeur d'ordinateurs généralistes de l'ère du supercalculateur, il n'était pas le plus grand concepteur et constructeur d'un type de matériel donné. Le monopole du téléphone, détenu par l'American Telephone & Telegraph (AT&T), était en fait plus important que celui d'IBM, mais il utilisait ses produits en interne. Et c'est aux célèbres Bell Labs, départements de la recherche de la compagnie détenant le monopole du téléphone, à la fin des années 60, que les développements de langages décrits précédemment ont donné naissance à un système d'exploitation appelé Unix.

L'idée d'Unix était de créer un système d'exploitation simple, s'adaptant à toutes les échelles, pour tous les ordinateurs, des petits aux grands, que la compagnie du téléphone construisait pour elle-même. Atteindre ce but impliquait d'écrire un système d'exploitation, ni dans un langage machine ni dans un assembleur dont la forme linguistique était intégrante à une conception particulière du matériel, mais dans un langage plus expressif et généraliste. Celui choisi était aussi une invention des Bell Labs, appelé « C »[22]. Le langage C est devenu commun et même dominant, pour de nombreuses tâches de programmation, et à la fin des années 1970, le système d'exploitation Unix écrit dans ce langage a été transféré (ou « porté », dans le jargon professionnel) sur des ordinateurs de nombreux constructeurs et de nombreuses conceptions différentes.

AT&T a largement distribué Unix, et en raison de la conception même du système d'exploitation, la compagnie devait effectuer cette distribution sous forme de code source C. Mais AT&T a conservé la propriété du code source et a contraint les utilisateurs à acheter des licences qui ont interdit la redistribution et la création de travaux dérivés. Les gros centres informatiques, industriels ou académiques, pouvaient se permettre d'acheter de telles licences, mais pas les individus. En même temps, les restrictions des licences interdisaient à la communauté des utilisateurs qui utilisaient Unix de

l'améliorer d'une manière évolutionnaire plutôt qu'épisodique. Et comme les programmeurs à travers le monde commençaient à aspirer à une révolution de l'ordinateur personnel (et même à l'attendre), le statut « non libre » d'Unix a commencé à devenir une source de problèmes.

Entre 1981 et 1984, un homme a imaginé une croisade pour changer la situation. Richard M. Stallman, alors employé au laboratoire d'intelligence du MIT, a conçu le projet d'une reconception et d'une implémentation indépendantes et coopératives d'un système d'exploitation qui serait constitué de vrais logiciels libres. De la bouche de Stallman, le logiciel libre serait une question de liberté, pas de prix. Tout le monde pourrait librement modifier et redistribuer de tels logiciels ou même les vendre, avec juste la restriction de ne pas essayer de réduire les droits de ceux à qui ils sont redistribués. De cette manière, le logiciel libre pourrait devenir un projet auto-organisé, dans lequel aucune innovation ne serait perdue à travers l'exercice des droits de la propriété. Le système, tel que Stallman l'a décidé, serait appelé GNU, qui signifie (c'est un des exemples initiaux des acronymes récursifs qui ont caractérisé le logiciel libre depuis) « GNU N'est pas Unix ». Malgré des doutes sur la conception fondamentale d'Unix aussi bien que sur ses termes de distribution, GNU était conçu pour bénéficier de la large (bien que non libre) distribution de sources d'Unix. Stallman a commencé le projet GNU en écrivant des composants du système final qui étaient aussi conçus pour fonctionner sans modification sur les systèmes Unix existants. Le développement des outils GNU pouvait ainsi se faire directement dans l'environnement des universités et des autres centres de calcul avancé à travers le monde.

L'échelle d'un tel projet était immense. D'une manière ou d'une autre, des programmeurs volontaires devaient être recrutés, organisés et incités à construire tous les outils nécessaires à la construction ultime. Stallman lui-même était l'auteur initial de plusieurs outils fondamentaux. De petites ou grandes équipes de programmeurs contribuaient ailleurs aux autres outils et les attribuaient au projet de Stallman ou les distribuaient directement. Quelques endroits à travers le réseau du développement sont devenus des archives pour le code source de ces composants de GNU, et au long des années 80, les outils GNU ont gagné la reconnaissance et l'acceptation par les utilisateurs d'Unix à travers le monde. Les outils GNU sont devenus synonymes de stabilité, de fiabilité et de bonne maintenance, tandis que les capacités profondes de Stallman à concevoir ont continué à devancer et à fournir des buts au processus. La récompense d'un prix de la fondation MacArthur, accordée en 1990 à Stallman, était une reconnaissance appropriée pour ses innovations conceptuelles et techniques et leurs conséquences sociales.

Le projet GNU et la Free Software Foundation, à qui il a donné vie en 1985, n'étaient pas les seules sources d'idées sur le logiciel libre. Plusieurs formes de copyright ont commencé à se développer dans la communauté universitaire, pour la plupart autour de l'environnement Unix. L'université Berkeley de Californie a entamé la conception et l'implémentation d'une autre version d'Unix pour une libre distribution dans la communauté universitaire. L'Unix BSD, tel que nous le connaissons maintenant, a aussi considéré l'Unix d'AT&T comme un standard conceptuel. Le code était largement distribué et constituait un réservoir d'outils et de techniques, mais les termes de sa licence limitaient la portée de son utilisation, alors que l'élimination de code propriétaire spécifique au matériel de la distribution signifiait que personne ne pouvait vraiment construire un système d'exploitation fonctionnel pour un ordinateur particulier à partir de BSD. D'autres travaux universitaires se sont aussi terminés par des logiciels presque libres ; par exemple, l'interface graphique (ou GUI, pour *Graphical User Interface*) des systèmes Unix, appelée X Window System, a été créée au MIT et distribuée avec le code source sur des termes permettant la libre modification. Et en 1989-1990, un étudiant en informatique de

l'université d'Helsinki, Linus Torvalds, a commencé le projet qui a complété le circuit et a vraiment impulsé de l'énergie à la vision du logiciel libre.

Ce que Torvalds a fait était de commencer à adapter un outil informatique pédagogique pour un usage réel. Le noyau MINIX d'Andrew Tannenbaum[23] était une base des cours de systèmes d'exploitation, fournissant un exemple de solutions de base à des problèmes de base. Lentement, et d'abord sans en reconnaître l'intention, Linus a commencé à transformer le noyau MINIX en un vrai noyau Unix pour les processeurs Intel x86, qui fonctionnaient sur les ordinateurs personnels de base du monde entier. Au fur et à mesure que Linus commençait à développer son noyau, qu'il appela Linux, il réalisa que la meilleure manière de faire fonctionner le projet était d'ajuster ses décisions conceptuelles, afin que les composants GNU soient compatibles avec son noyau.

Le résultat du travail de Torvalds aboutit, en 1991, à la distribution sur le réseau d'une esquisse de modèle fonctionnel d'un noyau libre pour un système d'exploitation sur PC semblable à Unix, entièrement compatible, et même conçu de manière convergente pour l'énorme ensemble de composants systèmes de haute qualité créés par le projet GNU de Stallman et distribués par la Free Software Foundation. Puisque Torvalds a décidé de distribuer le noyau Linux sous la Licence Générale Publique de la Free Software Foundation (sur laquelle je m'étendrai plus bas), les centaines et finalement milliers de programmeurs à travers le monde qui ont décidé de contribuer par leurs efforts au développement futur du noyau pouvaient être sûrs que leurs efforts auraient pour résultat un logiciel perpétuellement libre, que personne ne pourrait transformer en produit propriétaire. Tout le monde savait que d'autres personnes seraient capables de tester, d'améliorer et de redistribuer leurs modifications. Torvalds acceptait les contributions volontiers, et avec un style génialement efficace a maintenu la direction globale sans refroidir l'enthousiasme. Le développement du noyau Linux a prouvé que l'Internet a rendu possible l'agrégation d'ensembles de programmeurs bien plus grands que n'importe quel éditeur commercial ne pourrait se le permettre, rassemblés de manière pratiquement non hiérarchique dans un projet de développement faisant finalement intervenir plus d'un million de lignes de code source (une échelle de collaboration entre des volontaires non payés et dispersés géographiquement, auparavant inimaginable dans l'histoire humaine[24]).

En 1994, Linux a atteint la version 1.0, représentant un noyau utilisable en production. Le niveau 2.0 a été atteint en 1996. En 1998, avec le noyau à la version 2.2.0 et disponible non seulement pour les machines à base de x86 mais pour toute une variété d'autres architectures de machines, GNU/Linux (la combinaison du noyau Linux et le bien plus grand corps des composants du projet GNU) et Windows NT étaient les deux seuls systèmes d'exploitation du monde à gagner des parts de marché. Une évaluation interne à Microsoft de la situation a filtré en octobre 1998 (et ultérieurement reconnue comme authentique par la compagnie) concluait que « Linux représente un UNIX qui sort du rang, en qui on fait confiance pour des missions d'applications critiques et (ce qui est en partie dû au code source [sic] ouvert) et qui a une crédibilité sur le long terme qui excède celle de nombreux systèmes d'exploitation compétitifs. »[25] Les systèmes GNU/Linux sont maintenant utilisés à travers le monde, fonctionnant partout, faisant office de serveurs web dans des sites de commerce électronique majeurs ou en tant que clusters dédiés à l'infrastructure réseau de centres de paiement de banques. On trouve GNU/Linux dans la navette de l'espace ou même fonctionnant côté jardin chez (eh oui) Microsoft. Les évaluations de l'industrie sur la fiabilité des systèmes Unix ont montré de manière répétée que Linux est de loin le noyau Unix le plus stable et le plus fiable, dont la fiabilité est seulement dépassée par les outils GNU eux-mêmes. GNU/Linux ne dépasse pas seulement les versions propriétaires d'Unix pour

les PC dans les tests de performances, mais est renommé pour sa capacité à fonctionner, sans perturbation et sans plainte, pendant des mois sur des environnements de haut volume et de haute sollicitation sans se planter.

