Annexes de la conférence sur « Le FreeSoftware dans l'entreprise »

Table des matières

Définition du « Free Software » selon la « Free Software Foundation »

(http://www.fsf.org)

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:

0) **La liberté d'exécuter le programme, pour tous les usages** (liberté 0).
1) **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 condition requise.
2) **La liberté de redistribuer des copies**, donc d'aider votre voisin, (liberté 2).
3) **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 libre de redistribuer des copies, avec ou sans modification, 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 vos 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 ultérieurement avec le développeur ou tout autre entité spécifique.

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. Il y a une exception s'il n'y a pas moyen de produire une version binaire ou exécutable, 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'accès au code source du programme. Par conséquent, **l'accessibilité du code source est une condition requise pour un logiciel 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 manière 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 programme, 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.

Ainsi, vous pouvez avoir à payer pour obtenir une copie d'un logiciel du projet GNU ou vous pouvez l'obtenir 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 usage commercial. **Le développement commercial de logiciel libre n'est plus l'exception ; de tels programmes**

sont des logiciels commerciaux libres.

Les règles sur la manière d'empaqueter une version modifiée sont acceptables si elles n'entravent pas votre liberté de la publier. Les règles disant «si vous publier le programme par ce moyen, vous devez le faire par ce moyen aussi» sont acceptables aux mêmes conditions (notez que de telles règles doivent vous laisser le choix de publier ou non le programme).

Dans le projet GNU, nous utilisons le «copyleft» 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, mais si votre programme est libre non-copylefté, nous pouvons tout de même l'utiliser.

[...]

Parfois le contrôle gouvernemental des exportations ou des sanctions économiques peuvent vous priver de la liberté de distribuer des copies de programmes à l'étranger. Les développeurs de logiciels 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'utilisation des programmes**. De cette manière, les restrictions n'affecteront pas les activités et les personnes se trouvant hors de la juridiction de leurs gouvernements.

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 pour un essai sur l'utilisation de ces termes. Nous avons aussi une liste de traductions de «free software» dans de nombreuses langues.

Enfin, notez 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 est définie comme libre, nous la jugeons sur ces critères pour déterminer si elle convient à leur esprit tout comme à leur formulation précise. Si une licence inclut des restrictions innaceptables, nous la rejetons même si nous n'avons pas anticipé le problème dans ces critères.

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

Autres textes à lire

Un autre groupe à commencé à utiliser **le terme «open source» pour exprimer quelque chose de proche (mais pas d'identique) au «logiciel libre».**

[...]

Les demandes de renseignements et les questions sur la FSF & le projet GNU sont a adresser à gnu@gnu.org. Autres moyens de contacter la FSF.

Les commentaires sur ces pages web sont a envoyer à webmasters@www.gnu.org, envoyez toute autre question à gnu@gnu.org.

Logiciel Libre

Un logiciel libre est un logiciel qui est fourni avec **l'autorisation pour quiconque de l'utiliser, de le copier, et de le distribuer, soit sous une forme conforme à l'original, soit avec des modifications, ou encore gratuitement ou contre un certain montant**. Ceci signifie en particulier que **son code source doit être disponible**. «S'il n'y a pas de sources, ce n'est pas du logiciel.» Ceci est une définition simplifiée; voir aussi la définition complète.

[...]

Du moment qu'il est libre, tout programme peut, potentiellement, faire partie d'un système d'exploitation libre tel que GNU, ou les versions libres du système GNU/Linux.

Il existe de nombreuses façons de rendre un logiciel libre---beaucoup de détails peuvent être définis de différentes façons, tout en gardant au logiciel son caractère libre. Certaines de ces variations sont décrites ci-après.

Un logiciel est libre du point de vue de la liberté, et non du prix. Mais les sociétés éditrices de logiciels propriétaires utilisent parfois le terme «logiciel libre» pour parler de logiciels gratuits. Ce qui veut parfois dire que vous pouvez en obtenir une copie binaire gratuitement, ou qu'une copie de ce logiciel est comprise dans le prix d'achat de votre ordinateur. Ceci n'a rien à voir avec le terme de logiciel libre, tel que nous le définissons dans le projet GNU.

A cause de cette confusion potentielle il serait souhaitable, chaque fois qu'une société informatique annonce que son produit est un logiciel libre, de vérifier les conditions de distribution, afin de s'assurer que les usagers disposent de toutes les libertés associées au logiciel libre. Parfois il s'agit, effectivement, d'un logiciel libre, parfois non.

L'anglais utilise le même mot «free» pour «libre» et «gratuit». C'est pourquoi il y a souvent confusion sur la nature des termes "free software". Nous tenons à souligner qu'il ne s'agit pas du prix mais de la liberté d'utilisation.

Le logiciel libre est souvent plus fiable que le logiciel non-libre.

Logiciel Open Source

Le terme logiciel «open source» (littéralement à source ouvert) est utilisé par certaines personnes pour signifier plus ou moins la même chose que logiciel libre. Nous préférons le terme «logiciel libre»; [...]

Logiciel du domaine public

Logiciel du domaine public veut dire logiciel non soumis aux droits d'auteurs. C'est un cas particulier de logiciel libre "non-copylefté", ce qui veut dire que certaines copies, ou certains versions modifiées, peuvent ne pas être du tout libres.

Parfois, on utilise le terme «domaine public» d'une façon peu précise pour dire «libre» ou «disponible gratuitement». Toutefois, «domaine public» est un terme légal qui signifie précisément que le logiciel n'est pas «soumis au copyright». Afin d'être plus précis, nous conseillons d'utiliser le terme «domaine public» dans ce cas uniquement, et d'utiliser d'autres termes dans les autres cas.

Logiciel copylefté (sous gauche d'auteur)

Le logiciel sous copyleft (littéralement, gauche d'auteur) est un logiciel libre, dont les conditions de distribution interdisent aux nouveaux distributeurs d'ajouter des restrictions supplémentaires lorsqu'ils redistribuent ou modifient le logiciel. Ceci veut dire que chaque copie du logiciel, même si elle a été modifiée, doit être un logiciel libre.

Dans le projet GNU, presque tous les logiciels que nous créons sont soumis au copyleft, car notre but est de donner à chaque utilisateur les libertés garanties par le terme «logiciel libre». Voir Qu'est-ce que le copyleft

pour plus d'explications sur le fonctionnement du copyleft et savoir pourquoi nous l'utilisons.

Le copyleft est un concept général; pour l'appliquer à un programme, vous avez besoin d'un ensemble de termes relatifs à sa distribution. Il y a de nombreuses façons d'écrire ces conditions de distribution.

Logiciel libre non-copylefté

Le logiciel libre non-copylefté est diffusé par son auteur avec la permission de le redistribuer et de le modifier, mais aussi d'y ajouter d'autres restrictions.

Si un programme est libre, mais non-copylefté, alors certaines copies ou versions modifiées peuvent ne plus être libres du tout. Une société informatique peut compiler ce programme, avec ou sans modifications, et distribuer le fichier exécutable sous forme de produit logiciel propriétaire.

Le Système X Window illustre bien ce cas. Le Consortium X distribue X11 avec des conditions de distribution qui en font un logiciel libre non-copylefté. Si vous le souhaitez, vous pouvez en obtenir une copie qui possède de tels termes de distribution et qui est libre. Toutefois, il existe aussi des versions non-libres, et il y a des stations de travail ainsi que des cartes graphiques pour PC pour lesquelles les versions non-libres sont les seules qui fonctionnent. Si vous utilisez ce matériel-là, pour vous, X11 n'est pas un logiciel libre.

Logiciel couvert par la GPL

La GNU GPL (Licence Publique Générale GNU) (20 K octets) est un ensemble spécifique de conditions de distribution pour copylefter un programme. Le projet GNU l'utilise comme conditions de distribution de la plupart des logiciels GNU.

Le système GNU

Le système GNU est un système d'exploitation libre complet façon Unix.

Un système d'exploitation comparable à UNIX contient de nombreux programmes. Le système GNU comprend tous les logiciels GNU, ainsi que bien d'autres paquetages tels que le X Window System et TeX, qui ne sont pas des logiciels GNU.

Nous développons et accumulons des composants pour ce système depuis 1984; la première mise à disposition en test d'un «système GNU complet» remonte à 1996. Aujourd'hui, en 2001, le système fonctionne correctement, et des gens travaillent pour faire fonctionner GNOME et PPP avec ce système. Dans le même temps, le système GNU/Linux, une ramification du système GNU utilisant Linux comme kernel, a obtenu un grand succès.

Puisque le but du GNU est d'être libre, chacun de ses moindres composants doit être un logiciel libre. Tous ne doivent cependant pas être copyleftés; n'importe quel type de logiciel libre pourra y figurer légalement, s'il permet d'atteindre les objectifs techniques. Nous pouvons et nous utilisons des logiciels libres non-copylefté comme le X Window System.

Programmes GNU

Les «programmes GNU» sont équivalents aux Logiciels GNU. Un programme Toto est un programme GNU s'il est un logiciel GNU.

Logiciel GNU

Un logiciel GNU est un logiciel diffusé sous les auspices du projet GNU. La plupart des logiciels GNU sont soumis à un copyleft, mais pas tous; cependant, tous les logiciels GNU doivent être des logiciels libres.

Si un programme est un logiciel GNU, nous disons aussi qu'il est un programme GNU.

Certains des logiciels GNU sont réalisés par le personnel de la Free Software Foundation, mais la plus grande

partie des logiciels GNU est apportée par des volontaires. Certaines contributions sont sous copyright de la Free Software Foundation; d'autres appartiennent aux auteurs du logiciel.

Logiciel semi-libre

Le logiciel semi-libre est un logiciel qui n'est pas libre, mais qui s'accompagne de la permission pour les personnes physiques de l'utiliser, de le copier, de le distribuer, et de le modifier (y compris pour la distribution des versions modifiées) dans un but non lucratif. PGP est un exemple de programme semi-libre.

Un logiciel semi-libre est toujours mieux qu'un logiciel propriétaire, mais cela pose toujours des problèmes, et nous ne pouvons l'utiliser dans un système d'exploitation libre.

Les restrictions du copyleft sont conçues pour protéger les libertés fondamentales pour tous les utilisateurs. Pour nous, la seule justification à la définition d'une restriction substantielle sur l'utilisation d'un programme est d'empêcher l'ajout d'autres restrictions par d'autres personnes. Les programmes semi-libres possèdent des restrictions supplémentaires, motivées par des buts purement égoïstes.

Il est impossible d'inclure du logiciel semi-libre dans un système d'exploitation libre. Ceci est dû au fait que les conditions de distribution du système d'exploitation dans son entier sont la somme des conditions de distribution de tous les programmes qui le composent. Y ajouter un seul logiciel semi-libre rendrait le système tout entier seulement semi-libre. Il y deux raisons pour lesquelles nous ne voulons pas que cela se produise:

Nous pensons que le logiciel libre doit l'être pour tout le monde -- y compris les entreprises, et pas seulement les écoles et les amateurs. Nous voulons inviter l'entreprise à utiliser le système GNU en entier et, par conséquent nous ne devons pas y inclure de logiciel semi-libre.

La distribution commerciale de systèmes d'exploitation libres, incluant le système GNU/Linux, est très importante, et les utilisateurs apprécient la possibilité d'acheter des distributions commerciales sur CD-ROM. L'inclusion d'un seul programme semi-libre dans un système d'exploitation supprimerait la distribution commerciale de CD-ROM pour ce système.

La Free Software Foundation étant elle-même non-commerciale, elle aurait donc le droit d'utiliser légalement un programme semi-libre «en interne». Mais nous ne le faisons pas, parce que cela minerait nos efforts pour obtenir un programme que nous pourrions alors inclure dans GNU.

Si un travail nécessite l'utilisation d'un logiciel, alors tant qu'il n'existe pas de programme libre permettant de le réaliser, le système GNU contient une lacune. Nous devons dire aux volontaires, «Nous n'avons pas encore de programme pour faire ce travail dans GNU, et nous espérons donc que vous en écrirez un». Si nous mêmes nous utilisions un programme semi-libre pour faire le travail en question, cela minerait notre discours; et annulerait la nécessité (pour nous, et pour ceux qui suivraient notre point de vue) de développer un équivalent libre. C'est pourquoi nous ne le faisons pas.

Logiciel propriétaire

Le logiciel propriétaire est un logiciel qui n'est ni libre, ni semi-libre. Son utilisation, sa redistribution ou sa modification sont interdites, ou exigent une autorisation spécifique, ou sont tellement restreintes que vous ne pouvez en fait pas le faire librement.

La Free Software Foundation suit une règle consistant à ne jamais installer un logiciel propriétaire sur nos ordinateurs, sauf à titre temporaire dans le but spécifique d'élaborer un remplacement libre à ce même logiciel. Exception faite de ce cas, nous pensons qu'il n'existe aucune excuse pour l'installation d'un programme propriétaire.

Par exemple, nous estimons justifiée l'installation d'Unix sur nos ordinateurs dans les années 1980, parce que nous l'utilisions pour écrire une version libre en remplacement d'Unix.

Actuellement, puisque des systèmes d'exploitation libres sont disponibles, l'excuse n'est plus valable; nous avons éliminé tous nos systèmes d'exploitation non-libres, et chaque ordinateur que nous installons doit fonctionner avec un système d'exploitation complètement libre.

Nous n'insistons pas pour que les utilisateurs de GNU, ou ses contributeurs, suivent cette règle. C'est une règle que nous nous imposons nous-mêmes. Mais nous espérons que vous déciderez de la suivre également.

Freeware

Le terme «freeware» n'a pas de définition claire communément acceptée, mais elle est utilisée couramment pour des paquetages qui autorisent la redistribution mais pas la modification (et dont le code source n'est pas disponible). Ces paquetages ne sont pas des logiciels libres, donc n'utilisez pas, s'il vous plaît, «freeware» pour parler de logiciel libre.

Shareware (partagiciel)

Le partagiciel est un logiciel qui s'accompagne de la permission de redistribuer des copies, mais qui mentionne que toute personne qui continue à en utiliser une copie est obligée de payer des royalties.

Les sharewares ne sont pas des logiciels libres ou même semi-libres. Pour deux raisons:

* Pour les sharewares, le code source n'est pratiquement jamais fourni; et donc vous ne pouvez pas du tout modifier le programme.

Avec le shareware, il ne vous est pas permis d'effectuer une copie et de l'installer sans vous acquitter du paiement d'un droit licence, même pour des individus impliqués dans des activités non lucratives. (En pratique, ces termes de distribution sont en général peu appréciés, et les gens le font quand même, même si ce n'est pas permis.)

Logiciel commercial

Le logiciel commercial est un logiciel développé par une entreprise dont le but est de gagner de l'argent sur l'utilisation du logiciel. «Commercial» et «propriétaire» ne sont pas synonymes ! La plupart des logiciels commerciaux sont propriétaires, mais il y a des logiciels libres commerciaux, et il y a des logiciels non-commerciaux non-libres.

Par exemple, GNU Ada est toujours distribué sous les termes de la GPL GNU, et chaque copie est un logiciel libre; mais ses développeurs vendent des contrats de support. Quand leurs commerciaux parlent à de futurs clients, quelquefois ceux-ci disent, «Nous nous sentirions plus en sécurité avec un compilateur commercial.». Le représentant répond, «GNU Ada est un compilateur commercial; il est également un logiciel libre.»

Pour le Projet GNU, l'accent est mis sur l'autre composante: la chose importante est que GNU Ada est un logiciel libre; que ce soit un logiciel commercial n'est pas un point crucial. Cependant, le développement supplémentaire de GNU Ada qui résulte de ce commerce est certainement bénéfique.

Aidez-nous s'il vous plaît à faire prendre conscience que le logiciel libre commercial est possible. Vous pouvez y contribuer en faisant un effort pour ne pas dire «commercial» lorsque vous voulez dire «propriétaire».

Qu'est-ce que le copyleft ?

La manière la plus simple de rendre un programme libre est de le distribuer dans le domaine public, sans copyright. Cela autorise les gens à partager le programme et leurs améliorations si le coeur 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". 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 terme général, il y a cependant beaucoup de variations qui rentrent plus dans le détail. Dans le projet GNU, les conditions de distribution spécifiques que nous utilisons sont contenues dans la GNU General Public License (disponible au formt HTML, texte, et Texinfo). La GNU General Public License est appelée la GNU GPL.

Une forme alternative de copyleft, la GNU Lesser General Public License (LGPL) (disponible au format HTML, texte, et Texinfo), 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.

La GNU Library General Public License (27k d'HTML ou 25k de texte) est toujours disponible bien qu'elle ait été remplacée par la licence précédente.

La GNU Free Documentation License (FDL) (disponible au format HTML, texte et Texinfo) 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.

La GNU GPL est conçue de façon à pouvoir être appliquée à votre programme si vous en détenez le copyright. Vous n'aurez pas à modifier la GNU GPL pour le faire, mais seulement à ajouter des références appropriées à la GNU GPL au programme.

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 comme conseil.

L'utilisation des mêmes conditions de distribution pour plusieurs programmes différents facilite la copie de code entre ces programmes. Avec les mêmes conditions de distribution, il n'y a plus de souci d'incompatibilité. La LGPL 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.

[...]

Traduction de la GPL

Ces versions ne sont pas officielles. Légalement parlant, la version originale (anglaise) de la GPL est ce qui spécifie les vraies conditions de distribution pour les programmes GNU.

Les raisons pour lesquelles la FSF n'autorise pas les traductions à être officiellement valides est que les vérifier serait difficile et coûteux (ceci nécessitant l'aide de juristes bilingues dans d'autres pays). Encore pire, si une erreur se glissait dans les traductions, les conséquences seraient désastreuses pour la communauté du logiciel libre toute entière. Tant que les traductions ne sont pas officielles, elles ne peuvent pas faire de mal, et nous espérons qu'elles aideront plus de personnes à comprendre la GPL.

Nous permettons la publication de traductions de la GPL ou de la LGPL dans d'autres langues, du moment où vous (1) spécifiez dans vos traductions qu'elles ne sont pas officielles (voir plus loin comment faire), informez que vous ne comptez pas légalement comme substitution à la version originale, et (2) vous autorisez à ajouter des modifications à notre demande, si nous apprenons par d'autres amis de GNU que des modifications sont nécessaires pour rendre la traduction plus claire.

Pour spécifier que votre traduction n'est pas officielle, nous voudrions que vous ajoutiez le texte suivant au début, à la fois en anglais et à la fois dans la langue de la traduction, en remplaçant
«language» par le nom de cette langue :

This is an unofficial translation of the GNU General Public License into language. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL--only the original English text of the GNU GPL does that. However, we hope that this translation will help language speakers understand the GNU GPL better.

Soit en français:

Ceci est une traduction non officielle de la GNU General Public License en français. Elle n'a pas été publiée par la Free Software Foundation, et ne détermine pas les termes de distribution pour les logiciels qui utilisent la GNU GPL--seul le texte anglais original de la GNU GPL en a le droit. Cependant, nous espérons que cette traduction aidera les francophones à mieux comprendre la GNU GPL.

GNU GENERAL PUBLIC LICENSE (TRADUCTION NON OFFICIELLE)

Version 2, juin 1991

Préambule

Les licences relatives à la plupart des logiciels sont destinées à supprimer votre liberté de les partager et de les modifier. **Par contraste, la licence** publique générale **GNU General Public License veut garantir votre liberté de partager et de modifier les logiciels libres**, pour qu'ils soient vraiment libres pour tous leurs utilisateurs. La présente licence publique générale s'applique à la plupart des logiciels de la Free Software Foundation, ainsi qu'à tout autre programme dont les auteurs s'engagent à l'utiliser. (Certains autres logiciels sont couverts par la Licence Publique Générale pour Bibliothèques GNU à la place). Vous pouvez aussi l'appliquer à vos programmes.

Quand nous parlons de logiciels libres, nous parlons de liberté, non de gratuité. Nos licences publiques générales veulent vous garantir :

* que **vous avez toute liberté de distribuer des copies des logiciels libres** (et de facturer ce service, si vous le souhaitez);

* que **vous recevez les codes sources** ou pouvez les obtenir si vous le souhaitez ;

* que **vous pouvez modifier les logiciels ou en utiliser des éléments dans de nouveaux programmes libres**;

* et que vous savez que vous pouvez le faire.

Pour protéger vos droits, nous devons apporter des restrictions, qui vont interdire à quiconque de vous dénier ces droits, ou de vous demander de vous en désister. Ces restrictions se traduisent par certaines responsabilités pour ce qui vous concerne, si vous distribuez des copies de logiciels, ou si vous les modifiez.

Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou contre une rémunération, vous devez transférer aux destinataires tous les droits dont vous disposez. Vous devez vous garantir qu'eux-mêmes, par ailleurs, reçoivent ou peuvent recevoir le code source. Et vous devez leur montrer les présentes dispositions, de façon qu'ils connaissent leurs droits.

Nous protégeons vos droits en deux étapes:

1. Nous assurons le droit d'auteur (copyright) du logiciel, et

2. Nous vous proposons cette licence, qui vous donne l'autorisation légale de dupliquer, distribuer et/ou modifier le logiciel.

De même, pour la protection de chacun des auteurs, et pour notre propre protection, nous souhaitons nous assurer que tout le monde comprenne qu'**il n'y a aucune garantie** portant sur ce logiciel libre. **Si le logiciel est modifié** par quelqu'un d'autre puis transmis à des tiers, nous souhaitons **que les destinataires sachent que ce qu'ils possèdent n'est pas l'original**, de façon que tous problèmes introduits par d'autres ne se traduisent pas par une répercussion négative sur la réputation de l'auteur original.

Enfin, **tout programme libre est** en permanence **menacé par des brevets de logiciels**. Nous souhaitons éviter le danger que des sous-distributeurs d'un programme libre obtiennent à titre individuel des licences de brevets, avec comme conséquence qu'ils ont un droit de propriété sur le programme. Pour éviter cette situation, nous avons fait tout ce qui est nécessaire pour que **tous brevets doivent faire l'objet d'une concession de licence qui en permette l'utilisation libre par quiconque**, ou bien qu'il ne soit pas concédé

du tout.

Nous présentons ci-dessous les clauses et dispositions concernant la duplication, la distribution et la modification.

Conditions d'exploitation portant sur la duplication, la distribution et la modification

1. Le présent contrat de licence s'applique à tout programme ou autre ouvrage contenant un avis, apposé par le détenteur du droit de propriété, disant qu'il peut être distribué au titre des dispositions de la présente Licence Publique Générale. Ci-après, le "Programme" désigne l'un quelconque de ces programmes ou ouvrages, et un "ouvrage fondé sur le programme" désigne soit le programme, soit un ouvrage qui en dérive au titre de la loi sur le droit d'auteur ; plus précisément, il s'agira d'un ouvrage contenant le programme ou une version de ce dernier, soit mot à mot, soit avec des modifications et/ou traduit en une autre langue (ci-après, le terme "modification" englobe, sans aucune limitation, les traductions qui en sont faites). Chaque titulaire de licence sera appelé "concessionnaire".

