# Philosophie du Libre

## À propos du logiciel libre

Le logiciel libre est une question de liberté : tout le monde devrait pouvoir être libre d'utiliser des logiciels de toutes les façons qui sont utiles socialement. Le logiciel diffère des objets matériels -- tels que des chaises, des sandwiches, et de l'essence -- en ce qu'il peut être copié et modifié beaucoup plus facilement. Ces possibilités rendent le logiciel aussi utile qu'il peut l'être; nous croyons que les utilisateurs de logiciels devraient pouvoir les utiliser.

## Qu'est-ce qu'un Logiciel Libre?

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

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

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

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

Un programme est un logiciel libre si les utilisateurs ont toutes ces libertés. Ainsi, vous êtes 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 à ce sujet avec le développeur ou tout autre entité spécifique. Dans cette liberté, il est question de *l'utilisateur*, pas du développeur ; en tant qu'utilisateur, vous êtes libre d'exécuter un programme comme bon vous semble et si vous le redistribuer à quelqu'un d'autre, cette personne est libre de l'exécuter comme bon lui semble, mais vous n'êtes pas autorisé à lui imposer vos conditions.

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

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

Un moyen important de modifier un programme est de le fusionner avec des modules ou des sous-programmes disponibles libres. Si la licence du programme indique que vous ne pouvez pas le fusionner dans un module existant, telle que la nécessité que vous soyez le détenteur du copyright de tout code que vous ajoutez, alors la licence est trop restrictive pour être qualifiée de libre.

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

Cependant, certains types de règles sur la 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 programmes, vous ne pouvez ajouter de restrictions pour retirer les libertés fondamentales au public. Cette règle ne rentre pas en conflit avec les libertés fondamentales; en fait, elle les protège.

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

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

Les règles sur la manière d'empaqueter une version modifiée sont acceptables si elles n'entravent pas votre liberté de la publier, ou votre liberté de faire et d'utiliser pour votre usage personnel des versions modifiées. Les règles disant «si vous publiez 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). Les règles qui nécessitent que le code source soit publié pour les utilisateurs pour les versions que vous rendez publiques sont aussi acceptables. Il est également acceptable que la licence l'exige, si vous avez distribué une version modifiée et qu'un développeur précédent vous en demande une copie, vous devez lui envoyer, ou que vous indiquiez vos modifications.

Dans le projet GNU, nous utilisons le «copyleft» 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.

Lisez Les catégories de Logiciel Libre (18k), où sont décrites les relations entre «logiciel libre», «logiciel copylefté» et les autres catégories de logiciel.

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.

La plupart des licences de logiciels sont basées sur le droit d'auteur, or les types d'exigences que le droit d'auteur peut imposer ont des limites. Si une licence basée sur le droit d'auteur respect la liberté de la manière décrite plus haut, il est improbable que nous ayons une autre sorte de problème que nous n'ayons pas anticipé (bien que cela arrive parfois). Cependant, certaines licences de logiciels sont basées sur le droit du contrat et les contrats impliquent un champ bien plus large de restrictions. Cela signifie qu'il y a bien plus de possibilités pour qu'une licence de ce type puisse restreindre de manière inacceptable la liberté des utilisateurs et ainsi devenir non libre.

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

Quand vous parlez des logiciels libres, il est préférable de ne pas utiliser de termes comme «donner» ou «gratuit», car ils laissent supposer que la finalité des logiciels libres est le prix et non la liberté. Certains termes répandus comme «piratage» comportent des idées auxquelles nous espérons que vous n'adhérerez pas. Lisez Termes prêtant à confusion, que vous devriez éviter 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 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 inacceptables, nous la rejetons même si nous n'avons pas anticipé le problème dans ces critères. Quelquefois les conditions d'une licence soulève un problème qui nécessite des réflexions intenses, y compris des discussions avec un juriste, avant que nous puissions décider si la condition est acceptable. Quand nous arrivons à une conclusion concernant un nouveau problème, nous mettons souvent à jour ces critères pour rendre plus facile le fait de savoir si une licence est une licence logicielle libre ou non.

Si vous voulez savoir si une licence spécifique est définie comme «libre», reportez-vous à notre 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>.

Si vous envisager d'écrire une nouvelle licence, veuillez contacter la FSF en écrivant à cette adresse. La prolifération de différentes licences de logiciels libres implique un travail grandissant pour les utilisateurs dans la

compréhension des licences; nous pouvons vous aider à trouver une licence existante de logiciel libre qui réponde à vos besoins.

Si ce n'est pas possible, si vous avez vraiment besoin d'une nouvelle licence, avec notre aide, vous pouvez vous assurer que la licence est vraiment une licence de logiciel libre et éviter divers problèmes pratiques.

### Open Source?

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

# Pourquoi les logiciels ne doivent pas avoir de propriétaire

par **Richard Stallman**

**Les techniques numériques de l'information contribuent à l'intérêt général en rendant plus commodes la copie et la modification de l'information. Les ordinateurs apportent la promesse de faciliter ces opérations pour tous.**

Tout le monde ne veut pas de cette simplification. Le système du droit de copie attribue aux programmes informatiques des «propriétaires», qui pour la plupart souhaitent en garder pour eux les bénéfices potentiels et non les ouvrir au public. Ils veulent être seuls à pouvoir copier et modifier les logiciels que nous utilisons.

Le système du droit de copie s'est développé en même temps que l'imprimerie, une technique de copie à grande échelle. Le droit de copie était adapté à cette technologie parce qu'il ne limitait que la copie à grande échelle. Il ne privait pas les lecteurs de livres de leurs libertés : le lecteur moyen ne possédait pas de presse à imprimer, et il lui arrivait de recopier des livres avec sa plume et son encrier. Les lecteurs ne se voyaient pas traînés devant les tribunaux parce qu'ils avaient ainsi recopié des livres.

Les techniques numériques sont plus souples que la presse d'imprimerie. Une fois sous forme numérique, il devient facile de recopier l'information pour en faire profiter d'autres personnes. Cette souplesse place le support numérique en porte-à-faux dans un système comme le droit de copie. C'est pour cette raison que de plus en plus souvent des mesures sévères et désagréables sont prises afin de renforcer le droit de copie pour les logiciels. Par exemple les quatre pratiques suivantes de l'Association des Éditeurs de Logiciels (Software Publishers Association, SPA) :

- Une propagande massive clamant qu'il est mal de désobéir aux propriétaires afin d'aider ses amis.
- Une incitation à la dénonciation de ses camarades ou de ses collègues.
- Des visites surprises, avec l'aide de la police, dans les bureaux et dans les écoles, au cours desquelles on exige des personnes la preuve qu'elles sont innocentes du délit de copie illégale.
- Des poursuites menées par le gouvernement américain à la demande de la SPA, à l'encontre de personnes comme David LaMacchia, du , non pour avoir recopié un logiciel, mais simplement pour avoir laissé des moyens de recopie sans surveillance et ne pas avoir réussi à en empêcher l'utilisation.

Ces pratiques rappellent toutes les quatre celles de l'ancienne Union Soviétique. Sous ce régime à présent aboli, chaque photocopieuse était gardée pour empêcher la copie interdite, et les individus étaient obligés de recopier l'information en secret et de la diffuser de la main à la main sous forme de «samizdats». Évidemment, les motifs de cette restriction n'étaient pas les mêmes : en Union Soviétique ils étaient politiques, aux États-Unis c'est le profit. Mais ce sont les mesures qui nous affectent, et non pas les motifs. Toute tentative de blocage de la diffusion de l'information pour quelque raison que ce soit conduit aux mêmes méthodes et à la même brutalité.

Les propriétaires ont inventé divers arguments pour justifier leur prise de contrôle de la manière dont nous utilisons l'information :

- Les insultes.

  Les propriétaires emploient des expressions péjoratives comme «pirate» ou «vol» en les associant à une terminologie plus technique comme «propriété intellectuelle» ou «préjudice». Ils conduisent ainsi le public à penser comme ils le veulent, par une analogie simpliste entre les programmes d'ordinateurs et les objets du monde physique.

  Nos idées et nos intuitions sur la propriété des objets matériels se rapportent à la question de savoir s'il est juste d'*emporter un objet* qui appartient à quelqu'un d'autre. Elles ne s'appliquent pas directement à la *recopie* de quelque chose. Mais les propriétaires nous demandent de les appliquer quand même.

- L'exagération.

  Les propriétaires disent subir des «dommages» ou des «pertes économiques» du fait que les utilisateurs recopient eux-mêmes les programmes. Pourtant le fait de la copie n'a aucun effet direct pour le propriétaire et ne fait de mal

à personne. Le propriétaire ne subit une perte que dans la mesure où la personne qui fait cette copie aurait été prête à payer au propriétaire le prix d'un autre exemplaire.

Or en y réfléchissant un petit peu, on conclut vite que la plupart de ces personnes n'auraient pas acheté le logiciel. Ce qui n'empêche nullement les propriétaires de calculer leurs «pertes» comme si toutes ces personnes avaient été des acheteurs potentiels. Le moins qu'on puisse dire c'est qu'ils exagèrent.

- Le droit.

  Les propriétaires parlent souvent des dispositions légales et des pénalités dont ils peuvent nous menacer. Implicitement, ils veulent nous dire là que les lois d'aujourd'hui reflètent un point de vue moral incontestable, et en même temps nous invitent à considérer les pénalités encourues comme des faits de nature, dont personne ne porte la responsabilité.

  Ce type d'argumentation n'a pas été taillé pour résister au raisonnement critique mais pour venir renforcer une pensée routinière.

  En aucune façon les lois ne sont des arbitres du bien et du mal. Tout Américain devrait savoir qu'il y a quarante ans, dans de nombreux États, il était illégal pour un Noir de s'asseoir à l'avant d'un autobus. Cependant seuls les racistes diront que c'était mal de le faire.

- Les droits naturels.

  Souvent les auteurs revendiquent leur attachement affectif aux programmes qu'ils ont écrits et nous en font déduire que leurs désirs et leurs intérêts au sujet de ces programmes sont plus importants que ceux de toutes les autres personnes, plus importants même que ceux du monde entier. Il faut ici remarquer que la plupart du temps ce sont les sociétés et non les auteurs qui détiennent les droits de copie sur les logiciels, mais nous sommes censés négliger cette incohérence.

  À ceux qui énoncent comme un axiome moral l'idée que l'auteur est plus important que le public, je peux seulement répondre que pour ma part, bien qu'auteur de logiciels très connu, je dis que c'est du chiqué.

  Si les gens ont tellement tendance à sympathiser avec les arguments du type «droits naturels», c'est généralement pour l'une des deux raisons suivantes.

  La première de ces raisons repose sur une analogie abusive avec les objets matériels. Si je prépare un plat de spaghetti, cela ne me va pas me convenir que quelqu'un d'autre que moi les mange, parce qu'alors je ne pourrai plus les manger moi-même. En mangeant mes spaghetti, l'autre personne me cause un dommage dans l'exacte mesure de son bénéfice à elle. D'elle ou de moi, une seule personne peut avoir les spaghetti, la question est de savoir qui. La plus petite distinction entre nous deux suffit à faire pencher la balance morale.

  Mais le cas d'un programme que j'ai écrit est très différent. Si vous le faites fonctionner ou si vous le modifiez, cela vous profite directement mais ne m'affecte que d'une manière indirecte. Le fait que vous en donniez ou non une copie à un ami vous profite beaucoup plus, à vous et à votre ami, qu'il ne me dérange, moi. Il ne faut pas que j'aie le pouvoir de vous en empêcher. Personne ne doit avoir ce pouvoir.

  Deuxièmement, les gens ont entendu dire que les droits naturels des auteurs constituaient une tradition universellement acceptée et jamais remise en question par notre société.

  Pourtant historiquement c'est l'inverse qui est vrai.

  Au moment de la rédaction de la Constitution des États-Unis, l'idée de droits naturels pour les auteurs fut proposée mais finalement rejetée. C'est pourquoi la Constitution se borne à *autoriser* les systèmes de droit de copie, sans toutefois les rendre *obligatoires*. La Constitution énonce aussi que de tels sytèmes doivent être temporaires, que leur but est de favoriser le progrès et non pas de récompenser les auteurs. Le droit de copie récompense certes les auteurs, et les éditeurs encore plus, mais toujours en tant que moyen d'infléchir leur comportement.

  La véritable tradition de notre société, c'est que le droit d'auteur empiète sur les droits naturels du public et ne peut se justifier que dans l'intérêt de ce même public.

- L'Économie.

  Finalement, on nous dit qu'il faut aux logiciels des propriétaires parce que cela encourage leur production.

  Contrairement aux autres arguments, celui-ci repose du moins sur une approche légitime du sujet. Le but est valable en effet: satisfaire les utilisateurs de logiciels. L'expérience montre que les gens produiront plus de logiciels s'ils sont mieux payés pour le faire.

  Mais cet argument économique a le défaut d'être basé sur l'hypothèse discutable que la différence se fera en fonction de l'argent que nous voulons bien donner. On tient pour acquis que ce que nous voulons c'est de la «production de logiciels», avec ou sans propriétaire.

  Le public accepte facilement cette hypothèse parce qu'elle correspond à son expérience des objets matériels. Si nous prenons l'exemple du sandwich, on peut sans doute réussir à en trouver deux qui soient identiques, l'un gratuit, l'autre payant. Si c'est le cas la somme payée ou non est la seule différence entre les deux. Que vous deviez

ou non le payer, le sandwich aura le même goût, la même valeur nutritive et en tous cas vous ne pourrez le manger qu'une seule fois. Le fait le sandwich vous ait été fourni, ou non, par un propriétaire n'a d'influence directe que sur la somme d'argent dont vous disposez par la suite.

Ceci est vrai pour tout type d'objet matériel. Le fait qu'ils aient ou non un propriétaire ne change pas leur *nature*, ni leur utilité au cas où vous en faites l'acquisition.

Pour un programme c'est différent. Le fait qu'il ait un propriétaire modifie nettement sa nature et ce que vous pouvez en faire si vous en achetez un exemplaire. Et pas seulement pour une question d'argent, car le système de propriétaires de logiciels encourage ces derniers à produire un bien qui n'est pas celui dont la société a besoin. Il est la cause d'une pollution morale imperceptible qui nous affecte tous.

En effet, de quoi la société a-t-elle besoin ? D'une information vraiment disponible pour ses citoyens. Par exemple, des programmes que les gens peuvent réparer, modifier, adapter, améliorer, et pas seulement faire marcher. Or ce que les propriétaires de logiciels livrent généralement est une boîte noire que personne ne peut étudier ni modifier.

La société a aussi besoin de liberté. Quand un programme a un propriétaire, ses utilisateurs perdent la liberté de contrôler un des aspects de leur vie.

Surtout, la société a besoin d'encourager l'esprit de coopération volontaire de ses citoyens. Quand les propriétaires de logiciels qualifient de «pillage» le fait d'aider notre voisin d'une façon simple et naturelle, ils polluent l'esprit civique de notre société.

C'est pourquoi nous disons que le logiciel libre est une question de liberté, pas de prix.

L'argument économique des propriétaires est un faux argument, mais le problème économique est un vrai problème. Certaines personnes écrivent des logiciels utiles pour le plaisir ou pour conquérir l'admiration et la reconnaissance, mais si nous voulons plus de logiciels que ceux que ces personnes écrivent il nous faut récolter des fonds.

Depuis maintenant dix ans, les développeurs de logiciels libres essaient, avec un certain succès, diverses méthodes pour trouver des financements. Il n'est pas indispensable pour cela d'enrichir quelqu'un. Le revenu d'une famille américaine moyenne, autour de 35 mille dollars, a fait la preuve de sa capacité suffisante comme stimulant pour beaucoup de métiers moins satisfaisants que la programmation.

Pendant des années, jusqu'à ce que la création d'une association le rende superflu, j'ai gagné ma vie avec les améliorations que je faisais ponctuellement aux logiciels que j'avais écrits. Chacune de ces améliorations était ajoutée à la version livrée en standard, devenant ipso facto disponible au public. Les clients me payaient pour travailler aux améliorations dont ils avaient besoin et qui ne coïncidaient pas forcément avec les fonctionnalités que j'aurais autrement considérées comme prioritaires.

La Free Software Foundation (FSF), une association exemptée d'impôt pour le développement du logiciel libre, récolte des fonds par la vente de CD-ROMs GNU, de T-shirts, de manuels et de distributions de luxe - que chaque utilisateur a la possibilité de copier ou de modifier librement, ainsi que par les dons. Nous avons maintenant cinq programmeurs et l'équipe compte aussi trois employés pour traiter les commandes par courrier.

Certains développeurs de logiciels libres gagnent leur vie en vendant des services de support technique. Cygnus Support, avec 50 employés [au moment de la rédaction de cet article], estime que 15 pourcent de l'activité de ses équipes est consacrée au développement de logiciels libres -- pourcentage fort respectable pour une société de logiciels.