D'autres composants du mouvement du logiciel libre ont également été couronnés de succès. Apache, de loin le serveur web dominant, est un logiciel libre, de même que Perl, le langage de programmation qui est la langue véhiculaire des programmeurs qui construisent des sites web sophistiqués. Netscape Communications distribue désormais son navigateur Netscape Communicator 5.0 en tant que logiciel libre, sous une licence proche de la Licence Générale Publique de la Free Software Foundation. Des constructeurs de PC majeurs, ce qui inclut IBM, ont annoncé des projets de distribution de GNU/Linux ou sont déjà en train en train de le distribuer en tant qu'option pour le client sur leurs PC de haut de gamme destinés à une utilisation en tant que serveurs web ou de partage de fichiers. Samba, un programme qui permet aux ordinateurs GNU/Linux de se comporter en tant que serveurs de fichiers Windows NT, est mondialement utilisé comme une alternative à Windows NT Server, et lui assène une compétition efficace à court terme dans son propre marché réservé. Grâce aux standards de la qualité des logiciels qui ont été reconnus par l'industrie depuis des décennies (et dont la pertinence ininterrompue sera claire pour vous la prochaine fois que votre PC sous Windows se plantera), les nouvelles à la fin de ce siècle ne sont pas ambiguës. La corporation accumulant le plus de profits et la plus puissante au monde arrive de loin seconde, ayant tout exclut sauf le vrai vainqueur de la course. Le propriétarianisme joint à la vigueur du capitalisme a détruit toute compétition commerciale digne de ce nom, mais lorsqu'il est question de faire de bons logiciels, l'anarchisme gagne.

L'anarchisme comme mode de production

C'est une jolie histoire, et si seulement l'IPdroïde et l'écononain n'avaient pas été aveuglés par la théorie, ils l'auraient vu venir. Mais comme certains d'entre nous y travaillaient et l'avaient prédite depuis des années, les conséquences théoriques sont si subversives pour les cheminements de la pensée qu'ils maintiennent nos nains et nos droïdes dans la situation confortable où l'on pourrait difficilement les blâmer d'avoir refusé de voir. Les faits prouvent qu'il y avait une erreur dans la métaphore de « l'incitation » qui étaye les raisonnement classiques sur la propriété intellectuelle.[26] Mais ils ont fait plus. Ils ont fourni un aperçu initial du futur de la créativité humaine dans un monde d'interconnexion globale. Et ce n'est pas un monde fait pour les nains et les droïdes.

Mon argumentation, avant que nous ayons fait une pause pour nous rafraîchir dans le monde réel, peut se résumer de cette manière: le logiciel (qu'il s'agisse de programmes exécutables, de musique, d'art visuel, de liturgie, d'armement etc.) consiste en des flux de bits, qui bien qu'essentiellement indistinguables sont traités par une multiplicité déroutante de catégories légales. Cette multiplicité est instable sur le long terme pour des raisons intégrées au processus légal. La diversité instable des règles est causée par le besoin d'effectuer des distinctions entre les différentes sortes d'intérêts sur les flux de bits. Ce besoin est principalement ressenti par ceux qui essayent d'obtenir du profit des formes socialement acceptables de monopole créées en traitant les idées comme de la propriété. Ceux d'entre nous qui se soucient de l'inégalité sociale et de l'hégémonie culturelle créées par ce régime intellectuellement insatisfaisant et moralement répugnant sont hués. Ceux qui nous huent, les nains et les droïdes, croient que ces lois sur la propriété sont nécessaires, non pas pour aspirer ouvertement à vivre dans le monde de Murdoch (bien qu'un petit peu de cooptation luxueuse soit toujours la

bienvenue) mais parce que la métaphore de la motivation, qu'ils ne prennent pas juste pour une image mais pour un argument, prouve que ces règles (en dehors de leurs conséquences lamentables) sont nécessaires si nous voulons écrire de bons logiciels. La seule façon de continuer à le croire est d'ignorer les faits. Au centre de la révolution numérique, avec des flux de bits qui rendent tout possible, les régimes propriétaristes ne rendent non seulement pas les choses meilleures, mais ils peuvent en plus rendre les choses radicalement pires. Les concepts de la propriété, quoi qu'on puisse leur reprocher, ne permettent pas (et en fait retardent) le progrès.

Mais quelle est cette mystérieuse alternative ? Le logiciel libre existe, mais quels sont ses mécanismes et comment se généralise-t-ils à travers une théorie non propriétariste de la société numérique ?

La théorie légale du logiciel libre

Un mythe, qui comme la plupart des mythes est partiellement fondé sur la réalité, dit que les programmeurs de logiciels sont tous des libertaires. Ceux de droite sont des capitalistes, agrippés à leurs stock-options et dédaignant les lois sur les impôts, les syndicats et les droits civils ; ceux de gauche haïssent le marché et les gouvernements, croient à l'encryption forte quel que soit le terrorisme nucléaire que cela puisse causer[27] et n'aiment pas Bill Gates parce qu'il est riche. Cette croyance est sans doute fondée. Mais la différence la plus importante entre les courants politiques au sein de la cyberélite et les courants politiques extérieurs est que dans la société du réseau, l'anarchisme (ou plus correctement, l'individualisme anti-possessif) est une philosophie politique viable.

Le coeur du succès du mouvement du logiciel libre et la grande réalisation de Richard Stallman n'est pas un logiciel. Le succès du logiciel libre, ce qui inclut l'écrasant succès de GNU/Linux, est le résultat de la possibilité d'exploiter d'extraordinaires quantités d'efforts de grande qualité pour des projets de taille immense et d'une complexité profonde. Et cette possibilité est à son tour le résultat du contexte légal dans lequel la main-d'oeuvre est mobilisée. En tant qu'architecte visionnaire, Richard Stallman a créé plus qu'Emacs, GDB ou GNU. Il a créé la Licence Publique Générale[28].

La GPL[29], aussi connue sous le nom de copyleft, utilise le copyright (pour paraphraser Toby Milsom) pour contrefaire le phénomène de l'anarchisme. Comme le préambule de la licence l'exprime[30]:

> Lorsque nous parlons de free software, nous entendons free dans le sens de liberté, et non pas de gratuité. Notre licence est conçue pour s'assurer que vous avez la liberté de distribuer des copies des programmes, gratuitement ou non, et que vous recevez ou pouvez obtenir le code source, que vous pouvez modifier les programmes ou en utiliser des parties dans d'autres programmes libres, en sachant que vous pouvez le faire.
>
> Afin de protéger vos droits, nous devons faire des restrictions qui interdisent à quiconque de vous refuser ces droits ou de vous demander d'y renoncer. Ces restrictions vous imposent par conséquent certaines responsabilités si vous distribuez des copies des programmes protégés par la Licence Publique Générale ou si vous les modifiez.
>
> Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou non, vous devez transmettre aux utilisateurs tous les droits que vous possédez. Vous devez vous assurer qu'ils reçoivent ou qu'il peuvent se procurer le code source. Vous devez leur montrer cette licence afin qu'ils soient eux aussi au courant de leurs droits.

Plusieurs variantes de cette idée de base du logiciel libre ont été exprimées dans plusieurs licences de différents types, comme je l'ai déjà indiqué. La GPL est différente des autres manières d'exprimer ces valeurs sur un plan crucial. La section 2 de la licence en fournit un extrait pertinent:

> Vous pouvez modifier votre copie ou vos copies du programme ou toute partie de celui-ci, ou travail basé sur ce programme, et copier et distribuer ces modifications ou ce travail ..., à condition que vous vous conformiez également aux conditions suivantes :
>
> ...
>
> b) C'est sous les termes de la Licence Publique Générale que vous devez distribuer l'ensemble de toute réalisation contenant tout ou partie du programme, avec ou sans modifications.

La section 2(b) de la GPL est parfois appelée « restrictive », mais son intention est de libérer. Elle crée un pot commun auquel chacun peut ajouter quelque chose mais duquel personne ne peut rien retirer. À cause du paragraphe 2(b), chaque contributeur d'un projet sous GPL est assuré qu'il poura, ainsi que tous les autres utilisateurs, exécuter, modifier et redistribuer le programme indéfiniment, que le code source sera toujours disponible et (contrairement aux logiciels commerciaux) que sa longévité ne pourra pas être limitée par les contingences du marché ou les décisions des développeurs futurs. L'« hérédité » de la GPL a parfois été critiquée comme un exemple du parti-pris anti-commercial du mouvement du logiciel libre. Rien ne peut être plus éloigné de la vérité. L'effet du paragraphe 2(b) est de rendre les distributeurs de logiciels libres meilleurs que les commerces du logiciel propriétaire. Pour confirmer ce point, on ne peut pas faire mieux que de le demander aux concurrents propriétaires. Comme l'auteur de la « note de service Halloween », Vinod Vallopillil l'exprime:

> La GPL et son aversion pour la séparation divergente du code rassurent les clients sur le fait qu'ils ne s'engagent pas dans une impasse technologique en souscrivant à une distribution commerciale particulière de Linux.
>
> L'« impasse technologique » est le coeur de l'argument du FUD logiciel[31].