Les activités autres que la duplication, la distribution et la modification ne sont pas couvertes par la présente licence; elles n'entrent pas dans le cadre de cette dernière. **L'exécution du programme n'est soumise à aucune restriction**, et les résultats du programme ne sont couverts que si son contenu constitue un ouvrage fondé sur le programme (indépendamment du fait qu'il a été réalisé par exécution du programme). La véracité de ce qui précède dépend de ce que fait le programme.

2. Le concessionnaire **peut dupliquer et distribuer des copies** mot à mot **du code source du programme tel qu'il les reçoit**, et ce sur un support quelconque, **du moment qu'il appose**, d'une manière parfaitement visible et appropriée, **sur chaque exemplaire, un avis approprié de droits d'auteur (Copyright)** et de renonciation à garantie; qu'il maintient intacts tous les avis qui se rapportent à la présente licence et à l'absence de toute garantie; et qu'**il transmet à tout destinataire du programme un exemplaire de la présente licence en même temps que le programme.**

Le concessionnaire **peut facturer** l'acte physique de **transfert d'un exemplaire**, et **il peut**, à sa discrétion, **proposer en échange d'une rémunération une protection en garantie**.

3. Le concessionnaire **peut modifier son ou ses exemplaires du programme** ou de toute portion de ce dernier, en formant ainsi un ouvrage fondé sur le programme, **et dupliquer et distribuer ces modifications** ou cet ouvrage selon les dispositions de la section 1 ci-dessus, du moment que le concessionnaire satisfait aussi à toutes ces conditions:

a. Le concessionnaire doit faire en sorte que **les fichiers modifiés portent un avis, parfaitement visible, disant que le concessionnaire a modifié les fichiers, avec la date de tout changement**.

b. Le concessionnaire doit faire en sorte que **tout ouvrage qu'il distribue ou publie, et qui, en totalité ou en partie, contient le programme ou une partie quelconque de ce dernier ou en dérive, soit concédé en bloc, à titre gracieux, à tous tiers au titre des dispositions de la présente licence**.

c. Si le programme modifié lit normalement des instructions interactives lors de son exécution, le concessionnaire doit, quand il commence l'exécution du programme pour une telle utilisation interactive de la manière la plus usuelle, faire en sorte que ce programme imprime ou affiche une annonce, comprenant un avis approprié de droits d'auteur, et un avis selon lequel il n'y a aucune garantie (ou autrement, que le concessionnaire fournit une garantie), et que les utilisateurs peuvent redistribuer le programme au titre de ces
dispositions, et disant à l'utilisateur comment visualiser une copie de cette licence (exception: si le programme par lui-même est interactif mais n'imprime normalement pas une telle annonce, l'ouvrage du concessionnaire se fondant sur le programme n'a pas besoin d'imprimer une annonce).

Les exigences ci-dessus s'appliquent à l'ouvrage modifié pris en bloc. **Si des sections identifiables de cet ouvrage ne dérivent pas du programme et peuvent être considérées raisonnablement comme représentant des ouvrages indépendants et distincts par eux-mêmes, alors la présente licence**, et ses dispositions, **ne s'appliquent pas à ces sections quand le concessionnaire les distribue sous forme d'ouvrages distincts**. Mais quand le concessionnaire distribue ces mêmes sections en tant qu'élément d'un

tout qui représente un ouvrage se fondant sur le programme, la distribution de ce tout doit se faire conformément aux dispositions de la présente licence, dont les autorisations, portant sur d'autres concessionnaires, s'étendent à la totalité dont il est question, et ainsi à chacune de ces parties, indépendamment de celui qu'il a écrite.

Ainsi, cette section n'a pas pour but de revendiquer des droits ou de contester vos droits sur un ouvrage entièrement écrit par le concessionnaire; bien plus, **l'intention est d'exercer le droit de surveiller la distribution d'ouvrages dérivée ou collective se fondant sur le programme**.

De plus, un simple assemblage d'un autre ouvrage ne se fondant pas sur le programme, avec le programme (ou avec un ouvrage se fondant sur le programme) sur un volume d'un support de stockage ou distribution, ne fait pas entrer l'autre ouvrage dans le cadre de la présente licence.

4. Le concessionnaire peut dupliquer et distribuer le programme (ou un ouvrage se fondant sur ce dernier, au titre de la Section 2), en code objet ou sous une forme exécutable, au titre des dispositions des Sections 1 et 2 ci-dessus, du moment que le concessionnaire effectue aussi l'une des opérations suivantes :

a. Lui joindre le code source complet correspondant, exploitable par une machine, code qui doit être distribué au titre des Sections 1 et 2 ci-dessus sur un support couramment utilisé
pour l'échange de logiciels ; ou bien

b. Lui joindre une offre écrite, dont la validité se prolonge pendant au moins 3 ans, de transmettre à un tiers quelconque, pour un montant non supérieur au coût pour le concessionnaire, de réalisation physique de la distribution de la source, un exemplaire complet, exploitable par une machine, du code source correspondant, qui devra être distribué au titre des dispositions des Sections 1 et 2 ci-dessus sur un support couramment utilisé pour l'échange des logiciels ; ou bien

c. Lui joindre les informations que le concessionnaire a reçues, pour proposer une distribution du code source correspondant (cette variante n'est autorisée que pour la distribution non commerciale, et seulement si le concessionnaire a reçu le programme sous forme exécutable ou sous forme d'un code objet, avec une telle offre, conformément à l'alinéa b) ci-dessus).

Le code source d'un ouvrage représente la forme préférée de l'ouvrage pour y effectuer des modifications. Pour un ouvrage exécutable, le code source complet représente la totalité du code source pour tous les modules qu'il contient, plus tous fichiers de définitions d'interface associés, plus les informations en code machine pour commander la compilation et l'installation du programme exécutable. Cependant, à titre d'exceptions spéciales, le code source distribué n'a pas besoin de comprendre quoi que ce soit qui est normalement distribué (sous forme source ou sous forme binaire) avec les composants principaux (compilateur, noyau de système d'exploitation, etc.) du système d'exploitation sur lequel est exécuté le programme exécutable, à moins que le composant, par lui-même, soit joint au programme exécutable.

Si la distribution de l'exécutable ou du code objet est réalisée de telle sorte qu'elle offre d'accéder à une copie à partir d'un lieu désigné, alors le fait d'offrir un accès équivalent à la duplication du code source à partir de ce même lieu s'entend comme distribution du code source, même si des tiers ne sont pas contraints de dupliquer la source en même temps que le code objet.

5. Le concessionnaire ne peut dupliquer, modifier, concéder en sous-licence ou distribuer le programme, sauf si cela est expressément prévu par les dispositions de la présente licence. Toute tentative pour autrement dupliquer, modifier, concéder en sous-licence ou distribuer le programme est répétée nulle, et met automatiquement fin aux droits du concessionnaire au titre de la présente licence. Cependant, les parties qui ont reçu des copies, ou des droits, de la part du concessionnaire au titre de la présente licence, ne verront pas expirer leur contrat de licence, tant que ces parties agissent d'une manière parfaitement conforme.

6. Il n'est pas exigé du concessionnaire qu'il accepte la présente licence, car il ne l'a pas signée. Cependant, rien d'autre n'octroie au concessionnaire l'autorisation de modifier ou de distribuer le programme ou ses ouvrages dérivés. Ces actions sont interdites par la loi si le concessionnaire n'accepte pas la présente licence. En conséquence, par le fait de modifier ou de distribuer le programme (ou un ouvrage quelconque se fondant sur le programme), le concessionnaire indique qu'il accepte la présente licence, et qu'il a la volonté de se conformer à toutes les clauses et dispositions concernant la duplication, la distribution ou la modification du programme ou d'ouvrages se fondant sur ce dernier.

7. Chaque fois que le concessionnaire redistribue le programme (ou tout ouvrage se fondant sur le programme), le destinataire reçoit automatiquement une licence de l'émetteur initial de la licence, pour dupliquer, distribuer ou modifier le programme, sous réserve des présentes clauses et dispositions. Le concessionnaire ne peut imposer aucune restriction plus poussée sur l'exercice, par le destinataire, des droits octroyés au titre des présentes. Le concessionnaire n'a pas pour responsabilité d'exiger que des tiers se conforment à la présente licence.

8. Si, en conséquence une décision de justice ou une allégation d'infraction au droit des brevets, ou pour toute autre raison (qui n'est pas limitée à des problèmes de propriétés industrielles), des conditions sont imposées au concessionnaire (par autorité de justice, par convention ou autrement), qui entrent en contradiction avec les dispositions de la présente licence, elles n'exemptent pas le concessionnaire de respecter les dispositions de la présente licence. Si le concessionnaire ne peut procéder à la distribution de façon à satisfaire simultanément à ces obligations au titre de la présente licence et à toutes autres obligations pertinentes, alors, en conséquence de ce qui précède, le concessionnaire peut ne pas procéder du tout à la distribution du programme. Par exemple, si une licence de brevet ne permettait pas une redistribution du programme, sans redevances, par tous ceux qui reçoivent des copies directement ou indirectement par l'intermédiaire du concessionnaire, alors le seul moyen par lequel le concessionnaire pourrait satisfaire tant à cette licence de brevet qu'à la présente licence, consisterait à s'abstenir complètement de distribuer le programme.

Si une partie quelconque de cette section est considérée comme nulle ou non exécutoire dans certaines circonstances particulières, le reste de cette section est réputé s'appliquer, et la section dans son ensemble est considérée comme s'appliquant dans les autres circonstances.

La présente section n'a pas pour objet de pousser le concessionnaire à enfreindre tous brevets ou autres revendications à droit de propriété, ou encore à contester la validité de une ou plusieurs quelconques de ces revendications; **la présente section a pour objet unique de protéger l'intégrité du système de distribution des logiciels libres**, système qui est mis en oeuvre par les pratiques liées aux licences publiques. De nombreuses personnes ont apporté une forte contribution à la gamme étendue des logiciels distribués par ce système, en comptant sur l'application systématique de ce système; **c'est à l'auteur/donateur de décider s'il a la volonté de distribuer le logiciel** par un quelconque autre système, **et un concessionnaire ne peut imposer ce choix**.

La présente section veut rendre parfaitement claire ce que l'on pense être une conséquence du reste de la présente licence.

9. Si la distribution et/ou l'utilisation du Programme est restreinte dans certains pays, sous l'effet de brevets ou d'interfaces présentant un droit d'auteur, le détenteur du droit d'auteur original, qui soumet le Programme aux dispositions de la présente licence, pourra ajouter une limitation expresse de distribution géographique excluant ces pays, de façon que la distribution ne soit autorisée que dans les pays ou parmi les pays qui ne sont pas ainsi exclus. Dans ce cas, la limitation fait partie intégrante de la présente licence, comme si elle était écrite dans le corps de la présente licence.

La Free Software Foundation peut, de temps à autre, publier des versions révisées et/ou nouvelles du General Public License. Ces nouvelles versions seront analogues, du point de vue de leur esprit, à la présente version, mais pourront en différer dans le détail, pour résoudre de nouveaux problèmes ou de nouvelles situations.

Chaque version reçoit un numéro de version qui lui est propre. Si le programme spécifie un numéro de version de la présente licence, qui s'applique à cette dernier et "à toute autre version ultérieure", le concessionnaire a le choix de respecter les clauses et dispositions de cette version, ou une quelconque version ultérieure publiée par la Free Software Foundation. Si le programme ne spécifie pas de numéro de version de la présente licence, le concessionnaire pourra choisir une version quelconque publiée à tout moment par la Free Software Foundation.

10. Si le concessionnaire souhaite incorporer des parties du programme dans d'autres programmes libres dont les conditions de distribution sont différentes, il devrait écrire à l'auteur pour demander son autorisation. Pour un logiciel soumis à droit d'auteur par la Free Software Foundation, il devra écrire à la Free Software Foundation; nous faisons quelquefois des exceptions à cette règle. Notre décision va être

guidée par le double objectif de protéger le statut libre de tous les dérivés de nos logiciels libres, et de favoriser le partage et la réutilisation des logiciels en général.

ABSENCE DE GARANTIE

11. COMME LA LICENCE DU PROGRAMME EST CONCEDEE A TITRE GRATUIT, IL N'Y AUCUNE GARANTIE S'APPLIQUANT AU PROGRAMME, DANS LA MESURE AUTORISEE PAR LA LOI EN VIGUEUR. SAUF MENTION CONTRAIRE ECRITE, LES DETENTEURS DU DROIT D'AUTEUR ET/OU LES AUTRES PARTIES METTENT LE PROGRAMME A DISPOSITON "EN L'ETAT", SANS AUCUNE GARANTIE DE QUELQUE NATURE QUE CE SOIT, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS LIMITATION, LES GARANTIES IMPLICITES DE COMMERCIALISATION ET DE L'APTITUDE A UN OBJET PARTICULIER. C'EST LE CONCESSIONNAIRE QUI PREND LA TOTALITE DU RISQUE QUANT A LA QUALITE ET AUX PERFORMANCES DU PROGRAMME. SI LE PROGRAMME SE REVELAIT DEFECTUEUX, C'EST LE CONCESSIONNAIRE QUI PRENDRAIT A SA CHARGE LE COUT DE L'ENSEMBLE DES OPERATIONS NECESSAIRES D'ENTRETIEN, REPARATION OU CORRECTION.

12. EN AUCUN CAS, SAUF SI LA LOI EN VIGUEUR L'EXIGE OU SI UNE CONVENTION ECRITE EXISTE A CE SUJET, AUCUN DETENTEUR DE DROITS D'AUTEUR, OU AUCUNE PARTIE AYANT LE POUVOIR DE MODIFIER ET/OU DE REDISTRIBUER LE PROGRAMME CONFORMEMENT AUX AUTORISATIONS CI-DESSUS, N'EST RESPONSABLE VIS-A-VIS DU CONCESSIONNAIRE POUR CE QUI EST DES DOMMAGES, Y COMPRIS TOUS DOMMAGES GENERAUX, SPECIAUX, ACCIDENTELS OU INDIRECTS, RESULTANT DE L'UTILISATION OU DU PROGRAMME OU DE L'IMPOSSIBILITE D'UTILISER LE PROGRAMME (Y COMPRIS, MAIS SANS LIMITATION, LA PERTE DE DONNEES, OU LE FAIT QUE DES DONNEES SONT RENDUES IMPRECISES, OU ENCORE LES PERTES EPROUVEES PAR LE CONCESSIONNAIRE OU PAR DES TIERS, OU ENCORE UN MANQUEMENT DU PROGRAMME A FONCTIONNER AVEC TOUS AUTRES PROGRAMMES), MEME SI CE DETENTEUR OU CETTE AUTRE PARTIE A ETE AVISE DE LA POSSIBILITE DE TELS DOMMAGES.

FIN DES CONDITIONS D'EXPLOITATION

Le Projet GNU

par Richard Stallman

publié à l'origine dans le livre "Open Sources" (traduction Sébastien Blondeel)

La première communauté qui partageait le logiciel

En 1971, quand j'ai commencé à travailler au laboratoire d'intelligence artificielle (IA) du MITN.d.T. : Institut de Technologie du Massachusetts, l'une des universités les plus prestigieuses des États-Unis d'Amérique., j'ai intégré **une communauté qui partageait le logiciel depuis de nombreuses années déjà**. **Le partage du logiciel** n'était pas limité à notre communauté ; c'**est une notion aussi ancienne que les premiers ordinateurs**, tout comme on partage des recettes depuis les débuts de la cuisine. Mais nous partagions davantage que la plupart.

Le laboratoire d'IA utilisait un système d'exploitation à temps partagé appelé ITS (Incompatible Timesharing System, ou système à temps partagé incompatible) que les hackers de l'équipe avaient écrit et mis au point en langage d'assemblage pour le Digital PDP-10, l'un des grands ordinateurs de l'époque. En tant que membre de cette communauté, hacker système de l'équipe du laboratoire d'IA, mon travail consistait à améliorer ce système.

Nous ne qualifiions pas nos productions de «logiciels libres», car ce terme n'existait pas encore ; c'est pourtant ce qu'elles étaient. Quand d'autres universitaires ou quand des ingénieurs souhaitaient utiliser et porter l'un de nos programmes, nous les laissions volontiers faire. Et quand on remarquait que quelqu'un utilisait un programme intéressant mais inconnu, on pouvait toujours en obtenir le code source, afin de le lire, le modifier, ou d'en réutiliser des parties dans le cadre d'un nouveau programme.

(1) L'utilisation du mot «hacker» dans le sens de «qui viole des systèmes de sécurité» est un amalgame instillé par les mass media. Nous autres hackers refusons de reconnaître ce sens, et continuons d'utiliser ce mot dans le sens «qui aime programmer et apprécie de le faire de manière astucieuse et intelligente.» (N.d.T. : en français, on peut utiliser le néologisme «bitouilleur» pour désigner l'état d'esprit de celui qui «touille des bits»).

L'effondrement de la communauté

La situation s'est modifiée de manière drastique au début des années 1980 quand la société Digital a mis fin à la série des PDP-10. Cette architecture, élégante et puissante dans les années 60, ne pouvait pas s'étendre naturellement aux plus grands espaces d'adressage qu'on trouvait dans les années 80. Cela rendait obsolètes la quasi-totalité des programmes constituant ITS.

La communauté des hackers du laboratoire d'IA s'était effondrée peu de temps auparavant. La plupart d'entre eux avaient été engagés par une nouvelle société, Symbolics, et ceux qui étaient restés ne parvenaient pas à maintenir la communauté (le livre Hackers, écrit par Steve Levy, décrit ces événements et dépeint clairement l'apogée de cette communauté). Quand le laboratoire a, en 1982, choisi d'acheter un nouveau PDP-10, ses administrateurs ont décidé de remplacer ITS par le système de temps partagé de la société Digital, qui n'était pas libre.

Les ordinateurs modernes d'alors, tels que le VAX ou le 68020, disposaient de leurs propres systèmes d'exploitation, mais aucun d'entre eux n'était un logiciel libre: **il fallait signer un accord de non divulgation pour en obtenir ne serait-ce que des copies exécutables.**

Cela signifiait que **la première étape de l'utilisation d'un ordinateur était de promettre de ne pas aider son prochain**. On **interdisait toute communauté coopérative**. La règle qu'édictaient ceux qui détenaient le monopole d'un logiciel propriétaire était «Qui partage avec son voisin est un pirate. **Qui souhaite la moindre modification doit nous supplier de la lui faire**».

L'idée que **le système social du logiciel propriétaire** - le système qui vous interdit de partager ou d'échanger le logiciel -**est antisocial, immoral, et qu'il est tout bonnement incorrect**, surprendra peut-être certains lecteurs. Mais comment qualifier autrement un système fondé sur la division et l'isolement des

utilisateurs ? Les lecteurs surpris par cette idée ont probablement pris le système social du logiciel propriétaire pour argent comptant, ou l'ont jugé en employant les termes suggérés par les entreprises vivant de logiciels propriétaires. **Les éditeurs de logiciels travaillent durement, et depuis longtemps, pour convaincre tout un chacun qu'il n'existe qu'un seul point de vue sur la question - le leur.**

Quand les éditeurs de logiciels parlent de «faire respecter» leurs «droits» ou de «couper court au piratage», le contenu réel de leur discours passe au second plan. Le véritable message se trouve entre les lignes, et il consiste en des hypothèses de travail qu'ils considèrent comme acquises ; nous sommes censés les accepter les yeux fermés. Passons-les donc en revue.

La première hypothèse est que **les sociétés éditrices de logiciels disposent d'un droit naturel, incontestable, à être propriétaire du logiciel et asseoir ainsi leur pouvoir sur tous ses utilisateurs** (si c'était là un droit naturel, on ne pourrait formuler aucune objection, **indépendamment du tort qu'il cause à tous**). Il est intéressant de remarquer que la constitution et la tradition juridique des États-Unis d'Amérique rejettent toutes deux cette idée; **le copyright n'est pas un droit naturel, mais un monopole artificiel, imposé par l'État, qui restreint le droit naturel qu'ont les utilisateurs de copier le logiciel.**

Une autre hypothèse sous-jacente est que seules importent les fonctionnalités du logiciel - et que **les utilisateurs d'ordinateurs n'ont pas leur mot à dire quant au modèle de société qu'ils souhaitent voir mettre en place.**

Une troisième hypothèse est qu'**on ne disposerait d'aucun logiciel utilisable** (ou qu'on ne disposerait jamais d'un logiciel qui s'acquitte de telle ou telle tâche en particulier) **si on ne garantissait pas à une société l'assise d'un pouvoir sur les utilisateurs du programme**. Cette idée était plausible, jusqu'à ce que **le mouvement du logiciel libre démontrât qu'on peut produire toutes sortes de logiciels utiles sans qu'il soit nécessaire de les barder de chaînes**.

Si l'on se refuse à accepter ces hypothèses, et si on examine ces questions en se fondant sur une moralité dictée par le bon sens, qui place les utilisateurs au premier plan, on parvient à des conclusions bien différentes. **Les utilisateurs des ordinateurs doivent être libres de modifier les programmes pour qu'ils répondent mieux à leurs besoins, et libres de partager le logiciel, car la société est fondée sur l'aide à autrui**.

La place me manque ici pour développer le raisonnement menant à cette conclusion, aussi renverrai-je le lecteur au document Pourquoi le logiciel ne devrait appartenir à personne.

Une profonde prise de décision

Avec la fin de ma communauté, il m'était impossible de continuer comme de par le passé. J'étais au lieu de cela confronté à une profonde prise de décision.

La solution de facilité était de rejoindre le monde du logiciel propriétaire, de signer des accords de non divulgation et promettre ainsi de ne pas aider mon ami hacker. J'aurais aussi été, très probablement, amené à développer du logiciel qui aurait été publié en fonction d'exigences de non divulgation, augmentant la pression qui en inciterait d'autres à trahir également leurs semblables.

J'aurais pu gagner ma vie de cette manière, et peut-être me serais-je même amusé à écrire du code. Mais je savais qu'**à la fin de ma carrière, je n'aurais à contempler que des années de construction de murs pour séparer les gens, et que j'aurais l'impression d'avoir employé ma vie à rendre le monde pire**.

J'avais déjà eu l'expérience douloureuse des accords de non divulgation, quand quelqu'un m'avait refusé, ainsi qu'au laboratoire d'IA du MIT, l'accès au code source du programme de contrôle de notre imprimante (l'absence de certaines fonctionnalités dans ce programme rendait l'utilisation de l'imprimante très frustrante). Aussi ne pouvais-je pas me dire que les accords de non divulgation étaient bénins. J'avais été très fâché du refus de cette personne de partager avec nous ; je ne pouvais pas, moi aussi, me comporter d'une telle manière à l'égard de mon prochain.