D'autres sociétés, parmi lesquelles Intel, Motorola, Texas Instruments et Analog Devices, se sont regroupées pour financer le développement du compilateur libre GNU pour le langage C. Dans le même temps, le compilateur libre GNU pour le langage Ada est soutenu financièrement par la US Air Force, car celle-ci pense que c'est le moyen le plus économique d'obtenir un compilateur de haute qualité. [Les subventions de US Air Force ont cessé il y a quelque temps; le compilateur GNU pour Ada est maintenant en service et sa maintenance est subventionnée par des sociétés commerciales.]

Tous ces exemples sont à petite échelle. Le mouvement du logiciel libre est encore petit et encore jeune. Mais l'exemple de la radio financée par les auditeurs dans ce pays [les États-Unis] montre qu'il est possible de soutenir une activité importante sans forcer chaque utilisateur à payer.

En tant qu'utilisateur d'ordinateur aujourd'hui, vous utilisez peut-être un programme propriétaire. Si votre ami vous en demande une copie, ce serait mal de refuser. La coopération est plus importante que le droit de copie. Cependant, dans le fond, la coopération en circuit fermé ne permettra pas de bâtir une bonne société. Chacun devrait aspirer à une vie honnête menée ouvertement et avec fierté, et pour cela il faut dire «non!» au logiciel propriétaire.

Vous méritez de pouvoir coopérer ouvertement et librement avec les autres personnes qui utilisent les logiciels. Vous méritez de pouvoir apprendre comment marche un logiciel et de l'enseigner à vos élèves. Vous méritez de pouvoir engager votre programmeur favori pour réparer le logiciel quand il est cassé.

Vous méritez le logiciel libre.

# Pourquoi les logiciels devraient être libres

par **Richard Stallman** **(Version du 24 avril 1992)**

## Introduction

L'existence de logiciels soulève forcément la question de la façon dont devraient être prises les décisions concernant leur usage. Par exemple, supposons qu'une personne ayant une copie d'un programme, rencontre une autre personne qui en voudrait une copie. Il est possible pour eux de copier le programme; qui devrait décider si c'est possible? Les personnes elles-mêmes? Ou une tierce personne, son «propriétaire»?

Les développeurs de logiciels considèrent typiquement que le critère de réponse supposé est la maximisation de leurs profits. Le pouvoir politique des affaires a poussé le gouvernement à adopter à la fois ce critère et la réponse proposée par les développeurs : c'est-à-dire qu'un programme a un propriétaire, en général une société associée à son développement.

J'aimerais considérer la même question en utilisant un critère différent : la prospérité et la liberté du public en général.

Cette réponse ne peut pas être décidée par la loi actuelle - la loi devrait se conformer à la morale, pas l'inverse - ni par l'usage, bien qu'ils puissent suggérer des réponses possibles. La seule façon d'en juger, est d'observer qui bénéficie et qui est lésé en reconnaissant un propriétaire au logiciel, pourquoi et dans quelle mesure. En d'autres termes, nous devrions en évaluer le pour et le contre pour la société dans son ensemble, en prenant en compte aussi bien la liberté individuelle que la production de biens matériels.

Dans cet article, je décrirai les effets d'avoir des propriétaires, et je montrerai que les résultats sont préjudiciables. Ma conclusion est que les programmeurs ont le devoir d'encourager les autres à partager, distribuer, étudier et améliorer les logiciels que nous écrivons : autrement dit, d'écrire des logiciels libres.(1)

## Comment les propriétaires justifient leur pouvoir

Ceux-là, qui bénéficient du système actuel où les programmes sont propriétaires, proposent deux arguments en leur faveur : l'argument affectif et l'argument économique.

L'argument affectif ressemble à ceci : «j'ai mis ma sueur, mon coeur, mon âme dans ce programme. Il vient de *moi*, c'est *le mien*!»

Cet argument ne nécessite pas de réfutation sérieuse. Les programmeurs peuvent cultiver ce sentiment de possession quand ça les arrange; ce n'est pas inévitable. Considérez, par exemple, comment ces mêmes programmeurs cèdent volontiers leurs droits à une grosse entreprise moyennant salaire; mystérieusement, l'attachement affectif disparaît. Faites le contraste avec ces grands artistes et artisans des temps médiévaux, qui ne signaient même pas de leur nom leurs travaux. Pour eux, le nom de l'artiste n'avait pas d'importance. Ce qui importait, c'était que le travail, et son but afférent, aient été faits. Cette façon de voir à prévalu pendant des centaines d'années.

L'argument économique est du style : «je veux devenir riche» (ce qui en général, se dit incorrectement «je veux gagner ma vie»), et si vous ne me permettez pas de devenir riche en programmant, eh bien je ne programmerai pas. Comme tout le monde me ressemble, personne n'écrira de programmes. Et vous serez coincés, car pas de programme du tout!» Cette menace est généralement déguisée en conseil amical venant de la bouche d'un sage.

J'expliquerai plus tard pourquoi cette menace est du bluff. J'aimerai d'abord mettre le doigt sur une supposition implicite qui est plus évidente dans une autre formulation de l'argument.

Cette formulation part de la comparaison entre l'utilité sociale d'un programme propriétaire et celle où il n'y a pas de programme, pour alors conclure que le développement de logiciels propriétaires est, globalement, bénéfique et qu'il devrait être encouragé. L'erreur, ici, vient de ne comparer que deux résultats - logiciel propriétaire contre pas de logiciel - et de supposer qu'il n'y a pas d'autres possibilités.

Dans un système reconnaissant la propriété intellectuelle, le développement logiciel est habituellement lié à l'existence d'un propriétaire qui contrôle l'utilisation du logiciel. Tant que ce lien existe, nous devons souvent faire face au choix d'un programme propriétaire ou pas de programme. Cependant, ce lien n'est pas inhérent ou inévitable; c'est une conséquence d'une décision politique, légale et sociale spécifique, que nous contestons : la décision qu'il y ait des propriétaires. Formuler le choix entre logiciel propriétaire contre pas de logiciel, c'est faire une pétition de principe.

## L'argument contre le fait qu'il y ait des propriétaires

La question qu'il faut poser, c'est : «Est-ce que le développement de logiciels doit être lié à un propriétaire qui en restreint l'usage?»

Pour pouvoir en décider, il nous faut juger chaque effet des deux activités suivantes sur la société, *indépendamment* : l'effet du développement de logiciels (indépendamment de ses termes de diffusion), et l'effet de la restriction de son emploi (supposant que le logiciel ait été développé). Si l'une de ces activités est utile et l'autre nocive, nous devrions les dissocier et nous lancer uniquement dans la première.

En d'autres termes, si restreindre la distribution d'un logiciel déjà développé est préjudiciable à la société dans son ensemble, alors un développeur moral rejettera cette activité.

Pour déterminer l'effet de la restriction du partage, nous avons besoin de comparer la valeur, pour la société, d'un programme restreint (par ex. propriétaire) avec le même programme, mais disponible pour tout le monde. Ce qui signifie comparer deux mondes possibles.

Cette analyse met aussi en exergue le simple contre-argument qui est parfois soulevé que «le bénéfice pour le voisin en lui donnant une copie d'un logiciel est annulé par le préjudice fait au propriétaire». Ce contre-argument suppose que les inconvénients et bénéfices sont équivalents dans leur ampleur. L'analyse implique de comparer ces étendues et elle montre que les bénéfices sont bien plus importants.

Pour mettre en lumière cet argument, prenons un autre domaine d'application : la construction routière.

Il serait possible de financer la contruction de toutes les routes avec des péages. Ce qui entraînerait d'avoir des postes de péage à tous les coins de rues. Un tel système inciterait grandement à améliorer l'état des routes. Il aurait aussi comme vertu de faire payer l'usager de la route concernée. Cependant, le péage n'est qu'une entrave artificielle à la fluidité du trafic, artificielle dans le sens où elle n'est pas une conséquence du fonctionnement des routes et des voitures.

Si on compare les routes avec ou sans péage, nous voyons que (si tout se déroule normalement) les routes sans péage sont meilleur marché à construire et à maintenir, plus sûres et plus efficaces à emprunter. (2) Dans les pays pauvres, les postes de péage rendent les routes inaccessibles à bien des citoyens. Les routes sans péage offrent ainsi plus de bénéfices à moindre coût ; elles sont préférables pour la société. C'est pourquoi la société devrait trouver d'autres moyens de financer les routes, sans recourir aux péages. L'usage des routes, une fois construites, devrait être libre.

Quand les partisans des postes de péage les proposent comme étant *simplement* une façon de lever des fonds, ils déforment le choix offert. Les postes de péage, effectivement, permettent de récolter des fonds, mais elles occasionnent aussi autre chose : elles dégradent les routes. La route à péage n'est pas aussi bonne que celle sans péage; nous donner plus de routes ou supérieures sur le plan technique, n'est peut-être pas une amélioration si cela signifie substituer aux routes gratuites des routes à péage.

Bien entendu, construire des routes sans péage coûte de l'argent, que, d'une façon ou d'une autre, le public doit payer. Mais ce n'est pas ce qui, inévitablement, implique des postes de péage. Nous, qui dans un cas comme dans l'autre, devrons payer, aurons tout intérêt pour notre porte-monnaie à acheter des routes sans péage.

Je ne suis pas en train de dire que les routes à péage sont pires que pas de route du tout. Ce serait vrai si les péages étaient tels que presque personne n'emprunterait la route - mais ce n'est pas une politique plausible pour un collecteur de péage. Cependant, tant que les postes de péage causeront des gaspillages et des inconvénients significatifs, il est plus avantageux de lever des fonds de façon moins obstructive.

Pour appliquer ce même argument au développement logiciel, je vais maintenant montrer que d'avoir des «postes de péage» sur des logiciels utiles, coûte cher à la société : cela rend le programme plus coûteux à élaborer, plus cher à distribuer et moins satisfaisant et efficace à utiliser. Je poursuivrai en disant que la construction du programme devra être encouragée autrement. Puis je m'attacherai à présenter d'autres méthodes d'encouragement et de financement du développement logiciel (dans la mesure réellement nécessaire).

## Le tort fait en entravant le logiciel

Considérez un instant un programme qui a été développé et dont tous les financements nécessaires à son élaboration ont été pris ; maintenant, la société doit faire le choix entre le rendre propriétaire ou le rendre libre d'utilisation et de partage. Supposez qu'il est désiré que ce programme existe et soit disponible.(3)

Les restrictions sur la distribution et la modification du programme ne facilitent pas son utilisation. Elles ne peuvent qu'interférer. Ainsi, l'effet ne peut avoir qu'un impact négatif. Mais jusqu'à quel point? Et de quelle manière?

On distingue trois niveaux de préjudice matériel dans ce genre d'entrave :

- Moins de gens utilisent le programme.
- Aucun des utilisateurs ne peut adapter ou corriger le programme.
- Les autres développeurs ne peuvent rien apprendre du programme ou encore démarrer un nouveau travail en se basant dessus.

Chaque niveau de préjudice matériel est concomitant à un préjudice psycho-social. Cela se réfère aux effets qu'ont les décisions des gens sur leurs sentiments, attitudes et prédispositions futurs. Ces changements dans la façon de penser des gens auront par la suite un effet sur leurs relations avec leurs concitoyens et peuvent avoir des conséquences matérielles.

Les trois niveaux de préjudice matériel gaspillent une partie de la valeur que le programme pourrait offrir, mais ne peuvent la réduire à zéro. S'ils gaspillent la presque totalité de la valeur du programme, alors l'écriture du programme cause du tort à la société, au plus à hauteur de l'effort qu'il a fallu fournir pour écrire ce programme. En effet, un programme dont la vente génère des profits fournit sûrement un net bénéfice matériel direct.

Cependant, tenant compte des préjudices psycho-sociaux concomitants, il n'y a pas de limite au préjudice que peut

provoquer le développement d'un logiciel propriétaire.

### L'entrave à l'utilisation de programmes

Le premier niveau de préjudice gêne le simple emploi d'un programme. Une copie d'un programme a pratiquement un coût marginal de zéro (et ce prix, vous pouvez le payer en faisant le travail vous-même), ce qui veut dire que, dans un marché libre, elle aurait un prix avoisinant zéro. Une taxe via une licence est un découragement significatif à l'utilisation d'un programme. Si un logiciel largement utile est propriétaire, beaucoup moins de gens l'utiliseront.

Il est facile de montrer que la contribution totale d'un programme à la société est réduite si on lui assigne un propriétaire. Chaque utilisateur potentiel du logiciel, face à la nécessité de le payer pour l'utiliser, peut choisir de payer ou de renoncer à son usage. Si un utilisateur fait le choix de payer le programme, il y a un simple transfert de richesses entre deux parties. Mais chaque fois qu'une personne choisit d'outrepasser l'utilisation du programme, cela cause du tort à cette personne sans que quelqu'un y trouve son compte. La somme de nombres négatifs avec zéro, ça doit être négatif...

Mais cela ne réduit pas la somme de travail qu'il faut pour *développer* le programme. Donc l'efficacité du processus, calculée en divisant la satisfaction des utilisateurs par le coût du développement, en est diminuée.

Ceci reflète la différence cruciale entre la copie de programmes et celle de voitures, de chaises ou de sandwiches. À part dans la science-fiction, il n'existe pas de machine pouvant reproduire les objets matériels. Mais les programmes sont faciles à copier; n'importe qui peut faire autant de copies que nécessaire, sans gros effort. Ce qui n'est pas vrai dans le cas des objets matériels, vu que la matière est conservée : chaque copie nécessite des matières premières, tout comme la première.

En ce qui concerne les objets matériels, décourager leur usage est logique, car moins d'objets signifie moins de matières premières, moins de travail pour les construire. C'est vrai qu'il faut un coût pour démarrer, pour développer, coût réparti sur l'ensemble de la production. Mais tant que le coût marginal est significatif, y ajouter un pourcentage des coûts de développement ne provoque pas de différence qualitative. Et cela ne nécessite pas de restrictions sur la liberté des utilisateurs.

Cependant, imposer un prix à quelque chose qui, autrement, aurait pu être gratuit, c'est un changement qualitatif. Une taxe imposée, centralisée, sur la distribution de logiciels devient fortement décourageante.

De plus, la production centrale, telle qu'elle est pratiquée de nos jours, est inefficace, y compris dans son rôle de fournisseur de copies de logiciels. Ce système implique d'empaqueter des disquettes ou des bandes dans un emballage superflu, de les envoyer en nombre dans le monde entier puis de les stocker avant leur vente. Ces coûts sont présentés comme étant le prix à payer pour faire des affaires; en vérité, ils font partie du gaspillage causé par la présence d'un propriétaire.

### Endommager la cohésion sociale

Supposons que vous et votre voisin trouviez utile de faire tourner un certain programme. Dans un souci moral pour votre voisin, vous devriez sentir que la façon correcte d'appréhender la situation, est de permettre une utilisation de ce logiciel par vous deux. S'il n'y avait la possibilité que pour un seul d'entre vous de faire tourner ce programme, il y aurait division; ni vous, ni votre voisin ne trouveriez cela acceptable.

Signer un accord de licence logicielle typique revient à trahir votre voisin : «je fais la promesse de priver mon voisin de ce programme, ainsi je peux en avoir une copie pour moi-même». Les gens qui font de tels choix ressentent une pression psychologique interne pour se justifier, en diminuant l'importance d'aider leur voisin - c'est comme cela que le sens civique souffre. C'est un préjudice psycho-social associé au préjudice matériel de décourager l'utilisation du logiciel.

Beaucoup d'utilisateurs reconnaissent inconsciemment le tort de refuser le partage; ils décident alors d'ignorer les licences et les lois, et de partager tout de même les programmes. Mais ils s'en sentent souvent coupables. Ils savent qu'ils doivent enfreindre la loi pour être de bons voisins, mais ils continuent de penser que les lois font autorité et concluent qu'être un bon voisin (ce qu'ils sont), c'est vilain et honteux. C'est aussi une forme de préjudice psycho-social, mais on peut y échapper en prenant la décision que ces licences et ces lois n'ont pas de force morale.

Les programmeurs souffrent aussi de préjudices psycho-sociaux, sachant que bien des usagers ne pourront utiliser leurs travaux. Ceci conduit à une attitude de cynisme ou de dénégation. Un programmeur peut faire une description enthousiaste du travail qu'il pense techniquement excitant; puis, quand on lui demande «Est-ce qu'il me sera permis de l'utiliser?», son visage se ferme et il doit bien admettre que non. Pour éviter de se sentir découragé, soit il ignore ce fait la plupart du temps, soit il adopte une attitude cynique afin d'en minimiser l'importance.

Depuis la période reaganienne, la plus grande pénurie, aux États-Unis, ce n'est pas l'innovation technique, mais plutôt la volonté de travailler ensemble pour le bien public. Cela n'a pas de sens d'encourager l'un au détriment de l'autre.

### L'entrave à l'adaptation sur mesure des programmes

Le deuxième niveau de préjudice matériel est l'impossibilité d'adapter les programmes. La facilité de modification du logiciel est un de ses grands avantages sur les technologies plus anciennes. Mais la plupart des logiciels commerciaux ne

peuvent être modifiés, même après les avoir achetés. Ils sont à prendre ou à laisser, comme une boîte noire, un point c'est tout.