Traduit en Microlangue, cela signifie que la stratégie par laquelle l'éditeur propriétaire dominant éloigne les clients des concurrents (en semant le doute, la peur et l'incertitude à propos de la viabilité sur le long terme de leurs logiciels) n'est pas efficace sur les logiciels sous GPL. Les utilisateurs de code sous GPL, ce qui inclut ceux qui achètent des logiciels et des systèmes à un distributeur commercial, savent que les améliorations futures et les corrections seront accessibles par les pots communs et n'ont pas besoin d'avoir peur de la disparition de leur fournisseur ou que quelqu'un utilisera une amélioration attractive ou une réparation désespérément nécessaire comme un moyen de « s'approprier le programme ».

Cette utilisation des règles de la propriété intellectuelle pour créer un pot-commun dans le cyber-espace est la structure institutionnelle permettant le triomphe anarchiste. Garantir l'accès libre et permettre la modification à chaque étape du processus signifie que l'évolution du logiciel s'effectue selon le mode rapide Lamarckien: chaque caractéristique favorable acquise du travail des autres peut être directement héritée. D'où la vitesse avec laquelle le noyau Linux, par exemple, dépassa tous ces prédécesseurs

propriétaires. Comme la défection est impossible, les cavaliers seuls sont les bienvenus, ce qui résout un des casse-tête centraux des actions collectives dans un système social propriétarien.

La production non-propriétarienne est aussi directement responsable de la fameuse stabilité et de la robustesse du logiciel libre, qui émerge de ce qu'Éric Raymond appelle « la loi de Linux »: avec suffisamment d'yeux, tous les bugs disparaissent. D'un point de vue pratique, l'accès au code source signifie que si j'ai un problème, je peux le résoudre. Comme je peux le résoudre, je n'ai pratiquement jamais à le faire car quelqu'un d'autre l'a pratiquement toujours vu et résolu avant.

Pour la communauté du logiciel libre, l'engagement à la production anarchiste peut être un impératif moral ; comme Richard Stallman l'a écrit, il s'agit de liberté, pas de prix. Ou il pourrait s'agir d'utilité, de chercher à produire de meilleurs logiciels que les modes de travail propriétariens n'autoriseront jamais. Du point de vue du droïde, le copyleft représente la perversion de la théorie, mais il résout les problèmes de l'application du copyright aux caractéristiques fonctionnelles et expressives inextricablement mêlées des programmes d'ordinateur mieux que tous les autres propositions à travers les décennies passées. Qu'il produise des logiciels meilleurs que son opposé n'implique pas que les règles traditionnelles du copyright devraient être désormais interdites à ceux qui veulent posséder et commercialiser des produits logiciels inférieurs ou (plus charitablement) ceux dont les produits sont trop limités en attrait pour la production commune. Mais notre histoire devrait servir d'avertissement pour les droïdes: le monde du futur aura peu de relation avec le monde du passé. Les lois se penchent aujourd'hui dans les deux directions. Les entreprises propriétaires d'« icônes culturelles » et d'autres capitaux qui cherchent des termes encore plus longs pour les auteurs de l'entreprise, en convertissant le « temps limité » de l'article I, §8 en une propriété foncière disponible ont naturellement siffloté de la musique à l'oreille de l'androïde[32]. Après tout, qui achète aux droïdes leurs places de concert ? Mais comme la conception propriétarienne cherche à s'inclure encore plus fort dans une conception du copyright libérée des problèmes mineurs tels que les clauses de terminaison et d'usage loyal, au centre même de notre système du « logiciel culturel », la contre-attaque anarchiste a commencé. Le pire est encore d'arriver aux droïdes, comme nous le verrons. Mais d'abord, nous devons nous acquitter de nos devoirs finaux envers les nains.

Car c'est là: l'aimant de Faraday et la créativité humaine

Après tout, ils méritent une réponse. Pourquoi des personnes font des logiciels libres s'ils n'en tirent pas de profit ? On a habituellement donné deux réponses. Une est à moitié vraie et la deuxième est fausse, mais les deux sont insuffisamment simples.

La réponse fausse est inscrite dans de nombreuses références à la « culture hacker du partage ». Cette utilisation du jargon ethnographique commençait à être commune dans le domaine, il y a quelques années, et est devenue rapidement, si ce n'est trompeuse, du moins omniprésente. Cela nous rappelle seulement que les économéretriciens[33] ont tellement corrompu notre processus de réflexion que toute forme de comportement économique non marchand semble égal à tout autre sorte. Mais l'échange de dons, comme l'échange réciproque du marché est une institution propriétarienne. La réciprocité est centrale à ces règlements symboliques de dépendance mutuelle et si les ignames ou le poisson ne font pas le même poids, il y a un problème. Le logiciel libre, au risque de le répéter, est un pot commun : aucune réciprocité rituelle n'est réglementée ici. Quelques personnes donnent du code que d'autres vendent, utilisent, modifient ou y en empruntent le principal afin d'en tirer des parties pour quelque

chose d'autre. Malgré le très grand nombre de contributeurs à GNU/Linux (des dizaines de milliers tout au plus), c'est un ordre de grandeur inférieur au nombre d'utilisateurs qui ne contribuent pas de toute façon[34].

Une partie de la vraie réponse est suggérée par l'hypothèse que le logiciel libre est fait par ceux qui cherchent en compensation une réputation pour leur activité. La théorie est que les développeurs noyau célèbres sont connus à travers la planète comme des dieux de la programmation. Ils en tirent soit une estime personnelle plus haute, soit des avantages matériels indirects[35]. Mais les dieux de la programmation, tout autant qu'ils aient contribué au logiciel libre, n'ont pas réalisé la totalité du travail. Les réputations, comme Linus Torvalds l'a lui-même souvent fait remarquer, sont faites en reconnaissant volontairement que tout a été fait par quelqu'un d'autre. Et comme beaucoup d'observateurs l'ont remarqué, le mouvement du logiciel libre a aussi produit de la documentation échappant aux superlatifs. L'écriture de documentation n'est pas ce que les hackers font pour s'amuser, et la plupart de la documentation a été écrite par des personnes qui n'ont pas écrit le code. Nous ne devons pas non plus limiter les avantages matériels indirects de la paternité à l'augmentation du capital réputation. La plupart des auteurs de logiciels libres que je connais ont des emplois dans les industries technologiques, et les qualifications qu'ils acquièrent dans travail plus créatif qu'ils font, en dehors du marché, améliorent sans doute souvent considérablement leur valeur à l'intérieur du marché. Et comme les produits du logiciel libre ont gagné une masse critique, et sont devenus la base de tous nouveaux ensembles de business models construits autour de la distribution commerciale de ce que les gens peuvent aussi obtenir pour rien, un nombre croissant de personnes sont employées spécifiquement pour écrire du logiciel libre. Mais pour pouvoir être employés dans le secteur, ils doivent déjà y être établis par eux-mêmes. De manière ordinaire, alors cette motivation est présente, mais ce n'est pas l'explication entière.

En fait, le reste de la réponse est trop simple pour avoir reçu l'écho qui lui revient. La meilleure manière de comprendre ce qui suit est de suivre la carrière brève et inédite d'un auteur de logiciel libre initialement désavantagé. Vinod Vallopillil de Microsoft, en train d'écrire l'analyse compétitive de Linux qui filtrera comme étant la deuxième des fameuses « notes de service Halloween », acheta et installa un système Linux sur un des ordinateurs qu'il utilisait au travail. Il eut des problèmes car la distribution (commerciale) de Linux qu'il a installée ne contenait pas de daemon pour gérer le protocole DHCP qui assigne une adresse IP dynamiquement. Le résultat était suffisamment important pour que nous risquions un autre examen au style d'écriture Microsoft:

> Un petit nombre de sites web et de FAQs plus tard, j'ai trouvé un site FTP avec un client DHCP Linux. Le client DHCP était développé par un ingénieur employé par Fore Systems (comme le montre son adresse email ; je suppose, cependant, qu'il l'a développé dans son propre temps libre). Un autre jeu de documentations et de manuels a été écrit pour le client DHCP par un hacker de *Hongrie* qui fournit des instructions relativement simples pour installer/charger le client.
>
> J'ai téléchargé et décompressé le client et tapé les deux commandes simples:
>
> Make - qui compile les binaires du client.
>
> Make install - qui installe les binaires en tant que daemon Linux.

> En tapant « DHCPD » (pour DHCP Client Daemon) sur la ligne de commande, j'ai déclenché le processus de découverte de DHCP et voilà, j'ai obtenu un réseau IP qui fonctionne.
>
> Comme j'avais juste téléchargé le code du client DHCP, j'ai un peu joué avec sur un coup de tête. Bien que le client n'était pas aussi extensible que le client que nous distribuons avec NT5 (par exemple, il n'effectuera pas de requêtes sur des options arbitraires pour stocker les résultats), il était évident de comprendre comment je pourrais écrire du code additionnel pour implémenter cette fonctionnalité. Le client complet consistait en 2600 lignes de code.
>
> Un exemple de fonctionnalité ésotérique et étendue qui était clairement patchée par un ensemble de routines pouvait ajouter aux requêtes DHCP des chaînes de caractères spécifiques à l'hôte requises par les sites Modem Câble et ADSL.
>
> Quelques autres étapes étaient requises pour configurer le client DHCP pour auto-démarrer et auto-configurer mon interface Ethernet au démarrage du système, mais elles étaient documentées dans le code du client et dans la documentation DHCP écrite par le développeur Hongrois.
>
> Je suis un mauvais programmeur UNIX, mais il m'a semblé immédiatement évident d'améliorer le code du client DHCP (le sentiment était grisant et « accoutumant »).
>
> De plus, directement en raison de la GPL et de la disponibilité de l'environnement de travail complet en face de moi, j'étais dans une position où je pouvais écrire mes modifications et les envoyer par email en quelques heures (par rapport à la manière dont de telles choses auraient été faites sous NT). M'engager dans ce processus m'aurait préparé pour un projet Linux plus grand, plus ambitieux dans le futur[36].