Une autre possibilité, radicale mais déplaisante, était d'abandonner l'informatique. De cette manière, mes capacités ne seraient pas employées à mauvais escient, mais elles n'en seraient pas moins gaspillées. Je ne me rendrais pas coupable de diviser et de restreindre les droits des utilisateurs d'ordinateurs, mais cela se

produirait malgré tout.

Alors, j'ai cherché une façon pour un programmeur de se rendre utile pour la bonne cause. Je me suis demandé si je ne pouvais pas écrire un ou plusieurs programmes qui permettraient de souder à nouveau une communauté.

La réponse était limpide: **le besoin le plus pressant était un système d'exploitation**. C'est le **logiciel le plus crucial** pour commencer à utiliser un ordinateur. Un système d'exploitation ouvre de nombreuses portes; sans système, l'ordinateur est inexploitable. Un système d'exploitation libre rendrait à nouveau possible une communauté de hackers, travaillant en mode coopératif - et tout un chacun serait invité à participer. Et tout un **chacun pourrait utiliser un ordinateur sans devoir adhérer à une conspiration visant à priver ses amis des logiciels qu'il utilise**.

En tant que développeur de système d'exploitation, j'avais les compétences requises. Aussi, bien que le succès ne me semblât pas garanti, j'ai pensé être le candidat de choix pour ce travail. **J'ai décidé de rendre le système compatible avec Unix de manière à le rendre portable**, et pour le rendre plus accessible aux utilisateurs d'Unix. J'ai opté pour le nom GNU, fidèle en cela à une tradition des hackers, car c'est un acronyme récursif qui signifie «GNU's Not Unix» (GNU N'est pas Unix).

Un système d'exploitation ne se limite pas à un noyau, qui suffit à peine à exécuter d'autres programmes. Dans les années 1970, tout système d'exploitation digne de ce nom disposait d'interpréteurs de commandes, d'assembleurs, de compilateurs, d'interpréteurs, de débogueurs, d'éditeurs de textes, de logiciels de courrier électronique, pour ne citer que quelques exemples. C'était le cas d'ITS, c'était le cas de Multics, c'était le cas de VMS, et c'était le cas d'Unix. Ce serait aussi le cas du système d'exploitation GNU.

Plus tard, j'ai entendu ces mots, attribués à Hillel:

If I am not for myself, who will be for me ?
If I am only for myself, what am I ?
If not now, when ?

N.d.T.: on peut rendre l'esprit de ce poème comme suit : Si je ne suis rien pour moi-même, qui sera pour moi ?
Si je suis tout pour moi-même, que suis-je ?
Si ce n'est pas aujourd'hui, alors quand ?

C'est dans cet état d'esprit que j'ai pris la décision de lancer **le projet GNU.**

(1) En tant qu'athée, je ne suis les pas d'aucun meneur religieux, mais j'admire parfois ce que l'un d'entre eux a dit.

Free comme liberté

N.d.T. : en anglais, le «libre» de «logiciel libre» se dit «free». Malheureusement, ce mot a une autre acception, indépendante et incorrecte ici, il signifie également «gratuit». Cette ambiguïté a causé énormément de tort au mouvement du logiciel libre., **le terme «free software» est mal compris - il n'a rien à voir avec le prix. Il parle de liberté**. Voici donc la définition d'un logiciel libre : un programme est un logiciel libre pour vous, utilisateur en particulier, si:

* Vous avez la **liberté de l'exécuter**, pour quelque motif que ce soit
* Vous avez **la liberté de modifier le programme** afin qu'il corresponde mieux à vos besoins (dans la pratique, pour que cette liberté prenne effet, il vous faut pouvoir accéder au code source, puisqu'opérer des modifications au sein d'un programme dont on ne dispose pas du code source est un exercice extrêmement difficile)
* Vous disposez de la **liberté d'en redistribuer des copies**, que ce soit gratuitement ou contre une somme d'argent
* Vous avez la **liberté de distribuer des versions modifiées** du programme, afin que la communauté puisse bénéficier de vos améliorations.

Puisque le mot «free» se réfère ici à la liberté, et non au prix, il n'est pas contradictoire de vendre des copies

de logiciels libres. En réalité, cette liberté est cruciale : les compilations de logiciels libres vendues sur CD-ROM sont importantes pour la communauté, et le produit de leur vente permet de lever des fonds pour le développement du logiciel libre. C'est pourquoi on ne peut pas qualifier de logiciel libre un logiciel qu'on n'a pas la liberté d'inclure dans de telles compilations.

Le mot «free» étant ambigu, on a longtemps cherché des solutions de remplacement, mais personne n'en a trouvé de convenable. La langue anglaise compte plus de mots et de nuances que toute autre langue, mais elle souffre de l'absence d'un mot simple, univoque, qui ait le sens de «free», comme liberté - «unfettered» (terme littéraire signifiant «sans entrave») étant le meilleur candidat, d'un point de vue sémantique, des mots comme «liberated» («libéré»), «freedom» («liberté»), et «open» («ouvert») présentant tous un sens incorrect ou un autre inconvénient.

Les logiciels et le système du projet GNU

C'est une gageure que de développer un système complet. Pour mener ce projet à bien, j'ai décidé d'adapter et de réutiliser les logiciels libres existants, quand cela était possible. J'ai par exemple décidé dès le début d'utiliser LaTex comme formateur de texte principal; quelques années plus tard, j'ai décidé d'utiliser le système de fenêtrage X plutôt que d'écrire un autre système de fenêtrage pour le projet GNU.

Cette décision a rendu le système GNU distinct de la réunion de tous les logiciels GNU. Le système GNU comprend des programmes qui ne sont pas des logiciels GNU, ce sont des programmes qui ont été développés par d'autres, dans le cadre d'autres projets, pour leurs buts propres, mais qu'on peut réutiliser, car ce sont des logiciels libres.

La genèse du projet

En janvier 1984, j'ai démissionné de mon poste au MIT et j'ai commencé à écrire les logiciels du projet GNU. Il était nécessaire que je quitte le MIT pour empêcher ce dernier de s'immiscer dans la distribution du projet GNU en tant que logiciel libre. Si j'étais resté dans l'équipe, le MIT aurait pu se déclarer le propriétaire de mon travail, et lui imposer ses propres conditions de distribution, voire en faire un paquetage de logiciels propriétaires. Je n'avais pas l'intention d'abattre autant de travail et de le voir rendu inutilisable pour ce à quoi il était destiné: **créer une nouvelle communauté qui partage le logiciel**.

Cependant, le professeur Winston, qui dirigeait alors le laboratoire d'IA du MIT, m'a gentiment invité à continuer à utiliser les équipements du laboratoire.

Les premiers pas

Peu de temps avant de me lancer dans le projet GNU, j'avais entendu parler du Free University Compiler KitN.d.T. : En anglais, le placement des mots ne permet pas de déterminer s'il s'agit d'un «kit compilateur libre de l'université» ou d'un «kit compilateur de l'université libre»., plus connu sous le nom de VUCK (en néerlandais, le mot «free» commence par un V). Ce compilateur avait été mis au point dans l'intention de traiter plusieurs langages, parmi lesquels C et Pascal, et de produire des binaires pour de nombreuses machines cibles. J'ai écrit à son auteur en lui demandant la permission d'utiliser ce compilateur dans le cadre du projet GNU.

Il répondit d'un ton railleur, en déclarant (en anglais) que l'université était «free». M. Stallman ne sait pas s'il voulait ici dire «libre» ou «gratuite»., mais pas le compilateur. J'ai alors décidé que le premier programme du projet GNU serait un compilateur traitant de plusieurs langages, sur plusieurs plates-formes.

En espérant m'épargner la peine d'écrire tout le compilateur moi-même, j'ai obtenu le code source du compilateur Pastel, qui était un compilateur pour plusieurs plates-formes, développé au laboratoire Lawrence Livermore. Il compilait, et c'était aussi le langage dans lequel il avait été écrit, une version étendue de Pascal, mise au point pour jouer le rôle de langage de programmation système. J'y ai ajouté une interface pour le C, et j'ai entrepris le portage de ce programme sur le Motorola 68000. Mais j'ai dû abandonner quand j'ai découvert que ce compilateur ne fonctionnait qu'avec plusieurs méga-octets d'espace de pile disponibles, alors que le système Unix du 68000 ne gérait que 64 Ko d'espace de pile.

C'est alors que j'ai compris que le compilateur Pastel avait été mis au point de telle manière qu'il analysait le fichier en entrée, en faisait un arbre syntaxique, convertissait cet arbre syntaxique en chaîne d'«instructions»,

et engendrait ensuite le fichier de sortie, sans jamais libérer le moindre espace mémoire occupé. J'ai alors compris qu'il me faudrait réécrire un nouveau compilateur en partant de zéro. Ce compilateur est maintenant disponible, il s'appelle GCC ; il n'utilise rien du compilateur Pastel, mais j'ai réussi à adapter et à réutiliser l'analyseur syntaxique que j'avais écrit pour le C. Mais tout cela ne s'est produit que quelques années plus tard; j'ai d'abord travaillé sur GNU Emacs.

GNU Emacs

J'ai commencé à travailler sur GNU Emacs en septembre 1984, et ce programme commençait à devenir utilisable début 1985. Cela m'a permis d'utiliser des systèmes Unix pour éditer mes fichiers ; vi et ed me laissant froid, j'avais jusqu'alors utilisé d'autres types de machines pour éditer mes fichiers.

C'est alors que j'ai reçu des requêtes de gens souhaitant utiliser GNU Emacs, ce qui a soulevé **le problème de sa distribution**. Je l'avais bien sûr proposé sur le serveur ftp de l'ordinateur du MIT que j'utilisais (cet ordinateur, prep.ai.mit.edu, a ainsi été promu au rang de site de distribution par ftp principal du projet GNU; quelques années plus tard, à la fin de son exploitation, nous avons transféré ce nom sur notre nouveau serveur ftp). Mais à l'époque, une proportion importante des personnes intéressées n'avaient pas d'accès à l'Internet et ne pouvaient pas obtenir une copie du programme par ftp. La question se posait en ces termes : que devais-je leur dire ?

J'aurais pu leur dire : «Trouvez un ami qui dispose d'un accès au réseau et qui vous fera une copie.» J'aurais pu également procéder comme j'avais procédé avec la version originale d'Emacs, sur PDP-10, et leur dire: «Envoyez-moi une bande et une enveloppe timbrée auto-adressée, et je vous les renverrai avec Emacs.» Mais j'étais sans emploi, et je cherchais des moyens de gagner de l'argent grâce au logiciel libre. C'est pourquoi j'ai annoncé que j'enverrais une bande à quiconque en désirait une, en échange d'une contribution de 150 USD. De cette manière, je mettais en place une **entreprise autour du marché de la distribution du logiciel libre**, entreprise **précurseur des sociétés** qu'on trouve aujourd'hui et **qui distribuent des systèmes GNU entiers fondés sur Linux**.

Un programme est-il libre pour chacun de ses utilisateurs ?

Si un programme est un logiciel libre au moment où il quitte les mains de son auteur, cela ne signifie pas nécessairement qu'il sera un logiciel libre pour quiconque en possédera une copie. **Le logiciel relevant du domaine public**, par exemple (**qui ne tombe sous le coup d'aucun copyright**), **est du logiciel libre**; mais t**out un chacun peut en produire une version propriétaire** modifiée. De façon comparable, de nombreux programmes libres sont couverts par des copyrights mais distribués sous des licences permissives qui autorisent la création de versions modifiées et propriétaires.

L'exemple le plus frappant de ce problème est le système de fenêtrage X. Développé au MIT, et distribué sous forme de logiciel libre sous une licence permissive, il a rapidement été adopté par divers constructeurs. Ils ont ajouté X à leurs systèmes Unix propriétaires, sous forme binaire uniquement, en le frappant du même accord de non divulgation. Ces exemplaires de X n'étaient en rien du logiciel plus libre que le reste d'Unix.

Les développeurs du système de fenêtrage X ne voyaient là nul problème - ils s'attendaient à cela et souhaitaient un tel résultat. Leur but n'était pas la liberté, mais la simple **«réussite», définie comme le fait d'«avoir beaucoup d'utilisateurs.» Peu leur importait la liberté de leurs utilisateurs, seul leur nombre revêtait de l'importance à leurs yeux.**

Cela a conduit à une situation paradoxale, où deux différentes façons d'évaluer la liberté répondaient de manières différente à la question «Ce programme est-il libre ?» Qui fondait son jugement sur la liberté fournie par les conditions de distribution de la distribution du MIT, concluait que X était un logiciel libre. Mais qui mesurait la liberté de l'utilisateur type de X, devait conclure que X était un logiciel propriétaire. La plupart des utilisateurs de X exécutaient des versions propriétaires fournies avec des systèmes Unix, et non la version libre.

Le copyleft et la GPL de GNU

Le but du projet GNU était de rendre les utilisateurs libres, pas de se contenter d'être populaire. Nous avions besoins de conditions de distribution qui empêcheraient de transformer du logiciel GNU en logiciel propriétaire. Nous avons utilisé la méthode du copyleft (1), ou «gauche d'auteur».

Le gauche d'auteur utilise les lois du droit d'auteur, en les retournant pour leur faire servir le but opposé de ce pour quoi elles ont été conçues: ce n'est pas une manière de privatiser du logiciel, mais une manière de le laisser «libre».

L'idée centrale du gauche d'auteur est de donner à quiconque la permission d'exécuter le programme, de le copier, de le modifier, et d'en distribuer des versions modifiées - mais pas la permission d'ajouter des restrictions de son cru. C'est ainsi que **les libertés cruciales qui définissent le «logiciel libre» sont garanties pour quiconque en possède une copie; elles deviennent des droits inaliénables.**

Pour que le gauche d'auteur soit efficace, **il faut que les versions modifiées demeurent libres**, afin de s'assurer que **toute oeuvre dérivée de notre travail reste disponible à la communauté en cas de publication**. Quand des programmeurs professionnels se portent volontaires pour améliorer le logiciel GNU, c'est le gauche d'auteur qui empêche leurs employeurs de dire : «Vous ne pouvez pas partager ces modifications, car nous allons les utiliser dans le cadre de notre version propriétaire du programme.»

Il est essentiel d'imposer que les modifications restent libres si on souhaite garantir la liberté de tout utilisateur du programme. Les sociétés qui ont privatisé le système de fenêtrage X faisaient en général quelques modifications pour le porter sur leurs systèmes et sur leur matériel. Ces modifications étaient ténues si on les comparait à X dans son ensemble, mais elles n'en étaient pas pour autant faciles. Si le fait de procéder à des modifications pouvait servir de prétexte à ôter leur liberté aux utilisateurs, il serait facile pour quiconque de s'en servir à son avantage.

Le **problème de la réunion d'un programme libre avec du code non libre** est similaire. Une telle combinaison serait indubitablement non libre ; les libertés absentes de la partie non libre du programme ne se trouveraient pas non plus dans l'ensemble résultat de cette compilation. Autoriser de telles pratiques ouvrirait une voie d'eau suffisante pour couler le navire. C'est pourquoi il est crucial pour le gauche d'auteur d'exiger qu'un programme couvert par le gauche d'auteur ne puisse pas être inclus dans une version plus grande sans que cette dernière ne soit également couverte par le gauche d'auteur.

L'implémentation spécifique du gauche d'auteur que nous avons utilisée pour la plupart des logiciels GNU fut la GNU General Public License (licence publique générale de GNU), ou GNU GPL en abrégé. Nous disposons d'autres types de gauche d'auteur pour des circonstances particulières. Les manuels du projet GNU sont eux aussi couverts par le gauche d'auteur, mais en utilisent une version très simplifiée, car il n'est pas nécessaire de faire appel à toute la complexité de la GNU GPL dans le cadre de manuels.

(1) En 1984 ou 1985, Don Hopkins (dont l'imagination était sans bornes) m'a envoyé une lettre. Il avait écrit sur l'enveloppe plusieurs phrases amusantes, et notamment celle-ci : «Copyleft - all rights reversed.» (N.d.T. : «couvert par le gauche d'auteur, tous droits renversés.»). J'ai utilisé le mot «copyleft» pour donner un nom au concept de distribution que je développais alors.

La Free Software Foundation, ou Fondation pour le Logiciel Libre

Emacs attirant de plus en plus l'attention, le projet GNU comptait un nombre croissant de participants, et nous avons décidé qu'il était temps de repartir à la chasse aux fonds. **En 1985**, nous avons donc **créé la fondation du logiciel libre (FSF)**, une association à but non lucratif, exemptée d'impôts, pour le développement de logiciel libre.

La FSF a récupéré le marché de la distribution de logiciel libre sur bandes, auxquelles elle ajouta ensuite d'autres logiciels libres (GNU ou non), et par la vente de manuels libres. La FSF accepte les dons, mais la plus grande partie de ses recettes est toujours provenue des ventes - de copies de logiciel libre ou d'autres services associés. De nos jours, elle vend des CD-ROM de code source, des CD-ROM de binaires, des manuels de qualité (tout cela, en autorisant la redistribution et les modifications), et des distributions Deluxe (dans lesquelles nous construisons tous les logiciels pour la plate-forme de votre choix).

Les employés de la fondation du logiciel libre ont écrit et maintenu un grand nombre de paquetages logiciels du projet GNU, en particulier la bibliothèque du langage C et l'interpréteur de commandes. La bibliothèque du langage C est ce qu'utilise tout programme fonctionnant sur un système GNU/Linux pour communiquer avec Linux. Elle a été développée par Roland McGrath, membre de l'équipe de la fondation du logiciel libre. L'interpréteur de commandes employé sur la plupart des systèmes GNU/Linux est BASH, le Bourne-Again

Shell, qui a été développé par Brian Fox, employé de la FSF.

Nous avons financé le développement de ces programmes car le projet GNU ne se limitait pas aux outils ou à un environnement de développement. Notre but était la mise en place d'un système d'exploitation complet, et de tels programmes étaient nécessaires pour l'atteindre.

(1) «Bourne-Again Shell» est un clin d'oeil au nom «Bourne Shell», qui était l'interpréteur de commandes habituel sur Unix (N.d.T. : le mot anglais bash a le sens de «coup, choc» et la signification de cet acronyme est double ; c'est à la fois une nouvelle version de l'interpréteur de commandes Bourne, et et une allusion aux chrétiens qui se sont sentis renaître dans cette religion, et qu'aux États-Unis d'Amérique on qualifie de born again Christians)

Assistance technique au logiciel libre

La philosophie du logiciel libre rejette une pratique spécifique, **très répandue dans l'industrie du logiciel, mais elle ne s'oppose pas au monde des affaires. Quand des entreprises respectent la liberté des utilisateurs, nous leur souhaitons de réussir.**

La vente de copies d'Emacs est une forme d'affaires fondées sur du logiciel libre. Quand la FSF a récupéré ce marché, j'ai dû chercher une autre solution pour gagner ma vie. Je l'ai trouvée sous la forme de **vente de services associés au logiciel libre** que j'avais développé. Cela consistait à enseigner des thèmes tels que la programmation de GNU Emacs et la personnalisation de GCC, et à développer du logiciel, principalement en portant GCC sur de nouvelles plates-formes.

De nos jours, chacune de ces **activités lucratives fondées sur le logiciel libre est proposée par de nombreuses sociétés**. Certaines distribuent des compilations de logiciel libre sur CD-ROM; d'autres **vendent de l'assistance technique** en répondant à des questions d'utilisateurs, en **corrigeant des bogues**, et en **insérant de nouvelles fonctionnalités** majeures. On commence même à observer des **sociétés de logiciel libre fondées sur la mise sur le marché de nouveaux logiciels libres**.

Prenez garde, toutefois - certaines des sociétés qui s'associent à la dénomination «open source» N.d.T.: littéralement, «[logiciel dont le] code source est ouvert». C'est une périphrase lourde et inélégante en français, mais qui résout en anglais l'ambiguïté discutée plus haut, bien que l'auteur rejette cette solution, pour des raisons expliquées à la fin de cet article. Il s'agit ici de sociétés qui font peu de cas du logiciel libre et choisissent un slogan calculé pour s'attirer les faveurs du public. fondent en réalité leur activité sur du logiciel propriétaire, qui interagit avec du logiciel libre. Ce ne sont pas des sociétés de logiciel libre, ce sont des sociétés de logiciel propriétaire dont les produits détournent les utilisateurs de leur liberté. Elles appellent cela de la «valeur ajoutée», ce qui reflète quelles valeurs elles souhaitent nous voir adopter: préférer la facilité à la liberté. Si nous faisons passer la liberté au premier plan, il nous faut leur donner le nom de produits à « liberté soustraite ».

Objectifs techniques

L'objectif principal du projet GNU était le logiciel libre. Même si GNU ne jouissait d'aucun avantage technique sur Unix, il disposerait d'un **avantage social, en autorisant les utilisateurs à coopérer**, et d'un **avantage éthique, en respectant la liberté de l'utilisateur**.

Mais il était naturel d'appliquer à ce travail les standards bien connus du développement logiciel de qualité - en utilisant par exemple des structures de données allouées dynamiquement pour éviter de mettre en place des limites fixées arbitrairement, et en gérant tous les caractères possibles encodables sur 8 bits, partout où cela avait un sens.

De plus, nous rejetions l'accent mis par Unix sur les petites quantités de mémoire, en décidant de ne pas nous occuper des architectures 16 bits (il était clair que les architectures 32 bits seraient la norme au moment de la finalisation du système GNU), et en ne faisant aucun effort pour réduire la consommation mémoire en deçà d'un méga-octet. Dans les programmes pour lesquels il n'était pas crucial de manipuler des fichiers de tailles importantes, nous encouragions les programmeurs à lire le fichier en entrée, d'une traite, en mémoire, et d'analyser ensuite son contenu sans plus se préoccuper des entrées/sorties.

Ces décisions ont rendu de nombreux programmes du projet GNU supérieurs à leurs équivalents sous Unix

en termes de fiabilité et de vitesse d'exécution.

Les ordinateurs offerts

La réputation du projet GNU croissant, on nous offrait des machines sous Unix pour nous aider à le mener à bien. Elles nous furent bien utiles, car le meilleur moyen de développer les composants de GNU était de travailler sur un système Unix, dont on remplaçait les composants un par un. Mais cela a posé un problème éthique: était-il correct ou non, pour nous, de posséder des copies d'Unix ?

Unix était (et demeure) du logiciel propriétaire, et la philosophie du projet GNU nous demandait de ne pas utiliser de logiciels propriétaire. Mais, en appliquant le même raisonnement que celui qui conclut qu'il est légitime de faire usage de violence en situation de légitime défense, j'ai conclu qu'il était légitime d'utiliser un paquetage propriétaire quand cela était crucial pour développer une solution de remplacement libre, qui en aiderait d'autres à se passer de ce même paquetage propriétaire.