Un programme que vous pouvez exécuter se compose d'une série de nombres dont le sens est obscur. Personne, même un bon programmeur, ne peut aisément changer les nombres afin que le logiciel exécute autre chose.

Normalement, les programmeurs travaillent sur le «code source» d'un programme, écrit dans un langage de programmation comme le Fortran ou le C. Ils utilisent des noms pour désigner les données utilisées et des parties du programme, et ils représentent les opérations par des symboles comme le «+» pour une opération ou le «-» pour une soustraction. Le langage est conçu pour aider les programmeurs à déchiffrer et modifier les programmes. Voici un exemple : il s'agit d'un programme qui calcule la distance entre deux points d'un plan :

```
float
distance (p0, p1)
     struct point p0, p1;
{
  float xdist = p1.x - p0.x;
  float ydist = p1.y - p0.y;
  return sqrt (xdist * xdist + ydist * ydist);
}
```

Voici le même programme sous sa forme exécutable, sur l'ordinateur que j'utilise habituellement :

```
1314258944      -232267772      -231844864      1634862
1411907592      -231844736      2159150         1420296208
-234880989      -234879837      -234879966      -232295424
1644167167      -3214848        1090581031      1962942495
572518958       -803143692      1314803317
```

Le code source est utile (au moins potentiellement) pour chaque utilisateur d'un programme. Mais la plupart des utilisateurs n'ont pas la permission d'avoir des copies du code source. Normalement, le code source d'un programme propriétaire est tenu secret par son propriétaire, empêchant quiconque d'en apprendre quelque chose. L'utilisateur reçoit simplement des fichiers de nombres incompréhensibles que l'ordinateur exécutera. Cela signifie que seul le propriétaire du logiciel peut changer le programme.

Un jour, une amie me dit qu'elle travaillait comme programmeur dans une banque depuis environ six mois, écrivant un programme similaire à quelque chose de commercialement disponible. Elle croyait que, si elle avait accès au code source de ce programme commercial, elle pourrait facilement l'adapter à ses besoins. La banque souhaitait payer pour cela, mais elle n'y fut pas autorisée - le code source était un secret. Elle a dû alors travailler d'arrache-pied pendant six mois, un travail qui compte pour le PNB, mais qui était en fait du gaspillage.

Le Laboratoire d'Intelligence Artificielle du MIT reçut une imprimante graphique comme cadeau de la part de Xerox, aux alentours de 1977. Elle était pilotée par un logiciel libre, auquel nous avons ajouté de nombreuses caractéristiques bien commodes. Par exemple, le logiciel avertissait immédiatement l'utilisateur de la fin du processus d'impression. Si l'imprimante venait à rencontrer un problème, comme un bourrage ou un manque de papier, le programme avertissait de suite tous ceux qui avaient des travaux d'impression en cours. Ces fonctionnalités facilitaient la vie.

Plus tard, Xerox offrit au labo d'IA une nouvelle imprimante, plus rapide, une des premières laser. Elle était pilotée par un logiciel propriétaire qui tournait sur un poste dédié et séparé, nous ne pouvions donc ajouter aucune de nos fonctionnalités favorites. On pouvait s'arranger pour envoyer un message quand le travail d'impression se lançait sur le poste dédié, mais pas quand celui-ci se faisait effectivement (et les délais étaient habituellement importants). Il n'y avait aucun moyen de savoir si l'impression était faite; il fallait deviner. Et personne n'était informé d'un bourrage papier, l'imprimante attendait ainsi souvent une heure avant d'être rechargée.

Les programmeurs système du labo d'IA étaient capables de corriger de tels problèmes, probablement tout aussi capables que les auteurs du programme. Xerox n'avait pas envie de les corriger et choisit de nous en empêcher, nous avons donc été forcés de subir les problèmes. Ils n'ont jamais été corrigés.

La plupart des bons programmeurs ont fait l'expérience de cette frustration. La banque pouvait se permettre de résoudre son problème en écrivant un nouveau programme depuis le début, mais un utilisateur lambda, quelle que soit son habileté, ne peut que laisser tomber.

Laisser tomber provoque un préjudice psycho-social - sur l'esprit d'indépendance. C'est démoralisant d'habiter une maison qu'on ne peut réarranger selon ses besoins. Cela conduit à la résignation et au découragement, ce qui peut gagner d'autres aspects de la vie. Les gens qui se sentent ainsi ne sont pas heureux et ne font pas du bon travail.

Imaginez ce que ce serait si les recettes étaient logées à la même enseigne que les logiciels. Vous vous diriez «Voyons, comment modifier cette recette pour qu'il n'y ait plus de sel?», et le chef cuistot de vous répondre «Comment oses-tu insulter ma recette, fruit de mon cerveau et de mon palais, en tentant de la modifier? Tu n'as pas le jugement pour changer ma recette afin qu'elle marche mieux.»

«Mais mon docteur m'a recommandé de ne pas manger salé. Que puis-je faire? Pouvez-vous en ôter le sel pour moi?»

«Je serais heureux de le faire; mes honoraires ne sont que de 50000 dollars». À partir du moment où le propriétaire a le monopole sur les modifications, les honoraires tendent à gonfler. «De toute façon, je n'ai pas le temps maintenant. Je

suis pris par une commission afin de créer une nouvelle recette de biscuits marins pour le Département de la Marine. Je reprendrai contact avec toi d'ici à peu près deux ans».

### L'entrave au développement logiciel

Le troisième niveau de préjudice matériel touche le développement logiciel. Il était autrefois un processus évolutif, où quelqu'un prenait un programme existant et en réécrivait une partie pour ajouter une nouvelle fonctionnalité; puis une autre personne en réécrivait aussi une partie pour y ajouter une autre fonctionnalité. Dans certains cas, cela a continué ainsi sur une période d'une vingtaine d'années. Entre-temps, certaines parties du programme auront été «cannibalisées» pour créer les prémices d'autres programmes.

L'existence de propriétaires empêche ce genre d'évolution, rendant nécessaire de repartir de rien si on veut développer un programme. Cela empêche également les nouveaux praticiens d'étudier les programmes existants pour en apprendre des techniques utiles ou même apprendre comment on structure de gros programmes.

Les propriétaires entravent aussi l'éducation, l'apprentissage. J'ai rencontré de brillants étudiants en informatique qui n'avaient jamais vu le code source d'un gros logiciel. Ils peuvent être bons à écrire de courts programmes, mais ils ne peuvent commencer à apprendre les techniques différentes de l'écriture d'un vaste programme, s'ils ne peuvent observer comment d'autres l'ont fait.

Dans tout domaine intellectuel, on peut atteindre de plus grandes hauteurs en se tenant sur les épaules des autres. Mais ce n'est généralement plus permis dans le domaine logiciel - vous ne pouvez vous tenir sur les épaules que de ceux qui font partie de *votre propre compagnie*.

Le préjudice psycho-social qui s'y rattache affecte l'esprit de coopération scientifique, qui était autrefois si fort que les scientifiques coopéraient même quand leurs pays étaient en guerre. C'est dans cet esprit que les océanographes japonais, abandonnant leur labo dans une île du Pacifique, ont soigneusement conservé leurs travaux pour les Marines qui commençaient à débarquer, et laissèrent un mot leur demandant d'en prendre bien soin.

Les conflits de profits ont détruit ce que les conflits internationaux avaient épargné. Aujourd'hui, les scientifiques de nombreuses disciplines ne donnent pas assez de détails dans leurs publications, qui permettraient aux autres de reproduire leur expérience. Ils ne publient que ce qui permet au lecteur d'être impressionnés par l'étendue de leurs travaux. C'est particulièrement vrai pour la recherche informatique, où le code source des programmes décrits dans les publications est en général secret.

### Peu importe comment le partage est restreint

J'ai parlé des effets d'empêcher les gens de copier, de modifier ou de se baser sur un programme existant. Je n'ai pas précisé comment cette obstruction était réalisée, car cela n'affecte pas la conclusion. Que ce soit par protection contre la copie, copyright, licences, cryptage, cartes ROM ou encore un numéro de série sur le matériel, si cela *réussit* à empêcher l'utilisation, alors il y a préjudice.

Les utilisateurs considèrent certaines de ces méthodes comme plus odieuses que d'autres. Je suggère que les méthodes les plus détestées sont celles qui accomplissent leur objectif.

### Les logiciels devraient être libres

J'ai montré comment le fait d'être propriétaire d'un programme, le pouvoir de restreindre sa modification ou sa copie, est une entrave. Ses retombées négatives sont vastes et importantes. Il s'ensuit que la société devrait se passer de propriétaires de logiciels.

Une autre façon de comprendre cela, est que ce dont a besoin la société, c'est de logiciels libres et que les logiciels propriétaires ne sont qu'un pauvre substitut. Encourager le substitut n'est pas une façon rationnelle d'obtenir ce dont nous avons besoin.

Vaclav Havel nous a conseillé de «travailler pour une chose parce qu'elle est bien, pas parce qu'elle a des chances de réussir». Un marché créant des logiciels propriétaires a des chances de réussir selon son propre point de vue, mais ce n'est pas ce qui est bon pour la société.

### Pourquoi les gens développeront des logiciels

Si nous éliminons la propriété intellectuelle comme un moyen d'encourager les gens à développer des logiciels, au début, peu de programmes seront développés, mais ils seront plus utiles. Difficile de dire si la satisfaction d'ensemble des utilisateurs sera moindre. Mais si c'est le cas, ou si nous voulons malgré tout l'augmenter, il y a d'autres moyens d'encourager le développement, tout comme il y a des alternatives aux postes de péage pour tirer de l'argent des routes. Mais avant de parler de la façon dont cela peut se faire, je vais d'abord me demander dans quelle mesure un encouragement artificiel est vraiment nécessaire.

## Programmer, c'est fun

Certains domaines professionnels trouvent peu de candidats, sauf pour l'argent; la construction routière, par exemple. Il en est d'autres, touchant aux études ou à l'art, dans lesquelles il y a peu de chance de devenir riche, mais où les gens s'engagent par fascination ou à cause de sa valeur perçue pour la société. On peut y inclure par exemple, les mathématiques logiques, la musique classique et l'archéologie; puis l'organisation politique au sein des travailleurs. Les gens concourent, plus tristement qu'âprement, pour les quelques situations assises disponibles, aucune d'entre elles n'étant vraiment bien solide. Ils paieraient pour avoir la chance de travailler dans un de ces domaines, s'ils le pouvaient.

Un domaine peut se transformer du jour au lendemain, s'il commence à offrir la possibilité de devenir riche. Si un travailleur devient riche, les autres réclament la même opportunité. Bientôt tous demanderont de fortes sommes d'argent pour ce qu'ils avaient l'habitude de faire pour le plaisir. Puis quelques années passent, chaque personne en relation avec ce domaine tournera en dérision l'idée que le travail pourrait être fait sans salaires mirobolants. Ils conseilleront aux acteurs sociaux de s'assurer que de tels salaires soient possibles, en prescrivant des privilèges spéciaux et les pouvoirs, monopoles nécessaires pour que cela puisse se faire.

Ce changement est apparu dans le domaine de la programmation cette dernière décennie. Il y a quinze ans, des articles parlaient d'«accros à l'informatique» : les utilisateurs étaient «connectés» et vivaient modestement. Il était généralement compris que les accros de la programmation pouvaient briser leur couple. Aujourd'hui, il est généralement admis que personne ne ferait de la programmation, sans salaire élevé. Les gens ont oublié ce qu'ils savaient il y a quinze ans.

Même si à un moment précis, la plupart des gens travailleront dans un certain domaine uniquement pour un haut salaire, cela ne durera pas forcément. La tendance peut s'inverser, si la société donne une impulsion. Si nous laissons de côté les possibilités de gros gains, peu de temps après, quand les gens auront réajusté leurs attitudes, ils auront à nouveau à coeur de travailler dans leur domaine pour la joie de le faire.

La question «comment payer un programmeur» devient plus simple, quand nous réalisons que ce n'est pas la peine de les payer une fortune. Il est plus facile de leur assurer un niveau de vie correct sans plus.

## Financer le logiciel libre

Les institutions qui payent les programmeurs n'ont pas besoin d'être des «boîtes à logiciels». Beaucoup d'autres institutions existantes peuvent le faire.

Les constructeurs de matériels informatiques pensent qu'il est essentiel de supporter le développement logiciel, même s'ils ne peuvent contrôler l'usage du logiciel. En 1970, la plupart de leurs logiciels étaient libres, car ils ne pensaient pas à les entraver. Aujourd'hui, la volonté croissante de se joindre à des consortiums montre leur compréhension que de posséder le logiciel n'est pas ce qui est vraiment important pour eux.

Les universités mènent de nombreux projets de programmation. Aujourd'hui, elles en vendent souvent les résultats, mais, dans les années 70, elles ne le faisaient pas. Douterait-on que les universités développeraient des logiciels libres si elles n'étaient pas autorisées à vendre des logiciels? Ces projets pourraient recevoir le soutien de contrats gouvernementaux et de bourses qui soutiennent actuellement le développement de logiciels propriétaires.

Il est commun de nos jours que les chercheurs universitaires reçoivent une bourse pour développer un système, qu'ils le développent presque jusqu'à la finalisation, qu'ils le déclarent «fini», puis qu'ils créent des sociétés où ils le finiront effectivement et le rendront utilisable. Parfois, ils déclarent «libre» la version non terminée; s'ils sont vraiment corrompus, ils obtiendront une licence exclusive de la part de l'université. Ce n'est pas un secret, c'est ouvertement admis par les personnes concernées. Pourtant, si les chercheurs n'étaient pas exposés à ces tentations, ils continueraient quand même leurs recherches.

Les programmeurs qui écrivent des logiciels libres peuvent gagner leur vie en vendant des services liés au logiciel. J'ai reçu des honoraires pour le portage du compilateur GNU C sur un nouveau matériel et pour faire des extensions d'interfaces utilisateurs pour GNU Emacs. (J'ai offert ces améliorations au public, une fois qu'elles ont été réalisées). Je suis aussi payé pour enseigner dans des classes.

Je ne suis pas seul à travailler de cette façon; il existe maintenant une entreprise fructueuse, grandissante qui ne fait pas autrement. Plusieurs autres compagnies offrent aussi un support commercial aux logiciels libres issus du système GNU. C'est le début d'une industrie indépendante du support du logiciel libre, une industrie qui pourrait prendre de fortes proportions, si le logiciel libre devenait courant. Elle offre aux utilisateurs des options qui sont généralement indisponibles dans le cas de logiciels propriétaires, sauf pour les plus riches.

De nouvelles institutions, comme la Free Software Foundation, peuvent aussi financer les programmeurs. La majorité des fonds de la Fondation vient de l'argent récolté dans la vente de bandes par correspondance. Le logiciel présent sur la bande est libre, ce qui signifie que chaque utilisateur est libre de le copier et de le modifier, mais néanmoins, beaucoup payent pour en obtenir des copies. (Rappelez-vous que «free software» veut dire libre, et non gratuit). Certains utilisateurs achètent des bandes, alors qu'ils en possèdent déjà, simplement parce qu'ils sentent que nous méritons cette contribution. La Fondation reçoit aussi des donations considérables de la part de fabricants d'ordinateurs.

La Free Software Foundation est une organisation caritative et ses revenus sont dépensés en employant le plus possible de programmeurs. Si elle avait été érigée en business, distribuant les mêmes logiciels libres au public pour la

même somme, elle pourrait maintenant offrir un très bon niveau de vie à son fondateur.

Parce que la Fondation est une organisation caritative, les programmeurs travaillent souvent pour la moitié de qu'ils pourraient toucher ailleurs. Ils le font parce qu'ils sont libres de toute bureaucratie et parce qu'ils ressentent de la satisfaction à savoir que leur travail pourra être utilisé par tous. Et, par-dessus tout, ils le font parce que programmer, c'est passionnant. Ajoutons que des volontaires nous ont écrit nombre de programmes (même des rédacteurs techniques ont récemment commencé à se proposer).

Ce qui confirme que la programmation est parmi les domaines les plus fascinants, au même titre que la musique et les arts. Nous n'avons pas à craindre que plus personne ne veuille programmer.

## Que doivent les utilisateurs aux programmeurs?

Il y a une bonne raison à ce que les utilisateurs de logiciels se sentent obligés moralement à contribuer à leur soutien. Les développeurs de logiciels libres contribuent à l'activité des utilisateurs, c'est à la fois loyal et - sur le long terme - aussi dans l'intérêt des utilisateurs que de les financer.

Cependant, ceci ne s'applique pas aux développeurs de logiciels propriétaires, vu que l'obstructionnisme appelle plutôt une sanction qu'une récompense.

Nous nous trouvons ainsi face à un paradoxe : le développeur de logiciels utiles a droit au soutien des utilisateurs, mais n'importe quelle tentative de transformer cette obligation morale en une exigence, détruit les bases de l'obligation. Un développeur peut soit recevoir une récompense, soit l'exiger, mais pas les deux.