« Le sentiment était grisant et accoutumant ». Arrêtez les presses: Microsoft vérifie expérimentalement la métaphore de Moglen, corollaire à la loi de Faraday. Enroulez l'Internet autour de chaque cerveau de la planète et faites tourner la planète. Le logiciel parcourt les câbles. La création est une propriété émergente de l'esprit humain. « Directement en raison de la GPL », comme Vallopillil le remarque justement, le logiciel libre lui a fourni une augmentation grisante de sa propre créativité, d'une manière qui n'est pas possible dans son travail coutumier dans la Plus Grande Entreprise de Programmation sur Terre. Si seulement il avait envoyé par mail cette première amélioration « accoutumante », qui sait où il serait maintenant ?

Ainsi, mes amis nains, c'est juste une question humaine. Semblable à la raison pour laquelle Figaro chante, pour laquelle Mozart a écrit pour lui la musique qu'il chante, et pour laquelle nous construisons tous de nouveaux mots: parce que nous pouvons. L'Homo ludens rencontre l'Homo faber. La condition sociale de l'interconnexion globale que nous appelons l'Internet rend possible la créativité pour chacun d'entre nous dans des voies nouvelles, et que nous n'apercevions même pas en rêve. À moins que nous n'autorisions la « propriété » à interférer. Répétez après moi, vous les nains et les hommes: résistez à la résistance !

Nos seigneurs meurent dans le noir ?

Pour l'IPdroïde, tout droit sorti de l'avion au retour d'une semaine à Bellagio[37] payée par Dreamworks SKG, c'est assez pour causer une indigestion.

Libérer les possibilités de la créativité humaine en connectant tout le monde à tous les autres ? Mettre le système de la propriété à l'écart afin que nous puissions tous ajouter nos voix au choeur même, si cela signifie coller notre chant par dessus le tabernacle Mormon et envoyer le résultat à un ami ? Plus personne assis, la mâchoire pendante, devant une mixture télévisée de violence et de copulation imminente, conçue attentivement pour attirer l'intérêt de l'oeil du jeune mâle sur une publicité pour une bière ? Que va devenir notre civilisation ? Ou du moins nos enseignants sur le copyright ?

Mais peut-être est-ce prématuré. J'ai seulement parlé de logiciel. De vrai logiciel, l'ancien, celui qui fonctionne sur des ordinateurs. Pas celui qui tourne sur les lecteurs de DVD ou celui fait par Grateful Dead (ah oui, Grateful Dead, il y avait quelque chose de bizarre avec eux, n'est ce pas ? Il n'interdisaient pas les enregistrements à leurs concerts. Ils se moquaient que leurs fans agacent l'industrie du disque. Il semble qu'ils aient bien fait, en fait, vous devez l'admettre. Le sénateur Patrick Leahy n'est il pas un ancien Deadhead ? Je me demande s'il votera pour étendre la propriété de l'entreprise à 125 ans afin que Disney ne perde pas Mickey Mouse en 2004). Et ces lecteurs de DVD, ce sont des ordinateurs, n'est-ce pas ?

Dans la société numérique, tout est connecté. Nous ne devons pas dépendre sur le long terme de la distinction d'un flux de bits d'un autre afin de savoir quelles règles s'appliquent. Ce qui est arrivé au logiciel est déjà arrivé à la musique. Nos seigneurs de l'industrie de la copie sont désormais en train de remuer férocement afin de conserver le contrôle sur la distribution, quand à la fois les musiciens et les auditeurs réalisent que les intermédiaires ne sont plus nécessaires. Le « grand village Potemkine » de 1999, la bien nommée « initiative de musique numérique sécurisée » (Secure Digital Music Initiative[38]) aura disparu bien avant que le premier président de l'Internet soit inauguré, pour de simples raisons techniques aussi évidentes à ceux qui savent qu'à ceux qui ont dicté le triomphe du logiciel libre[39]. La révolution anarchiste dans la musique est différente de celle du logiciel tout court[40], mais ici aussi (comme tout adolescent avec une collection de MP3 de musique auto-produite d'artistes indépendants vous le dira) la théorie a été tuée par les faits. Que vous soyez Mick Jagger, un grand artiste national du tiers monde cherchant un public global ou un artiste réinventant la musique dans votre coin, l'industrie du disque n'aura bientôt plus rien à vous offrir que vous n'auriez mieux et gratuitement. Et la musique ne sonne pas moins bien quand elle est distribuée gratuitement, payez ce que vous voulez directement à l'artiste et ne payez plus rien si vous ne le voulez pas. Donnez votre argent à vos amis, ils l'apprécieront peut-être.

Ce qui est arrivé à la musique est aussi en train d'arriver aux informations. Les services par câblés, comme chaque étudiant en droit américain l'apprend avant même de prendre le quasi-obligatoire cours sur le copyright pour droïdes, ont un intérêt protectible de propriété dans leur expression des informations, même si ce n'est pas le cas dans les faits, sur les rapports d'informations[41]. Alors pourquoi est-ce qu'ils font maintenant cadeau de toute leur production ? Parce que dans le monde du réseau, la plupart des informations sont des informations de commodité. Et l'avantage original des collecteurs d'informations (c'est-à-dire qu'ils étaient très fortement inter-connectés, ce qui n'était pas le cas des autres quand les communications étaient coûteuses) n'est plus. Maintenant, ce qui compte c'est de collecter de l'attention des globes oculaires pour la livrer aux publicitaires. Ce ne sont pas les

services câblés qui ont avantage à couvrir le Kosovo, c'est sûr. Encore moins les parangons de la propriété « intellectuelle », nos seigneurs de la télévision. Ils, avec leur gentil personnel surpayé et leur massive infrastructure technique, sont à peu près les seules organisations au monde qui ne peuvent se permettre d'être partout à la fois. Et alors ils doivent se limiter eux-mêmes à quatre-vingt-dix secondes par sujet, ou les chasseurs d'yeux iront voir ailleurs. Alors qui fait la meilleure information, les propriétariens ou les anarchistes ? On verra bientôt.

Oscar Wilde dit quelque part que le problème avec le socialisme est qu'il prend trop de lendemains à venir. Les problèmes de l'anarchisme comme système social sont aussi les coûts de transaction. Mais la révolution numérique modifie deux aspects de l'économie politique qui ont étés par ailleurs invariants dans le reste de l'histoire humaine. Tous les logiciels ont un coût marginal nul dans le cas du monde du réseau, alors que les coûts de coordination sociale ont étés réduits suffisament pour permettre la rapide formation et dissolution de groupes sociaux de grande échelle et très divers sans aucune limitation géographique[42]. De tels changements fondamentaux dans les circonstances matérielles de la vie produisent nécessairement des changements également fondamentaux dans la culture. Vous pensez que non ? Dites-le aux Iroquois. Et, bien sûr, de tels changements dans la culture sont des menaces pour les relations de pouvoir existantes. Vous pensez que non ? Demandez au Parti communiste chinois. Ou alors attendez vingt-cinq ans et voyez si vous pouvez le trouver pour leur poser la question.

Dans ce contexte, l'obsolescence de l'IPdroïde n'est pas inoubliable ni tragique. En fait, il pourrait se trouver lui-même cliquetant dans le désert, expliquant toujours lucidement à une audience imaginaire les règles profitablement compliquées d'un monde qui n'existe plus. Mais, au moins, il aura une compagnie familière, reconnaissable, de toutes ces fêtes scintillantes à Davos, Hollywood et Bruxelles. Nos seigneurs des Médias sont maintenant en prise avec le destin, bien que la plupart d'entre eux doivent sentir que la Force est avec eux. Les règles sur les flux de bits sont maintenant d'une utilité incertaine pour maintenir le pouvoir en cooptant la créativité humaine. Vus clairement à la lumière du jour, ces empereurs ont encore moins de vêtements que les modèles qu'ils utilisent pour attirer nos regards. À moins d'être soutenus par des technologies d'immobilisation de l'utilisateur, par une culture de surveillance totalitaire permettant de noter et de facturer chaque lecteur de chaque « propriété » et un écran de fumée d'haleine de droïde assurant chaque jeune personne que la créativité humaine aurait disparu sans l'aristocratie bienveillante de BillG le Créateur, de Lord Murdoch de Partout, le Spielmeister et de Lord High Mouse, leur règne est pratiquement fini. Mais ce qui est en jeu est le contrôle de la ressource la plus rare de toutes: notre attention. L'occuper est ce qui produit tout l'argent du monde de l'économie digitale, et les seigneurs actuels de la terre se battront pour ça. Seuls les anarchistes sont ligués contre eux: ceux qui ne sont personne, les hippies, les amateurs, les amoureux et les artistes. L'inégale compétition qui en résulte est le grand enjeu politique et légal de notre temps. L'aristocratie semble difficile à battre, mais c'est aussi comme cela qu'elle semblait en 1788 et en 1913. Il est, comme Chou En-Lai l'a dit à propos du sens de la Révolution française, trop tôt pour le dire.

Article préparé pour la publication à la conférence internationale Buchmann sur le droit, la technologie et l'information, à l'université de Tel Aviv en mai 1999. Mes remerciements aux organisateurs pour leur gracieuse invitation. Je dois plus que jamais à Pamela Karlan pour sa perspicacité et ses encouragements. Merci à Jérôme Saltzer, Richard Stallman et bien d'autres qui ont largement contribué aux corrections et aux améliorations de cet article. J'aimerais spécialement remercier les programmeurs à travers le monde qui ont rendu le logiciel libre possible. Traduction française par Jérôme Dominguez (taz@gnu.org).