Mais ce mal avait beau être justifiable, il n'en restait pas moins un mal. De nos jours, nous ne possédons plus aucune copie d'Unix, car nous les avons toutes remplacées par des systèmes d'exploitation libres. Quand nous ne parvenions pas à substituer au système d'exploitation d'une machine un système libre, nous remplacions la machine.

La GNU Task List, ou liste des tâches du projet GNU

Le projet GNU suivant son cours, on trouvait ou développait un nombre croissant de composants du système, et il est finalement devenu utile de faire la liste des parties manquantes. Nous l'avons utilisée pour recruter des développeurs afin d'écrire ces dernières. Cette liste a pris le nom de GNU task list. En plus des composants manquants d'Unix, nous y avons listé plusieurs autres projets utiles, de logiciel et de documentation, que nous jugions nécessaires au sein d'un système réellement complet.

De nos jours, on ne trouve presque plus aucun composant d'Unix dans la liste des tâches du projet GNU - ces travaux tous ont été menés à bien, si on néglige certains composants non essentiels. Mais la liste est pleine de projets qu'on pourrait qualifier d'«applications». Tout programme qui fait envie à une classe non restreinte d'utilisateurs constituerait un ajout utile à un système d'exploitation.

On trouve même des jeux dans la liste des tâches - et c'est le cas depuis le commencement. Unix proposait des jeux, ce devait naturellement être également le cas de GNU. Mais il n'était pas nécessaire d'être compatible en matière de jeux, aussi n'avons-nous pas suivi la liste des jeux d'Unix. Nous avons plutôt listé un spectre de divers types de jeux qui plairont vraisemblablement aux utilisateurs.

La GNU Library GPL, ou licence publique générale de GNU pour les bibliothèques

La bibliothèque du langage C du projet GNU fait appel à un gauche d'auteur particulier, appelé la GNU Library General Public License (licence publique générale de GNU pour les bibliothèques, ou GNU LGPL), qui autorise la liaison de logiciel propriétaire avec la bibliothèque. Pourquoi une telle exception ?

Ce n'est pas une question de principe ; aucun principe ne dicte que les logiciels propriétaires ont le droit de contenir notre code (pourquoi contribuer à un projet qui affirme refuser de partager avec nous ?). L'utilisation de la LGPL dans le cadre de la bibliothèque du langage C, ou de toute autre bibliothèque, est un choix stratégique.

La bibliothèque du langage C joue un rôle générique; tout système propriétaire, tout compilateur, dispose d'une bibliothèque du langage C. C'est pourquoi limiter l'utilisation de notre bibliothèque du langage C au logiciel libre n'aurait donné aucun avantage au logiciel libre - cela n'aurait eu pour effet que de décourager l'utilisation de notre bibliothèque.

Il existe une exception à cette règle: sur un système GNU (et GNU/Linux est l'un de ces systèmes), la bibliothèque du langage C de GNU est la seule disponible. Aussi, ses conditions de distribution déterminent s'il est possible de compiler un programme propriétaire sur le système GNU. **Il n'existe aucune raison éthique d'autoriser des applications propriétaires sur le système GNU, mais d'un point de vue stratégique,** il semble que **les interdire découragerait plus l'utilisation d'un système GNU que cela n'encouragerait le développement d'applications libres.**

C'est pourquoi l'utilisation de la GPL pour les bibliothèques (ou LGPL) est une bonne stratégie dans le cadre de la bibliothèque du langage C. En ce qui concerne les autres bibliothèques, il faut prendre la décision stratégique au cas par cas. Quand une bibliothèque remplit une tâche particulière qui peut faciliter l'écriture de certains types de programmes, la distribuer sous les conditions de la GPL, en limitant son utilisation aux programmes libres, est une manière d'aider les développeurs de logiciels libres et de leur accorder un avantage à l'encontre du logiciel propriétaire.

Considérons GNU Readline, une bibliothèque développée dans le but de fournir une édition de ligne de commande pour l'interpréteur de commandes BASH. Cette bibliothèque est distribuée sous la licence publique générale ordinaire de GNU, et non pas sous la LGPL. Cela a probablement pour effet de réduire l'utilisation de la bibliothèque Readline, mais cela n'induit aucune perte en ce qui nous concerne. Pendant ce temps, on compte au moins une application utile qui a été libérée, uniquement dans le but de pouvoir utiliser la bibliothèque Readline, et c'est là un gain réel pour la communauté.

Les développeurs de logiciel propriétaire jouissent des avantages que leur confère l'argent; les développeurs de logiciel libre doivent compenser cela en s'épaulant les uns les autres. J'espère qu'un jour nous disposerons de toute une collection de bibliothèques couvertes par la GPL, et pour lesquelles il n'existera pas d'équivalent dans le monde du logiciel propriétaire. Nous disposerons ainsi de modules utiles, utilisables en tant que blocs de construction de nouveaux logiciels libres, et apportant un avantage considérable à la continuation du développement du logiciel libre.

Gratter là où ça démange

Eric Raymond affirme que «Tout bon logiciel commence par gratter un développeur là où ça le démange.». Cela se produit peut-être, parfois, mais **de nombreux composants essentiels du logiciel GNU ont été développés dans le but de disposer d'un système d'exploitation libre complet. Ils ont été inspirés par une vision et un projet à long terme, pas par un coup de tête.**

Nous avons par exemple développé la bibliothèque du langage C de GNU car un système de type Unix a besoin d'une bibliothèque du langage C, nous avons développé le Bourne-Again Shell (BASH) car un système de type Unix a besoin d'un interpréteur de commandes, et nous avons développé GNU tar car un système de type Unix a besoin d'un programme d'archivage. Il en va de même pour les programmes que j'ai développé, à savoir le compilateur C de GNU, GNU Emacs, GDB, et GNU Make.

Certains programmes du projet GNU ont été développés pour répondre aux menaces qui pesaient sur notre liberté. C'est ainsi que nous avons développé gzip en remplacement du programme Compress, que la communauté avait perdu suite aux brevets logiciels déposés sur LZW. Nous avons trouvé des gens pour développer LessTif, et plus récemment nous avons démarré les projets GNOME et Harmony, en réponse aux problème posés par certaines bibliothèques propriétaires (lire ci-après). Nous sommes en train de développer le GNU Privacy Guard (le gardien de l'intimité de GNU, ou GPG) pour remplacer un logiciel de chiffrement populaire mais pas libre, car les utilisateurs ne devraient pas devoir choisir entre la préservation de leur intimité et la préservation de leur liberté.

Bien sûr, les gens qui écrivent ces programmes se sont intéressés à ce travail, et de nombreux contributeurs ont ajouté de nouvelles fonctionnalités car elles comblaient leurs besoins ou les intéressaient. Mais ce n'est pas là la raison première de ces programmes.

Des développements inattendus

Au commencement du projet GNU, j'ai imaginé que nous développerions le système GNU dans sa globalité avant de le publier. Les choses se sont passées différemment.

Puisque chaque composant du système GNU était implémenté sur un système Unix, chaque composant pouvait fonctionner sur des systèmes Unix, bien avant que le système GNU ne soit disponible dans sa globalité. Certains de ces programmes sont devenus populaires, et leurs utilisateurs ont commencé à travailler sur des extensions et des ports - vers les diverses versions d'Unix, incompatibles entre elles, et parfois, sur d'autres systèmes encore

Ce processus a rendu ces programmes bien plus complets, et a drainé des fonds et des participants vers le

projet GNU. Mais il a probablement eu également pour effet de retarder de plusieurs années la mise au point d'un système en état de fonctionnement, puisque les développeurs du projet GNU passaient leur temps à s'occuper de ces ports et à proposer des nouvelles fonctionnalités aux composants existants, plutôt que de continuer à développer peu à peu les composants manquants.

Le GNU Hurd

En 1990, le système GNU était presque terminé ; le seul composant principal qui manquait encore à l'appel était le noyau. Nous avions décidé d'implémenter le noyau sous la forme d'une série de processus serveurs qui fonctionneraient au-dessus de Mach. Mach est un micro-noyau développé à l'université Carnegie-Mellon puis à l'université de l'Utah; le GNU Hurd est une série de serveurs (ou une «horde de gnous») qui fonctionnent au-dessus de Mach, et remplissent les diverses fonctions d'un noyau Unix. Le développement a été retardé car nous attendions que Mach soit publié sous forme de logiciel libre, comme cela avait été promis.

L'une des raisons qui ont dicté ce choix était d'éviter ce qui semblait être la partie la plus difficile du travail: déboguer un programme de noyau sans disposer pour cela d'un débogueur au niveau du code source. Ce travail avait déjà été fait, dans Mach, et nous pensions déboguer les serveurs du Hurd en tant que programmes utilisateur, à l'aide de GDB. Mais cela prit beaucoup de temps, et les serveurs à plusieurs processus légers, qui s'envoyaient des messages les uns aux autres, se sont révélés très difficiles à déboguer. La consolidation du Hurd s'est étalée sur plusieurs années.

Alix

À l'origine, le noyau du système GNU n'était pas censé s'appeler Hurd. Son premier nom était Alix - du nom de celle qui à l'époque était l'objet de ma flamme. Administratrice de systèmes Unix, elle avait fait remarquer que son prénom ressemblait aux noms typiques des versions de systèmes Unix; elle s'en était ouverte auprès d'amis en plaisantant : «Il faudrait baptiser un noyau de mon nom.» Je suis resté coi, mais ai décidé de lui faire la surprise d'appeler Alix le noyau du système GNU.

Mais les choses ont changé. Michael Bushnell (maintenant, il s'agit de Thomas), le développeur principal du noyau, préférait le nom Hurd, et a confiné le nom Alix à une certaine partie du noyau - la partie qui se chargeait d'intercepter les appels système et de les gérer en envoyant des messages aux serveurs du Hurd.

Finalement, Alix et moi mîmes fin à notre relation, et elle a changé de nom; de manière indépendante, le concept du Hurd avait évolué de telle sorte que ce serait la bibliothèque du langage C qui enverrait directement des messages aux serveurs, ce qui a fait disparaître le composant Alix du projet.

Mais avant que ces choses ne se produisissent, un de ses amis avait remarqué le nom Alix dans le code source du Hurd, et s'en était ouvert auprès d'elle. Finalement, ce nom avait rempli son office.

Linux et GNU/Linux

Le GNU Hurd n'est pas encore utilisable de manière intensive. Heureusement, on dispose d'un autre noyau. **En 1991, Linus Torvalds a développé un noyau compatible avec Unix et lui a donné le nom de Linux. Aux alentours de 1992, la jonction de Linux et du système GNU, qui était presque complet, a fourni un système d'exploitation libre et complet** (ce travail de jonction était lui-même, bien sûr, considérable). **C'est grâce à Linux qu'on peut désormais employer une version du système GNU.**

On appelle cette version du système «GNU/Linux» pour signaler qu'il est composé du système GNU et du noyau Linux.

Les défis à venir

Nous avons fait la preuve de notre capacité à développer un large spectre de logiciel libre. Cela ne signifie pas que nous sommes invincibles et que rien ne peut nous arrêter. Certains défis rendent incertain l'avenir du logiciel libre; et il faudra des efforts et une endurance soutenus pour les relever, pendant parfois plusieurs années. **Il faudra montrer le genre de détermination dont les gens font preuve quand ils accordent de la valeur à leur liberté et qu'ils ne laisseront personne la leur voler.**

Les quatres sections suivantes discutent de ces défis.

Le matériel secret

Les fabricants de matériel tendent de plus en plus à garder leurs spécifications secrètes. Cela rend plus difficile l'écriture de pilotes de périphériques libres afin de permettre à Linux et au projet XFree86 de reconnaître de nouveaux matériels. Nous disposons aujourd'hui de systèmes entièrement libres, mais cela pourrait ne plus être le cas dans l'avenir, si nous ne pouvons plus proposer des pilotes pour les ordinateurs de demain.

On peut résoudre ce problème de deux manières. Les programmeurs peuvent analyser l'ensemble afin de deviner comment prendre en compte le matériel. Les autres peuvent choisir le matériel qui est reconnu par du logiciel libre; **plus nous serons nombreux, plus la politique de garder les spécifications secrètes sera vouée à l'échec.**

L'ingénierie à l'envers est un travail conséquent ; disposerons-nous de programmeurs suffisamment déterminés pour le prendre en main ? Oui - si nous avons construit un sentiment puissant selon lequel **le logiciel libre est une question de principe**, et que **les pilotes non libres sont inacceptables**. Et serons-nous nombreux à dépenser un peu plus d'argent, ou à passer un peu de temps, pour que nous puissions utiliser des pilotes libres ? Oui - si la détermination afférente à la liberté est largement répandue.

Les bibliothèques non libres

Une bibliothèque non libre qui fonctionne sur des systèmes d'exploitation libres se comporte comme un piège vis-à-vis des développeurs de logiciel libre. Les fonctionnalités attrayantes de cette bibliothèque sont l'appât; si vous utilisez la bibliothèque, vous tombez dans le piège, car votre programme ne peut pas être utilisé de manière utile au sein d'un système d'exploitation libre (pour être strict, on pourrait y inclure le programme, mais on ne pourrait pas l'exécuter en l'absence de la bibliothèque incriminée). Pire encore, si un programme qui utilise une bibliothèque propriétaire devient populaire, il peut attirer d'autres programmeurs peu soupçonneux dans le piège.

Ce problème s'est posé pour la première fois avec la boîte à outils Motif, dans les années 80. Même s'il n'existait pas encore de systèmes d'exploitation libres, il était limpide que Motif leur causerait des problèmes, plus tard. Le projet GNU a réagi de deux manières : en demandant aux projets de logiciel libre de rendre l'utilisation de Motif facultative en privilégiant les gadgets de la boîte à outils X, libre, et en recherchant un volontaire pour écrire une solution de remplacement libre à Motif. Ce travail prit de nombreuses années ; LessTif, développé par les Hungry Programmers (les «Programmeurs affamés»), n'est devenu suffisamment étendu pour faire fonctionner la plupart des applications utilisant Motif qu'en 1997.

De 1996 à 1998, une compilation conséquente de logiciel libre, le bureau KDE, a fait usage d'une autre bibliothèque non libre de boîte à outils pour l'interface graphique utilisateur, appelée Qt. Les systèmes GNU/Linux libres ne pouvaient pas utiliser KDE, car nous ne pouvions pas utiliser la bibliothèque.

Les systèmes GNU/Linux libres ne pouvaient pas utiliser KDE, car nous ne pouvions pas utiliser la bibliothèque. Cependant, certains distributeurs commerciaux de systèmes GNU/Linux n'ont pas été assez stricts pour coller au logiciel libre et ont ajouté KDE dans leurs systèmes - produisant un système disposant d'un plus grand nombre de fonctionnalités, mais souffrant d'une liberté réduite. Le groupe KDE encourageait activement un plus grand nombre de programmeurs à utiliser la bibliothèque Qt, et des millions de «nouveaux utilisateurs de Linux» n'ont jamais eu connaissance du fait que tout ceci posait un problème. La situation était sinistre.

La communauté du logiciel libre a répondu à ce problème de deux manières : GNOME et Harmony.

GNOME, le GNU Network Object Model Environment (environnement de GNU de modèle d'objets pour le réseau), est le projet de bureau de GNU. Démarré en 1997 par Miguel de Icaza, et développé avec l'aide de la société Red Hat Software, GNOME avait pour but de fournir des fonctionnalités de bureau similaires, en utilisant exclusivement du logiciel libre. Il jouit aussi d'avantages techniques, comme le fait de collaborer avec toute une variété de langages, et de ne pas de se limiter au C++. Mais son objectif principal est la liberté : ne pas imposer l'utilisation du moindre logiciel non libre.

Harmony est une bibliothèque compatible de remplacement, conçue pour permettre l'utilisation des logiciels de KDE sans faire appel à Qt.

En novembre 1998, les développeurs de Qt ont annoncé une modification de leur licence qui, quand elle sera effective, fera de Qt un logiciel libre. On ne peut pas en être sûr, mais je pense que cette décision est en partie imputable à la réponse ferme qu'a faite la communauté au problème que Qt posait quand il n'était pas libre (la nouvelle licence n'est pas pratique ni équitable, aussi demeure-t-il préférable d'éviter d'utiliser Qt).

[Note ultérieure: en septembre 2000, Qt fut distribuée sous la GPL de GNU, ce qui résolvait essentiellement ce problème.]

Comment répondrons-nous à la prochaine bibliothèque non libre mais alléchante ? La communauté comprendra-t-elle dans son entier la nécessité de ne pas tomber dans le piège ? Ou serons-nous nombreux à préférer la facilité à la liberté, et à produire un autre problème majeur ? Notre avenir dépend de notre philosophie.

Les brevets sur les logiciels

La pire menace provient des brevets sur les logiciels, susceptibles de placer des algorithmes et des fonctionnalités hors de portée des logiciels libres pendant une période qui peut atteindre vingt ans. Les brevets sur l'algorithme de compression LZW ont été déposés en 1983, et nous ne pouvons toujours pas diffuser des logiciels libres qui produisent des images au format GIF correctement compressées. En 1998, la menace d'une poursuite pour cause de violation de brevets a mis fin à la distribution d'un programme libre qui produisait des données sonores compressées au format MP3.

Il existe plusieurs manières de répondre au problème des brevets: on peut rechercher des preuves qui invalident un brevet, et on peut rechercher d'autres solutions pour remplir une tâche. Mais chacune de ces méthodes ne fonctionne que dans certains cas; quand les deux échouent, il se peut qu'un brevet empêche le logiciel libre de disposer de fonctionnalités souhaitées par les utilisateurs. Que ferons-nous dans ce genre de situation ?

Ceux d'entre nous qui prêtent de la valeur au logiciel libre par amour de la liberté continueront à utiliser du logiciel libre dans tous les cas. On pourra travailler sans utiliser de fonctionnalités protégées par des brevets. Mais ceux d'entre nous qui prêtent de la valeur au logiciel libre car ils s'attendent à trouver là des logiciels techniquement supérieurs sont susceptibles de critiquer l'idée même du logiciel libre quand un brevet l'empêchera de progresser plus avant. Ainsi, même s'il est utile de discuter de l'efficacité, dans la pratique, du modèle de développement de type «cathédrale», et de la fiabilité et de la puissance de certains logiciels libres, il ne faut pas s'en tenir là. Il nous faut parler de liberté et de principes.

La documentation libre

Il ne faut pas chercher les lacunes les plus graves de nos systèmes d'exploitations libres dans le logiciel - c'est l'absence de manuels libres corrects qu'on puisse inclure dans nos systèmes qui se fait le plus cruellement sentir. La documentation est essentielle dans tout paquetage logiciel; quand un paquetage logiciel important ne dispose pas d'un bon manuel libre, il s'agit d'un manque crucial. On en compte de nombreux aujourd'hui.

La documentation libre, tout comme le logiciel libre, est une question de liberté, pas de prix N.d.T.: ici encore, les anglais sont victimes de l'absence de mot adéquat pour signifier «libre». La raison d'être d'un manuel libre est très proche de celle d'un logiciel libre: il s'agit d'offrir certaines libertés à tous les utilisateurs. Il faut autoriser la redistribution (y compris la vente commerciale), en ligne et sur papier, de telle sorte que le manuel puisse accompagner toute copie du programme.

Il est également crucial d'autoriser les modifications. En règle générale, je ne pense pas qu'il soit essentiel d'autoriser tout un chacun à modifier toutes sortes d'articles et de livres. Je ne pense pas, par exemple, que vous ou moi soyons tenus de donner la permission de modifier des textes comme le présent article, qui expose nos actions et nos idées.

Mais il existe une raison particulière, pour laquelle il est crucial de disposer de la liberté de modifier la documentation afférente au logiciel libre. Quand on jouit de son droit de modifier le logiciel, et d'ajouter des

fonctionnalités ou de modifier les fonctionnalités présentes, le programmeur consciencieux mettra immédiatement à jour le manuel - afin de fournir une documentation précise et utilisable aux côtés du programme modifié. Un manuel qui n'autorise pas les programmeurs à être consciencieux et à terminer leur travail, ne remplit pas les besoins de notre communauté.

Il est acceptable d'apposer certaines limites sur la manière dont les modifications sont faites. Il est par exemple envisageable d'exiger de préserver la notice de copyright de l'auteur original, les conditions de distribution, ou la liste des auteurs. D'exiger que les versions modifiées contiennent une notice qui stipule qu'elles ont été modifiées, et même d'interdire de modifier ou d'ôter des sections entières, pourvu que ces sections ne traitent pas de considérations techniques, ne pose pas non plus de problèmes, car cela n'interdit pas au programmeur consciencieux d'adapter le manuel afin qu'il corresponde au programme modifié par ses soins. En d'autres termes, cela n'empêche la communauté du logiciel libre d'utiliser pleinement le manuel.

En revanche, il faut autoriser la modification des portions *techniques* du manuel, et la distribution du résultat de ces modifications par tous les médias habituels, à travers tous les canaux habituels; sans quoi, les restrictions font obstruction à la communauté, le manuel n'est pas libre, et il nous en faut un autre.

Les développeurs de logiciels libres seront-ils déterminés, auront-ils conscience du fait qu'il est nécessaire de produire tout un spectre de manuels libres ? Une fois de plus, notre avenir dépend de notre philosophie.

Il nous faut faire l'apologie de la liberté

On estime aujourd'hui à dix millions le nombre d'utilisateurs de systèmes GNU/Linux et Red Hat Linux de par le monde. **Le logiciel libre propose tant d'avantages pratiques que les utilisateurs s'y ruent pour des raisons purement pratiques**.

Cet état de fait a des **conséquences heureuses**, qui n'échapperont à personne: **on voit plus de développeurs intéressés par la production de logiciels libres, les entreprises de logiciels libres comptent plus de clients**, et il est **plus facile d'encourager les sociétés à développer des logiciels libres commerciaux**, plutôt que des produits logiciels propriétaires.

Mais l'intérêt pour le logiciel libre croît plus vite que la prise de conscience de la philosophie sur laquelle il se fonde, et cela provoque des problèmes. Notre capacité à relever les défis et à répondre aux menaces évoqués plus haut dépend de notre volonté à défendre chèrement notre liberté. Pour nous assurer que notre communauté partage cette volonté, **il nous faut répandre ces idées auprès des nouveaux utilisateurs au fur et à mesure qu'ils rejoignent notre communauté**.

Mais nous négligeons ce travail; on dépense bien plus d'efforts pour attirer de nouveaux utilisateurs dans notre communauté qu'on n'en dépense pour leur enseigner l'éducation civique qui lui est attachée. Ces deux efforts sont nécessaires, et il nous faut les équilibrer.

«Open Source»

En 1998, il est devenu plus difficile de sensibiliser les nouveaux utilisateurs à la notion de liberté dans le logiciel, quand une portion de notre communauté a choisi d'arrêter d'utiliser le terme «Free Software» pour lui préférer la dénomination «Open Source software» N.d.T. : encore et toujours cette ambiguïté de la langue anglaise. «software» signifie «logiciel». «free» signifie à la fois «libre», sens qui est pertinent ici, et «gratuit», qualité qui n'est qu'un effet de bord des logiciels libres. «open source» signifie «dont le code source est ouvert».