Je crois qu'un développeur moral faisant face à ce paradoxe doit agir de manière à mériter la récompense, mais devrait aussi encourager les utilisateurs à donner volontairement. Tôt ou tard, les utilisateurs apprendront à soutenir les développeurs d'eux-mêmes, tout comme ils ont appris à soutenir les radios et les stations télé indépendantes.

## Qu'est-ce que la productivité logicielle?

Si les logiciels étaient libres, il y aurait toujours des programmeurs, mais peut-être en nombre moindre. Est-ce que cela serait mauvais pour la société?

Pas forcément. Aujourd'hui, les nations riches ont moins de fermiers qu'en 1900, mais nous ne pensons certainement pas que cela est mauvais pour la société, car ceux qui restent produisent plus de nourriture pour les consommateurs que tous ceux, plus nombreux, jadis. Nous appelons cela l'amélioration de la productivité. Le logiciel libre devrait demander moins de programmeurs pour satisfaire la demande, à cause de l'augmentation de la productivité logicielle à tous niveaux :

- Une utilisation plus large de chaque programme développé.
- La possibilité d'adapter un programme existant pour le personnaliser au lieu de repartir de zéro.
- Une meilleure instruction des programmeurs.
- L'élimation des redondances dans l'effort de développement.

Ceux qui font objection à la coopération dans la mesure où elle diminuerait l'emploi des programmeurs s'opposent en fait à l'accroissement de la productivité. Pourtant les mêmes acceptent souvent la croyance largement répandue que l'industrie logicielle a besoin d'accroître sa productivité. Comment cela se fait-il?

La «productivité logicielle» peut vouloir dire deux choses : la productivité générale de tout développement logiciel ou la productivité de projets individuels. La productivité générale, c'est ce que la société aimerait améliorer et la voie la plus directe pour le faire est d'éliminer les obstacles artificiels à la coopération qui la réduisent. Mais les chercheurs qui se penchent sur la «productivité logicielle» se focalisent uniquement sur le deuxième sens, limité, du terme, où l'amélioration demande des avancées technologiques difficiles.

## Est-ce que la compétition est inévitable?

Est-il inévitable que les gens se mettent en concurrence, qu'ils essayent de dépasser leurs rivaux dans la société? Peut-être, oui. Mais la compétition en elle-même n'est pas nocive : ce qui est nocif, c'est le *combat*.

Il y a plusieurs façons de concourir. La compétition peut se présenter comme essayer d'aller plus loin, comme surpasser ce que d'autres ont déjà réalisé. Par exemple, jadis, il y avait concurrence entre les meilleurs programmeurs, afin que l'ordinateur fasse les choses les plus incroyables possible, ou encore à qui écrira le programme le plus court, le plus rapide pour une tâche donnée. Ce genre de compétition est bénéfique pour tous, *tant que* l'esprit de saine émulation est maintenu.

Une compétition constructive est suffisante pour pousser les gens à de grands efforts. Certains concourent pour être les premiers à avoir visité tous les pays du globe; il y en a même qui dépensent des fortunes pour cela. Mais ils ne corrompent pas les capitaines de navire pour que leurs rivaux soient échoués une île déserte. Ils se satisfont de laisser le meilleur gagner.

La compétition devient un combat quand les participants commencent à entraver les autres plutôt que de

progresser eux-mêmes - c'est-à-dire quand le «que le meilleur gagne»fait la place au «laissez-moi gagner, que je sois le meilleur ou non». Le logiciel propriétaire est nocif, non parce qu'il est une forme de compétition, mais parce qu'il est une forme de combat au sein des citoyens de notre société.

La compétition dans les affaires n'est pas forcément un combat. Par exemple, lorsque deux épiceries sont en compétition, tout leur effort tend vers l'amélioration de leurs propres services, pas à saboter leur rival. Mais ce n'est pas ce qui démontre un engagement moral dans les affaires; plutôt qu'il existe une faible zone de combat dans ce genre de business, à la limite de la violence physique. Tous les domaines des affaires ne partagent pas cette caractéristique. La rétention d'information utile à tous, c'est une forme de combat.

L'idéologie, dans les affaires, ne prépare pas les gens à résister à la tentation de combattre, dans une compétition. Certaines formes de combat ont été interdites avec les lois anti-trusts, l'interdiction de la publicité mensongère, etc., mais plutôt que de généraliser le rejet du combat, les dirigeants inventent en général d'autres formes de combat qui ne sont pas spécifiquement prohibées. Les ressources de la société sont gaspillées économiquement, à l'instar d'une guerre civile entre factions.

## «Pourquoi n'irais-tu pas en Russie?»

Aux États-Unis, toute personne favorable à autre chose que le plus flagrant laissez-faire égoïste a souvent entendu ce genre de réflexion. On l'entend, par exemple, à l'encontre des partisans d'un système de santé publique, comme on en trouve dans les autres nations industrialisées. Ou encore à propos des partisans d'un soutien public des arts, tout aussi universel chez les nations avancées. L'idée que les citoyens aient une quelconque obligation de participer au bien public est considérée comme du communisme, aux États-Unis. Mais ce terme est-il bien approprié?

Le communisme, comme il était pratiqué en Union Soviétique, était un système de contrôle centralisé, où toutes les activités étaient passées au crible du régime, soi-disant pour le bien public, mais en fait pour le bien des membres du Parti Communiste. Système où les appareils permettant les copies étaient étroitement gardés, pour empêcher les copies illégales.

Le système américain de la propriété intellectuelle exerce un contrôle central sur la distribution d'un programme et surveille les copieurs grâce à des systèmes automatiques de protection contre la copie, pour empêcher les copies illégales.

Par opposition, je travaille à bâtir un système où les gens sont libres de décider de leurs propres actions; en particulier, libres d'aider leur voisin, de modifier et d'améliorer les outils qu'ils utilisent dans leur vie quotidienne. Un système basé sur la coopération volontaire et la décentralisation.

Du coup, si on doit juger ces points de vue par leurs ressemblances au communisme soviétique, alors ce sont les propriétaires de logiciels qui sont les communistes.

## La question des prémisses

Je fais la supposition, dans cet article, que l'utilisateur d'un logiciel n'est pas moins important qu'un auteur ou même l'employeur d'un auteur. Autrement dit, lorsqu'on décide quelle est la meilleure marche à suivre, leurs intérêts et leurs besoins ont autant d'importance.

Cette prémisse n'est pas universellement acceptée. Beaucoup maintiennent que l'employeur d'un auteur est fondamentalement plus important que n'importe qui d'autre. Ils disent, par exemple, que le but, d'avoir des propriétaires de logiciels, est de donner à l'employeur les avantages qui lui sont dûs - indépendamment de l'effet sur le public.

Cela ne sert à rien de prouver ou non ces prémisses. Prouver demande des prémisses partagées. C'est pourquoi la majorité de mon discours s'adresse à ceux qui partagent les prémisses que j'utilise, ou qui, au moins, sont intéressés par leurs conséquences. Pour ceux qui croient que les propriétaires sont plus importants que tous les autres, pour ceux-là, cet article n'est tout simplement pas pertinent.

Mais pourquoi un grand nombre d'Américains accepteraient une prémisse qui élèverait certaines personnes au-dessus des autres? En partie à cause de la croyance que cette prémisse fait partie des traditions légales de la société américaine. Il y a des gens qui pensent que douter de la prémisse, c'est défier les bases de la société.

Il est important pour ces gens de savoir que cette prémisse ne fait pas partie de notre tradition légale. Ne l'a jamais été.

Ainsi, la Constitution dit que le but du copyright est de «promouvoir le progrès des sciences et des arts utiles». La cour Suprême l'a élaboré ainsi, énonçant dans la «Fox Film contre Doyal» que «l'intérêt unique des États-Unis ainsi que l'objet principal du monopole [du copyright], résident dans les bénéfices que retire le public du travail des auteurs».

Nous ne sommes pas obligés d'approuver la Constitution ou la Cour Suprême (il fut un temps où les deux ont approuvé l'esclavage). Leurs positions ne réfutent donc pas la prémisse de la suprématie du propriétaire. Mais j'espère que la conscience qu'il s'agit d'une supposition de la droite radicale, plutôt que d'une tradition reconnue, affaiblira son intérêt.

### Conclusion

Nous aimons penser que notre société encourage à aider son voisin ; mais chaque fois que nous récompensons quelqu'un pour son obstructionnisme ou que nous l'admirons pour les richesses qu'il a accumulé ainsi, nous renvoyons le message contraire.

La thésaurisation de logiciels est un exemple de notre volonté d'ignorer le bien-être de la société pour le gain personnel. On peut en voir la trace depuis Ronald Reagan jusqu'à Dick Cheney, depuis Exxon à Enron, en passant par les échecs des banques et des écoles. Nous pouvons la mesurer à l'aune des sans-abri et de la population dans les prisons. L'esprit antisocial se nourrit de lui-même, parce que plus on voit que les autres ne nous tendront pas la main, plus il nous semble futile de les aider. Ainsi, notre société dégénère en jungle.

Si nous ne voulons pas vivre dans une jungle, nous devons changer nos attitudes. Nous devons commencer à lancer le message qu'un bon citoyen est un citoyen qui coopère quand il le faut, que ce n'est pas celui qui réussit en volant les autres. J'espère que le mouvement du logiciel libre contribuera à cela : au moins dans un domaine, nous remplacerons la jungle par un système plus efficace qui encouragera et se nourrira la coopération volontaire.

**Notes**

1. Le mot «free» dans «free software» signifie libre, et non gratuit [free désigne les deux termes, en anglais]; le prix payé pour une copie d'un programme libre peut être nul, faible, ou (rarement) assez élevé.
2. Les problèmes de pollution et de congestion du trafic ne modifient pas cette conclusion. Si nous désirons rendre plus coûteuse la conduite afin de la décourager, il n'est pas avantageux de le faire en mettant en place des péages, qui participent et à la pollution et à la congestion. Une taxe sur l'essence serait bien mieux. Pareillement, le désir de renforcer la sécurité en limitant la vitesse maximale n'est pas pertinent; un accès gratuit aux routes améliore la vitesse moyenne en évitant arrêts et retards, quelle que soit la limitation de vitesse.
3. On peut voir un logiciel particulier comme une chose nocive, qui ne devrait être accessible à personne, à l'instar de la base de données d'informations personnelles de Lotus (Marketplace), qui a été retirée des ventes suite à la désapprobation du public. La plus grande partie de mon discours ne s'applique pas à ce cas, mais préférer un propriétaire dans la mesure où cela rendrait le programme moins disponible n'est pas très sensé. Le propriétaire ne le rendra pas *complètement* indisponible, comme on pourrait le souhaiter pour un programme considéré comme nocif.

# Logiciels et manuels libres



Le plus grand défaut des systèmes d'exploitation libres n'est pas dans le logiciel, mais dans le manque de bons manuels libres que nous pouvons y inclure. Beaucoup de nos programmes les plus importants ne sont pas fournis avec des manuels complets. La documentation est une partie essentielle de tout logiciel; quand un logiciel libre important n'est pas fourni avec un manuel libre, c'est un manque majeur. Aujourd'hui, nous avons de nombreux manques importants.

Il y a de nombreuses années, j'ai voulu essayer d'apprendre Perl. J'avais une copie d'un manuel libre, mais je ne l'ai pas trouvé facile d'accès. Lorsque j'ai demandé à des utilisateurs de Perl s'il existait une alternative, ils me dirent qu'il y avait de meilleurs manuels d'introduction, mais que ceux-ci n'étaient pas libres.

Mais pourquoi cela ? Les auteurs de ces bons manuels les ont écrit pour O'Reilly Associates, qui les publie sous des termes restrictifs (pas de copie, pas de modification, les sources ne sont pas disponibles). Cela les exclut de la communauté du logiciel libre.

Ce n'était pas la première fois que ce genre de choses se produisait, et (malheureusement pour notre communauté) c'en est loin d'être fini. Les éditeurs de manuels propriétaires ont encouragé un grand nombre d'auteurs à restreindre leurs manuels depuis. J'ai souvent entendu un utilisateur de GNU me parler d'un manuel qu'il était en train d'écrire et avec lequel il comptait aider le projet GNU, puis décevoir mes espoirs en expliquant qu'il avait signé un contrat avec un éditeur qui restreindrait son manuel de telle manière que nous ne pourrions pas l'utiliser.

Etant donné la rareté de bons rédacteurs en langue anglaise parmi les programmeurs, nous ne pouvons pas nous permettre de perdre des manuels de cette manière.

L'intérêt d'une documentation libre (tout comme pour un logiciel libre) est la liberté, pas le prix. Le problème avec ces manuels n'était pas que O'Reilly Associates vende les versions imprimées de ses manuels, ce qui est bon en soi. (La Free Software Foundation vend aussi des impressions des manuels GNU). Mais les manuels GNU sont disponibles sous forme de code source, alors que ces manuels-là ne sont disponibles que sous forme imprimée. Les manuels de GNU sont distribués avec la permission de les copier et de les modifier; mais pas ces manuels de Perl. Ces restrictions sont le problème.

Les conditions à remplir pour un manuel libre sont à peu près les mêmes que pour un logiciel; il s'agit de donner à tous les

utilisateurs certaines libertés. La redistribution (y compris une distribution commerciale) doit être autorisée afin que le manuel accompagne chaque copie du programme, de manière électronique ou imprimée. Permettre les modifications est crucial aussi.

En règle générale, je ne crois pas qu'il soit essentiel que nous ayons la permission de modifier toutes sortes d'articles ou de livres. Les problèmes de l'écriture ne sont pas forcément les mêmes que ceux du logiciel. Par exemple, je ne crois pas que vous ou moi devrions nous sentir obligés de donner la permission de modifier des articles tels que celui-ci, qui décrivent nos actions et nos positions.

Mais il y a une raison particulière pour laquelle la liberté de modifier des documentations libres traitant de logiciels libres est cruciale. Lorsque les programmeurs exercent leur droit de modifier un logiciel et d'ajouter ou de modifier des fonctionnalités, s'il sont consciencieux, ils changeront aussi le manuel afin de pouvoir fournir une documentation précise et utilisable avec leur propre version du programme. Un manuel qui interdirait aux programmeurs d'être consciencieux et de finir leur travail, ou qui leur imposerait d'écrire un nouveau manuel à partir de zéro s'ils modifient le programme ne répond pas aux besoins de notre communauté.

Même si un refus total des modifications est inacceptable, quelques limites sur la manière de modifier une documentation ne pose pas de problème. Par exemple, il est normal d'avoir des injonctions de préserver la notice de copyright originale, les termes de distribution ou la liste des auteurs. Il n'y a pas non plus de problème à demander que les versions modifiées incluent une notice expliquant qu'il s'agit d'une version modifiée, et même d'avoir des sections entières qui ne puissent ni être supprimées ni être modifiées, du moment qu'il ne s'agit pas de sections ayant trait à des sujets techniques (certains manuels GNU en ont).

Ce type de restrictions n'est pas un problème, car d'une manière pratique elles n'empêchent pas le programmeur consciencieux d'adapter le manuel pour correspondre au programme modifié. En d'autres termes, elles n'empêchent pas la communauté du logiciel libre de faire son oeuvre à la fois sur le programme et sur le manuel.

De toutes façons, il doit être possible de modifier toute la partie *technique* du manuel ; sinon ces restrictions bloquent la communauté, le manuel n'est pas libre, et nous avons besoin d'un autre manuel.

Malheureusement, il est souvent difficile de trouver quelqu'un pour écrire un autre manuel quand un manuel propriétaire existe déjà. L'obstacle est que la plupart des utilisateurs pensent qu'un manuel propriétaire est suffisament bon, alors ils ne ressentent pas le besoin d'écrire un manuel libre. Il ne voient pas qu'un système d'exploitation libre a une fissure qui nécessite un colmatage.

Pourquoi ces utilisateurs pensent-ils que les manuels propriétaires sont suffisament bons ? La plupart ne se sont pas penchés sur le problème. J'espère que cet article sera utile dans ce sens.

D'autres utilisateurs considèrent les manuels propriétaires acceptables pour la même raison qu'énormément de personnes considèrent le logiciel propriétaire acceptable : ils pensent en termes purement pratiques, sans mettre la liberté en compte. Ces personnes sont attachées à leurs opinions, mais comme ces opinions découlent de valeurs qui n'incluent pas la liberté, ils ne sont pas un modèle pour ceux d'entre nous qui s'attachent à la liberté.

Je vous encourage à parler de ce problème autour de vous. Nous continuons à perdre des manuels au profit d'éditions propriétaires. Si nous faisons savoir que les manuels propriétaires ne sont pas suffisants, peut-être que la prochaine personne qui voudra aider GNU en écrivant de la documentation réalisera, avant qu'il soit trop tard, qu'elle devra avant tout la rendre libre.

Nous pouvons de plus encourager les éditeurs à vendre des manuels libres et dénués de copyright au lieu de manuels propriétaires. Une façon de le faire est de vérifier les termes de distribution de manuels avant de les acheter, et de préférer les manuels sans copyright à des manuels copyrightés.