[1] La distinction était seulement approximative dans son contexte original. À la fin des années 1960, certaines portions des opérations de base du matériel étaient contrôlées par des programmes codés numériquement dans l'électronique de l'équipement informatique, figés après que les éléments quittent la fabrique. De tels composants utilisant un symbolisme, mais non modifiables, étaient connus dans le commerce comme des « microcodes », mais il est devenu courant de s'y référer comme le « microprogramme » (firmware). Le logiciel, comme le terme « microprogramme » le démontrait, faisait primairement référence à la possibilité pour les utilisateurs de modifier les symboles déterminant le comportement de la machine. Comme la révolution numérique fût le résultat de l'utilisation massive d'ordinateurs par des incompétents techniques, la plupart des logiciels traditionnels (des applications, des systèmes d'exploitation, des instructions de contrôle numérique etc.) est, pour la plupart de ses utilisateurs, du microprogramme. Ces logiciels peuvent être symboliques plutôt qu'électroniques dans leur construction, mais les utilisateurs ne peuvent pas les modifier même s'ils le voulaient, ce qu'ils souhaitent souvent, impuissants et amers. Cette « microprogrammisation du logiciel » est une condition primordiale de l'approche propriétarienne de l'organisation légale de la société numérique, qui est le sujet de cet article.

[2] Au sein de la génération actuelle, la conception même d'un « développement » social est en train de glisser de la possession d'une industrie lourde basée sur le moteur à explosion à une industrie « post-industrielle » basée sur les communications numériques et sur les formes de l'activité économique « basées sur la connaissance ».

[3] En fait, un moment de réflexion révélera que nos gènes sont des microprogrammes. L'évolution a fait la transition entre l'analogique et le numérique avant les premiers fossiles recensés. Mais nous n'avons jamais possédé le pouvoir d'exercer des modifications directes contrôlées. Jusqu'à hier. Dans le prochain siècle, les gènes aussi deviendront des logiciels et, bien que je ne parle pas du problème plus loin dans cet article, les conséquences politiques de la libération du logiciel, dans ce contexte, sont encore plus dérangeantes qu'elles ne le sont en comparaison avec les aspects culturels.

[4] Voir, par exemple, J. M. Balkin, *Cultural Software: a Theory of Ideology*(New Haven: Yale University Press, 1998).

[5] Voir Henry Sumner Maine, *Ancient Law: Its Connection with the Early History of Society, and its Relation to Modern Idea*, 1st edn. (London: J. Murray, 1861).

[6] En général, je déteste l'intrusion de l'autobiographie dans une publication universitaire. Mais comme il est de mon triste devoir et mon grand plaisir de mettre en doute la qualification ou la bonne foi, tout simplement, de pratiquement tout le monde, je dois permettre ma propre évaluation. J'ai été confronté, pour la première fois, à l'art de la programmation informatique en 1971. J'ai commencé à gagner ma vie en tant que programmeur commercial en 1973 (à l'âge de treize ans) et continué ainsi jusqu'en 1985, dans différents services informatiques, d'ingénierie, et des entreprises de technologie

multinationales. En 1975, j'ai aidé à écrire un des premiers systèmes de courrier électronique en réseau des États-Unis ; à partir de 1979, j'ai été engagé dans les services de recherche et de développement de langages informatiques avancés chez IBM. Ces activités ont rendu économiquement possible, pour moi, l'étude des arts, de l'érudition historique et de l'ingéniosité légale. Mon salaire était suffisant pour payer mes cours, non pas (pour anticiper un argument qui sera énoncé plus loin par les « écononains » parce que mes programmes étaient la propriété intellectuelle de mon employeur, mais plutôt parce qu'ils faisaient mieux fonctionner le matériel que mon employeur vendait. La plupart des logiciels que j'ai écrits étaient effectivement des logiciels libres, comme nous le verront. Bien que j'ai ultérieurement apporté quelques contributions techniques insignifiantes au vrai mouvement du logiciel libre, que cet article décrit, mes activités principales en sa faveur ont été légales: j'ai servi, ces cinq dernières années (bénévolement, naturellement), d'avocat général à la Free Software Foundation.

[7] Le lecteur, bien sûr, a des entrées et des sorties secondaires: les boutons et les télécommandes infrarouges sont des entrées, et l'affichage du temps et du morceau sont des sorties.

[8] NdT: l'ingénierie inverse est légale en Europe pour des besoins d'interopérabilité et de compatibilité, en cas de mauvaise volonté ou d'impuissance de l'éditeur ou du constructeur.

[9] NdT: Ruppert Murdoch, magnat de la presse et de la finance.

[10] NdT: Gucci-gulch est le surnom donné à la chambre des finances du gouvernement américain et aux couloirs alentours, où se rencontrent les lobbyistes porteurs de chaussures Gucci et prompts aux affaires.

[11] Ce n'est pas une idée unique à notre entreprise actuelle. Une idée proche forme un des plus importants principes de l'histoire de la loi anglo-américaine, parfaitement décrite par Toby Milsom en ces termes:

> The life of the common law has been in the abuse of its elementary ideas. If the rules of property give what now seems an unjust answer, try obligation; and equity has proved that from the materials of obligation you can counterfeit the phenomena of property. If the rules of contract give what now seems an unjust answer, try tort. ... If the rules of one tort, say deceit, give what now seems an unjust answer, try another, try negligence. And so the legal world goes round.
>
> (La vie du droit coutumier est constituée de l'abus de ses idées élémentaires. Si les règles de la propriété donnent ce qui semble maintenant une réponse injuste, essayez l'obligation ; l'équité a prouvé qu'à partir de la substance de l'obligation, vous pouvez contrefaire le phénomène de la propriété. Si les lois du contrat donnent ce qui semble à présent une réponse injuste, essayez le délit...Si les lois d'un délit, disons la tromperie, donnent ce qui semble une réponse injuste, essayez autre chose, essayez la négligence. Et ainsi le monde du droit tourne rond.)

S. F. C. Milsom, *Historical Foundations of the Common Law*, 2nd edn. (London: Butterworths, 1981), 6.

[12] Voir Isaiah Berlin, *The Hedgehog and the Fox; an Essay on Tolstoy's View of History* (New York: Simon and Schuster, 1953).

[13] Voir « The Virtual Scholar and Network Liberation ».

[14] Un peu de vocabulaire de base est essentiel. Les ordinateurs numériques exécutent réellement des instructions numériques: des chaînes de bits qui contiennent de l'information dans le langage « natif » créé par les concepteurs de la machine. On y fait habituellement référence en tant que « langage machine ». Les langages machine du matériel sont conçus pour une haute vitesse d'exécution au niveau matériel, et ne sont pas appropriés à une utilisation directe par les êtres humains. C'est pourquoi, parmi les composants centraux d'un système informatique, il y a les « langages de programmation » qui traduisent des expressions adaptées aux humains en langage machine. La forme la plus commode et la plus appropriée (mais ce n'est pas la seule) de langage informatique est le « compilateur ». Le compilateur effectue une translation statique, afin qu'un fichier contenant des instructions lisibles par les humains, appelé « code source », résulte en la génération d'un fichier (ou plus) de langage machine exécutable, appelé « code objet ».

[15] NdT: COBOL a été bien entendu écrit dans une langue proche de l'anglais. La phrase précédente pourrait se traduire par: MULTIPLIER PRIX PAR QUANTITÉ DONNANT EXPANSION. ».

[16] C'était, je devrais dire, le chemin que la plupart de mes recherches et de mon développement ont suivi, grandement en rapport avec un langage appelé APL (« A Programming Language ») et ses successeurs. Ce n'était pas, cependant, l'approche finalement dominante, pour des raisons qui seront suggérées plus bas.

[17] Cette description élude quelques détails. Dans le milieu des années 1970, IBM a fait face à une compétition sérieuse dans le domaine des supercalculateurs, pendant que des actions anti-monopolistiques de grande envergure menées contre la société par le gouvernement américain ont incité la décision de « séparer » (ou vendre séparément) les logiciels. Ou dans un sens moins important, les logiciels ont cessé d'être libres. Mais (sans entrer dans la controverse dorénavant éteinte mais autrefois brûlante sur la politique tarifaire d'IBM), la révolution de cette séparation a eu moins d'effets sur les pratiques sociales de la fabrication de logiciels qu'on pourrait le supposer. En tant que co-responsable des améliorations techniques d'un langage informatique produit chez IBM entre 1979 et 1984, par exemple, j'étais capable de considérer le produit comme « presque libre », c'est-à-dire de discuter avec les utilisateurs des modifications qu'ils avaient proposé ou effectué dans les programmes, et de m'engager avec eux dans le développement coopératif du produit pour tous les utilisateurs.

[18] Cette description est très succinte et semblera à la fois très simplifiée et excessivement « rose » pour ceux qui ont travaillé dans l'industrie pendant cette période de son développement. La protection par le copyright des logiciels a été un sujet controversé dans les années 1970, menant à la célèbre commission CONTU et à ses recommandations de 1979, aboutissant à un consensus mou pour le copyright. Et IBM semblait bien moins coopérative avec ses utilisateurs à l'époque que ce portrait ne le fait paraître. Mais l'élément le plus important est le contraste avec le monde créé par le PC, l'Internet et la domination de Microsoft, et l'impulsion résultante pour le mouvement du logiciel libre. Et je me concentre ici sur les caractéristiques qui expriment ce contraste.

[19] Je parle de l'importance du logiciel pour ordinateurs personnels dans ce contexte, de l'évolution du « marché des globes oculaires » et de la « vie sponsorisée » dans d'autres chapitres de mon livre en cours d'écriture, le *Barbecue invisible*, dont cet essai est une partie

[20] Cette même forme de dualité, dans laquelle la mauvaise programmation menant à une instabilité largement distribuée de la nouvelle technologie, est à la fois effrayante et rassurante pour les

incompétents techniques, peut aussi être vue dans le phénomène initié par les américains de l'hystérie du bug de l'an 2000.