Certains de ceux qui ont choisi ce nouveau nom avaient en tête de mettre fin à la confusion souvent constatée entre les mots «free» et «gratuit» - ce qui est un objectif valable. D'autres, au contraire, avaient pour objectif de **laisser de côté le principe qui a depuis toujours motivé le mouvement du logiciel libre et le projet GNU, afin de cibler les cadres et les utilisateurs professionnels, dont beaucoup ont une idéologie où la liberté, la communauté, et les principes, cèdent le pas aux profits**. Ainsi, la rhétorique de **l'«Open Source» met l'accent sur le potentiel pour faire du logiciel puissant et de grande qualité, mais occulte délibérément les idées de liberté, de communauté, et de principes.**

Les magazines «Linux» illustrent clairement cet exemple - ils sont bourrés de publicités pour des logiciels propriétaires qui fonctionnent sur le système GNU/Linux. Quand le prochain Motif ou Qt poindra, ces

magazines mettront-ils les programmeurs en garde en leur demandant de s'en éloigner, ou passeront-ils des publicités pour ces produits ?

La communauté a beaucoup à gagner de la participation des entreprises; toutes choses étant égales par ailleurs, cette contribution est utile. Mais sacrifier à cette aide les discours traitant de liberté et de principes peut avoir des conséquences désastreuses; cela déséquilibre encore plus la situation précédente, où on voit que l'éducation civique des nouveaux utilisateurs s'avère difficile lorsqu'ils affluent.

Les termes **«Free Software» et «Open Source» décrivent tous deux plus ou moins la même catégorie de logiciels, mais correspondent à des conceptions différentes du logiciel et des valeurs qui lui sont associées**. Le projet GNU continue d'utiliser le terme «Free Software» pour exprimer l'idée que la liberté est plus importante que la seule technique.

Jetez-vous à l'eau

La philosophie de Yoda (il ne faut pas essayer) est attirante, mais elle ne s'applique pas à moi. J'ai effectué la plupart de mes travaux sans savoir si j'étais capable de les mener à bien, et sans savoir si ces derniers, une fois menés à bien, suffiraient aux buts que je leur avais fixés. Mais j'ai tenté ma chance, car il n'y avait personne d'autre que moi entre l'ennemi et ma cité. À ma grande surprise, j'ai parfois réussi.

J'ai parfois échoué; certaines de mes cités sont tombées. Je trouvais alors une autre cité menacée, et je me préparais pour une nouvelle bataille. Avec le temps, j'ai appris à reconnaître les menaces et à m'interposer entre ces dernières et ma cité, en appelant mes amis hackers à la rescousse.

Maintenant, il arrive souvent que je ne sois pas seul. C'est pour moi un soulagement et une joie de constater que tout un régiment de hackers se mobilise pour faire front, et je réalise qu'il se peut que cette cité survive - pour le moment. Mais les dangers grandissent chaque année, et **maintenant la société Microsoft a explicitement pris notre communauté dans son collimateur. L'avenir de la liberté n'est pas un fait acquis.** Ne le considérez pas comme tel ! Si vous souhaitez conserver votre liberté, il vous faut vous préparer à la défendre.

Comparatif: La sécurité traitée par le logiciel propriétaire et par le libre.

Une faille dans Explorer menace le commerce électronique
Olivier Ménager, 01 Réseaux, le 06/09/2002 à 19h45

Le 5 août dernier une mauvaise implémentation de SSL dans Internet Explorer a été découverte. **Traitée à la légère par Microsoft**, la faille est jugée critique mais **est restée sans correctif** pour l'ensemble des plates-formes touchées. Pour les experts en sécurité d'Althes, ce sont **des pans majeurs de la sécurité du commerce électronique** qui **ont été mis en cause**.

Le 5 août dernier, Mike Benham, jeune développeur de San Francisco envoie à une liste de diffusion dédiée à la sécurité, une description d'une faille d'Internet Explorer (IE) liée à une **mauvaise implémentation de SSL** (Secure Sockets Layer). SSL est l'un des protocoles de sécurité les plus usités dans le monde du commerce électronique.

La conséquence de cette faille est que presque n'importe qui peut se glisser dans une session sécurisée SSL, donc sur un serveur de commerce électronique. Les utilisateurs du navigateur de Microsoft - soit 90 % des internautes - ont ainsi pu être piraté à leur insu, et se voir dérober leurs codes d'accès ou numéro de carte bancaire. D'autant que **Microsoft a mis un mois à fournir les premiers correctifs.**

Le jeune développeur démontre, dans son rapport et sur son site, un problème majeur de sécurité lié à la chaîne de validation de certificats par le navigateur de Microsoft, ce qui rend IE victime de l'attaque de l'« Intercepteur », plus connu sous le nom « The Man in the Middle ».

Explorer, une véritable passoire

Ainsi, IE (5, 5.5 et 6) accepte à tort une chaîne de certificats, même lorsque le certificat intermédiaire ne dispose pas d'une contrainte de base valide. Or, c'est la contrainte de base qui authentifie le certificat comme étant celui du détenteur de l'URL du site sécurisé. Le navigateur de Microsoft omet tout simplement de vérifier si ce certificat détient ou non une contrainte de base.

Le pirate se glisse dans la session sécurisée entre le serveur de commerce électronique et le client, et recrée deux sessions indépendantes de telle sorte que sa présence passe inaperçue. Il peut se contenter de saisir les données qui sont échangées ou bien modifier, au cours d'une autre session, les données personnelles du client (**faire un transfert de la banque du piraté vers son compte bancaire, par exemple**).

Comme l'explique Vincent Royer, directeur technique chez Althès : « Verisign délivre des certificats serveur SSL qui ne contiennent pas nécessairement de contraintes de base ». Et d'ajouter : « Les versions 5.0 et 5.5 de IE valident à tort une chaîne de certificats, y compris celles contenant un certificat intermédiaire dont la contrainte de base est notée comme fausse [Ndlr : ne correspondant pas à l'URL du serveur]. Un pirate peu donc utiliser n'importe quel certificat serveur SSL acheté auprès d'une autorité de certification commerciale reconnue par IE [Entrust,Thwate, etc.], pour fabriquer un faux certificat de serveur SSL. »

La sécurité des transactions mise à mal

Concrètement, vous pouvez, depuis votre bureau, récupérer un mot de passe et un numéro de carte bancaire d'un collègue qui se connecte à sa banque en ligne au travers d'une session SSL. **Sur le site Thoughtcrime.com, la description de l'attaque est complète.**

« Notre équipe technique a reproduit **cette attaque**, qui **contrairement aux premières assertions de Microsoft**, **est très simple à mettre en oeuvre** », note Gilles Abouy, PDG d'Althès.

Si la seule contrainte consiste pour le pirate à se brancher sur le commutateur par lequel transite la session SSL), Vincent Royer note que « l'attaque peut être aussi mise en oeuvre par du spoofing DNS, même si cela est plus compliqué ». Le spoofing DNS consiste à rediriger les internautes à leur insu vers des sites pirates.

Pour Vincent Royer « la signature électronique S/MIME est aussi falsifiable, ainsi que la signature Authenticode pour codes mobiles ActiveX ».

« Puisqu'aucun avertissement ne prévient l'utilisateur du danger, la confiance apportée par les infrastructures de clefs publiques [PKI], destinées à protéger le commerce électronique contre ce type d'attaque, s'effrite sérieusement », affirme Vincent Royer.

Aujourd'hui, 6 septembre, beaucoup de mises à jour dans de nombreuses langues sont désormais disponibles, y compris en français. Reste à les appliquer sur l'ensemble serveurs et postes clients.

Tout le monde est touché...

Toutes les plates-formes Windows - à partir de Windows 98 - **sont touchées**. Pour le monde Windows, il est nécessaire de corriger l'API de cryptographie de Microsoft (CryptoAPI).

Dans l'univers Apple, MS Office X ou l'édition 2001, Outlook Express 5.0.5, Internet Explorer (version Mac OS 8.1 à 9.x et X) pour Macintosh **sont vulnérables**. **Aucun correctif n'est disponible** pour l'heure **côté Macintosh et Windows 2000.**

Ajoutons que pour les utilisateurs de Linux, le problème a été corrigé en trois jours chrono (week-end inclus) pour KDE Konqueror.

Les correctifs commencent à tomber

En mars 2002, Bill Gates a lancé le concept de Trustworthy Computing. Objectif : sécuriser les produits de Microsoft avec une formation des développeurs à la sécurité. **Entre la théorie et la pratique, Microsoft a du chemin à faire, d'autant que les failles engendrent généralement un effet de domino.**

Stéphane Sabbague, responsable marketing produits chez Microsoft France rappelle que « pour les produits en cours de développement [notamment la gamme .NET], Microsoft s'est attaqué deux mois durant [en mars et avril dernier] à une revue de code complète ».

Le géant de Redmond sait bien que toute faiblesse d'un produit retarde son adoption. Reste que la nécessité de **mettre à jour serveurs et postes clients font de cette faille un gigantesque travail pour les administrateurs**.

Pierre Bugnon, architecte chez Microsoft France, rappelle que le service Software Update Service « permet de reproduire au sein des entreprises l'infrastructure de Windows Update pour simplifier la mise à jour des correctifs de manière centralisée et après validation de l'administrateur ».

Notons que Windows Update ne récupère pas encore les Services Packs des produits. En conclusion, il convient de surveiller les nouveaux correctifs mis à disposition par Microsoft.

Sun Community Source License Principles

by Richard P. Gabriel and William N. Joy

Executive Summary

Community Source creates a community of widely available software source code just as does the Open Source model, but with two significant differences requested by our licensees, as follows:

* compatibility among deployed versions of the software is required and enforced through testing
* proprietary modifications and extensions including performance improvements are allowed

These important differences and other details make Community Source a powerful combination of the best of the proprietary licensing and the more contemporary open source technology licensing models.

In the contemporary world of Internet business, the traditional principles of overly protective software ownership don't make as much sense as they used to. Today it is difficult for a single company to house all the expertise it needs to succeed. Especially when a company wishes to build infrastructure on which other businesses depend, it cannot presume to know how best to do that right out of the chute. The theory of separation of concerns and modularity work well once the dividing lines are known, but finding those lines is a collaborative effort, requiring cooperation between companies.

As business practice has matured in the Internet world, it has become clear that there is good reason to expect that companies will be able to know when to cooperate, and so it makes sense to look at how to license software to take advantage of this emerging understanding.

The Sun Community Source License (SCSL) is designed to balance the needs of organizations needing to innovate rapidly in order to grow with the needs of those organizations to leverage a community's expertise while maintaining proprietary advantages.

Historical Perspective

In the recent past there have been two alternative approaches to software licensing: traditional proprietary licensing and Open Source licensing. Open Source licensing is relatively new, being preceded by shareware and free software.

Proprietary Licensing

Proprietary licensing recognizes primarily binary use licenses which restrict software to execute-only formats, and only those who have bought a license are entitled to use the software. In some cases, the source to the code is available at extra charge and only for limited purposes. In this model, the value of the software is assumed to be contained entirely within that software, and as such, it is to be protected as dearly as any other company asset. Such a model has several advantages, as follows:

It provides protection for intellectual property.
* It guarantees structured innovation, which is innovation that is planned within a single responsible organization.
* It is clear who owns what.

On the other hand, proprietary licensing has a number of disadvantages revealed by the pace of Internet business, as follows:

* Innovation and releases are scheduled, and the schedule may be ill timed for companies that depend on the software. Further, even when a schedule is acceptable for dependent companies, schedules can slip.
* Binary-only products are classical black boxes, and the problem with a black box in an emerging sector is that the wrong things might be in the black box: code with a wrong performance profile, improper resource utilization, or over- or under-generalization.
* The quality of the software depends entirely on the owner organization; this can be a problem if the owner organization does not place the same value on quality as the using organization, the owner organization has

a different perspective on quality, or if quality resources are not allocated in ways that the using organization needs.

In short, the benefits naturally fall out of the observation that proprietary software is completely black box, and that when the boundaries of the black box are optimally placed, the benefits are clear and effective. But if those boundaries are ill-placed or releases are ill-timed, the result is frustration and lost opportunities.

Open Source Licensing

Open Source licensing recognizes that different organizations have different concerns, and therefore the source code is freely available for any party to do what it will. Open Source licensing grew out of the tradition of shareware and free software, which were movements that embodied political beliefs about what can and cannot be owned.

Open Source licensing, on the other hand, does not embody beliefs about ownership aside from recognizing the fact that some organizations wish to own software and others don't. However, Open Source licensing does recognize that certain common elements - for example, infrastructure - are important for a number of parties and that those elements should be freely and openly available.

With Open Source licensing, the source code is available for any use, but there are incentives to "give back" to the Open Source community those improvements that are for the common good. Most importantly, the Open Source approach recognizes that the primary value of a piece of software is the expertise represented by the people who developed it. Thus, even when source is out in the open, the value still remains with those who can expertly manipulate it.

Such a model has several advantages, as follows:

* The code is open with published and, often, specified interfaces.
* There are more developers looking and working on the common source code, so there is higher quality and more-rapid innovation.
* There is no central owning organization that sets schedules and priorities that might conflict with a using organization's schedules and priorities.
* There is a self-organizing effect in which the boundaries between proprietary concerns and community concerns are adaptively set.
* A participating organization can reap the benefits of expertise not in its employ.

There are disadvantages as well, as follows:

* There is no clear control over compatibility issues and there may, therefore, be fragmentation.
* There may be no responsible organization. Bugs introduced by another organization may be too difficult for a using organization to fix and of too low priority for the author to fix in a timely manner.
* Progress can be chaotic and undirected.
* There are limited financial incentives for improvements and innovations, leading commercial developers to use the proprietary model.

Community Source License

The Community Source licensing model takes the advantages from each of the proprietary and Open Source models and eliminates the disadvantages.

In Community Source licensing there is a community of common interests centered around an infrastructure which is provided by a particular organization, called the developing organization. A group of other organizations may join the community, and generally those organizations will have an interest in building businesses around the infrastructure.

For example, JiniTM connection technology licensing is based on a community of companies who wish to build and sell devices employing Jini technology. Sun Microsystems is the developing organization which invented, designed, and built the initial Jini technology infrastructure. This infrastructure is network based and provides a mechanism for devices and software services to gather into a spontaneous network of capabilities.

The community comprises those who have agreed to the license, and within this community there is a mostly Open Source model of interaction. However, because the members of the community are bound by a common license, intellectual property is maintained and there is no requirement to share openly everything developed for the infrastructure. Moreover, the community comprises only those who have agreed to the license, not the general public.

Three levels of participation in the license can be supported, as follows:

* Research Use, which is not only for real research - for example, to improve the infrastructure - but for evaluating the technology and prototyping potential products.
* Internal Deployment, which is both for (very) limited distribution and for testing before launching a product.
* Commercial Use, which is for selling products.

Principles

The following summarizes the principles behind the design of the Sun Community Source License.

Immediate, Open Access
In order to encourage members to join the community defined by the license, there is an easy and risk-free way for a company to get access to the infrastructure source code. Any company or developer can become a Research Use licensee with a simple click-through license which grants broad experimentation and evaluation rights. The licensee may use the source code and its specifications in any way whatsoever short of deployment.

With a Research Use license, a developer may take any approach to using the source code in order to determine whether the technology can be of use; in fact, the developer can do all the work leading up to deployment with such a license.

The license and source code are available through the Net, and so the transition from interest to participation is immediate. Access is clearly open.

Increased Innovation

In order to increase the rate of innovation, improvements and additions to the original source code may be incorporated into the source code regardless of where they come from. There are two basic types of source code improvements that can be made: improvements to the core technology and improvements or additions to surrounding technology. With Jini connection technology, for instance, there is the basic infrastructure and there are services. Each sort of service - storage services, for example - will have common elements which can be improved by community efforts. In this case, acceptable modifications and additions are determined by the subcommunity of organizations who are working in that area using an open process. In such cases it is expected that change can be very rapid, and indeed, must be in order to gain widespread agreement and deployment.

With infrastructure modifications and additions, change is expected to be less rapid because the base of organizations that depend on that infrastructure is large. However, because developers at organizations with different requirements are working with the core technology, over time it is expected that situations will arise that the original developers did not anticipate. These outside developers are able to evolve the technology for their products and markets more rapidly than the original developers would ordinarily be able to do using a traditional proprietary license.

In short, there is a spectrum from the innermost portions of the infrastructure where change should be slow and deliberate to the outermost portions - which might represent applications - where change should be as rapid as the market requires.

Increased Work Force

In order to increase the effective work force, participation is not limited: Any organization can participate in the Research Use License without paying or negotiation.

Increased Quality
Simply stated, with more eyes looking at the code, there is more chance that errors will be caught and repaired, particularly when that code is used in situations and contexts not anticipated by the original developers and researchers. It might be thought that with an Open-source type of model there would be increased likelihood of errors in code, but the Open Source experience is largely the opposite - that because the source is published, developers are generally very careful while developing code in order to avoid public embarrassment.

Moreover, the test code is also covered by the license, and the benefits of a larger community working accrue to the test suite as well as to the target technology.

Faster Commercialization

The Internal Deployment License makes the transition to commercial use easy. Once the technology has been evaluated and adapted through the Research License it can be deployed internally by conforming to the related test suites and specifications. In some cases we expect that Internal Deployment will be allowed without a fee, as it is for the core of the Jini connection technology. The Internal Deployment License is made available with the Research Use License, which means it is also immediate and easy.

The Commercial License is tailored to provide a dependable platform for all third parties and to provide revenues to the developing organization, which can be used to fund both support for the entire community of licensees and also further development.

Access to Students

Researchers, teachers, and their students are particularly attuned to using technology at its limits, and therefore such people are especially encouraged to participate by special conditions of the SCSL. It is easy to use the source code in laboratories and in the classroom.

Protection for Intellectual Property

Licensees rightly enter into the community expecting that their intellectual property rights will be respected. Therefore it is not required that a participant give up intellectual property rights when joining the community. In order that the community remain open, however, the SCSL does require that any programming interfaces which extend the platform or infrastructure technology be open and specified, even while implementations and intellectual property embedded in the implementations may be held proprietary.

The Best of Both Worlds

The Sun Community Source License blends the best aspects of the proprietary and Open Source license models. The SCSL was designed to support a developing organization and a surrounding community of participants. The developing organization is generally building an infrastructure that will spawn a number of new marketplaces or business opportunities for the community of participants. The developing organization might itself act also as an ordinary participant in the community if, for example, that organization desires to take advantage of a new marketplace or business opportunity. Nevertheless, the two roles are separated in the SCSL. The developing organization deserves to reap some benefits for developing the infrastructure, and the SCSL recognizes that.

The following lists the benefits of the SCSL that are shared with or derived from the proprietary model:

* It provides protection for intellectual property.

* While error corrections to the licensed technology must be given back to the community, other modifications can remain proprietary at the discretion of the participating organization that created them.

* It guarantees structured innovation within a single responsible organization.

* The developing organization is responsible for the original code base and the community is responsible for the contributed portions. Ultimately, the developing organization is responsible for the entire source code

base including moving it forward.

It is clear who owns what.

* The infrastructure part of the original code and upgrades to it are owned by the developing organization and shared with the community. Error corrections are required to be given back to the community by all licensees. Community members may contribute shared modifications to the community, granting rights to the community to use these modifications, including intellectual property, specifications, and test suites.

* Licensees may make modifications, such as performance enhancements or platform adaptations which are compatible with the licensed technology, or make extensions - these modifications and extensions may be kept proprietary, though extensions are required to have open interfaces.

* There is clear control over compatibility.

* For the infrastructure part of the original code and upgrades to it, the developing organization provides specifications and test suites. These test suites must be passed by all internally distributed code. Commercial distributions have an additional requirement to use relatively recent upgraded code and pass these test suites, so that the platform can evolve.

* For extensions, the community member making the extension is required to supply specifications and, ultimately, test suites.

In some cases neither a test suite nor a good specification may be good enough to guarantee that a modification, extension, or bug fix is compatible. For some purposes a reference implementation can provide not only a base on which to build a commercial version of a specified piece of functionality but a guide to ensure correctness. Although the SCSL does not mention reference implementations per se, reference implementations as part of a process of defining and accepting specifications for interfaces, for example, are compatible with the SCSL principles and philosophy.

Benefits SCSL shares with or are dervied from the Open Source model are as follows:

* The platform is open with published and specified interfaces.

* The SCSL recognizes the strong community interest in an open platform. A primary mechanism for extension within the community is publishing the specifications for new and often layered interfaces. These interfaces may be best - or in rare cases, only - implemented by using proprietary techniques and technology; this is permitted, but under the SCSL, the programming interfaces themselves and related specifications and test suites must be open.

* There are more developers looking and working on the common source code, so there is higher quality and more-rapid innovation.

* The community pulls organizations and developers into a circle of shared concerns, and this community can effectively self-organize with only a little assistance from the developing organization.

* There is no central owning organization that sets schedules and priorities that might conflict with a using organization's schedules and priorities.

* Though there is a schedule for the developing organization's structured innovation, the SCSL provides every participant with all the freedom and authority to move forward independently.

* There is a self-organizing effect in which the boundaries between proprietary concerns and community concerns are adaptively set.

* There is a force tending to keep the infrastructure stable because all the participants depend on it, but innovation can take place any place in the technology base. The binding requirements of the SCSL such as openness and compatibility also tend to keep the infrastructure stable by preventing predatory modifications and extensions.

* There is also a spectrum of concerns from the developing organization's point of view that ranges from extensions of high importance to those of less importance; by initiating work for those high importance items itself, the developing organization can exert beneficial stability on them.

* A participating organization can reap the benefits of expertise not in its employ.

* There is no limit on who can participate.

Additional benefits to the SCSL are as follows:

* Proprietary modifications including performance improvements are allowed.

* Unlike the proprietary licensing model, all community members can make such changes to the original source base (or upgrades to it) because they have access to it. Unlike the Open Source model, such changes are not required to be given back to the community, though it is encouraged. (Note however, that error corrections to the original source code base must always be returned to the community.)

* Innovation and releases are not subject to a centrally planned schedule.

* Each participating organization may innovate or release at any time with any combination of private and community sources. In order to ensure that other independent releases are compatible, a conformance suite must be passed before release, and published specifications must be respected. For simplicity, any necessary testing and certification can be performed by the releasing participant.

* The quality of the software can be improved throughout the community.

* Each participant can determine appropriate quality levels and work toward them independent of any other participant or in conjunction with others.

* Progress is always made on the original source base, and progress on the peripheries depends on participant needs.

* The original source base is fully the responsibility of the developing organization which can move that base forward; the community may participate in those activities, accelerating them as necessary. Because commercial users of the technology are required to update to newer base technology in new products, after a reasonable delay, the platform can evolve.