# Vendre des logiciels libres

Beaucoup de personnes croient que l'esprit du projet GNU est de ne pas faire payer la distribution de copies de logiciels, ou alors le moins possible : juste assez pour couvrir les frais.

En fait, nous encourageons ceux qui distribuent des logiciels libres à les faire payer le prix qu'ils veulent ou peuvent. Si cela vous semble surprenant, continuez à lire.

Le mot anglais «free» (libre) a deux sens, il peut aussi bien faire référence au prix qu'à la liberté. Quand nous parlons de «free software» (logiciel libre) nous parlons de la liberté, pas du prix. Plus particulièrement, il signifie qu'un utilisateur est libre d'utiliser un programme, de le modifier et de le redistribuer, avec ou sans modifications.

Les logiciels libres sont parfois distribués gratuitement, et parfois contre rémunération. Un même programme est souvent disponible sous ces deux versions à partir de sources différentes. Le programme est libre en dépit de son prix, car les utilisateurs ont toute liberté dans son utilisation.

Les logiciels propriétaires sont souvent vendus à un prix élevé, mais parfois un revendeur peut vous en donner une copie gratuite. Cela n'en fait pas pour autant un logiciel libre. Qu'il soit gratuit ou payant, le programme n'est pas libre car

les utilisateurs n'ont aucune liberté.

Puisque le prix n'a pas d'importance lorsque nous parlons de logiciel libre, un prix bas ne rend pas un logiciel plus «libre». Ainsi, si vous redistribuez des copies de logiciels libres, vous pouvez aussi bien fixer un prix élevé que *rentrer dans vos frais*. La redistribution de logiciels libres est une activité honorable et totalement légale; si vous l'exercez, vous pouvez très bien en tirer du profit.

Le logiciel libre est le projet de toute une communauté, et tous ceux qui en dépendent devraient chercher à soutenir la communauté. Pour un distributeur, la manière d'y contribuer est de donner une part de ses bénéfices à la FSF ou à un autre projet de développement de logiciels libres. En soutenant des équipes de développement, vous faites avancer le logiciel libre.

**La distribution de logiciels libres est une chance de rassembler des fonds pour le développement. Ne la laissez pas passer!**

Pour faire un don, vous devez avoir des fonds en réserve. Si vous fixez un prix trop faible, vous n'aurez pas de réserve pour soutenir le développement.

### Est-ce qu'un prix plus élevé lésera des utilisateurs?

Certains s'inquiètent parfois qu'un prix trop élevé mette le logiciel libre hors de portée des utilisateurs n'ayant pas beaucoup de moyens financiers. En ce qui concerne les logiciels propriétaires, c'est exactement ce que fait un prix élevé, mais c'est différent pour le logiciel libre.

La différence est que le logiciel libre tend naturellement à se répandre, et qu'il y a différentes façons de se le procurer.

Les rapaces du logiciel vendent leur âme au diable pour vous empêcher d'utiliser un programme propriétaire sans payer le prix fort. Si le prix est élevé, il devient vraiment difficile pour certains utilisateurs d'utiliser le programme.

Avec le logiciel libre, les utilisateurs n'ont pas à payer la distribution pour utiliser le logiciel. Ils peuvent copier le programme à partir de la copie d'un ami, ou avec son aide s'il a accès au réseau. Plusieurs utilisateurs peuvent également se cotiser pour acheter un CD-ROM et installer le logiciel chacun à son tour. Un prix élevé n'est pas un obstacle majeur si le logiciel est libre.

### Est-ce qu'un prix plus élevé découragera l'utilisation du logiciel libre ?

Une autre inquiétude est souvent exprimée à propos de la popularité du logiciel libre. Certains pensent qu'un prix élevé réduira le nombre d'utilisateurs ou qu'un prix faible encouragera certainement ces derniers.

C'est vrai dans le cas d'un logiciel propriétaire, mais c'est différent dans le cas d'un logiciel libre. Il y a tellement de façons d'avoir des copies que le prix payé en échange de la distribution a beaucoup moins d'effet sur la popularité.

À longue échéance, le nombre d'utilisateurs du logiciel libre est déterminé par *les capacités du logiciel libre* et par sa simplicité d'utilisation. De nombreux utilisateurs continueront à utiliser des logiciels propriétaires si le logiciel libre ne peut pas faire tout ce qu'ils veulent. Ainsi, si nous voulons faire augmenter le nombre d'utilisateurs à longue échéance, nous devons avant tout *développer plus de logiciels libres*.

La façon la plus directe est d'écrire vous-même des logiciels libres ou des manuels qui manquent. Mais si vous assurez la distribution plutôt que la création, la meilleure façon de nous aider est de lever des fonds pour aider les autres à en écrire.

### Les mots «vendre des logiciels» peut aussi induire en erreur

*Stricto sensus*, «vendre» signifie échanger des biens contre de l'argent. Vendre une copie d'un logiciel libre est légal, et nous encourageons cette pratique.

Cependant, quand les gens pensent à «vendre des logiciels», ils l'imaginent habituellement de la même manière que la plupart des entreprises : rendre le logiciel propriétaire plutôt que libre.

Alors à moins que vous ne soyez prˆts à faire des distictions précises, comme le fait cet article, nous vous suggérons d'éviter le terme «vendre des logiciels» et de choisir un autre vocabulaire à la place. Par exemple, vous pourriez dire «distribuer des logiciels libres contre rémunération», ce qui lève toute ambiguité.

### Prix élevés ou bas et la GNU GPL

En dehors d'une seule situation spécifique, la Licence Publique Générale GNU (GNU GPL) n'a pas d'exigences en ce qui concerne le prix que vous pouvez demander pour la distribution d'un logiciel libre. Vous pouvez ne rien demander, ou alors un centime, un franc, un million de francs. Cela ne dépend que de vous, de l'offre et de la demande, alors ne venez pas vous plaindre si personne ne veut payer un million de francs pour une copie.

La seule exception est le cas où les binaires sont distribués sans le code source complet. Ceux qui font cela sont obligés par la GNU GPL de donner le code source sur toutes demandes ultérieures. Sans limite fixée au prix du code source, ils peuvent fixer un prix trop élevé pour que quelqu'un puisse payer (un million de francs, par exemple), et ainsi prétendre distribuer le code source alors qu'ils le dissimulent. Ainsi, nous devons dans ce cas limiter le prix du code source

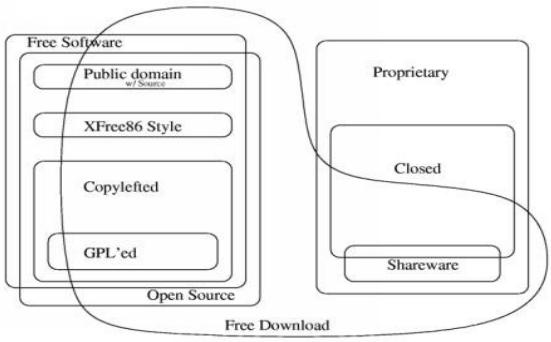
pour assurer la liberté de l'utilisateur. Cependant, dans des situations normales il n'y a pas de telle justification pour limiter le prix de distribution, alors nous ne le faisons pas.

Parfois des entreprises, dont les activités franchissent la limite de ce que la GNU GPL permet, implorent des autorisations, en disant qu'ils «ne vont pas faire payer les logiciels GNU», ou d'autres choses du même style. Cela ne les ménera pas loin. Le logiciel libre c'est avant tout la liberté, et faire respecter la GPL c'est défendre cette liberté. Quand nous défendons la liberté des utilisateurs, nous ne nous occupons pas de problèmes annexes comme le prix de la distribution. La liberté est la question, la seule et l'unique.

# Catégories de logiciels libres et non libres

Voici un glossaire des différentes catégories de logiciels qui sont le plus souvent citées dans les discussions sur les logiciels libres. Il explique les catégories qui englobent, ou qui font partie d'autres catégories.

Ce diagramme de Chao-Kuei explique les différentes catégories de logiciels. Il est disponible (en français) sous forme de fichier XFig, d'image JPEG et une image PNG grossie 1,5 fois, sous les termes de la GNU GPL v2 ou ultérieure, de la GNU FDL v1.2 ou supérieure ou de la Creative Commons Attribution-Share Alike v2.0 ou ultérieure.




**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 systèmes 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. Pour des informations sur des licences de logiciels libres spécifiques, consultez la page liste des licences.

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.

À 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.

Beaucoup de langues ont deux mots séparés pour «libre» pour liberté et «libre» pour gratuité. Par exemple, le français a «libre» et «gratuit». Pas l'anglais; il y a le mot «gratis» qui se réfère sans ambiguïté au prix, mais pas d'adjectif commun se référant sans équivoque à la liberté. Aussi, si vous parlez une autre langue, nous vous suggérons de traduire «free» dans votre propre langue pour le rendre plus clair. Consultez notre liste de traductions du terme «free software» dans diverses langues.

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 qualifier plus ou moins la même catégorie que le logiciel libre. Ce n'est pas exactement la même catégorie de logiciel : ils acceptent certaines licences que nous considérons trop restrictives et il y a des licences de logiciel libre qu'ils n'ont pas acceptées. Toutefois, les différences entre les deux catégories sont minimes : pratiquement tous les logiciels libres sont «open source» et presque tous les logiciels «open source» sont libres. Nous préférons le terme «logiciel libre»; car il se réfère à la liberté -- ce que ne fait pas le terme «open source».

**Logiciel du domaine public**

Logiciel du domaine public veut dire logiciel non soumis aux droits d'auteurs. Si le code source est dans le domaine public, 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.

Dans certains cas, un programme exécutable peut être dans le domaine public sans que le code source ne soit disponible. Il ne s'agit pas de logiciel libre, puisque le logiciel libre nécessite l'accès au code source.

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.

Selon la convention de Berne, que la plupart des pays ont signé, tout ce qui est écrit est automatiquement sous copyright. Cela comprend les programmes. Par conséquent, si vous voulez que le programme que vous avez écrit soit dans le domaine public, vous devez faire des démarches juridiques pour enlever son copyright; sinon le programme demeure sous copyright.

**Logiciel *copylefté* (sous *gauche d'auteur*)**

Le logiciel sous copyleft (littéralement, gauche d'auteur) est un logiciel libre, dont les conditions de distribution assurent que toutes les copies de toutes les versions sont des logiciels libres. Cela signifie par exemple, que les licences de copyleft ne permettent pas à d'autres d'ajouter des conditions supplémentaires au logiciel (bien qu'un ensemble limité de conditions sûres complémentaires puissent être ajoutées) et nécessitent que le code source soit disponible. Certaines licences de copyleft, comme la GPL version 3, bloquent d'autres moyens de rendre le logiciel propriétaire.

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, donc en théorie de nombreuses licences de logiciels libres sous copyleft peuvent exister. Néanmoins, dans la pratique actuelle quasiment tous les logiciels sous copyleft utilisent la Licence Publique Générale GNU. Deux licences différentes avec copyleft sont généralement «incompatible», ce qui signifie qu'il est illégal de mélanger du code utilisant une de ces licences avec du code utilisant l'autre à partir de là, il est bon pour la communauté de n'utiliser qu'une seule licence avec copyleft.

**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. Les développeurs d'X11 ont même rendu X11 non-libre pour un bon moment.

**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, qui est entièrement libre, et que nous avons développé au sein du projet GNU depuis 1984.

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 le système GNU depuis 1984; la première mise à disposition en test d'un «système GNU complet» remonte à 1996. Ceci inclut GNU Hurd, notre noyau, développé depuis 1990. En 2001, le système GNU (y compris Hurd) a commencé à fonctionner de façon relativement fiable, mais il manque d'importantes fonctionnalités à Hurd, c'est pourquoi il n'est pas largement utilisé. Dans le même temps, le système GNU/Linux, une ramification du système GNU utilisant Linux comme noyau plutôt que GNU Hurd, est devenu un grand succès dans les annéees 90.

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. Et il n'est pas nécessaire que tous les composants soient des logiciels GNU, individuellement. GNU peut et utilise des logiciels libres non-copyleftés, comme le système X Window, qui sont développés par d'autres projets.

**Programmes GNU**

Les «programmes GNU» sont équivalents aux Logiciels GNU. Un programme Toto est un programme GNU s'il est un logiciel GNU. Nous l'appelons parfois aussi «paquet GNU».

**Logiciel GNU**

Un logiciel GNU est un logiciel diffusé sous les auspices du Projet GNU.

Si un programme est un logiciel GNU, nous disons aussi qu'il est un programme GNU ou un paquetage GNU. Le fichier README ou le manuel d'un paquetage GNU devrait le spécifier. Le Répertoire des logiciels libres identifie également tous les paquetages GNU

La plupart des logiciels GNU sont soumis à un copyleft, mais pas tous; cependant, tous les logiciels GNU doivent être des logiciels libres.

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 non-libre**

Les logiciels non-libres sont tous les logiciels qui ne sont pas libres. Ceci inclut les logiciels semi-libres et les logiciels propriétaires.

**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 éthiquement 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 les systèmes GNU basés sur 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 privé**

Les logiciels privés sont développés pour un utilisateur (typiquement pour une organisation ou une société). Cet utilisateur le garde et ne publie ni les fichiers sources ni les fichiers binaires.

Un programme privé est dans un certain sens un logiciel libre si son seul utilisateur a tous les droits dessus. Cependant, en y réfléchissant un peu, cela n'a pas vraiment de sens de se poser la question de savoir si un tel programme est un logiciel libre ou non.

En général, nous ne pensons pas qu'il soit mauvais de développer un programme et de ne pas le publier. Il y a des fois où un programme est si utile que le garder pour soi est égoïste. Toutefois, la plupart des programmes ne sont pas si merveilleux que ça, et les garder pour soi n'est pas particulièrement dommageable. Donc, le développement de logiciel pour un usage privé, qui représente une grande part de l'emploi dans la programmation, ne viole pas les principes du mouvement du logiciel libre.

Pratiquement tous les emplois de programmeurs se situent dans le développement de logiciels privés; par conséquent la plupart des emplois de programmeurs sont, ou pourraient être faits d'une manière compatible avec le mouvement du logiciel libre.

**Logiciel commercial**

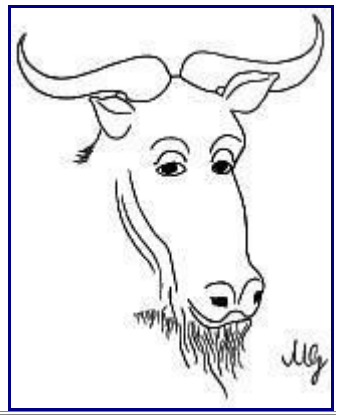
Le logiciel commercial est un logiciel développé par une entreprise dont le but est de gagner de l'argent sur l'utilisation du logiciel. «Commercial» et «propriétaire» ne sont pas synonymes! La plupart des logiciels commerciaux sont propriétaires, mais il y a des logiciels libres commerciaux, et il y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

# Le logiciel libre est plus fiable!



Ceux qui font l'apologie du logiciel propriétaire (18k caractères) aiment dire, «le logiciel libre est un beau rêve, mais nous savons tous que seul un système propriétaire peut produire des produits fiables. Un groupe de hackers ne peut faire la même chose.»

Cependant, l'évidence empirique ne concorde pas, des tests scientifiques ont montré que le logiciel libre est *plus* fiable qu'un logiciel propriétaire comparable.

Cela ne devrait pas être une surprise. Il y a de bonnes raisons pour la haute fiabilité des logiciels GNU, de bonne raisons de s'attendre à ce que le logiciel libre aura souvent (bien que pas toujours) une haute fiabilité.

Les logiciels GNU plus sûrs!

---

Barton P. Miller et ses collègues ont testé la fiabilité des programmes utilitaires Unix en 1990 et 1995. Chaque fois, les utilitaires GNU arrivèrent en tête de façon considérable. Ils testèrent sept systèmes Unix commerciaux ainsi que le système GNU. En leur soumettant un flux d'entrées aléatoires, ils ont pu «planter (avec un core dump) ou bloquer (boucle infinie) plus de 40% (dans le pire des cas) des programmes utilitaires de base...»

Ces chercheurs trouvèrent que les systèmes Unix commerciaux avaient un taux d'échec de 15% à 43%. Par contraste, le taux d'échec pour le système GNU était seulement de 7%.

Miller disait également : «les trois systèmes commerciaux que nous avons comparés, à la fois en 1990 et en 1995, se sont améliorés sensiblement en fiabilité, mais ils ont toujours des taux d'échec significatifs (les utilitaires de base de GNU/Linux étaient toujours sensiblement meilleurs que les utilitaires des systèmes commerciaux)».

Pour les détails, voir leur document : Fuzz Revisited: A Re-examination of the Reliability of Unix Utilities and Services (postscript 146k) par Barton P. Miller

<bart@cs.wisc.edu>, David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, et Jeff Steidl.

## Pourquoi le logiciel libre est plus fiable

Ce n'est pas le fait du hasard si les utilitaires de GNU sont si fiables. Il y a de bonnes raisons pour lesquelles le logiciel libre tend vers un très haut niveau de qualité.