[21] Les implications critiques de cette simple observation à propos de nos métaphores sont dissertées dans « Comment ne pas penser à l'Internet », dans *Le barbecue invisible*, en cours.

[22] Les lecteurs ayant un bagage technique observeront encore que cela compresse les développements de 1969 à 1973.

[23] Les systèmes d'exploitation, même Windows (qui cache le fait aux utilisateurs le plus minutieusement possible), sont vraiment des ensembles de composants plutôt que des touts indivisibles. La plupart de ce qu'un système d'exploitation fait (gérer les systèmes de fichiers, contrôler l'exécution des processus, etc.) peut être abstrait des détails matériels réels de l'ordinateur sur lequel le système fonctionne. Seul un petit ensemble au cur du système doit réellement gérer les particularités excentriques d'un matériel particulier. Une fois que le système d'exploitation est écrit dans un langage générique comme le C, seul ce cur, connu dans le milieu comme le noyau, sera hautement spécifique à une architecture particulière.

[24] Une analyse prudente et imaginative de la manière dont Torvalds a fait fonctionner ce processus et ce qu'il implique dans les processus sociaux de création de logiciels a été fournie par Eric S. Raymond dans son article très original « La cathédrale et le bazar », qui a lui-même joué un rôle significatif dans l'expansion de l'idée du logiciel libre.

[25] C'est une citation de ce qui est connu dans le milieu comme la « note de service Halloween » qui peut être trouvée, annotée par Eric Raymond, à qui elle a filtré, à l'URL http://www.opensource.org/halloween1.html.

[26] Ce n'est pas plus anciennement qu'en 1994 qu'un universitaire en droit et économie, talentueux et compétent techniquement (bien qu'utilisateur de Windows) dans une école de droit majeure aux États-Unis m'informait confidentiellement que le logiciel libre ne pouvait possiblement pas exister, car personne n'aurait de motivation pour créer des programmes vraiment sophistiqués nécessitant un investissement d'efforts substantiel seulement pour les donner.

[27] Cette question mérite aussi un examen minutieux, incrustée comme elle est dans un plaidoyer particulier du côté de l'appareil d'État. Voir mon bref essai ``*So Much for Savages*: Navajo 1, Government 0 in Final Moments of Play''.

[28] NdT: General Public License, abrégée par GPL.

[29] Voir GNU General Public License, Version 2, June 1991.

[30] NdT: les paragraphes tirés de la GPL sont issus de la traduction non officielle située à l'URL http://www.linux-france.org/article/these/licence/gpl/gpl.html.

[31] V. Vallopillil, Open Source Software: A (New?) Development Methodology.

[32] L'imminente expiration de la propriété sur Mickey Mouse par Disney requiert par exemple, du point de vue de ce riche « contributeur à la campagne », une modification de la loi générale sur le copyright aux États-Unis. Voir « Not Making it Any More? Vaporizing the Public Domain » dans *The Invisible Barbecue*, bientôt disponible.

[33] NdT: jeu de mot intraduisible sur le mot « économétriciens », adeptes de l'économétrie statistique.

[34] Une récente estimation industrielle place le nombre mondial de systèmes Linux à 7,5 millions. Voir « Josh McHugh, Linux: The Making of a Global Hack », Forbes, 10 août 1998. Comme les logiciels libres sont librement accessibles à travers le Net, il n'y a aucune manière simple de chiffrer l'utilisation réelle.

[35] Éric Raymond est un partisan de la théorie du « boost de l'ego » à laquelle il ajoute une autre comparaison pseudo-ethnographique de la composition du logiciel libre et du potlatch chez les indiens Kwakiutl. Voir Eric S. Raymond, « Homesteading the Noosphere ».. Mais le potlatch, certainement une forme de compétition pour le statut, est différent du logiciel libre pour au moins deux raisons fondamentales: il est essentiellement hiérarchique, ce que n'est pas le logiciel libre et, comme nous le savons depuis que Thorstein Veblen a le premier porté attention à cette particularité, c'est une forme de remarquable gâchis. Voir Thorstein Veblen, *The Theory of the Leisure Class* (New York: Viking, 1967), (1st ed. 1899), 75. Ce sont précisément les bases qui distinguent les cultures anti-hiérarchiques et utilitaires du logiciel libre de ses équivalents propriétariens.

[36] Vinod Vallopillil, « Linux OS Competitive Analysis » (Halloween II). Notez la surprise de Vallopillil qu'un programme écrit en Californie puisse être ultérieurement documenté par un programmeur de Hongrie.

[37] NdT: Bellagio: centre des recherches et conférences en Italie.

[38] NdT: *secure* signifie en anglais à la fois sécurisée et bloquée.

[39] Voir « They're Playing Our Song: The Day the Music Industry Died, » dans *The Invisible Barbecue*, à venir.

[40] NdT: en français dans le texte.

[41] International News Service v. Associated Press, 248 U.S. 215 (1918). En regard des expressions vraiment brèves, purement fonctionnelles des dépêches réellement en jeu dans la bousculade des services câblés, ça a toujours été une distinction que seul un droïde pouvait aimer.

[42] Voir « No Prodigal Son: The Political Theory of Universal Interconnection, » dans *The Invisible Barbecue*, en cours.

Le Manifeste du Point-Communiste ("The dotCommunist Manifesto")

Un spectre hante le capitalisme international - le spectre de la libre information. Toutes les puissances de la "globalisation" ont conclu une alliance bien peu sacrée afin de l'exorciser : Microsoft et Disney, l'Organisation Mondiale du Commerce, le Congrès des Etats-Unis et la Commission Européenne.

Quel défenseur de la liberté dans la nouvelle société numérique n'a jamais été vilipendé comme pirate, anarchiste ou communiste ? N'a-t-on pas vu que nombre de ceux qui lancent ces épithètes sont de simples voleurs en puissance, dont le discours de "propriété intellectuelle" n'est qu'une tentative de préserver des privilèges injustifiables dans une société dont l'évolution est inexorable ? En tout état de cause, tous les Pouvoirs de la Globalisation admettent que le mouvement libertaire est de fait un pouvoir, et il est grand temps que nous affichions nos vues à la face du monde, pour enluminer d'un Manifeste de notre cru le conte enfantin du Spectre de la Libre Information.

Propriétaires et Créateurs

A travers le monde, le mouvement pour la libre information annonce l'avènement d'un nouvel ordre social, né de la transformation de la société industrielle bourgeoise par la technologie numérique qu'elle a elle-même engendrée.

L'histoire de toutes les sociétés ayant existé jusqu'ici révèle une histoire de lutte des classes.

Homme libre et esclave, patricien et plébéien, seigneur et serf, maître artisan et compagnon, bourgeois et prolétaire, colon et colonisé, en un mot, oppresseur et opprimé, sont demeurés en constante opposition et ont mené un combat ininterrompu, tantôt dans l'ombre, tantôt en pleine lumière, qui a souvent entraîné soit une reconstitution révolutionnaire de la société dans son ensemble, soit la ruine commune des protagonistes.

La société industrielle qui a germé sur l'expansion mondiale de la puissance Européenne, conduisant à la modernité, n'a pas éliminé les antagonismes de classes. Elle n'a fait qu'établir de nouvelles classes, de nouvelles conditions ou formes d'oppression et de lutte en lieu et place des anciennes. Toutefois, l'ère de la bourgeoisie a simplifié les antagonismes de classes. Globalement, la société semblait divisée en deux camps hostiles, deux vastes classes se faisant face : la Bourgeoisie et le Prolétariat.

Mais la révolution ne s'est pas produite à l'échelle globale, et la "dictature du prolétariat", là où elle est apparue ou a prétendu apparaître, s'est avérée incapable d'établir la liberté. A contrario, la technologie a permis au capitalisme de s'imposer comme un consensus nécessaire. Le travailleur moderne des sociétés avancées s'est élevé à mesure des progrès de l'industrie, plus qu'il ne s'est enfoncé dans une condition inférieure à celle de sa propre classe. La pauvreté ne s'est pas développée plus vite que la population et la richesse. L'industrie rationnelle de type Fordien n'a pas transformé les travailleurs en prolétariat appauvri, mais en consommateurs de masse de la production de masse. Et civiliser le prolétariat fait maintenant partie de la stratégie auto-protectrice de la bourgeoisie.

Dans le même sens, l'éducation universelle et le refus de l'exploitation industrielle des enfants, issus du

programme tant décrié de la révolution prolétarienne, devinrent une norme de la morale sociale bourgeoise. Avec l'éducation universelle, les travailleurs eurent accès aux média qui pouvaient les pousser à consommer toujours plus. Le développement du son enregistré, de la téléphonie, du cinéma, de la diffusion radiophonique et télévisée changea la relation du travailleur à la culture bourgeoise, au moins autant qu'il changea la culture elle-même.

La musique, par exemple, fut durant une grande partie de notre histoire un non-objet hautement périssable, un processus social se produisant à un endroit et un instant donnés, consommé sur place par un mélange de gens qui en étaient à divers degrés les consommateurs et les producteurs. Après l'avènement de l'enregistrement, la musique devint un objet non périssable qui pouvait être transporté sur de longues distances, et ceux qui la produisaient furent inéluctablement spoliés. La musique devint, en tant que bien de consommation, un moyen pour ses nouveaux "propriétaires" de susciter toujours plus de consommation, de créer des demandes de la part de la nouvelle classe de consommateurs de masse, et d'orienter ces demandes dans un sens profitable au système de propriété. De même avec le nouveau médium que constituèrent les images animées, qui pendant des décennies agirent sur la nature de la cognition humaine, en captant une fraction substantielle du temps quotidien de chaque travailleur pour lui transmettre des messages l'incitant à plus de consommation. Des dizaines de milliers de telles publicités passèrent, chaque année, devant les yeux de chaque enfant, réduisant à une nouvelle forme de servitude ceux-là même qu'on avait affranchis d'actionner les machines de la production : ils étaient maintenant enrôlés de force pour actionner les machines de la consommation.