* As interest builds in creating related technologies, those parts can move ahead at their own pace.

* There is a place for both incremental improvment and reward for invention.

* Not only does the developing organization have sufficient motivation to invent and innovate but for participating community members there are incentives to invent in the marketplace and protections for those inventions within the community. That is, by creating a more protected community than a completely public one, there can be protections tuned to encourage and safeguard innovation and invention within that community.

Effects

In the last few years there have been remarkable changes in the computer industry largely spurred by the Internet, the Web, and by alternative forms of business. For the first time we have seen business strategies based on giving away products as rapidly as possible, we've seen Open Source gain a large measure of success, and we've seen collaboration where common interests play a stronge role.

There are many benefits to the sort of license the Community Source License represents. The following sections talk about them a bit.

Increased Commitment

One of the difficulties of trying to get a new infrastructure technology adopted is the sense that a proprietary

standard might not make it or that a competitive situation will make choosing it difficult or impossible. Regardless of who the sponsoring company is, there is a risk associated with not knowing everything.

Because participants in the Community Source License have access to the source code and can reasonably expect that their innovations will appear in that source, those participants can see everything there is to see about the technology. By being part of a community of partners, the risk of putting all one's eggs in a single company's basket is reduced. The SCSL creates a context in which commitment is less risky, and the more participants there are, the less risky it is.

Closer Ties Between Development Groups

Developers have always gotten together at conferences and coffee houses, and when they do they are very good at two things: exchanging useful information in the form of techniques and know-how, and not revealing secrets. When developers get together within the community created by the SCSL, they are able to speak freely about particular issues represented by the shared source code. Thus, beneficial information exchange can take place more rapidly, and each participating organization benefits - because writing software is hard.

Rapid Spread of Influence

Because it is so easy for a participant to join the community and because there is a shared commitment to the technology once participants are fully engaged, the underlying infrastructure is spread more rapidly. And because there are more development groups pushing up both the innovation and quality, the new marketplace based on the infrastructure is created more rapidly than it would any other way.

Reduced Risk for Adopters

Frequently, black-box technology products pose too great a risk for many organizations because those organizations end up depending on another company for the development, maintenance, and evolution of parts of their own products. Most importantly, the priorities the owner company would assign to work related to the technology are not necessarily priorities a customer would choose. With SCSL, the participants share common source code and can set their own priorities, working as much or as little on the code as they wish.

Those organizations that require more infrastructure can take it from a platform provider as usual just as some organizations prefer to buy from Red Hat rather than getting Linux from an ftp site.

The Future

In writers' workshops, in design charrettes, in code reviews and walkthroughs, and in artists' studios there is one common theme: By working within a community, a better result is achieved than working alone - even virtuosos learned by criticism and sharing. And if you talk to almost any artist or software developer, you will hear that what they are doing is a "work in progress."

In defining the Sun Community Source License, Sun has tried very hard to get it right, but maybe it isn't perfect. Naturally, we believe so much in our open process that the license itself is subject to that same process, and as participants need changes in the license, we will make them.

Right now, in the context of an industry rapidly changing, it's our best attempt - it's a work in progress.

Linux makes a run for government

By Robert Lemos
Staff Writer, CNET News.com
August 16, 2002, 4:00 AM PT

SAN FRANCISCO--A technology policy think tank is campaigning to win Linux a greater role in government by offering to act as a central repository for a federally certified version of the open-source operating system.

The Cyberspace Policy Institute, established a decade ago at George Washington University, plans to push for Linux to be certified under the Common Criteria, a standard grading of technology required by the United States and other countries before products can be sold into sensitive government applications.

If successful, the initiative would lead to a single, standard version of Linux acceptable to the government, and hence make it easier for Linux companies to compete against Microsoft and other large software makers. Certification costs anywhere from $100,000 to millions of dollars and takes up to five years--Microsoft is just finishing the certification of Windows 2000--but the effort could be a boon for Linux companies.

"The government wants to get open-source certified, but they don't want to certify for any specific vendor," Tony Stanco, senior policy analyst for open-source and e-government at the Cyberspace Policy Institute, said at a panel discussion on promoting Linux to the government.

A single agency administering the certification process for Linux is a must, Stanco said. Otherwise, only a few companies would be able to offer products and the entire community wouldn't benefit from the effort.

"Only one company (Red Hat) has enough money to get certified," he said. "I don't think even United Linux has enough money to get Linux-certified."

The initiative would also add the United States to the list of national governments that are supporting open-source efforts to offer a second option, along with Microsoft software, within federal agencies. On Monday, the British government confirmed that it would consider open-source software alternatives to buying Microsoft applications. And, in June, the German government signed a deal with IBM and Linux vendor SuSE to provide an open-source alternative to Microsoft operating systems. Both China and Taiwan, two nations often at loggerheads, have also dipped their toes into Linux.

A better Linux Strong support for the open-source operating system within the government came from a surprising quarter in early 2001 with the release of Security-Enhanced Linux from the National Security Agency, which for decades stymied researchers' and technology companies' efforts to create broadly available strong encryption.

SE Linux adds military-strength architecture improvements to Linux, the most obvious security improvement being mandatory access controls, or MACs, based on technology developed by Secure Computing Corp. The Cyberspace Policy Institute plans to also add authentication and key management features to the operating system.
[...]
"With the access controls, the customer doesn't have to worry about the next buffer overflow that comes along," said Westerman at a panel discussion at this week's LinuxWorld Conference and Expo. "SE Linux gives you military grade security at open-source cost."
[...]
Sources familiar with events said that aggressive Microsoft lobbying efforts have contributed to a halt on any further work. "Microsoft was worried that the NSA's releasing open-source software would compete with American proprietary software," said a source familiar with the complaints against the NSA who asked not to be identified.
[...]
The debate over whether the government should fund open source projects has been raging for some time. In July, MITRE, a defense contractor and think tank, released a much-awaited report sponsored by the Department of Defense endorsing the use of open-source software in the government.

"Open source methods and products are well worth considering seriously in a wide range of government applications," the report concluded.
[...]
Despite the intense battle surrounding the open source, the NSA will still fund research on secure operating systems based on Linux as well as work with U.S. companies to create better security in their own operating systems.

Both Red Hat's CEO Matthew Szulik and Chief Technology Officer Michael Tiemann said the company is working with the NSA on security projects, but neither would give details about the initiatives. On Tuesday morning, Tiemann and other technologists from companies including Intel, IBM and Oracle met to discuss the future of Linux in the government, said a source familiar with the meeting.

Through the Composable High Assurance Trusted Systems (CHATS) fund, the Defense Advanced Research Projects Agency, an arm of the Department of Defense, funds open-source initiatives that improve security. A year ago, Network Associates received $1.2 million from the CHATS program to create a common set of security features for open-source operating systems.

Apple Computer also will push its own operating system, the Mac OS X, which is based on the open-source Unix variant, FreeBSD, for government certification. Apple and a coalition of 40 government agencies have formed the Secure Trusted Operating System (STOS) consortium to create security features for the base FreeBSD operating system known as Darwin.

Welcome to certification

The road to certification will not be easy, however. For one, the co-developer of SE Linux, Secure Computing, has indicated that it plans to enforce patent claims on part of the access control technology based on its research and development.

In addition, the Common Criteria process, run jointly by the NSA and the National Institute of Standards and Technology under the National Information Assurance Partnership (NIAP), is better suited to certify proprietary software coming from a single company. It's ill suited to deal with the myriad updates that the open-source community produces on a regular basis.

"The big issue is how you fit this wild community into the all the little boxes that the government bureaucracy wants," said CPI's Stanco.

NIAP Common Criteria certifications run from Evaluation Assurance Level 1 (EAL), the lowest level, to EAL 7, the highest. The first four levels can be obtained through commercial labs, but the levels 5 through 7 require certification from the NSA themselves.

Because it is Linux's first time through the process, the Cyberspace Policy Institute has modest aims: EAL 2.

"That way we get some validation of open-source security," said Stanco. "Going straight to EAL 4 would be tough."

Shooting for a modest target gives the open-source community time to work out some kinks--not in Linux, but in the government's certification process.

U.K. government backs open source

By Matt Loney
Special to CNET News.com
July 23, 2002, 3:15 PM PT

The U.K. government confirmed on Monday that it will consider open-source software as a way to avoid getting locked into proprietary information technology products.

Not only central government will be affected by the policy: so too will local governments and the wider public sector, including non-departmental public bodies and the National Health Service. Contracts will be awarded on a value-for-money basis.

The move is likely to be seen as a major boost to open-source software. Open source has been increasingly adopted by big software vendors since 1998, when companies such as IBM, Oracle and Computer Associates started to take it seriously. IBM has since said it has devoted $1 billion to the marketing and development of open-source software.

What marks open-source software out as different from proprietary code is the licence under which it is distributed. Open-source licenses--of which the GNU General Public Licence is the best-known example--allow organizations and individuals to use and modify code free of charge, as long as any modifications are released back to the programming community.

In the final draft of the U.K. government's policy on open-source software, published on Monday by the Office of Government Commerce (OGC), the government says that in all future IT developments where interoperability is an issue, it will only use products that support open standards and specifications. Furthermore, it will follow a recent European Commission policy document that suggested exploring the open-source route for all government-funded software research and development.

"OSS (open-source software) is indeed the start of a fundamental change in the software infrastructure marketplace, but it is not a hype bubble that will burst and U.K. government must take cognizance of that fact," said the OGC in the policy document.

The government is already pushing adoption of open standards through its e-Government Interoperability Framework (e-GIF) but, said the OGC, it is now considered necessary to have a more explicit policy on the use of OSS within the U.K. government and this document details that policy.

The OGC said it was satisfied that open-source software could provide good enough security for government systems. "Properly configured open-source software can be at least as secure as proprietary systems, and is currently subject to fewer Internet attacks," it said, adding that in some cases mainstream proprietary products may be significantly less secure than open-source alternatives.

The decision comes one week before Microsoft's deadline for customers to sign up to its controversial Software Assurance licensing scheme, which analysts say will substantially hike fees for large customers. Instead to being able to buy software upgrades when it suits them, as is common practice, companies will have to either pay Microsoft the full price when they upgrade, or pay an annual fee under the Software Assurance program for the right to upgrade.

Microsoft, which remains the only big software company not to port its applications to open source platforms such as Linux, is keen for governments to ignore the platform too.

In a speech delivered to the Government Leaders' Conference in Seattle earlier this year, Microsoft chairman Bill Gates likened the concept of software based on the General Public License--which much open-source software uses--to anti-capitalism, adding that those governments who put development time into it are denying themselves the benefits of essential taxes. "The so-called (Free Software Foundation)...says that these other countries other than the U.S. should devote R&D dollars in the so-called open approach, that means you can never commercialize that software," said Gates at the time.

Free speech, free beer and free software

By Simon Phipps
August 20, 2002, 4:00 AM PT

On the Internet, software wants to be free. But as the Free Software Foundation and many others point out, the word "free" here is not about price; it is about liberty.

"Free" is used as in the phrase "free speech" (a right we covet), rather than the phrase "free beer" (always too good to be true) or "free kitten" (which sounds good, but has a high overhead).

Confusion arises because free software mostly has a zero price tag as a natural consequence of the original license, the GPL, that enforces the liberty of developers to use code created by their peers. The innovation of the Open Source Initiative was to provide new, more business-friendly licenses. By suggesting alternatives to GPL licensing, it enabled hybrid open-source/closed-source works.

The early years of open source have thus focused on free (as in beer) software, so it is still possible to misunderstand. But we have seen a definite shift in thinking. The open-source community has welcomed companies that build commercial enterprises, as long as they act symbiotically rather than parasitically. Today it is clear that open source has matured.

The key values of the Internet flow from the mesh of participants, which Metcalfe's Law observes as leading to an exponentially growing pool of potential relationships. Complementing that are loosely coupled, open, royalty-free standards, allowing all to participate at the linear-growing cost of connection to the standards rather than the exponentially growing cost of negotiating each relationship.

This exponential-relationship, linear-cost world is termed the "Net Effect" and has been the primary energy source of the Internet revolution from its inception.

The Web resulted from the Net Effect, and today we need a development and deployment methodology that harnesses it. Open source provides the ideal, loosely coupled yet rigorous environment for the massively connected community.

What distinguishes projects like Apache, NetBeans and Linux is less the price tag but rather the eclectic inclusiveness. If a standard is a technology where the community affected by changes controls the changes, then open source will underpin standards in this century.

Open source is not without cost. Someone has to underwrite the community. Developers have to donate their time and expertise. Sun's experience of underwriting NetBeans.org and OpenOffice.org (and others) has involved commercial and individual end users and developers cooperating with generosity to build the platform on which their solutions can be delivered.

The evolution of these projects has revealed the existence of a commercially valid business model based on open source. It has become the development methodology of the Net Effect.

In the evolved open-source development model, a "community of code" maintains an open-source code base, preferably itself evolved "in the open." It uses behaviors and principles well documented elsewhere, which inherently lead to better code faster, not least because of the scrutiny of the community. These benefits are obtained as long as there is a viable community of interested parties to create, maintain and improve the code.

From the "community of code," a gatekeeper function emerges, operating with implicit community consent. The gatekeeper embodies the will of the community to set bounds for the project and creates the "reference implementation." Changes are made infrequently enough to ensure stability for solutions above, while paying regard to work in the community below. This gatekeeper is the key to success for the evolved open-source model, bridging community informality with the stability needed by commercial enterprise.

Above the reference implementation is the crucial dividing line between the code foundation and solution offerings that depend on it. The licenses used below the line foster and protect the community; those above the line facilitate commercial success. Drawing the line is the great art; the balance between community and

commerce differs for every project.

Open or closed ?

The experience of Sun and others is that open source provides ideal development and business models for today's Net Effect economy. It's not about free stuff; it's about enfranchising every user and development community member. Today's software innovations need this model more than ever before. With an open foundation, companies can gain their just compensation for their innovations "above the line," but the subtle lock-in offered by our traditional understanding of "standards" is largely avoided.

Most importantly, open source is not just about code; it is about community. You don't make a project open source just with a license. It takes the costly and time-consuming birthing of a community of code, a trusted gatekeeper function and a series of symbiotic commercial enterprises to make true open source. 21st century open source is not free.

But it is liberating.

Can Linux duck the Redmond death ray?

By Charles Cooper
August 16, 2002, 4:00 AM PT

Stuck in a corner of San Francisco's Moscone Center, Microsoft's contingent at the LinuxWorld Conference and Expo resembled more a band of spies than an army ready for battle.

This was Microsoft's first official attendance at a Linux convention--an appearance that spoke volumes about the company's debate over how to best respond to the "Penguinista" challenge. This is the same company, remember, whose senior execs have long treated the open-source movement with thinly veiled distrust, if not open contempt.

Yet at the same time, Microsoft understands that Linux may be the biggest threat to its domination of the desktop since Janet Reno and her legions at the Justice Department. Some Redmond insiders would love to crush Linux, but it's way too late for that. And so it becomes all the more important to engage the Linux community--if not co-opt it.

"We're getting a lot of good response from people here," said a predictably cheerful Microsoft staffer at the show. While handing out "information," he was only too willing to explain how Microsoft could help Linux work even better.

Maybe so, but it would not be a match made in software heaven, so to speak. While I was standing there, a convention attendee strolled by, taunting the Microsoft booth boys with cries of, "shame, shame, shame!"

Granted, Microsoft's been called worse over the years (how about predatory monopolist ?) so any temporarily hurt feelings won't deflect the company from finding a way to "embrace and extend" Linux in much the same way it tried to recast Java in Windows image.

That's where Microsoft wants to head, though I'm not sure it's possible--any more than it is to imagine Linus Torvalds entering into a coding partnership with Bill Gates. Perhaps even more than their Java compadres, Linux developers aren't ready to drink the Redmond Kool-Aid quite yet.

Linux represents a tradition that supports community control over code standards, while Microsoft prefers to own the crown jewels of its software empire. Each approach has its pros and cons, but one can't easily finesse the difference between the two points of view.

Microsoft's right when it says few big Windows NT customers are ripping out their expensive software infrastructure to install Linux systems. Of course, that's today--will it hold true tomorrow ? As more young companies grow, I'll bet lots of them will conclude that Linux is for them.

Network computer redux?

Sun Microsystems CEO Scott McNealy, who rolled into town this week for the LinuxWorld convention, would concur. But there's more Linux tricks up Sun's sleeve. The company is getting ready to introduce a Linux-based personal computer that would include Sun's StarOffice collection of software applications.

McNealy, who was wrong about the network computer, just might be on to something this time.

When Sun--and Oracle--first talked up the concept of the network computer in 1997, much was made of its potential as an alternative to a Microsoft-centric PC world. The general idea was to offer stripped-down access terminals that would connect to a network, where most of the computing muscle and complexity resides (a world presumably powered by big Sun servers, of course). Unfortunately for Sun, early customer enthusiasm faded after the price of many Windows-based computers fell below $500.

Sun may have been a victim of timing, but since then, a lot has changed. Microsoft, a convicted predatory monopolist, still grapples with its corporate image in the post-Enron era. Also, the software giant isn't winning many friends with a new, more expensive licensing plan that many customers find onerous. All the while, Linux is getting a serious look in corporate America as an alternative operating system.

Although a low-end Linux desktop wouldn't technically qualify as a network computer, it would be a means to the same end: Elbow aside Microsoft but keep rivals (such as Dell Computer) away from bread-and-butter accounts, such as call centers, which use Sun's more-expensive servers.

Will it work ? Assuming Sun doesn't trip on its delivery, corporate customers might just be receptive this time around. And the clincher would be Linux.

The Growing Politicization of Open Source

by Tim O'Reilly, Aug. 15, 2002

Editor's Note: Join the discussion at Slashdot generated by Tim's Weblog below, as well as by a Slashdot posting from Michael, titled "Tim O'Reilly Bashes Open Source Efforts in Government." Despite Michael's inflammatory title, Tim's posting has engendered quite a debate, with viewpoints coming down on all sides of this issue.

One Slashdot commenter had this to say: "[Tim is] just saying that the government should use the best tools for the job, and not belabor its choices with (more) bureaucracy." Another says, "We should be more concerned about pushing the government to use open *standards*, rather than open source software."

What do you think ?

I just received some thought-provoking mail in response to the recent news about the proposed Digital Software Security Act, which would require open source software to be used in California state agencies. According to news.com, " If enacted as written, state agencies would be able to buy software only from companies that do not place restrictions on use or access to source code."

This mail comes from someone I've worked with for the past several years in promoting open source in corporate America. He's a thoughtful advocate for the benefits of open source, but he's disturbed (as am I) by the growing politicization of open source. He asked me to keep his mail anonymous, because he doesn't have permission to speak for his employer (and it's not Microsoft!). He wrote:

Is it just me, or does it seem that a vocal portion of the Open Source community is starting to get radicalized and politicized ?

I'd be all in favor of legislation that mandates Open Standards where applicable, and requires government bodies to seriously consider Open Source alternatives. But what we're getting is attempts to *require* use of Open Source software - effectively criminalizing an official' s decision to buy commercial software to meet their needs. First in Peru, and now here. And at LinuxWorld it is apparently being preached and accepted as part of the "party doctrine."

Whether you want to argue on moral grounds of personal freedom, or the practical grounds of generating a backlash, this seems like an extremely counter-productive strategy for Open Source advocates to be pursuing. Sincere and understandable, but horribly misguided. By asking for government mandates on software purchasing, they're practically inviting the commercial software developers to lobby for legislation forbidding Open Source, in response to 'our' efforts to require it.

From my perspective, I would think the Open Source community would be much better served if it took the moral high ground and called for Openness in Software Procurement. If you feel you have to coerce people, it would be better to force them to increase their disclosure. Require officials to document their acquisition critieria, require companies to publish their licensing policies, insist on use of open file formats for publicly accessible documents. That is, increase the flow of information and the range of choices, rather than trying to decrease them. That's what Open Source is supposed to be about - increasing choices, right? And wouldn't that put the commercial companies on the defensive, rather than letting -them- wrap themselves in the flag of freedom of choice?

Does this make any sense? Wouldn't a legislative agenda of increasing openness, rather than mandating choices, be more in keeping with the philosophy and culture of Open Source? Shouldn't someone with credibility be advocating such a balanced approach, rather than letting the radicals drive the public agenda?

Does it bother you, too, or am I just being oversensitive?

Yes, it does bother me. When I first heard of the proposed legislation in Peru, I thought it was great theater, and that the open source advocates in Peru clearly made some telling points about its benefits. But the more I think about it, the more you I realize that having governments specify software licensing policies is a bad idea. My correspondent crystallized my feelings on the matter, and made me realize that support for such

legislation was a violation of what I have previously called "my version of Freedom zero". I think he's right. This is a slippery slope, and a really dangerous idea, which thoughtful free software and open source advocates ought to reject.

As T.S. Eliot said in Murder in the Cathedral: "This last temptation is the greatest treason: to do the right deed for the wrong reason." No one should be forced to choose open source, any more than they should be forced to choose proprietary software. And any victory for open source achieved through deprivation of the user's right to choose would indeed be a betrayal of the principles that free software and open source have stood for.

Tim O'Reilly is founder and president of O'Reilly & Associates and an activist for Internet standards and for open source software.

Lettre de l'état péruvien au directeur général de Microsoft Pérou.

Lima, le 8 avril 2002.

Monsieur JUAN ALBERTO GONZÁLEZ, Directeur Général de Microsoft Pérou

Cher Monsieur.

Avant toute chose, je vous remercie de votre lettre du 25 mars 2002 dans laquelle **vous exprimez la position officielle de Microsoft concernant le Projet de Loi N°1609, Logiciel Libre dans l'Administration Publique**, qui est inspirée sans aucun doute par le désir d'aider le Pérou à réussir à trouver sa place dans le contexte technologique global. Animé du même esprit et convaincu que nous trouverons les meilleures solutions par l'échange d'idées claires et ouvertes, je me permets de répondre, par la présente, aux commentaires contenus dans votre lettre.

Je reconnais que **des opinions comme les vôtres** constituent un apport significatif, mais elles m'**eussent été plus utiles si**, en plus de formuler des objections à caractère général (que nous analyserons en détail plus loin) **vous aviez rassemblé des arguments solides sur les avantages que le logiciel propriétaire peut apporter à l'État péruvien et à ses citoyens** en général, car cela aurait pu permettre un échange plus clair dans le respect des positions de chacun.

Dans le but de clarifier le débat, nous conviendrons que ce que vous appelez "logiciel à code source ouvert" est ce que le Projet définit comme "logiciel libre", sachant qu'il existe du logiciel dont le code source est distribué avec les programmes, mais qui n'est pas couvert par la définition établie dans le Projet; et que ce que vous appelez "logiciel commercial" est ce que le Projet définit comme "propriétaire" ou "non libre", sachant qu'**il existe du logiciel libre commercialisé sur le marché avec un prix comme tout autre bien ou service**.