Une raison est que le logiciel libre bénéficie des efforts de l'ensemble de la communauté qui travaille dans le but de corriger les problèmes. Les utilisateurs ne se contentent pas seulement de signaler les bogues, mais ils les corrigent aussi et envoient leurs corrections. Les utilisateurs travaillent ensemble, communiquant par l'intermédiaire du courrier électronique, pour venir à bout des problèmes et rendent le travail avec les logiciels plus serein.

Une autre raison est que les développeurs se soucient vraiment de la fiabilité. Les paquetages de logiciels libres ne cherchent pas toujours la compétition sur le marché, mais ils se battent toujours pour une bonne réputation, et un programme qui n'est pas satisfaisant ne réussira pas à obtenir la popularité que ses concepteurs veulent lui donner. Qui plus est, un auteur qui met son code source à la disposition de tous met sa réputation en jeu, et il vaudrait mieux pour lui que son logiciel soit propre et élégant, sous peine de subir le mécontentement de la communauté.

### La Clinique du cancer fait confiance au Logiciel Libre!

Le Centre du cancer Roger Maris à Fargo, Dakota du Nord (le même Fargo qui fut récemment le décor d'un film et d'une inondation) utilise des systèmes GNU/Linux précisément parce que la fiabilité est essentielle. Un réseau de machines GNU/Linux exécute le système d'information, coordonne les thérapies de drogues, et accomplit de nombreuses autres fonctions. Ce réseau doit être disponible pour le personnel du Centre à tout moment.

Selon le Dr. G.W. Wettstein <greg@wind.rmcc.com> :

> «le soin apporté à nos patients atteints de cancer ne serait pas ce qu'il est aujourd'hui sans [GNU]/Linux ... Les outils que nous avons été capables de déployer à partir de logiciels libres nous ont permis d'écrire et de développer des applications innovantes qui ... ne peuvent exister à partir de voies commerciales».

### Des utilitaires GNU blindés!

Scott Maxwell conduit une initiative visant à éliminer les «fuzz bugs» des logiciels GNU, les rendant ainsi encore plus fiables. Vous pouvez prendre connaissance de son projet sur http://home.pacbell.net/s-max/scott/bulletproof-penguin.html.

## Pourquoi «Free Software» est-il meilleur que «Open Source»

Bien que la liberté d'un logiciel ne soit pas dépendante du nom qu'on lui donne, il y a des différences dans les noms qu'on lui accorde : des mots différents apportent des idées différentes.

En 1998, quelques personnes de la communauté du logiciel libre ont commencé à utiliser le terme «logiciel ouvert» (*open source software*) au lieu de «logiciel libre» (*free software*). Le terme «open source» devint rapidement associé à une approche, à une philosophie, à des valeurs et même à des critères différents pour lesquels les licences sont acceptables. Le Mouvement du logiciel libre et le Mouvement open source sont aujourd'hui des mouvements séparés, avec des vues et des objectifs différents, bien que nous puissions travailler ensemble, et nous le faisons, sur quelques projets concrets.

La différence fondamentale entre les deux mouvements se situe dans leurs valeurs, leurs façons de voir le monde. Pour le Mouvement open source, la question de savoir si un logiciel devrait être open source est une question pratique, non éthique. Comme quelqu'un l'a dit : «le Mouvement open source est une méthodologie de développement; le Mouvement du logiciel libre, un mouvement social». Pour le Mouvement open source, un logiciel non libre est une solution sous-optimale. Pour le Mouvement du logiciel libre, le logiciel non libre est un problème social et le logiciel libre en est la solution.

### Relations entre le Mouvement du logiciel libre et le Mouvement open source

Le Mouvement du logiciel libre et le Mouvement open source sont comme deux partis politiques à travers notre communauté.

Les groupes radicaux des années 1960 sont connus pour leur factionalisme : les organisations se séparent à cause de divergences dans des détails de stratégie, et se détestent alors. Ils sont d'accord sur les principes de base, et n'ont de divergences que sur des recommandations pratiques; mais ils se considèrent comme des ennemis, et se battent sauvagement.

Pour le Mouvement du logiciel libre et le Mouvement open source, c'est tout simplement le contraire. Nous ne sommes pas d'accord sur les principes de base, mais nous sommes d'accord sur la plupart des recommandations pratiques. Nous travaillons ensemble sur de nombreux projets spécifiques.

Dans le Mouvement du logiciel libre, nous ne pensons pas que le Mouvement open source soit un ennemi. L'ennemi est le logiciel propriétaire. Mais nous voulons que les personnes de notre communauté sachent que nous ne sommes pas pareils qu'eux!

Alors veuillez mentionner le Mouvement du logiciel libre lorsque vous parlez du travail que vous avez accompli, et du logiciel que vous avez développé, tout comme du système d'exploitation GNU/Linux.

### Comparaison des deux termes

Le reste de cet article compare les termes «logiciel libre» et «open source». Il montre pourquoi le terme «open source» ne résout aucun problème, et en fait en crée même certains.

### Ambiguïté

Le terme «free software» pose un problème d'ambiguïté : une signification involontaire, «logiciel gratuit», (*NdT : free = gratuit en anglais*) est aussi valable que la signification voulue, «logiciel qui accorde des libertés à l'utilisateur». Nous résolvons ce problème en publiant une définition plus précise du logiciel libre, mais ce n'est qu'une solution partielle; elle ne peut pas éliminer complètement le problème. Un terme correct et non ambigu serait meilleur, en supposant que cela n'entraîne pas d'autres problèmes.

Malheureusement, toutes les alternatives ont leurs propres défauts. Nous avons étudié de nombreuses alternatives que l'on nous a suggérées, et alors que certaines évitaient ce problème, elles en entraînaient de nouveaux, aucune n'étant suffisamment «correcte» pour que ce soit une bonne idée de l'utiliser. Chaque alternative proposée pour «free software» posait les mêmes sortes de problèmes sémantiques, ou pire, ce qui est le cas de «open source».

La définition officielle de «logiciel open source», telle qu'elle est publiée par la Open Source Initiative, est très proche de notre définition de logiciel libre; cependant, elle est un peu vague sur certains aspects, et ils ont accepté quelques licences que nous considérons restrictives de façon inacceptable pour les utilisateurs. La signification évidente de «open source» est «Vous pouvez avoir le code source». Cette catégorie de logiciels n'est clairement pas la même que celle de logiciels libres. Elle inclut les logiciels libres, mais aussi les logiciels semi-libres tels que Xv, et même les logiciels non-libres tels que Qt sous sa licence d'origine.

La signification explicite d'«open source» n'est pas la signification que ses partisans désirent (leur définition «officielle» est plus proche de «logiciel libre»). Le résultat est que le public comprend généralement mal la signification de ces termes. Voici comment l'écrivain Neal Stephenson définit «open source» :

> Linux est un logiciel «open source» signifie, simplement, que n'importe qui peut obtenir des copies de ses fichiers sources.

Je ne pense pas qu'il avait l'idée de délibérément rejeter ou critiquer la définition «officielle». Il a simplement appliqué les conventions de la langue anglaise, et tiré la conclusion naturelle. L'état du Kansas a publié une définition similaire :

> De l'utilisation de logiciel open source (OSS). L'OSS est un logiciel pour lequel le code source est librement et publiquement accessible, bien que les accords de licence spécifiques varient sur ce que chacun est autorisé à faire avec ce code.

Bien sûr, il est possible de résoudre ce problème en publiant une définition précise, comme nous l'avons fait pour «free software».

Mais l'explication pour «logiciel libre» est simple -- une personne qui a saisi l'idée de «discours libre, pas bière gratuite» («free speech, not free beer») ne commettra plus l'erreur. Il n'y a pas de manière aussi succincte pour expliquer la signification de «open source» et pour montrer clairement pourquoi la définition naturelle n'est pas la bonne.

### Peur de la liberté

L'argument principal pour le terme «open source» est que «free software» rend certaines personnes méfiantes. C'est vrai: parler de liberté, de questions éthiques, de responsabilités aussi bien que de convenance, c'est demander aux gens de réfléchir à des choses qu'ils préféreraient ignorer; cela peut induire une certaine gêne. Mais nous ne rendrions pas la société meilleure si nous arrêtions de parler de ces choses.

Il y a quelques années, des développeurs de logiciels libres ont remarqué cette réaction, et certains ont commencé à explorer des voies pour l'éviter. Ils ont imaginé qu'en restant prudent à propos de l'éthique et de la liberté, et en ne parlant que des bénéfices pratiques et immédiats de certains logiciels libres, ils pouvaient «vendre» le logiciel libre plus efficacement à certains utilisateurs, principalement en entreprise. C'est pour cela qu'ils ont utilisé le terme «open source», pour être «plus acceptable en entreprise».

Cette approche a prouvé son efficacité, à sa manière. Aujourd'hui, beaucoup de personnes se mettent au logiciel libre pour des raisons purement pratiques. C'est une bonne chose, mais ce n'est pas la seule chose que nous devons entreprendre! Attirer des utilisateurs vers le logiciel libre n'est pas le but final que nous devons atteindre, mais juste le premier pas.

Tôt ou tard ces utilisateurs seront tentés de retourner au logiciel propriétaire pour d'autres avantages pratiques. Des entreprises mineures cherchent à profiter d'une telle tentation, et pourquoi les utilisateurs refuseraient-ils? Seulement s'ils ont compris *l'importance de la liberté* dans le logiciel libre pour elle-même. C'est à nous de répandre cette idée, et pour le faire nous devons parler de liberté. Une certaine partie de l'approche «prudente» est utile pour la communauté, mais nous devons également avoir beaucoup de discours sur la liberté.

À présent, nous avons beaucoup de partisans de l'approche «prudente», mais pas assez de discours sur la liberté. La plupart des personnes concernées par le logiciel libre parlent peu de la liberté, souvent parce qu'elle cherchent à être «plus acceptables en entreprise». Les distributeurs de logiciels présentent essentiellement cet aspect. Quelques distributions du système d'exploitation GNU/Linux ajoutent des paquetages propriétaires au système libre de base, et ils espèrent que les

utilisateurs considèrent cela comme un avantage, plutôt qu'un retour en arrière.

Nous n'arrivons pas à gérer le flux entrant des utilisateurs de logiciels libres, nous n'arrivons pas à leur parler de liberté et de notre communauté dès qu'ils y entrent. C'est pourquoi les logiciels non-libres (comme Qt, à l'époque où celui-ci devint populaire) et des distributions en partie non-libres de systèmes d'exploitation trouvent une terre si fertile. Abandonner maintenant le mot «libre» serait une erreur; nous devons parler plus (et pas moins) de la liberté.

Espérons que ceux qui utilisent le terme «open source» attireront en fait plus d'utilisateurs dans notre communauté mais s'ils y arrivent, le restant de notre communauté devra travailler encore plus durement pour porter le problème de la liberté aux oreilles de ces utilisateurs. Nous devons dire «C'est un logiciel libre et il vous donne la liberté» plus souvent et plus fort qu'auparavant.

### Est-ce qu'une marque déposée nous aiderait?

Les partisans du «logiciel open source» ont essayé d'en faire une marque déposée, en disant que ça leur permettra d'éviter les abus. L'essai échoua lorsque la démarche cessa de prendre effet en 1999; ainsi, le statut légal d'«open source» est le même que celui de «free software» il n'y a pas de contrainte *légale* à l'utiliser.

Mais est-ce que cela ferait une grosse différence d'utiliser un terme qui est déposé? Je n'en suis pas sûr. J'ai entendu parler d'entreprises appelant des logiciels «open source» même s'ils ne correspondaient pas à la définition officielle; j'ai personnellement observé quelques exemples.

Des entreprises font parfois des annonces qui donnent l'impression qu'un programme est un «logiciel open source» sans vraiment le dire explicitement. Par exemple, les annonces d'IBM à propos d'un programme qui ne correspond pas à la définition officielle dit ceci : *Comme c'est courant dans la communauté open source, les utilisateurs de la technologie ... vont aussi pouvoir collaborer avec IBM ...*

Ceci ne disait pas vraiment que le programme *était* «open source», mais beaucoup de lecteurs n'ont pas noté ce détail (je devrais noter qu'IBM essayait sincèrement de rendre son programme libre, et adopta plus tard une nouvelle licence qui le rendit libre et «open source»; mais quand l'annonce fut faite, le programme n'était ni l'un ni l'autre).

Et voici comment Cygnus Solutions, qui a été formée pour être une entreprise de logiciels libres et qui a étendu par la suite ses activités (pour ainsi dire) dans le monde du logiciel propriétaire, fit de la publicité pour des logiciels propriétaires : *Cygnus Solutions fait partie de la tête du marché de l'open source et vient juste de lancer deux nouveaux produits sur le marché Linux.*

Contrairement à IBM, Cygnus n'essayait pas de rendre ses logiciels libres, et ses logiciels étaient loin d'être des logiciels libres. Mais Cygnus ne dit pas vraiment que ce sont des «logiciels open source», ils écrivent juste une phrase vague pour essayer d'obtenir l'attitude favorable qui vient avec ce terme.

Ces observations suggèrent qu'une marque déposée n'aurait pas vraiment résolu les problèmes du terme «open source».

### Incompréhensions d'«Open Source»

La définition de l'Open Source est assez claire, et il est complètement clair que le programme typiquement non libre n'y correspond pas. Alors vous pourriez penser qu'une «entreprise Open Source» fait des logiciels libres, n'est-ce-pas? Hélas, de nombreuses entreprises essayent de lui donner une autre définition.

Au colloque «Open Source Developers Day» en août 1998, plusieurs des développeurs commerciaux invités dirent qu'ils n'avaient l'intention de rendre qu'une partie de leur travail libre (ou «open source»). La partie principale de leur activité est de développer des extensions propriétaires (logiciels ou manuels) à vendre aux utilisateurs de logiciel libre. Ils nous demandent de considérer ceci comme légitime en tant que part de notre communauté, car une partie des bénéfices est reversé au développement de logiciels libres.

En effet, ces entreprises cherchent à acquérir l'apparence favorable de l'«open source» pour leurs logiciels propriétaires, tout en sachant que ce ne sont pas des «logiciels open source», car ils ont quelques relations avec le logiciel libre ou parce que la même entreprise maintient des logiciels libres (le fondateur d'une entreprise dit explicitement qu'il consacrerait le minimum de travail que la communauté accepterait dans le logiciel libre qu'ils maintiennent).

À travers les années, de nombreuses entreprises ont contribué au développement de logiciels libres. Certaines de ces entreprises développaient d'abord des logiciels propriétaires, mais les deux activités étaient séparées; ainsi, nous pouvions ignorer leurs produits non libres, et travailler avec eux sur leurs projets libres. Alors, nous pouvions par la suite les remercier honnêtement pour leur contribution au logiciel lire, sans parler du reste de leurs activités.

Nous ne pouvons plus faire de même avec ces nouvelles entreprises, car elles n'avanceront pas ainsi. Ces entreprises essayent activement de conduire le public à considérer leurs activités en bloc; elles veulent que nous considérions leurs logiciels non libres aussi favorablement qu'une vraie contribution, même si ce n'en est pas une. Elles se présentent elles mêmes comme des «entreprises open source», en espérant que nous allons ressentir un sentiment positif à leur propos, et que nous allons l'appliquer aveuglément.

Cette pratique manipulatrice ne serait pas moins grave si elle était faite en utilisant le terme «logiciel libre». mais

des entreprises n'ont pas l'air d'utiliser le terme «logiciel libre» de cette manière; peut-être que l'association avec l'idéalisme le rend inapproprié. Le terme «open source» à ouvert la porte à ces dérives.

Fin 1998, à une exposition dédiée au système d'exploitation qu'on appelle souvent «Linux», l'intervenant était un cadre d'une importante entreprise de logiciels. Il a certainement été invité grâce à la décision de son entreprise de «supporter» ce système. Malheureusement, leur support consiste à éditer des logiciels non libres qui fonctionnent sur ce système, en d'autres mots, utiliser notre communauté comme marché mais sans y contribuer.

Il dit «Nous ne rendrons pas notre produit open source, mais peut-être le rendrons-nous open source en 'interne'. Si nous autorisons notre support client à avoir accès au code source, ils pourraient corriger des bugs pour le client, et nous pourrions fournir un meilleur produit et un meilleur service.» (Ce n'est pas la citation exacte puisque je ne l'ai pas notée, mais ç'en est l'essentiel).

Des personnes du public me dirent plus tard «Il n'a tout simplement rien compris.» Tant que ça? Quel point n'a t-il pas compris?

Il n'a pas raté tout ce qui est habituellement associé au terme «open source». Ceci ne dit rien sur la liberté, mais dit seulement qu'autoriser plus de personnes à accéder au code source et aider à l'améliorer rendra le développement plus rapide et plus efficace. Le cadre a complètement assimilé ce point; en négligeant pour d'autres raisons de conduire cette approche jusqu'à la fin, en incluant les utilisateurs, il a juste pensé à le mettre en oeuvre partiellement, à l'intérieur de l'entreprise.

Ce qu'il n'a pas compris est ce que l'«open source» n'est pas conçu pour relever : que les utilisateurs *méritent* la liberté.