Ainsi les conditions d'accès à la société bourgeoise furent rendues moins étroites, plus à-même d'englober la richesse qu'elle avaient créée. Ainsi fut traitée l'absurde épidémie de surproduction récurrente. Et il n'y eut soudain plus d'excédent de civilisation, de moyens de subsistance, d'industrie ou de commerce.

Mais la bourgeoisie ne peut exister sans constamment révolutionner les instruments de production, et de ce fait les rapports à la production, et à leur suite les rapports sociaux dans leur ensemble. La production en constante révolution, le dérangement initerrompu de tous les liens sociaux, l'incertitude et l'agitation permanentes distinguent l'ère bourgeoise des précédentes. Tous les repères fixes ou rapidement entérinés comme tels, avec leur lot de préjudices consentis et d'opinions antiques et vénérables, ont été balayés, et ceux qui tendent à se former deviennent caducs avant même de s'être ossifiés. Tout ce qui est solide se fond dans l'air.

Avec l'adoption de la technologie numérique, le système de la production de masse soutenue par une culture de consommation de masse a donné naissance à de nouvelles conditions sociales, où se cristallise une nouvelle structure d'antagonisme de classes.

La bourgeoisie, grâce à l'amélioration rapide de tous les instruments de production et aux moyens de communication immenséments facilités, entraîne toutes les nations, même les plus barbares, vers la civilisation. Les prix bas de ses biens et services sont l'artillerie lourde avec laquelle elle abat toutes les murailles de Chine, et force à la capitulation la haine obstinée et intense que les barbares éprouvent vis-à-vis de l'étranger. Elle contraint les nations, sous peine d'extinction, à adopter sa culture et ses principes de propriété intellectuelle ; elle les contraint à introduire ce qu'elle nomme civilisation en leur coeur, pour qu'elles deviennent elles mêmes bourgeoises. En un mot, elle crée un monde à son image. Mais les instruments même de sa communication et de sa manipulation culturelle constituent les moyens de résistance que l'on peut retourner contre elle.

La technologie numérique transforme l'économie bourgeoise. Les biens dominants dans le système de production (les articles culturels qui sont à la fois des objets commercialisés et des instructions pour le travailleur sur ce qu'il doit acheter et comment), ainsi que les autres formes de culture et de savoir, ont maintenant un coût marginal nul. N'importe qui peut bénéficier de n'importe quel travail culturel : musique, art, littérature, information technique, science, ou toute autre forme de connaissance. Les barrières de l'inégalité sociale et de l'isolement géographique se sont dissoutes. A la place des anciens isolement et auto-suffisance locaux et nationaux, il y a des liens dans toutes les directions, une interdépendance universelle entre les gens, non seulement au niveau matériel, mais aussi au niveau de la production intellectuelle. Les créations intellectuelles des individus deviennent propriété commune. La société bourgeoise moderne avec ses relations de production, d'échange et de propriété, société qui a su orchestrer d'aussi gigantesques moyens de production et d'échange, est comme l'apprenti sorcier incapable de contrôler les pouvoirs du monde occulte qu'il a lui-même convoqué par ses sortilèges.

Suite à cette évolution, l'homme est contraint de considérer avec lucidité ses véritables conditions de vie, et ses relations à sa propre espèce. La société nous confronte au simple fait que, lorsque toute réalisation intellectuelle qui soit belle ou utile peut être la propriété de tous au même coût que si elle était propriété d'un seul (chacun récoltant alors toute la valeur humaine de chaque augmentation de la connaissance), il n'est plus moral d'exclure. Que Rome eût le pouvoir de nourrir tout le monde largement sans qu'il ne lui coûte plus que de nourrir César, et un seul homme souffrant de famine eût justifié que César fût renversé. Mais le système bourgeois de propriété exige que la culture et la connaissance soient rationnés selon la capacité de payer. Les modèles traditionnels et alternatifs, rendus de nouveaux viables par les technologies d'interconnexion, ceci incluant les associations volontaires de ceux qui créent et ceux qui les soutiennent, doivent être contraints à une compétition inégale avec les écrasants systèmes propriétaires de communication de masse. Ces systèmes sont à leur tour basés sur l'appropriation des droits communs des peuples à disposer du spectre électromagnétique. Dans toute la société numérique, la classe des travailleurs de la connaissance (artistes, musiciens, écrivains, étudiants, spécialistes en technologie, etc..., qui essaient d'améliorer leurs conditions de vie en copiant et modifiant de l'information) est radicalisée par le conflit entre ce qu'elle sait possible et ce que l'idéologie bourgeoise la contraint à accepter. De cette discordance émerge la conscience d'une nouvelle classe, qui en prenant conscience d'elle-même amorce la chute de la propriété.

L'avance de la société numérique, dont la bourgeoisie est l'involontaire promoteur, remplace l'isolement des créateurs, dû à la concurrence, par leur combinaison révolutionnaire, due à l'association. Les créateurs de connaissance, de technologie et de culture découvrent qu'ils n'ont plus besoin de la structure de production basée sur la propriété ni de la structure de distribution basée sur la contrainte du paiement. L'association, et son modèle anarchique de production sans propriété, rend possible la création de logiciel libre, par lequel les créateurs renforcent leur contrôle de la technologie servant à son tour à produire.[1] Le réseau lui-même, libéré du contrôle des diffuseurs et autres propriétaires de bande passante, devient le locus d'un nouveau système de distribution, basé sur l'association entre homologues sans contrôle hiérarchique, qui remplace le système contraignant de distribution de la musique, de la vidéo, et autres biens immatériels. Les universités, les bibliothèques, et les institutions qui leur sont liées deviennent des alliés de la nouvelle classe, interprétant leur rôle historique de distributeurs du savoir comme obligation de fournir un accès toujours plus complet au savoir pour tous, librement. Libérer l'information du contrôle de la propriété libère le travailleur de son rôle imposé de gardien de la machine. La libre information permet au travailleur d'investir son temps, non dans la consommation induite par la culture bourgeoise, avec ses invites toujours plus pressantes à la

consommation stérile, mais plutôt dans la culture de son esprit et de ses compétences. A mesure qu'il prend conscience de son pouvoir de création, il cesse d'être un participant passif des systèmes de production et de consommation dans lesquels la société bourgeoise l'avait piégé.

Mais la bourgeoisie, partout où elle a la haute main, a mis fin à toutes les relations féodales, patriarcales, idylliques. Elle a impitoyablement déchiré les liens féodaux hétéroclites qui reliaient l'homme à ses "suzerains naturels", et n'a laissé entre l'homme et l'homme d'autre connexion que l'intérêt à l'état pur, que le "paiement cash" inhumain. Elle a noyé les extases les plus sacrées de la ferveur religieuse, de l'enthousiasme chevaleresque, du sentimentalisme philistin, dans les eaux glacées du calcul égotiste. Elle a réduit les valeurs personnelles à des valeurs marchandes. Et en lieu et place des innombrables gradations des libertés admises, elle a mis en place cette unique, inconcevable liberté : le Libre-Echange. En un mot, l'exploitation voilée d'illusions religieuses et politiques a été remplacée par l'exploitation pure, directe, brutale et dénuée de scrupules.

Face à la profonde libération à venir des classes de travailleurs, dont le pouvoir d'accès au savoir et à l'information transcende désormais l'ancien rôle étroit de consommateurs de culture de masse, le système bourgeois de propriété se trouve nécessairement poussé dans ses ultimes retranchements. Avec son instrument favori, le libre-échange, il tente de provoquer la profonde crise de surproduction qu'il craignait auparavant. Désespérant d'enfermer les créateurs dans leur rôle de consommateurs salariés, la propriété bourgeoise tente de transformer la pauvreté matérielle dans certaines régions du globe en source de biens bon marché avec lesquels ramener à la passivité culturelle non pas les barbares, mais ses propres possessions les plus chères - les travailleurs technologiques des sociétés les plus avancées.

A cet instant, travailleurs et créateurs forment encore une masse incohérente dispersée dans le monde entier, et demeurent divisés par leur mutuelle concurrence. Parfois les créateurs sont victorieux, mais provisoirement. Le véritable fruit de leur combat n'est pas son résultat immédiat, mais l'union qui s'étend sans cesse. Cette union est facilitée par les moyens de communication évolués créés par l'industrie moderne et qui placent travailleurs et créateurs provenant d'endroits divers en contact mutuel. C'était ce simple contact qui manquait pour centraliser les nombreuses luttes locales, toutes du même type, en une lutte globale entre classes. Mais toute lutte de classes est une lutte politique. Et cette union, que les bâtisseurs du moyen-âge auraient mis des siècles à réaliser avec leurs misérables voies de communication, les modernes travailleurs de la connaissance peuvent la réaliser en quelques années.