De même il est important de préciser que la proposition contenue dans **le Projet** auquel nous nous référons **n'est pas directement en relation avec l'économie directe** qui peut être **réalisée par l'emploi de logiciel libre** dans les institutions de l'État. Ceci est dans tous les cas, une valeur ajoutée marginale, mais en aucune manière l'objectif final du Projet. **Les principes élémentaires qui inspirent le Projet sont liés aux garanties fondamentales d'un État démocratique de droit**, telles que:

- **Libre accès du citoyen à l'information publique;**
- **Pérennité des données publiques;**
- **Sécurité de l'État et des citoyens.**

Pour garantir le libre accès des citoyens à l'information publique, il est indispensable que l'encodage des données ne soit pas lié à un fournisseur unique. L'utilisation de formats standards et ouverts permet de garantir ce libre accès, et d'obtenir, si nécessaire, la création de logiciel libre compatible.

Pour garantir **la pérennité des données publiques**, il est indispensable que **l'utilisation et le maintien du logiciel ne dépendent pas de la bonne volonté des fournisseurs, ni des conditions de monopole imposées par ceux-ci**. Pour cela l'État a besoin de **systèmes dont l'évolution puisse être garantie par la disponibilité du code source.**

Pour **garantir la sécurité de l'État ou sécurité nationale**, il est indispensable de se baser sur **des systèmes dépourvus d'éléments qui en permettent le contrôle à distance ou la transmission non désirée d'information à des tiers**. Par conséquent, il faut des systèmes dont le code source est librement accessible au public pour permettre son examen par l'État lui-même, les citoyens, et un grand nombre d'experts indépendants dans le monde. Notre proposition apporte un plus de sécurité, puisque **la connaissance du code source élimine le nombre croissant de programmes contenant** potentiellement **du *code espion***.

De cette façon, notre proposition renforce la sécurité de nos citoyens, à la fois en tant que détenteurs légitimes de l'information gérée par l'État, et en tant que consommateurs. Dans ce dernier cas, c'est en permettant l'apparition d'une offre étendue de logiciel libre dépourvu de potentiel *code espion* susceptible de mettre en péril la vie privée et les libertés individuelles.

En ce sens, le projet de loi se limite à établir les conditions dans lesquelles les organismes de l'État **acquerront du logiciel** dans le futur, à savoir, **de façon compatible avec la garantie de ces principes fondamentaux**.

A la lecture du projet il apparaîtra clairement qu'une fois approuvée :

- **la loi n'interdit pas la production de logiciel propriétaire ;**
- **la loi n'interdit pas le commerce de logiciel propriétaire ;**
- **la loi ne dicte pas quel logiciel utiliser concrètement ;**

- **la loi ne dicte pas chez quel fournisseur acheter le logiciel ;**
- **la loi ne limite pas les termes de la licence qui couvre un produit logiciel.**

Ce que le projet exprime clairement c'est que, **pour être acceptable** par l'État, **il ne suffit pas que le logiciel soit techniquement suffisant pour mener à bien une tâche, mais il faut en plus que ses conditions contractuelles satisfassent une série de pré-requis en matière de licence**, sans lesquelles l'État ne peut pas garantir au citoyen le traitement adéquat de ses données, **veiller à leur intégrité, leur confidentialité et leur accessibilité** au cours du temps, car ce sont des aspects critiques de son usage normal.

Nous sommes d'accord, Mr. González, sur le fait que **la technologie de l'information et des communications a un impact significatif sur la qualité de vie des citoyens** (sans que pour eux, l'impact soit toujours positif ou neutre d'effet). De même nous serons certainement d'accord pour dire que les valeurs de base que j'ai signalées plus haut sont fondamentales dans une nation démocratique comme le Pérou. Depuis longtemps nous cherchons une alternative permettant de garantir ces principes, qui ne consiste pas à recourir à l'emploi de logiciel libre dans les termes définis dans le projet de Loi.

Quant aux observations que vous formulez, nous allons maintenant les examiner dans le détail:

En premier lieu, vous signalez que : "1. Le projet établit l'obligation pour tout organisme public d'employer exclusivement du logiciel libre, c'est-à-dire à code source ouvert, ce qui transgresse les principes de l'égalité devant la loi, de non-discrimination et les droits à la libre initiative privée, liberté d'entreprendre et de contrat, protégés par la constitution.".

Cette appréciation est une erreur. En aucune façon le projet n'affecte les droits que vous énumérez: il se limite à établir les conditions pour l'emploi de logiciel au sein des institutions de l'État, sans s'immiscer d'aucune manière dans les transactions du secteur privé. C'est un principe bien établi que l'État n'a pas la grande liberté de contrat du secteur privé, précisément parce qu'il est limité dans ses actions par le devoir de transparence des actes publics; et en ce sens, la préservation de l'intérêt commun doit prévaloir lorsqu'il légifère en la matière.

Le projet protège l'égalité devant la Loi, et aucune personne physique ou morale n'est exclue du droit d'offrir ces biens à l'État dans les conditions fixées dans le projet et sans plus de limitations que celles établies dans la loi des Contrats et Acquisitions de l'État (T.U.O. par Décret Suprême No. 012-2001-PCM).

Le projet n'introduit aucune discrimination, puisqu'il établit uniquement *comment* ces biens doivent être fournis (ce qui est une prérogative d'État) et non *qui* doit les fournir (ce qui serait effectivement discriminatoire si les restrictions étaient fondées sur l'origine nationale, raciale, religieuse, idéologique, la préférence sexuelle, etc.) Au contraire, le projet est résolument anti-discriminatoire. Il en est ainsi parce qu'en déterminant, sans l'ombre d'un doute possible, les conditions de sélection d'un logiciel, il évite aux organismes de l'État d'utiliser des programmes dont la licence inclurait des conditions discriminatoires.

Il résulte de ce qui a été exposé dans les paragraphes précédents, que le projet n'attente pas à la libre initiative privée, puisque celle-ci peut choisir sous quelles conditions elle produit un logiciel; certaines d'entre elles seront acceptables pour l'État, et d'autres ne le seront pas parce qu'elles contrediront la garantie des principes fondamentaux énumérés plus haut. Cette libre initiative est compatible avec la liberté d'entreprendre et la liberté de contrat (dans les limites où l'État peut exercer cette dernière). Tout sujet privé peut produire du logiciel selon les conditions requises par l'État, ou peut s'abstenir de le faire. Personne n'est forcé d'adopter un modèle de production, mais si quelqu'un désire fournir du logiciel à l'État, il lui faudra mettre en oeuvre des mécanismes garantissant les principes qui sont décrits dans le projet.

En guise d'exemple : rien dans le texte du projet n'interdit à votre société d'offrir aux organismes de l'État sa "suite" bureautique, dans les conditions définies dans le projet et à un prix que vous jugerez convenable. Si vous ne le faites pas, cela ne sera pas dû à des restrictions imposées par la loi, mais à des décisions de votre société tenant compte du mode de commercialisation de ses produits, décisions auxquelles l'État ne participe pas.

En poursuivant, vous signalez que : "2. Le projet, en rendant obligatoire l'emploi de logiciel à code source ouvert, établira un traitement discriminatoire et non compétitif pour les contrats et les acquisitions des organismes publics..."

Cette affirmation est une réitération de la précédente, la réponse se trouve quelques lignes plus haut. Cependant, arrêtons nous un instant sur votre appréciation concernant le "traitement ... non compétitif."

A l'évidence, au moment de définir un quelconque type d'acquisition, l'acheteur se fixe des conditions liées à l'usage prévu pour le bien ou le service. A partir de là, il exclut certains fabricants qui n'auront pas la possibilité de rivaliser, sans pour autant les avoir exclus "a priori", mais sur la base d'une série de principes décidés par la volonté autonome de l'acheteur, si bien que le processus s'avère finalement conforme à la loi. Et dans le projet il est établit que ***personne*, n'est exclu de la compétition pour autant que la garantie des principes fondamentaux est satisfaite.**

De plus **le projet *stimule* la compétition**, du moins **il pousse à générer une offre de logiciel présentant de meilleures conditions d'utilisation**, et à optimiser les travaux déjà accomplis, dans un modèle de progrès continu.

D'un autre côté, **l'aspect central de la compétitivité est l'opportunité de proposer de meilleures options au consommateur**. Il est impossible d'ignorer que le marketing ne joue pas un rôle neutre au moment de la présentation d'une offre au marché (du moins admettre le contraire reviendrait à dire que les investissements réalisés par les entreprises en matière de marketing sont dépourvus de sens), et par conséquent une dépense significative dans ce domaine peut influencer les décisions de l'acheteur. Cette influence du marketing est dans une large mesure réduite par le projet que nous soutenons, puisque le choix proposé dans le marché se base sur le *mérite technique* du produit et sur l'effort de commercialisation du producteur; en ce sens, la compétitivité est accentuée, et même **le plus petit producteur de logiciel peut rivaliser sur un pied d'égalité avec la plus puissante des entreprises**.

Il est nécessaire de souligner qu'**il n'y a pas de position plus anti-compétitive que celle des grands producteurs de logiciel propriétaire**, qui fréquemment, **abusent de leur position dominante**, parce que dans d'innombrables cas **ils proposent comme unique solution** aux problèmes soulevés par les utilisateurs : "**mettez à jour vos logiciels vers la nouvelle version**" (à la charge de l'utilisateur évidemment) ; de plus, les **interruptions arbitraires d'assistance technique sur des produits, jugés "anciens"** par le fournisseur, sont communes; ensuite pour obtenir une quelconque assistance technique, l'utilisateur est contraint de migrer (avec un coût non trivial, en particulier lorsque la migration implique des changements de plate-forme matérielle) vers de nouvelles versions. Et comme **toute l'infrastructure est consolidée par des formats de données propriétaires, l'utilisateur reste "captif" de la nécessité de continuer à employer les produits du même fournisseur**, à moins de consentir un énorme effort pour passer à un autre environnement (probablement tout aussi propriétaire).

Vous ajoutez : "3. Ainsi, en obligeant l'État à favoriser un modèle de commerce qui s'appuie exclusivement sur le logiciel à code source ouvert, le projet ne fera que décourager les sociétés de fabrication locales et internationales qui sont celles qui réalisent les véritables investissements, créent un nombre significatif d'emplois directs et indirects et contribuent au PIB contrairement à un modèle de logiciel à code source ouvert qui tend à avoir un impact économique toujours moindre du fait qu'il crée principalement des emplois de service."

Je ne suis pas d'accord avec ce que vous affirmez. En partie à cause de ce que vous-même signalez dans le paragraphe 6 de votre lettre, concernant **le poids relatif des services dans le contexte de l'utilisation du logiciel**. Cette contradiction, par elle-même, invalide votre position. **Le modèle des services, adopté par un grand nombre d'entreprises de l'industrie informatique, est bien plus significatif, en termes économiques, et de façon croissante, que le commerce de licences sur les programmes**.

D'un autre côté, **le secteur privé dispose de la plus grande liberté pour choisir le modèle économique qui convient le mieux à ses intérêts, même si cette liberté de choix est souvent obscurcie de manière subliminale par les investissements disproportionnés dans le marketing des producteurs de logiciel propriétaire**.

De plus, à la lecture de votre opinion il ressort que le marché de l'État est crucial et indispensable pour l'industrie du logiciel propriétaire, à tel point que si l'État adopte ce projet, il éliminerait complètement ces sociétés du marché. En supposant, ce qui n'est pas le cas, que ce soit vrai, nous en déduisons que **l'État subventionne l'industrie du logiciel propriétaire**. Dans cette hypothèse peu probable, l'État aurait alors le droit d'**attribuer ses subventions au domaine qu'il considère comme ayant la plus grande valeur sociale**. il en résulterait que si l'État décide de subventionner le logiciel il devra le faire en préférant le libre par rapport au propriétaire, compte tenu de son effet social et de son **utilisation rationnelle de l'argent des contribuables**.

Concernant les emplois générés par le logiciel propriétaire dans des pays comme le nôtre, ceux-ci concernent majoritairement des tâches techniques de faible valeur ajoutée; au niveau local, **les techniciens qui offrent du support au logiciel propriétaire produit par des entreprises transnationales ne sont pas en mesure de corriger un bug**, pas nécessairement faute de capacité technique ou de talent, mais **parce qu'ils ne disposent pas du code source**. **Le logiciel libre crée des emplois techniquement plus qualifiés et on génère un cadre pour la libre concurrence où le succès n'est limité que par la capacité d'offrir du bon support technique et de la qualité de service**, on stimule le marché et on enrichit le patrimoine commun de la connaissance, en ouvrant des alternatives pour générer des services de grande valeur ajoutée et de meilleur profil de qualité profitant à tous les acteurs: producteurs, prestataires de services et consommateurs.

C'est un phénomène courant dans les pays en voie de développement que les industries locales de logiciel tirent la majeure partie de leurs revenus des services ou de la fabrication de logiciel "ad hoc". Par conséquent, l'éventuel impact négatif que l'application du projet pourrait avoir dans ce secteur sera compensé par la croissance de la demande de services (à condition que ceux-ci soient conformes aux exigences de qualité). Évidemment, il est probable que les entreprises transnationales de logiciel décidant de ne pas concourir conformément à ces règles du jeu, souffrent d'une perte de revenus en termes de facturation de licences; néanmoins, considérant que ces entreprises soutiennent que beaucoup de logiciels utilisés par l'État ont été copiés illégalement, on peut penser que l'impact ne sera pas très sérieux. Certainement, en tout cas, leur succès sera déterminé par les lois du marché dont les changements ne peuvent être

évités; **de nombreuses entreprises traditionnellement associées au logiciel propriétaire ont déjà franchi le pas** (au prix d'investissements importants) **pour offrir des services associés au logiciel libre, ce qui démontre que les modèles ne sont pas mutuellement exclusifs**.

Avec ce projet, **l'État décide de préserver certaines valeurs fondamentales**. Et il le décide sur la base de ses pouvoirs souverains, sans affecter par là aucune des garanties constitutionnelles. Si ces valeurs peuvent être garanties sans avoir à choisir un modèle économique donné, les effets de la loi seront plus bénéfiques encore. En tout cas, il doit rester clair que **l'État n'opte pas pour un modèle économique**; s'il s'avérait qu'il n'existe qu'un seul modèle économique capable de fournir du logiciel qui satisfasse la garantie de base de ces principes, cela relèverait de circonstances historiques et non d'une décision arbitraire en faveur d'un modèle donné.

Poursuivant votre lettre : "4. Le projet de loi impose l'utilisation de logiciel à code source ouvert sans considérer les dangers que ceci peut entraîner d'un point de vue de la sécurité, de la garantie et des possibles violations des droits de propriété intellectuelle de tiers."

Faire allusion de façon abstraite aux "dangers que ceci peut entraîner", sans spécifier un seul exemple de ces supposés dangers, dénote une méconnaissance du sujet. Aussi, permettez-moi d'illustrer quelques-uns de ces points.

Concernant la sécurité:

La sécurité nationale a déjà été évoquée dans les principes fondamentaux du projet. En termes plus précis concernant la sécurité du logiciel lui-même, il est bien connu que le logiciel (propriétaire ou libre) contient des erreurs de programmation ou "bugs" (en jargon informatique) dans ses lignes de code. De même, **il est de notoriété publique que les bugs dans le logiciel libre sont moins nombreux, et qu'ils sont réparés bien plus rapidement, que dans le logiciel propriétaire**. Ce n'est pas en vain que de **nombreux organismes publics responsables de la sécurité informatique des systèmes d'institutions de l'État dans les pays développés recommandent l'utilisation de logiciel libre dans des conditions égales de sécurité et d'efficacité.**

Il est impossible de prouver que le logiciel propriétaire est plus sûr que le libre, sauf par un examen détaillé, public et ouvert, par la communauté scientifique et les utilisateurs en général. Or, cette démonstration est impossible parce que le modèle même du logiciel propriétaire interdit cette analyse, si bien que la garantie de sécurité se base sur la parole ambiguë (mais vraisemblablement partiale) du producteur du logiciel ou de ses contractants.

Il faut se souvenir que, dans de nombreux cas, **les conditions de la licence incluent des clauses de confidentialité** [NdT : « NDA » ou non disclosure agreement] **qui interdisent aux utilisateurs de révéler ouvertement les failles de sécurité découvertes dans le produit propriétaire sous licence.**

Respect de la garantie:

Comme vous le savez parfaitement, ou pourrez le découvrir en lisant le "Contrat de Licence pour l'Utilisateur Final" [NdT : EULA] des produits dont vous commercialisez la licence, dans la très large majorité des cas, les **garanties sont limitées au remplacement du support de distribution** s'il est défectueux, mais en aucun cas elles ne prévoient de compensations pour les dommages directs ou indirects, manque à gagner, etc. si suite à un bug de sécurité dans un quelconque de vos produits, non réparé par vous, un attaquant parvenait à compromettre des systèmes cruciaux pour les services de l'État : quelle garantie, quelles réparations ou quelles compensations donneraient votre société en accord avec les conditions de votre licence ? **Les garanties du logiciel propriétaire**, comme les programmes sont livrés ``AS IS" [NdT : tel quel], ce qui veut dire dans l'état dans lequel ils se trouvent, sans aucune responsabilité additionnelle pour le fournisseur concernant sa fonctionnalité, **ne diffèrent aucunement de celles habituelles dans le logiciel libre**.

Sur la propriété intellectuelle:

Les questions de propriété intellectuelle dépassent le cadre de ce projet, et elles sont couvertes par d'autres lois spécifiques. Le modèle du logiciel libre n'implique en aucune façon l'ignorance de ces lois et en fait, la grande majorité du logiciel libre est couverte par le copyright. En réalité, la seule présence de cette question dans vos observations démontre votre confusion quant au cadre légal où vit le logiciel libre. **L'incorporation de la propriété intellectuelle d'autrui dans des travaux que l'on s'attribue par la suite n'est pas une pratique courante de la communauté du logiciel libre; en revanche, c'est malheureusement le cas sur le terrain du logiciel propriétaire.** Prenez comme exemple la **condamnation par le Tribunal de Commerce de Nanterre, France, le 27 septembre 2001, de Microsoft Corp., à 3 millions de francs en dommages et intérêts, pour violation de la propriété intellectuelle** (piratage, pour utiliser le terme malheureux que votre société utilise couramment dans ses publicités).

Vous poursuivez en disant que: "5. Le projet utilise de manière erronée les concepts du logiciel à code source ouvert, qui n'est pas nécessairement du logiciel libre ou de coût nul, aboutissant à des conclusions équivoques sur les économies pour l'État, sans une analyse des coûts et bénéfices pour étayer votre position."

Cette remarque est fausse, en principe **la gratuité et la liberté sont des concepts orthogonaux: il y a du logiciel propriétaire et onéreux (par exemple, MS Office)**, **du logiciel propriétaire et gratuit (MS Internet Explorer), du logiciel libre et onéreux (distributions RedHat, SuSE, etc. du système GNU/Linux), du logiciel libre et gratuit (Apache, OpenOffice, Mozilla), et du logiciel sous différentes modalités de licence (MySQL).**

Il est certain que **le logiciel libre n'est pas nécessairement gratuit**. Et le texte du projet ne dit pas qu'il doit l'être comme vous l'aurez bien noté après l'avoir lu. Les définitions incluses dans le projet déterminent clairement *quoi* considérer comme logiciel libre, sans jamais faire référence à la gratuité. Bien qu'il soit fait mention des économies réalisées en terme de non-paiement des licences de logiciel propriétaire, les fondements du projet mentionnent clairement les **garanties fondamentales qui doivent être préservées et la stimulation du développement technologique local**. Sachant qu'**un État démocratique doit respecter ces principes**, il ne lui reste aucune autre solution que d'**employer du logiciel dont le code source est publiquement disponible et d'échanger de l'information uniquement dans des formats standards**.

Si l'État n'employait pas de logiciel présentant ces caractéristiques, il violerait les principes républicains fondamentaux. Par chance, le logiciel libre implique en plus un coût global moindre; néanmoins, dans l'hypothèse (aisément réfutée) où il coûterait plus cher que le logiciel propriétaire, **la seule existence d'un outil logiciel libre efficace pour une fonction informatique déterminée obligerait l'État à l'utiliser**; non par force de ce projet de Loi, mais **pour les principes élémentaires** que nous avons énumérés au début et **qui émanent de l'essence même de l'État de droit démocratique.**

Vous poursuivez : "6. Il est faux de penser que le logiciel à code source ouvert est gratuit. Des études du Gartner Group (organisme étudiant le marché technologique reconnu au niveau mondial) ont révélé que le coût d'acquisition du logiciel (système d'exploitation et applications) ne représente que 8% du coût total que les entreprises et les institutions doivent assumer pour une utilisation rationnelle et réellement bénéfique de la technologie. Les autres 92% sont constitués des coûts d'installation, de déploiement, de support, de maintenance, d'administration et d'indisponibilité."

Cet argument répète celui déjà donné au paragraphe 5 et contredit en partie le paragraphe 3. Aussi nous nous en remettrons aux précédents commentaires à des fins de brièveté. Nonobstant, permettez-moi de signaler que votre conclusion est fausse d'un point de vue logique: que le coût du logiciel selon le Gartner Group ne soit que de 8% du coût total d'utilisation, n'invalide d'aucune manière l'existence de logiciel gratuit, c'est-à-dire, dont le coût de la licence est zéro.

De plus dans ce paragraphe vous indiquez fort justement que **les composants de service et les pertes pour indisponibilité forment une partie substantielle du coût total d'utilisation du logiciel**; ce qui, vous le noterez, entre en contradiction avec votre affirmation de la valeur mineure des services suggérée dans le paragraphe 3. En réalité, l'utilisation de logiciel libre contribue significativement à la diminution des coûts restants du cycle de vie du logiciel. Cette réduction de l'impact économique de l'installation, du support, etc. se note dans de nombreux domaines; d'un côté, le **modèle compétitif de services autour du logiciel libre**, dont il est **possible d'acheter le support et la maintenance auprès d'une offre variée qui rivalise sur le rapport qualité/prix. Ceci est valable pour l'installation, le déploiement, et le support, et en grande partie pour la maintenance.** En second lieu, la caractéristique de reproductibilité du modèle fait que **la maintenance effectuée pour une application est facilement réutilisable**, sans impliquer des coûts importants (c'est-à-dire, **sans payer plus d'une fois pour la même chose**) **car les modifications**, si on le souhaite, **peuvent être incorporées au patrimoine commun de la connaissance**. Troisièmement, **l'énorme coût d'indisponibilité** ("écrans bleus de la mort", code mal-intentionné tel que les virus, les vers et les chevaux de Troie, exceptions, fautes générales de protection et nombre d'autres maux connus) **est considérablement réduit par l'emploi de logiciel plus stable; et il est bien connu qu'une des vertus les plus remarquables du logiciel libre est sa stabilité.**

Vous affirmez plus loin que : "7. L'un des arguments derrière le projet de loi est la prétendue gratuité du logiciel à code source ouvert, comparée au coût du logiciel commercial, sans tenir compte qu'il existe des modalités de licence en volume qui peuvent être très avantageuses pour l'État, comme cela se fait dans d'autres pays."