Faire progresser l'idée de la liberté est un gros travail (qui a besoin de votre aide). Le projet GNU s'attachera au terme «logiciel libre». Si vous croyez que la liberté et la communauté sont importantes en tant que telles, pas juste pour la commodité qu'elles apportent, rejoignez-nous en utilisant le terme «logiciel libre».

---

Joe Barr a écrit un article intitulé Live and let license (Croissez et faites des licences); qui donne son point de vue sur le problème.

Le document sur la motivation des développeurs de logiciels libres de Lakhani et Wolf, dit qu'une part considérable d'entre eux est motivée par l'idée que le logiciel devrait être libre. Ce, en dépit du fait qu'ils ont interrogé les développeurs sur SourceForge, un site qui ne soutient pas l'idée qu'il s'agit d'un problème éthique.

## Linux, GNU, et liberté

par **Richard M. Stallman**

Puisque l'article de Joe Barr a critiqué mes négociations avec SIGLINUX, j'aimerais faire une mise au point sur ce qui s'est vraiment passé, et donner mes raisons.

Quand SIGLINUX m'a invité à parler, il s'agissait d'un «groupe d'utilisateurs Linux» (LUG); c'est-à-dire un groupe pour utilisateurs du système GNU/Linux qui appelle l'ensemble du système d'exploitation «Linux». J'ai alors répondu poliment que s'ils voulaient que quelqu'un du projet GNU fasse un discours pour eux, ils devraient traiter le Projet GNU correctement, et appeler le système «GNU/Linux». Le système est une variante de GNU, et le Projet GNU en est le principal développeur, aussi, la correction veut qu'il soit appelé par le nom que nous avons choisi. À moins qu'il n'y ait des raisons impérieuses de faire une exception, je refuse habituellement de faire des discours pour des organisations qui ne donneraient pas à GNU le crédit qui lui revient. Je respecte leur liberté d'expression, mais j'ai également la liberté de ne pas faire de discours.

Par la suite, Jeff Strunk de SIGLINUX a essayé de changer la politique du groupe, et a demandé à la FSF de placer son groupe sur notre page de groupes d'utilisateurs GNU/Linux. Notre webmestre lui a dit que nous ne listerions pas son groupe sous le nom «SIGLINUX», car ce nom implique que le groupe concerne Linux. Strunk a proposé de changer le nom pour «SIGFREE», et notre webmestre a trouvé que ce serait bien. (l'article de Barr a dit que nous avions rejeté cette proposition). Quoi qu'il en soit, le groupe décida finalement de garder «SIGLINUX».

À ce moment-là, on me redemanda mon avis, et je suggérais de réfléchir à d'autres noms possibles. Il y a plusieurs noms qu'ils pourraient choisir qui n'appelleraient pas le système «Linux», et j'espère qu'ils en trouveront un qui leur plaise. Pour l'instant, on en est là , pour autant que je sache.

Est-il vrai, comme l'écrit Barr, que certains voient dans ces actions une «utilisation de la force» comparable au pouvoir du monopole de Microsoft? Probablement. Décliner une invitation n'est pas de la coercition, mais les gens qui sont déterminés à croire que le système entier est «Linux» développent parfois une vision étonnamment déformée. Pour justifier ce nom, ils prennent des vessies pour des lanternes. Si vous pouvez ignorer les faits et croire que Linus Torvalds a développé l'ensemble du système en commençant en 1991, ou si vous pouvez ignorer votre sens de la justice et croire que Torvalds devrait obtenir, seul, le crédit même s'il n'a pas fait cela, il n'y a qu'un pas pour croire que je vous dois un discours si vous le demandez.

Considérez ceci : le Projet GNU commence à développer un système d'exploitation, et des années plus tard Linus Torvalds y ajoute une partie importante. Le Projet GNU a dit, «Veuillez mentionner notre projet équitablement», mais Linus a dit, «Ne leur donnez pas une part du mérite, appelez l'ensemble de mon seul nom!». Imaginez maintenant l'état d'esprit d'une personne, qui, à la lumière de ces événements, accuse le Projet GNU d'égoïsme. Il faut de sacrés préjugés pour se méprendre à ce point.

Une personne ayant autant de préjugés peut dire toutes sortes de choses injustes au sujet du Projet GNU et les trouver justifiées; ses camarades la soutiendront, parce qu'ils veulent un soutien mutuel dans le maintien de leur méprise. Ceux qui ne sont pas d'accord peuvent se faire injurier; ainsi, si je refuse de participer à une activité sous la rubrique «Linux», ils peuvent trouver cela inexcusable, et me tenir pour responsable du malaise qu'ils ressentent après coup. Quand tant de gens veulent que j'appelle le système «Linux», comment puis-je, moi qui ai tout simplement lancé son développement, ne pas obéir? Et forcément, leur refuser un discours, les rendra forcément mécontents. C'est de la coercition, aussi néfaste que celle de Microsoft!

Maintenant, vous pourriez vous demander pourquoi je fais tant d'histoires et me donne tout ce mal. Quand SIGLINUX m'a convié à parler, j'aurais pu juste dire «Non, désolé» et l'affaire en serait restée là. Pourquoi n'ai-je pas fait cela? Je suis prêt à prendre le risque d'être personnellement injurié afin d'avoir une chance de corriger l'erreur qui sape les efforts du Projet GNU.

Appeler cette variante du système GNU «Linux», fait le jeu de ceux qui choisissent leur logiciel en fonction du seul critère de l'avantage technique, sans se demander s'il respecte leur liberté. Il y a des gens comme Barr, qui veulent leur logiciel «libre de toute idéologie» et qui critiquent quiconque parle de liberté. Il y a des gens comme Torvalds qui feront pression sur notre communauté pour utiliser des programmes non-libres, et qui défieront ceux qui s'en plaignent, de fournir immédiatement un programme (techniquement) meilleur ou de se taire. Il y a des gens qui disent que les décisions techniques ne devraient pas être «politisées» par la prise en compte de leurs conséquences sociales.

Dans les années 70, les utilisateurs d'ordinateurs ont perdu leurs libertés de redistribuer et modifier le logiciel parce qu'ils ne pensaient pas à défendre leur liberté. Les utilisateurs d'ordinateurs ont regagné ces libertés dans les années 80 et 90, parce qu'un groupe d'idéalistes, le Projet GNU, croyait que cette liberté est ce qui rend un programme meilleur, et voulait travailler pour ce en quoi nous croyons.

Nous avons une liberté partielle aujourd'hui, mais notre liberté n'est pas assurée. Elle est menacée par par la CBDTPA (anciennement SSSCA), par le Broadcast «Protection» Discussion Group (voir http://www.eff.org/) qui propose d'interdire l'utilisation de logiciel libre pour accéder aux télédiffusions numériques; par les brevets logiciels (l'Europe est en train de réfléchir aux brevets logiciels), par les accords de non-divulgation de Microsoft pour les protocoles essentiels, et par tous ceux qui nous incitent à utiliser des programmes non-libres qui sont «meilleurs» (techniquement) que les programmes libres disponibles. Nous pouvons perdre à nouveau notre liberté tout comme nous l'avons perdue la première fois, si nous ne faisons pas suffisamment attention pour la protéger.

Serons-nous assez à faire attention? Cela dépend de beaucoup de choses; parmi celles-ci, quelle influence a le Projet GNU, et quelle influence a Linus Torvalds. Le Projet GNU dit : «Appréciez votre liberté!». Joe Barr dit : «Choisissez entre des programmes libres et non-libres sur les seuls critères techniques!». Si les gens attribuent à Torvalds le mérite d'être le développeur principal du système GNU/Linux, ce n'est pas seulement inexact, cela donne aussi plus de poids à son message -- et ce message dit : «Les logiciels non-libres sont bien; je les utilise et les développe moi-même». S'ils reconnaissent notre rôle, ils nous écouteront plus, et le message que nous leur donnerons est, «Ce système existe grâce à des gens qui se soucient de la liberté. Rejoignez-nous, appréciez votre liberté et ensemble nous pouvons la préserver». Consulter http://www.gnu.org/gnu/the-gnu-project.fr.html pour l'historique.

Quand je demande aux gens d'appeler le système «GNU/Linux», certains donnent des excuses farfelues et des faux prétextes. Mais nous n'avons probablement rien perdu, car ils n'étaient sans doute pas convaincus au départ. Pendant ce temps, d'autres personnes reconnaissent les raisons que je donne, et utilisent ce nom. En le faisant, ils aident à rendre conscientes d'autres personnes de la vraie raison de l'existence du système GNU/Linux, et cela accroît notre capacité à répandre l'idée que la liberté est une valeur importante.

C'est pourquoi je continue à me battre contre les préjugés, la calomnie et l'accablement. Ils heurtent mes sentiments, mais quand il aboutit, cet effort aide la campagne du Projet GNU pour la liberté.

Puisque ceci est survenu dans le contexte de Linux (le noyau) et Bitkeeper, le système de contrôle de version non-libre que Linus Torvalds utilise maintenant, j'aimerais maintenant commenter également ce problème.

### Le problème Bitkeeper

L'utilisation de Bitkeeper pour les sources Linux a un sérieux effet sur la communauté du logiciel libre, parce que quiconque veut suivre de près les patches de Linux ne peut le faire qu'en installant ce programme non-libre. Il doit y avoir des dizaines et même des centaines de hackers du noyau qui ont fait cela. La plupart d'entre eux se convainc petit à petit que ce n'est pas grave d'utiliser des logiciels non-libres, pour ne pas reconnaître la schizophrénie qui consiste à accepter la présence de Bitkeeper sur leurs machines. Que pouvons-nous faire?

Une solution est de mettre en place un autre référentiel pour les sources Linux, utilisant CVS ou un autre système de contrôle de version, et de s'arranger pour y charger les nouvelles versions automatiquement. Il pourrait utiliser Bitkeeper

pour accéder aux dernières révisions, puis installer les nouvelles révisions dans le CVS. Ce processus de mise à jour pourrait être exécuté automatiquement et fréquemment.

La FSF ne peut pas faire cela, parce que nous ne pouvons pas installer Bitkeeper sur nos machines. Nous n'avons pas de systèmes ou d'applications non-libres installés sur nos machines, et nos principes nous dictent de continuer ainsi. La création de ce référentiel devra être faite par quelqu'un d'autre qui accepte d'avoir Bitkeeper sur sa machine, à moins que quelqu'un ne trouve ou ne crée un moyen de le faire en utilisant des logiciels libres.

Les sources Linux ont elles-mêmes un problème bien plus sérieux avec les logiciels non-libres : elles en contiennent en fait quelques-uns. Pas mal de pilotes de périphériques contiennent une série de nombres qui représentent des programmes de firmware à installer dans le périphérique. Ces programmes ne sont pas des logiciels libres. Quelques nombres à déposer dans un registre de périphérique sont une chose; un programme binaire consistant en est une autre.

La présence de ces programmes seulement binaires dans les fichiers «sources» de Linux crée un autre problème : cela remet en question la légalité de la redistribution. La GPL requiert le «code source complet correspondant», et une séquence d'entiers n'est pas un code source. Pour la même raison, ajouter de tels binaires aux sources Linux viole la GPL.

Les développeurs de Linux ont prévu de placer ces programmes de firmware dans des fichiers séparés; cela prendra des années avant d'y arriver, mais quand ce sera accompli, cela résoudra le deuxième problème; nous pourrions faire une version «Linux libre» qui ne contiendrait pas les fichiers de firmware non-libres. Ce ne sera pas très bénéfique en soi si la plupart des gens utilise la version «officielle» non-libre de Linux. Cela pourrait très bien arriver, car la version libre ne fonctionne pas sur bien des plateformes sans les programmes firmware non-libres. Le Projet «Linux libre» devra découvrir ce que font les programmes firmware et écrire le code source, peut-être en langage assembleur, sur un quelconque processeur intégré sur lequel il est exécuté. C'est un travail décourageant. Cela aurait été moins décourageant si nous l'avions fait petit à petit, au fil des années, plutôt que de le laisser s'accumuler. En recrutant des gens pour faire ce travail, nous devrons surmonter l'idée, répandue par quelques développeurs de Linux, que ce travail n'est pas nécessaire.

Linux, le noyau, est souvent vu comme le porte-drapeau du logiciel libre, cependant, il est en partie non-libre. Comment est-ce arrivé? Ce problème, tout comme la décision d'utiliser Bitkeeper, reflète l'attitude du développeur à l'origine de Linux, une personne qui pense que «techniquement mieux» est plus important que liberté.

Appréciez votre liberté, ou vous la perdrez, nous apprend l'histoire. «Ne nous ennuyez pas avec la politique», répondent ceux qui ne veulent pas apprendre.

## À propos de Gnutella

«Gnutella» est à présent le nom d'un protocole de partage de fichiers par le réseau, principalement des fichiers de musique. Le nom se réfère parfois au réseau lui-même aussi bien qu'au logiciel Gnutella original. La situation est assez confuse. Pour plus d'informations sur l'origine et l'histoire de Gnutella, veuillez vous référer à l'article Wikipedia sur le sujet.

Dans tous les cas, le nom était à l'origine un jeu de mot sur «GNU» (les développeurs originaux avaient pour but de distribuer leur code sous GNU GPL et avaient peut-être dans l'idée de contribuer au projet GNU) et sur «Nutella» (une friandise que les développeurs appréciaient). Cependant, ni le logiciel original ni aucun des projets relatifs ne sont officiellement des logiciels GNU. Nous avons demandé aux développeurs de Gnutella de changer le nom pour éviter la confusion, peut être le feront-ils dans le futur.

Il existe un certain nombre de programmes en développement qui sont conçus pour utiliser le protocole Gnutella. Tel que gtk-gnutella, mutella, and gnucleus. Mais aucun de ces deux programmes n'est officiellement un logiciel GNU. GNU a son propre programme de réseau peer-to-peer, GNUnet, dont la documentation inclut une comparaison des protocoles.

La Fondation pour le Logiciel Libre est soucieuse de la liberté de copier et de modifier le logiciel; la musique sort de notre champ d'action. Mais il y a une similarité partielle entre les questions éthiques de la copie de logiciels et la copie d'enregistrements musicaux. Certains articles dans notre répertoire philosophique traitent de ce qui entoure la liberté de copie dans d'autres domaines que celui du logiciel. Quelques articles écrits par d'autres personnes vers lesquels nous avons des liens sont tout aussi pertinents.

Quel que soit le type d'information publiée qui est partagé, nous invitons prestement le public à rejeter l'idée que quelque personne ou société ait le droit naturel d'interdire le partage ainsi que de dicter exactement comment le public peut l'utiliser. Même le système judiciaire des États-Unis. rejette formellement cette idée anti-sociale.

# Pourquoi les écoles devraient utiliser exclusivement des logiciels libres

par Richard Stallman

Il existe des raisons très générales pour lesquelles tout utilisateur d'ordinateur devrait se focaliser sur les logiciels libres. Ils donnent la possibilité de contrôler notre propre ordinateur, alors qu'avec les logiciels commerciaux l'ordinateur obéit au propriétaire du logiciel - l'éditeur - et non plus au propriétaire de l'ordinateur lui-même. Avec les logiciels libres, les utilisateurs ont la liberté de coopérer, de mieux diriger leur vie. Cela s'applique aux écoles comme à tout le monde.

Mais il existe des raisons spécifiques qui concernent les écoles.

D'abord, les logiciels libres permettent aux écoles d'économiser de l'argent. Même dans les pays riches, les écoles ont un budget très serré. Les logiciels libres donnent aux écoles, comme aux autres utilisateurs, la liberté de copier et de redistribuer les logiciels, si bien que l'Éducation nationale (ou tout système éducatif) peut faire des copies pour tous les ordinateurs de toutes les écoles. Dans les pays pauvres, cela peut aider à réduire la «fracture numérique».

Cet argument économique évident, quoi qu'important, n'a qu'une portée assez marginale. En effet les développeurs de logiciels propriétaires peuvent éliminer cet inconvénient en donnant des copies aux écoles. Mais attention : une école qui accepte ce «cadeau» risque de devoir payer les futures mises à jour.

Approfondissons donc la question.

L'École devrait enseigner aux élèves des comportements qui profiteront à la société toute entière. Elle devrait promouvoir l'utilisation des logiciels libres tout comme elle promeut le recyclage. Si l'École enseigne les logiciels libres aux élèves (et aux étudiants), ceux-ci les utiliseront encore après la fin de leurs études. Cela pourra aider la société toute entière à échapper à la domination (et à la lamination) par les multinationales. Ces entreprises offrent des versions gratuites de logiciels aux écoles pour la même raison que des compagnies de tabac américaines distribuent des cigarettes gratuites.

: pour rendre les enfants dépendants (1). Ils ne feront pas de remises à ces élèves et étudiants après leurs études une fois qu'ils auront grandi.