Liberté et Création

La bourgeoisie ne s'est pas contentée de forger les armes qui lui amènent la mort ; elle a aussi fait venir au monde les hommes chargés de servir ces armes, la classe des travailleurs du numérique : les créateurs.
Pétris des compétences et des savoirs créateurs de valeur sociale et marchande, réfractaires au fait d'être réduits à l'état de commodités, capables de produire collectivement toutes les technologies de la liberté, de tels travailleurs ne peuvent être réduits à de simples appendices de la machine. Là où autrefois l'ignorance associée à l'isolement géographique liaient indistinctement le prolétaire à l'armée industrielle dont il était un soldat transparent et corvéable, les créateurs qui contrôlent collectivement le réseau des communications humaines préservent leur individualité, et proposent la valeur engendrée par leur labeur intellectuel par divers moyens bien plus favorables à leur bien-être et à leur liberté que tout ce que le système de propriété bourgeoise leur avait jamais concédé.

Mais en proportion égale au succès des créateurs à établir la véritable économie libre, la bourgeoisie doit renforcer la structure de production et distribution par la contrainte que dissimule sa préférence supposée pour les "marchés libres" et le "libre échange". Bien que fermement décidée à défendre par la force des accords s'appuyant sur la force, quoique masquée, la bourgeoisie essaie d'abord de réimposer la contrainte par l'entremise de son instrument favori d'asservissement, les institutions de sa loi. Comme l'ancien régime en France, qui croyait que la propriété féodale pouvait être maintenue par la force conservatrice de la loi en dépit de la modernisation de la société, les possédants dans la culture bourgeoise attendent de leurs lois de propriété qu'elles constituent un rempart magique contre les forces qu'elles ont elles-mêmes libérées.

A un certain point du développement des moyens de production et d'échange, les conditions de la production et des échanges dans la société féodale, l'organisation féodale de l'agriculture et de l'industrie manufacturière, en un mot, les relations de propriété féodales devinrent incompatibles avec l'état de développement des forces productives; elles devinrent autant de chaînes. Elles devaient être mises en pièces, elles le furent.

A leur place s'installa la libre concurrence, accompagnée d'une constitution sociale et politique adaptée, et de l'influence sociale et politique de la classe bourgeoise. Mais la " libre concurrence " ne fut jamais qu'une aspiration de la société bourgeoise, qui privilégia constamment la préférence capitaliste pour le monopole. La propriété bourgeoise érigea en exemple le concept de monopole, abaissant au niveau d'arrangements de circonstance le dogme de liberté effrontément proclamé par la loi bourgeoise. Comme, dans la nouvelle société numérique, les créateurs établissent de véritables formes de libre activité économique, le dogme de la propriété bourgeoise entre en conflit ouvert avec le dogme de la liberté bourgeoise. Protéger la propriété des idées nécessite de supprimer la libre technologie, donc la liberté d'expression. La puissance des Etats est employée à interdire la libre création. Scientifiques, artistes, ingénieurs et étudiants se trouvent empêchés de créer ou de partager du savoir, sur la base du fait que leurs idées mettent en péril la propriété des possesseurs du système de production et de distribution culturelle. C'est dans les tribunaux des possédants que les créateurs rencontrent le plus clairement leur identité de classe, et c'est logiquement là que le conflit commence.

Mais les lois de la propriété bourgeoise ne sont pas des amulettes magiques contre les conséquences de la technologie bourgeoise : le balai de l'apprenti sorcier peut continuer d'éponger, l'eau continue de monter. C'est dans le domaine de la technologie que la défaite de la propriété s'accomplit finalement, là où les modes de production et de distribution font tomber les chaînes de la loi démodée.

Toutes les précédentes classes ayant eu la haute main sur la société ont cherché à fortifier leur statut acquis en soumettant la société au sens large à leurs conditions d'appropriation. Les travailleurs du savoir ne peuvent se rendre maîtres des forces productives de la société, sauf à abolir leurs propres modes d'appropriation actuels, et par conséquent tout autre mode plus ancien d'appropriation. Il leur appartient de choisir le dévouement révolutionnaire à la liberté, à l'abolition de la propriété des idées, à la libre circulation du savoir, à la restauration de la culture comme fondement symbolique que partagent tous les êtres humains.

Aux propriétaires de la culture, nous disons : vous êtes horrifiés par notre intention de nous débarrasser de la propriété privée appliquée aux idées. Mais dans votre société, les neuf dixièmes de la population sont déjà débarrassés de la propriété privée. Leurs employeurs s'approprient immédiatement ce qu'ils créent, revendiquant le fruit de leur intelligence au nom du droit des brevets, du copyright, du secret

commercial et d'autres formes de "propriété intellectuelle". Leur droit naturel dans le spectre électromagnétique, qui peut permettre à toute personne de communiquer et d'apprendre des autres, librement, presque sans limites de capacité et à un coût nominal, leur a été enlevé par la bourgeoisie, et leur est restitué au prix fort sous forme de biens de consommation (culture audiovisuelle et services de télécommunication). Leur créativité ne trouve pas d'issue : leur musique, leur art, leur tradition orale sont noyés dans les articles de la culture capitaliste, amplifiée par toute la puissance de cet oligopole qu'est la "télédiffusion", face à laquelle ils sont censés demeurer passifs et consommer plutôt que créer. En un mot, la propriété que vous revendiquez procède du vol : son existence pour un petit nombre est uniquement due à sa non-existence pour tous les autres. Vous nous reprochez, donc, d'essayer de nous débarrasser d'une forme de propriété qui ne peut exister qu'à condition d'être hors de portée d'une immense majorité de la société.

D'aucuns ont objecté qu'en l'absence de propriété privée des idées et de la culture tout travail créatif cesserait, par manque d' "incitation" ou de motivation, et que nous serions saisis d'une paresse universelle.

De ce point de vue, il n'y aurait dû avoir ni musique, ni art, ni technologie, ni apprentissage avant l'avènement de la bourgeoisie, qui seule élabora le concept de placer l'ensemble de la culture et du savoir sous l'emprise du cash. Confronté à l'éventualité d'une production et d'une distribution libre, avec le logiciel libre et le développement de technologies de libre distribution qui en découle, cet argument contredit simplement des faits visibles et incontestables. Les faits sont occultés au profit du dogme selon lequel l'ordre des choses qui a brièvement caractérisé la production intellectuelle et la distribution culturelle durant la courte apogée de la bourgeoisie serait le seul ordre possible, et ce malgré l'évidence tant passée que présente.

Ainsi, nous disons aux possédants : cette méprise qui consiste à considérer les structures sociales qui naissent de votre mode présent de production et de propriété (ces relations historiques qui naissent et meurent au fil des progrès de la production) comme lois éternelles de la nature et de la raison, cette méprise donc, vous l'avez en commun avec toutes les classes dirigeantes qui vous ont précédés. Ce que vous percevez clairement concernant la propriété antique, ce que vous admettez concernant la propriété féodale, vous refusez évidemment de l'admettre concernant votre propre forme de propriété bourgeoise.

Nos conclusions théoriques ne sont aucunement basées sur des idées ou principes inventés ou découverts par tel ou tel pseudo-réformateur universel. Elles expriment simplement, en termes généraux, les relations réelles qui jaillissent d'une lutte des classes actuelle, d'un mouvement historique qui se déroule sous nos propres yeux.

Lorsque les gens parlent d'idées qui révolutionnent la société, ils ne font qu'exprimer le fait qu'au sein de l'ancienne société, les éléments d'une société nouvelle ont été créés, et que la dissolution des anciennes idées va de pair avec le dissolution des anciennes conditions d'existence.

Nous, créateurs de la société de la libre information, prétendons arracher graduellement à la bourgeoisie le patrimoine commun de l'humanité. Nous aspirons à la restitution de l'héritage culturel qui nous a été volé au gré de la "propriété intellectuelle", au même titre que l'ont été les moyens électromagnétiques de transmission. Nous sommes engagés dans la lutte pour la libre expression, le libre savoir, et la libre technologie. Les moyens par lesquels nous mènerons ce combat seront bien entendu différents dans les différents pays, mais les règles suivantes seront assez généralement applicables :

1. Abolition de toute forme de propriété privée appliquée aux idées.
2. Retrait de toute licence exclusive ou droit privilégié d'exploiter le spectre électromagnétique. Annulation de toute concession permanente de fréquence électromagnétique.
3. Développement d'infrastructures de transmission dans le spectre électromagnétique qui garantissent à toute personne un droit égal à communiquer.
4. Développement social commun des programmes informatiques et de toute autre forme de logiciel, information génétique incluse, considérés comme biens publics.
5. Respect sans restriction de la liberté d'expression, incluses toutes formes d'expression technique.
6. Protection de l'intégrité des oeuvres de création.
7. Accès libre et égalitaire à toute information produite par le secteur public et tout contenu éducatif en usage dans toute branche du système éducatif public.

Par ces moyens, entre autres, nous nous engageons dans la révolution qui libère l'esprit humain. En rejetant le système de propriété privée appliquée aux idées, nous jetons les bases d'une société réellement juste, dans laquelle le libre développement de chacun est la condition du libre développement de tous.

[1] Le mouvement du Logiciel Libre s'est appuyé sur des programmeurs (certains payés, d'autres non) depuis les années 1980 pour créer le système d'exploitation GNU/Linux et les programmes qui lui sont associés qui peuvent être copiés, modifiés et redistribués par tous leurs utilisateurs. Cet environnement technique, maintenant omniprésent et plus efficace que ses concurrents propriétaires issus de l'industrie, libère les utilisateurs d'ordinateurs de la forme de contrôle technologique monopolistique qui aurait dû dominer la révolution des ordinateurs personnels, du point de vue du capitalisme. En changeant le cours de la production propriétaire du plus puissant monopole sur terre, le mouvement du logiciel libre démontre que des associations de travailleurs du numérique sont capables de produire des biens de meilleurs qualité, distribués à coût nominal, que le système de production capitaliste, en dépit des si fameuses "incitations" découlant de la propriété et de lois favorisant une "propriété intellectuelle" exclusive.

French translation: Pierre-Yves Gibello.