J'ai déjà indiqué que ce qui est en question n'est pas le coût du logiciel, mais les principes de liberté d'information, d'accessibilité et de sécurité. Ces arguments ont été largement traités dans les paragraphes précédents, auxquels je vous prie de vous référer.

D'autre part, il existe certainement des modalités de licence en volume (malheureusement, le logiciel propriétaire ne satisfait pas les principes de base). Mais, comme vous l'avez noté dans le paragraphe immédiatement antérieur de votre lettre, cela ne permet que de réduire l'impact d'un composant qui ne pèse pas plus de 8% du coût total.

Vous poursuivez : "8. De plus, l'alternative adoptée pour le projet (i) est clairement plus coûteuse du fait des coûts élevés de la migration logicielle, et (ii) met en péril la compatibilité et la possibilité d'interopérabilité des plates-formes informatiques au sein de l'État, et entre l'État et le secteur privé, compte tenu des centaines de versions de logiciel à code source ouvert sur le marché."

Analysons votre affirmation en deux parties. Le premier argument, celui de la migration qui implique des coûts élevés, est en fait un argument en faveur du projet. En effet, **plus le temps passe et plus la migration vers une autre technologie sera onéreuse**; et dans le même temps, **les risques de sécurité associés au logiciel propriétaire augmenteront aussi**. De cette manière, l'**utilisation de systèmes et de formats propriétaires rendra l'État encore plus dépendant des fournisseurs**. Au contraire **une fois implantée la politique d'utilisation du logiciel libre** (implantation qui, certes, a un coût), **la migration d'un système vers un autre se fait facilement, puisque toutes les données sont stockées dans des formats ouverts**. D'autre part, la migration vers un environnement de logiciel ouvert n'implique pas plus de coûts que celle entre deux environnements distincts de logiciel propriétaire, ce qui invalide complètement votre argument.

Le second argument se réfère à "l'interopérabilité des plates-formes informatiques au sein de l'État, et entre l'État et le secteur privé". Cette affirmation démontre **une ignorance des mécanismes de fabrication du logiciel libre, qui ne maximise pas la dépendance de l'utilisateur par rapport à une plate-forme donnée, comme c'est habituellement le cas dans le domaine du logiciel propriétaire**. Même lorsqu'il existe plusieurs distributions d'un logiciel libre et plusieurs programmes susceptibles d'être employés pour une même fonction, **l'interopérabilité reste garantie autant par l'emploi de formats standards, exigé dans le projet, que par la possibilité de créer un logiciel interopérable à partir du code source disponible.**

Vous dites plus loin que : "9. La majeure partie du logiciel à code source ouvert n'offre pas de niveaux de service adéquats, pas plus que de garantie de fabricants reconnus pour favoriser une grande productivité de la part des utilisateurs, ce qui a conduit différentes organisations publiques à revenir sur leur décision d'utiliser du logiciel à code source ouvert et à utiliser du logiciel commercial en lieu et place."

Cette observation n'est pas fondée. Compte tenu de la garantie, votre argument est réfuté par la réponse au paragraphe 4. Concernant les services de support, il est possible d'utiliser du logiciel libre sans eux (de la même manière qu'on le fait avec du logiciel propriétaire), mais quiconque le souhaite peut obtenir du support séparément, soit de la part d'une entreprise locale, soit de sociétés internationales, de la même manière que pour le logiciel propriétaire.

D'autre part, vous contribueriez beaucoup à notre analyse si vous pouviez nous donner des informations concernant les **projets de logiciel libre *implantés* dans des entités publiques et qui ont été abandonnés en faveur de logiciel propriétaire. Nous connaissons un bon de nombre de cas où l'inverse s'est produit, mais n'avons pas d'information au sujet des cas auxquels vous faites référence.**

Vous continuez en observant que : "10. Le projet décourage la créativité de l'industrie péruvienne du logiciel, qui a un chiffre d'affaires de 40 millions de dollars US par an, exporte pour 4 millions de dollars US (10e au rang des produits d'exportation non traditionnels, plus que l'artisanat) et est une source d'emplois hautement qualifiés. Avec une loi qui incite à l'utilisation du logiciel à code source ouvert, les programmeurs de logiciel perdent leurs droits de propriété intellectuelle et leur principale source de revenus."

Il est assez clair que **personne n'est obligé de commercialiser son code sous forme de logiciel libre**. La seule chose à prendre en compte est que, **si cela n'est pas fait, on ne pourra pas le vendre au secteur public**. Ce n'est en aucun cas le principal marché pour l'industrie nationale du logiciel. Plus haut nous avions abordé quelques-unes des questions relatives à l'influence du projet sur la génération d'emplois techniques hautement qualifiés et dans de meilleures conditions de compétitivité, il n'est donc pas nécessaire d'insister sur ce point.

Ce qui suit dans votre affirmation est erroné. D'un côté, **aucun auteur de logiciel libre ne perd ses droits de propriété intellectuelle**, à moins qu'il n'ait exprimé sa volonté de placer son oeuvre dans le domaine public. **Le mouvement du logiciel libre a toujours été extrêmement respectueux de la propriété intellectuelle**, et a donné une **reconnaissance publique très large à ses auteurs. Des noms tels que ceux de Richard Stallman, Linus Torvalds, Guido van Rossum, Larry Wall, Miguel de Icaza, Andrew Tridgell, Theo de Raadt, Andrea Arcangeli, Bruce Perens, Darren Reed, Alan Cox, Eric Raymond, et bien d'autres, sont mondialement reconnus pour leurs contributions au développement de logiciel aujourd'hui utilisé par des millions de personnes partout dans le monde, alors que les noms des auteurs d'excellents composants logiciels propriétaires, demeurent dans l'anonymat**. D'un autre coté, affirmer que les revenus de droits d'auteur constituent la source principale de revenus des programmeurs péruviens est pour le moins risqué, en particulier quand on n'a apporté aucune preuve à cet effet, ni aucune démonstration de comment l'emploi de logiciel libre par l'État influencerait ces revenus.

Vous poursuivez en disant que : "11. Le logiciel à code source ouvert, puisqu'il peut être distribué gratuitement, ne permet pas de générer des revenus pour ses développeurs par le biais de l'exportation. De cette manière, on affaiblit la synergie de la vente de logiciel à d'autres pays et par conséquent la croissance de cette industrie, alors qu'au contraire les normes d'un gouvernement doivent stimuler l'industrie locale."

Cette affirmation démontre une fois de plus une méconnaissance totale des mécanismes et du marché du logiciel libre. Elle tente d'affirmer que **le marché de cession des droits non exclusifs d'utilisation à titre onéreux (vente de**

licence) est le seul possible pour l'industrie informatique alors que, comme vous l'avez signalé quelques paragraphes plus haut, il n'est en aucun cas le plus important. Les incitations, émanant de ce projet, à une meilleure offre de personnels qualifiés et à **une expérience du logiciel libre à grande échelle permettront aux techniciens nationaux de se placer à un niveau très compétitif sur le marché du travail international.**

Vous signalez plus loin que : "12. Dans le Forum on a discuté de l'importance de l'emploi de logiciel à code source ouvert dans l'éducation, sans commentaire sur le retentissant fracas de cette initiative dans un pays comme le Mexique, où précisément les fonctionnaires de l'État qui fondèrent le projet, déclarent aujourd'hui que le logiciel à code source ouvert ne permet pas d'offrir une expérience d'apprentissage aux écoliers, qu'il n'a pas eu la capacité au niveau national de fournir du support pour cette plate-forme, et qu'il n'a pas pris en compte l'intégration de la plate-forme existante dans les écoles."

Effectivement, le Mexique a fait marche arrière avec le projet Red Escolar. Cela est dû, précisément au fait que les initiateurs du projet mexicain utilisèrent le coût des licences comme principal argument, au lieu des autres raisons stipulées dans notre projet et qui sont plus fondamentales. Compte tenu de cette erreur conceptuelle, aggravée par l'absence d'appui effectif de la part du SEP (Secrétariat à l'Education Publique), ils décidèrent que l'implantation de logiciel libre dans les écoles consistait à suspendre le budget logiciel et en échange à leur envoyer un CD ROM contenant GNU/Linux. Bien sûr, ceci échoua et il ne pouvait en être autrement, de même qu'échouent les laboratoires scolaires qui utilisent des logiciels propriétaires sans disposer d'un **budget pour l'installation et la maintenance**. C'est précisément pour cela que notre projet de loi ne se limite pas à recommander l'emploi de logiciel libre, mais reconnaît la nécessité et ordonne la création d'un plan de migration viable, dans lequel l'État encadre précisément la transition technique pour bénéficier des avantages du logiciel libre.

Vous terminez par une question rhétorique : "13. Si le logiciel à code source ouvert satisfait tous les pré-requis des entités de l'État pourquoi une loi pour l'adopter ? Ne devrait-ce pas être le marché qui décide librement quels sont les produits qui donnent le plus de bénéfices ou de valeur ?".

Nous sommes d'accord sur le fait que pour le secteur privé, c'est le marché qui doit décider quel produit utiliser et il ne serait pas admissible que l'État interfère. Mais dans le secteur public, le raisonnement n'est pas le même : comme nous l'avons déjà dit, **l'État collecte, manipule et transforme de l'information qui ne lui appartient pas, mais qui lui a été confiée par les citoyens qui, par force de loi, n'ont pas d'autre choix que de le faire. En contrepartie de cette obligation légale, l'État doit mettre en oeuvre des mesures extrêmes pour sauvegarder l'intégrité, la confidentialité et l'accessibilité de ces informations. L'emploi de logiciel propriétaire soulève de sérieux doutes quant à l'accomplissement de ces missions, faute d'évidence concluante à ce propos, et par conséquent il n'est pas apte à être utilisé dans le secteur public.**

La nécessité d'une loi se fonde d'un côté sur la matérialisation des principes fondamentaux énoncés plus haut dans le domaine spécifique du logiciel ; d'un autre côté, il est un fait que l'État n'est pas une entité idéale homogène mais qu'il est composé de multiples organismes avec différents degrés d'autonomie de décision. Étant donné que l'emploi de logiciel propriétaire est inapproprié, le fait d'établir ces règles dans la loi évitera que la décision discrétionnaire d'un quelconque fonctionnaire mette en péril l'information qui appartient aux citoyens. Et, par-dessus tout, elle constitue une réaffirmation actualisée par rapport aux moyens de traitement et de communication de l'information employés aujourd'hui, du principe républicain du service public.

Conformément à ce principe universellement accepté, le citoyen a le droit de connaître toute l'information en possession de l'État qui ne soit pas couverte par une déclaration de secret conforme à la loi. Le logiciel traite de l'information et il est lui-même de l'information. Information dans un format spécial, susceptible d'être interprété par une machine pour exécuter des actions, mais sans l'ombre d'un doute information cruciale parce que le citoyen dispose d'un droit légitime de savoir, par exemple, comment se comptabilise son vote ou se calculent ses impôts. Et pour cela, il faut pouvoir accéder librement au code source et éprouver les programmes utilisés pour le comptage électoral ou le calcul des impôts.

Je vous salue avec l'expression de ma considération la meilleure, soyez assuré que mon bureau sera toujours ouvert à l'exposé de vos points de vue, à quelque niveau de détail que vous jugeriez convenable.

Veuillez agréer mes salutations distinguées,

DR. EDGAR DAVID VILLANUEVA NUÑEZ
Membre du Congrès de la République du Pérou.

Traduction et adaptation : Guy Brand <guybrand @ chimie.u-strasbg.fr>
Relectures et corrections : Cyril Chaboisseau, Georges Khaznadar,
Yves Ouvrard, Alain Riffart, Stéphane Casset.
$Id: rescon-fr.html,v 1.19 2002/05/14 07:41:26 bug Exp bug

Migration d'une entreprise vers un système libre: exemple concret.

Pourquoi Auchan a choisi Linux pour gérer ses serveurs départementaux

Le groupe Auchan a décidé de migrer l'ensemble de ses serveurs départementaux - ceux qui assurent les services de base comme la messagerie, le partage de fichiers et d'imprimantes dans ses magasins et ses zones de stocks- **de l'environnement Windows NT vers l'environnement Open Source Linux**. «Ce genre de décision n'est jamais facile à prendre, mais **Linux est aujourd'hui une alternative crédible**, vu l'étendue des fonctions et le nombre de packages stables qu'on trouve dans cet environnement, **ce qui n'était pas le cas il y a 3 ou 4 ans**», déclare Michel Barthélémy, le directeur technique d'Auchan International Technology (AIT), l'entité qui gère les infrastructures informatiques du groupe de distribution.

Chez Auchan, l'opportunité de migrer vers un environnement Open Source s'est fait jour avec l'arrivée de Windows NT4. «**La fin de vie de la gamme NT est déjà programmée par Microsoft pour 2003. Le coût d'une migration vers XP aurait été beaucoup plus élevé. La différence se chiffre en millions de francs, du seul fait des licences**. Nous étions d'autant plus motivés que notre infrastructure est majoritairement sous Unix. Cela répondait pour nous à un souci de rationnalisation», confie Michel Barthélémy.

Se libérer de la pression des éditeurs propriétaires

«Auchan a fait le **constat que ses serveurs départementaux dépendaient de «runtimes» propriétaires, ce qui ne leur permettait pas d'assurer une maîtrise parfaite de leurs solutions et des coûts**», analyse Nat Makarevitsch, le directeur technique d'Idealx, la SSII qui réalisé le prototypage de la migration pour le distributeur. «**Les licences des éditeurs propriétaires mettent les entreprises sous coupe réglée**. Et le ticket d'entrée d'une migration vers Linux est moins onéreux aujourd'hui qu'il ne le sera demain», plaide-t-il. De son côté, Michel Barthélémy constate que «le coût des licences des éditeurs propriétaires est maîtrisable si vous suivez leur politique de mise à jour. Mais ce n'est pas notre cas. Nous ne faisons une migration que tous les 3 ou 4 ans et nous trouvons de fait dans la situation la plus défavorable financièrement».

Nat Makarevitch fait le décompte des avantages et du retour sur investissement d'une telle opération de **migration vers Linux**. «En général, le backbone de l'infrastructure des entreprises est soit mainframe soit Unix. Ce type de migration leur **permet**, outre **de retrouver une maîtrise de leurs budgets, de simplifier et d'homogénéiser leur infrastructure**, de rapprocher la couche départermentale des autres ressources Unix comme les serveurs middleware, et **de favoriser la répartition et l'échange des compétences**».

Il met également l'accent sur le **gain en fiabilité des serveurs**. «Dans l'environnement Windows, chaque machine assure 1, 2 ou 3 services au maximum afin que le plantage d'un serveur n'obère pas du fonctionnement de l'ensemble des services. Cela a un coût matériel. Il faut une trentaine de micros pour assurer une cinquantaine de services sous Windows. **On peut tout à fait remplacer une quinzaine de serveurs Windows par deux serveurs Linux et assurer une meilleure continuité de service**», assure-t-il. **L'environnement Open Source est aussi plus personnalisable et offre une plus grande souplesse d'adaptation pour répondre aux besoins des utilisateurs**. La notion de **sécurité est également déterminante**. «**Les OS dérivés d'Unix sont beaucoup plus sûrs, ils ne sont pas atteints par les virus et sont plus difficiles à pirater**», rappelle Nat Makarevitch.

Un investissement relativement faible

Reste à garantir la compatibilité des plateformes matérielles et à s'assurer de l'engagement des constructeurs pour les maintenir dans cet environnement. «**La part de marché de Linux est croissante** et si cela reste plus problématique que dans le monde Windows, ça l'est de moins en moins», déclare le directeur technique d'Idealx, qui reconnaît que **3 ou 4 ans en arrière, aucun constructeur n'assurait de maintenance sous Linux**. «**Ce n'est plus vrai aujourd'hui** pour l'immense majorité d'entre eux. Des constructeurs comme **IBM, Dell, HP ou Compaq sont de plus en plus soucieux d'assurer ce service**», se réjouit-il.

La **migration vers Linux n'a pas suscité** chez Auchan **de développements spécifiques**. «C'est plus un travail d'intégration que de développement. En terme de plateforme technique, avec les services associés, on a trouvé facilement l'équivalent en packages Open Source», indique Michel Barthélémy qui refuse de communiquer l'**investissement nécessaire pour la migration**. On saura juste que le montant **est jugé relativement faible** et qu'il permettra un **retour sur investissement dès la première année**. «Nous n'avons pas à faire appel à des compétences extérieures, tout sera traité principalement en interne. Nos équipes ont les compétences de base nécessaires et sont très favorables au projet. Idealx nous fournit un package prêt à l'emploi qui ne nécessite pas de compétences particulières. Ce sont des services d'infrastructures et non d'applications», explique Michel Barthélémy. Il considère que cette étape intermédiaire offre in fine de nombreux avantages. De son côté, Nat Makarevitch affiche sa satisfaction de constater une **tendance des entreprises à aller vers Linux qui dépasse le simple intérêt souvent évoqué par les cabinets d'études** spécialisés. «Nombreuses sont les PME et les grands comptes chez qui la réflexion est très avancée», affirme-t-il.

IT managers cite security and competition when choosing a Linux system

Nick Selby Special to the International Herald Tribune
Monday, September 9, 2002

MUNICH A Cannes-based private investigator, Alain Stevens, recently **switched computer operating systems from Windows to Linux**. "It's a security issue," Stevens said. "Viruses which target Windows could send confidential documents from my machines to random people - and that could send me to prison."

Citing **cost savings, open standards and enhanced security, the German government in June reached a Linux deal with International Business Machines Corp. and SuSE Linux** AG of Germany for its local, state and federal computer infrastructure.

And as the City Council in Nottingham, England, plans a **new software application for 10,000 employee workstations**, it is seriously asking the question, "Are we going to run this on Windows or open-source, like Linux?"

Throughout Europe, companies and governments large and small have recently been asking the same thing. Information technology departments are looking at what they have and rethinking what they want.

The resulting groundswell could soon **make the Linux-based desktop more prevalent in Europe than anyone could have predicted even a year ago**. Dan Kusnetzky, an analyst for International Data Corp., said **Linux had a 3.9 percent share of desktops worldwide, outpacing Macintosh's 3.1 percent.**

Richard Heggs, Nottingham's systems analyst, described the process this way: "We're looking at **Linux as a possible replacement for Windows as council desktop standard**. It's looking favorable. Senior management is saying, 'We like this, but can it do what people say it can ?'"

The stimulus to find out has been manifold. A **new generation of user-friendly Linux products spearheaded by SuSE and MandrakeSoft** SA of France - both of which are small, as yet unprofitable companies - has eased migration.

Legislative incentives have put open-source on corporate tongue-tips. Countries including Britain, Germany, France, Italy, Norway and Malta have introduced a flurry of initiatives to give open-source software access to a level playing field - and **mandate the use of open standards for official communications**. And Microsoft Corp.'s unpopular license-fee revamping has contributed to a general re-evaluation of IT purchasing criteria: Some tech managers say their feasibility studies of Linux migration may be justified by reasoning that, at a minimum, the results are ammunition for negotiations with Microsoft.

Microsoft's Europe office would not comment. Companies still look for big names - like Microsoft's - behind any new software they might buy. Now, other **big names in computing are putting money behind Linux** products. **Sun Microsystems** Inc., which recently announced an Intel-based server pre-loaded with Sun Linux 5.0, contends that the concept of having "one folk to choke" support for an open-source product lends credibility to open-source. "The key value Sun's bringing to Linux isn't really 'on the tin,'" said Simon Tindall, volume products business manager for Sun in London, "but that we will support it directly as a vendor."

This type of Linux support means that corporate IT departments and purchasing managers, ever wary of getting stuck with something forever, can now say, "Well, Sun's providing support for it." For example, **BEA Systems Inc., IBM, Oracle Corp., SAP AG and Veritas Software Corp. have all ported their applications to run on Linux systems**. All this effort may raise costs (Linux costs typically have nowhere to go but up), but that may not be a deterring factor.

Consider StarOffice, Sun Microsystems' open-source challenge to Microsoft Office, its word-processing business software suite. Until recently, it cost nothing. Since release of version 6.0, Sun has begun charging up to $79 per license.

The price seems to make businesses trust it more, some analysts say - it is a real product with a viable revenue model, which is a lot easier to explain to your boss than a product supported only by eleemosynary efforts by some vaguely hippie-sounding "open-source community."

James Jarvie, **IT manager of the Central Scotland Police, said the £245,000** ($380,000) **they saved on licensing fees with StarOffice paid for more police on the streets**. Councils in Aberdeen and Penwrith have embraced it, and the British Office of Government Finance has now endorsed it, along with Office and Lotus's SmartSuite.

"Unless Microsoft makes significant concessions in its new Office licensing policies," **Gartner Inc. said in a research report, "StarOffice will gain at least 10 percent market share at the expense of Microsoft Office by year-end 2004**."

To stand a chance, an operating system must provide applications that allow users to seamlessly edit and exchange

documents with others (which often means "with Microsoft Office users"). **StarOffice is about 95 percent compatible with Microsoft Office** (macros don't translate, but **for everyday files it is more than adequate**). It runs on Windows, Linux and Solaris, and **since the user interface looks identical on Windows and Linux desktops, a major changeover for users would be easier**.

"Running StarOffice on Windows," said MandrakeSoft's chief executive, Jacques Le Marois, "is almost always a strategic migration choice."

Martijn Dekkers, chief enterprise architect for the prime minister's office in Malta, agrees. "The key barrier," Dekkers said, "is office suites and collaborative tools like e-mail and Web browsers. Interface similarities ease transitions between different operating systems."

Ten months ago, Malta began investigating the culture and benefits of open-source. Where big software vendors claim that open-source is unreliable, unsupported and untrustworthy, open-sourcers assert that its products are the solutions to the world's ills. The truth is perhaps neither, but **on the issue of support, Dekkers found open-source viable**.

"We have found," **Dekkers said, "that one of the major issues put forward - no support and no accountability - is false**.

"Small and large **open-source vendors offer support** which **is equal to or better than support from main commercial developers**."

While large organizations typically take a long time to weigh such issues, some smaller **businesses in Europe are switching to SuSE and MandrakeSoft for their desktops**.

Last year, SuSE implemented its SmartClient architecture on Linux for Debeka-Gruppe, a German insurance and financial services group.

More than 3,000 workstations in 230 German locations are administered from its corporate headquarters in Koblenz. Where governments deal with issues of open-source culture and monopoly-busting, small **companies indicate three main reasons for taking the plunge: reliability, security and cost**.

"**I switched**," said Mervyn Cottenden, an Essex accountant who runs two MandrakeSoft Linux machines, "**because Windows is unreliable. I can't afford to lose a client's work because a machine goes down in the middle of a job**."