Les logiciels libres permettent aux élèves et aux étudiants d'apprendre comment les logiciels fonctionnent. À l'adolescence, certains d'entre eux veulent tout apprendre au sujet de leur ordinateur et de ses logiciels. C'est à cet âge que les personnes qui deviendront de bons programmeurs devraient l'apprendre. Pour bien apprendre à écrire des logiciels, les élèves et les étudiants ont besoin de lire beaucoup de code source et d'écrire beaucoup de logiciels. Ils ont besoin de lire et de comprendre de vrais programmes que les gens utilisent réellement. Ils seront extrêmement curieux de lire le code source des programmes qu'ils utilisent.

Le logiciel propriétaire rejette cette soif de connaissance; il dit «le savoir que tu veux est un secret, l'apprendre est interdit!» Le logiciel libre encourage tout le monde à apprendre. La communauté du logiciel libre rejette ce «culte de la technologie», qui maintient le grand public dans l'ignorance du fonctionnement de la technologie; nous encourageons les élèves et les étudiants de tous âges et de toutes origines à lire du code source et à apprendre autant qu'ils veulent savoir. Les écoles qui utilisent les logiciels libres encouragent cela et permettent aux apprentis programmeurs doués de progresser.

La raison suivante est encore plus profonde. Nous attendons de l'École qu'elle enseigne aux élèves et étudiants des connaissances de base et des compétences utiles, mais ce n'est pas son unique mission. La mission la plus fondamentale de l'École est d'enseigner aux gens à être de bons citoyens et de bons voisins pour œuvrer avec ceux qui ont besoin d'aide. Dans le domaine de l'informatique, cela signifie enseigner à partager les logiciels. Les écoles élémentaires, par dessus tout, devraient dire à leurs élèves: «si tu apportes un logiciel à l'école, tu devras le partager avec les autres enfants». Bien entendu, l'école doit pratiquer ce qu'elle prêche: tous les logiciels installés par l'école devront être accessibles aux élèves pour être copiés, emportés à la maison et redistribués par la suite.

Enseigner l'utilisation des logiciels libres aux élèves et étudiants et prendre part à la communauté des logiciels libres est une forme d'éducation à la citoyenneté. Cela démontre aussi aux étudiants les avantages d'un modèle basé sur le service public plutôt que celui prôné par les ultralibéraux. Les logiciels libres devraient être utilisés à tous les niveaux de l'École.

1. *(1). La compagnie de tabac RJ Reynolds fut condamnée à 15 millions de dollars d'amende en 2002 pour avoir fourni des échantillons gratuits de cigarettes sur des événements ciblés sur les enfants. Voir http://www.bbc.co.uk/worldservice/sci_tech/features/health/tobaccotrial/usa.htm.*

# MyDoom et Vous

par **Richard Stallman**

J'ai grandi dans une communauté dont d'autres membres commettaient des crimes aussi graves que des meurtres. La ville de New York, avec ses 8 millions d'habitants, était le théâtre de centaines de meurtres chaque année, et la plupart étaient commis par des gens qui habitaient la ville. Les agressions et les vols étaient encore plus courants.

D'autres mauvaises actions ayant trait à l'information cette fois et non à la violence physique étaient également courants. Par exemple, des policiers de New York avaient l'habitude de mentir à la barre des témoins, et ils avaient même inventé un mot pour cela : «testilying»*. Certains programmeurs de New York s'adonnaient à une pratique légale mais socialement néfaste, celle du logiciel propriétaire : ils proposaient à d'autres personnes des ensembles logiciels séduisants sans le code source, et exigeaient d'eux la promesse de ne les partager avec personne d'autre.

Bien que ces méfaits aient été monnaie courante, je n'ai jamais vu dans ma vie quiconque essayer de condamner tous les New-yorkais pour les mauvaises actions commises par seulement quelques uns. Je n'ai vu personne supposer que tous les habitants de New York sont coupables de meurtre, de violence, de vol, de faux témoignage, ou d'avoir développé des logiciels propriétaires. Tout le monde sait bien que ce n'est pas parce que certains New-yorkais ont fait des choses pareilles qu'il faut nous considérer tous comme coupables. Ce serait de la «culpabilité par association» et chacun sait que c'est injuste. J'habite aujourd'hui Cambridge, dans le Massachussetts, qui est une ville plus petite. Des meurtres et des vols, il y en a ici aussi; je ne sais pas si les policiers de Cambridge ont l'habitude de mentir au tribunal, mais les logiciels propriétaires sont chose commune ici. Néanmoins, je n'ai jamais vu personne essayer de condamner toute la ville de Cambridge pour autant. Ici aussi les gens admettent que la culpabilité par association est une injustice.

Cependant, on ne pense pas toujours à appliquer ce principe. Ma communauté virtuelle, la communauté du logiciel libre dont je participe à la construction depuis 20 ans en développant le système d'exploitation GNU, est actuellement victime d'une campagne de culpabilité par association. Un certain nombre d'articles -- j'en ai vu quelques-uns -- essaient de faire passer toute notre communauté pour responsable du développement du virus MyDoom.

Nous pouvons être certains que des New-yorkais ont commis des meurtres, parce qu'ils ont été jugés et condamnés pour cela. Nous ne savons pas si quelqu'un dans la communauté du libre a participé au développement de MyDoom. Les développeurs n'ont pas été identifiés; ils savent qui ils sont, mais à part cela vous et moi en sommes réduits aux spéculations. Nous pouvons imaginer que des utilisateurs de GNU/Linux aient développé le virus pour s'attaquer à SCO. Nous pouvons imaginer que Microsoft ait développé le virus pour que nous soyons accusés. Nous pouvons imaginer que d'anciens employés de SCO, mécontents, aient développé le virus pour se venger. Mais aucune preuve ne vient apppuyer toutes ces spéculations.

Si l'on découvre un jour que ceux qui ont développé le virus étaient des utilisateurs de logiciels libres, alors ma communauté virtuelle sera dans la même situation que New York et Cambridge : on aura prouvé que certains de ses membres ont agi de façon destructrice.

Cela ne devrait étonner personne. La communauté du libre compte des dizaines de millions de membres, elle est plus grande que New York ou même Shanghaï. On peut difficilement imaginer que ces gens sont tous moralement irréprochables. Notre communauté est composée de gens qui décident eux-mêmes d'y appartenir parce qu'ils rejettent au moins partiellement une pratique immorale -- les logiciels propriétaires -- mais même cette attitude ne garantit pas la perfection. La présence de quelques brebis galeuses parmi des millions de gens n'est pas une surprise, et ce n'est pas une excuse pour la culpabilité par association.

Je suis persuadé que pratiquement tous les lecteurs de cet article n'ont rien à voir avec le développement du virus MyDoom. Alors si quelqu'un vous accuse, ne soyez pas sur la défensive. Vous n'avez pas plus à voir avec le virus que votre accusateur, alors gardez la tête haute et dites-le.

Si quelqu'un a des informations sur les développeurs de ce virus, j'espère qu'il ou elle va se manifester et accuser des gens bien précis avec des preuves bien précises. Mais personne ne devrait accuser sans preuves, et la culpabilité par association n'est pas excusable. Ni à New York, ni à Cambridge, ni dans le Monde (du) Libre.

* NDT: jeu de mots entre «testifying» (témoigner) et «lying» (mentir).

# 15 ans de logiciels libres

par **Richard M. Stallman**

Cela va faire juste 15 ans depuis le début du mouvement des logiciels libres et du projet GNU. Cela en fait du chemin!

En 1984, il était impossible d'utiliser un ordinateur moderne sans installer de système d'exploitation propriétaire, obtenu sous licence restrictive. Personne n'était autorisé à partager librement de logiciels avec des amis, et quasiment personne ne pouvait modifier de logiciels pour qu'ils s'adaptent à ses besoins. Les éditeurs de logiciels avaient érigé des

murs pour nous diviser les uns des autres.

Le projet GNU a été fondé pour changer tout cela. Son premier objectif: développer un système d'exploitation portable et compatible Unix qui serait à 100% libre. Pas libre à 95%, pas à 99.5%, mais à 100%. Les utilisateurs seraient ainsi libres de redistribuer le système entier, libres de modifier et contribuer à n'importe laquelle de ses parties. Le nom de ce système, GNU, est un acronyme récursif signifiant «GNU's Not Unix» (GNU n'est pas Unix), une manière de rendre hommage à Unix, tout en disant que GNU est quelque chose de différent. Techniquement, GNU est comme Unix. Mais contrairement à Unix, GNU accorde la liberté à ses utilisateurs.

Cela a pris des années de travail, des centaines de programmeurs, pour développer ce système d'exploitation. Certains étaient payés par la Fondation pour le logiciel libre et par des entreprises en logiciel libre, mais la plupart étaient des volontaires. Certains sont devenus célèbres; la plupart sont connus dans leur profession, par d'autres hackers qui utilisent ou travaillent sur leur code. Tous ensemble, ils ont aidé à libérer le potentiel du réseau informatique pour toute l'humanité.

En 1991, le dernier élément essentiel à un système semblable à Unix a été développé: Linux, le noyau libre écrit par Linus Torvalds. Aujourd'hui, la combinaison de GNU et de Linux est utilisée par des millions de gens dans le monde, et sa popularité ne fait qu'augmenter. Ce mois-ci, on annonçait la sortie de GNOME version 1.0, le bureau graphique GNU, qui, nous l'espérons, rendra le système GNU/Linux aussi facile à utiliser que n'importe quel autre système d'exploitation.

Mais notre liberté n'est pas assurée pour l'avenir. Le monde avance et nous ne pouvons pas être sûrs d'avoir les mêmes libertés dans cinq ans, juste parce que nous les avons aujourd'hui. Le logiciel libre fait face à de graves problèmes et dangers. Cela demandera des efforts pour préserver notre liberté, comme cela en a demandé pour l'obtenir. En attendant, le système d'exploitation n'est que le commencement. Nous avons maintenant besoin d'ajouter des applications libres pour couvrir l'ensemble des tâches que les utilisateurs voudraient effectuer.

Dans un futur article, j'écrirai à propos des défis spécifiques auxquels doit faire face la communauté du libre, et dans d'autres colonnes, à propos des problèmes relatifs à la liberté des utilisateurs ou au développement du système d'exploitation GNU/Linux.

# Le mouvement du logiciel libre

Les gens utilisent un système d'exploitation libre comme GNU/Linux pour des raisons variées. Nombre d'utilisateurs changent pour des raisons pratiques: parce que le système est puissant, parce qu'il est fiable, ou pour la capacité de modifier le logiciel soi-même pour qu'il réalise ce que l'on souhaite.

Ce sont de bonnes raisons, mais il y a plus en jeu que la seule commodité. L'enjeu c'est votre liberté et votre communauté.

L'idée du logiciel libre, c'est que les utilisateurs d'ordinateur méritent la liberté de former une communauté. Vous devriez avoir la liberté de vous venir en aide, en modifiant le code source pour faire ce que vous avez à faire, quelle que soit la tâche. Et la liberté d'aider votre voisin en redistribuant des copies des programmes à d'autres personnes. Et aussi la liberté d'aider à construire votre communauté, en publiant des versions améliorées afin que d'autres personnes puissent en bénificier.

Qu'un programme soit un logiciel libre ou non dépend de sa licence. Notre définition détaillée du logiciel libre montre comment nous évaluons une licence pour voir si elle produit des logiciels libres. Nous avons également des articles sur certaines licences spécifiques expliquant les avantages et les inconvénients de quelques licences qui répondent à cette définition, et pourquoi d'autres sont trop restrictives pour y répondre.

Un autre mouvement connexe fut démarré en 1998, le mouvement open source (NdT: code source ouvert). Les deux mouvements font un travail similaire et collaborent souvent sur des projets logiciels, mais nous avons des raisons philosophiques complètement différentes pour ce que nous faisons. Le mouvement open source parle seulement d'avantages pratiques du logiciel libre. C'est bon dans une certaine mesure, mais il ne fait qu'effleurer la surface du problème. Ils évitent soigneusement de soulever les questions de la liberté, de la communauté et du principe. Le mouvement du logiciel libre, lui, soulève ces questions plus profondes.

Si vous pensez que la liberté et la communauté sont importantes pour leurs propres enjeux, rejoignez-nous fièrement en utilisant le terme «logiciel libre», et aidez à faire passer le mot.

# À propos du projet GNU

Ceci est l'annonce originale du Projet GNU écrite par Richard Stallman en 1983.

L'histoire du Projet GNU diffère en beaucoup de points de ce plan initial. Par exemple, le début a été reporté en janvier 1984 et plusieurs détails en rapport avec le logiciel libre n'ont pas encore été clarifiés.

### Libérez Unix!

À partir de Thanksgiving je vais écrire un système logiciel complet compatible avec Unix appelé GNU (pour Gnu's Not Unix - Gnu N'est pas Unix) et le distribuer librement [1] à quiconque voudra l'utiliser. Il y a un grand besoin de contributions sous forme de temps, d'argent, de programmes et d'équipement.

Pour commencer, GNU comprendra un noyau ainsi que tous les utilitaires requis pour écrire et faire tourner des programmes C : éditeur, interpréteur de commandes, compilateur C, éditeur de liens, assembleur et quelques autres encore. Par la suite, nous ajouterons un formateur de texte, un YACC, un jeu Empire, un tableur et des centaines d'autres choses. Nous espérons suppléer par la suite à tout composant utile venant normalement avec un système Unix ainsi que n'importe quoi d'autre d'utile, incluant de la documentation en ligne et imprimée.

GNU sera capable de faire tourner des programmes Unix mais ne sera pas identique à Unix. Nous y apporterons toute amélioration pratique en nous basant sur notre expérience d'autres systèmes d'exploitation. En particulier, nous avons l'intention de mettre en oeuvre des noms de fichiers plus longs, des numéros de version sur les fichiers, un système de fichiers résistant aux plantages, la terminaison automatique des noms de fichiers peut-être, l'affichage indépendant du terminal et éventuellement un système de fenêtrage basé sur le Lisp grâce auquel plusieurs programmes Lisp ou programmes Unix ordinaires pourront se partager l'écran. Le C et le Lisp seront tous les deux disponibles comme langages de programmation système. Nous aurons des logiciels réseaux basés sur le protocole chaosnet du MIT, bien supérieur à UUCP. Nous pourrions aussi avoir quelque chose de compatible avec UUCP.

### Qui suis-je?

Je suis Richard Stallman, inventeur de la version originale de l'éditeur très imité Emacs, maintenant au Labo d'Intelligence Artificielle du MIT. J'ai travaillé intensément sur des compilateurs, des éditeurs, des débogueurs, des interpréteurs de commandes, sur l'Incompatible Timesharing System (Système à Temps partagé Incompatible) ainsi que sur le système d'exploitation de la Machine Lisp. J'ai été un pionnier lors du support de l'affichage indépendant du terminal dans l'ITS. De plus, j'ai implémenté un système de fichiers robuste et deux systèmes de fenêtrage pour machines Lisp.

### Pourquoi je dois écrire GNU

Je considère que la règle d'or requiert que si j'aime un programme je dois le partager avec d'autres personnes qui l'aiment. Je ne peux pas en bonne conscience signer un accord de non-révélation ni un accord de licence pour logiciel.

Afin de pouvoir continuer à utiliser les ordinateurs sans violer mes principes, j'ai décidé de rassembler une quantité suffisante de logiciels libres, de manière à ce que je puisse m'en tirer sans aucun logiciel qui ne soit pas libre.

### Comment vous pouvez participer

Je demande aux constructeurs d'ordinateurs des dons sous forme de machines et d'argent. Je demande aux individus une participation sous forme de programmes et de travail.

Un constructeur d'ordinateurs a déjà offert de nous fournir une machine. Mais nous pourrions en employer d'autres. Une conséquence à laquelle vous pouvez vous attendre si vous donnez des machines est que GNU tournera sur elles à une date proche. La machine devrait pouvoir fonctionner dans une zone résidentielle sans requérir des systèmes de courant ou de refroidissement sophistiqués.

Les programmeurs individuels peuvent apporter leur contribution en écrivant des clones de certains utilitaires Unix et en me les donnant. Pour la plupart des projets, un tel travail distribué à temps partiel serait très difficile à coordonner; les parties écrites indépendamment ne pourraient pas fonctionner ensemble. Mais pour la tâche particulière de remplacer Unix, ce problème est absent. La plupart des spécifications d'interface sont résolues par la compatibilité avec Unix. Si chacune des contributions peut fonctionner avec le reste de Unix, elle a de fortes chances de fonctionner avec le reste de GNU.

Si je recevais des dons d'argent, je pourrais engager quelques personnes à temps complet ou à temps partiel. Le salaire ne sera pas élevé mais je recherche des gens pour qui aider l'humanité est aussi important que l'argent. Je vois cela comme un moyen de permettre aux personnes dévouées de mettre toute leur énergie à travailler sur GNU en leur épargnant le besoin de gagner leur vie d'une autre manière.

Pour de plus amples informations, contactez-moi.

Courriel Arpanet:

RMS@MIT-MC.ARPA

Usenet:
…!mit-eddie!RMS@OZ
…!mit-vax!RMS@OZ

Adresse postale (États-Unis):
Richard Stallman
166 Prospect St
Cambridge, MA 02139

### [1]Piètre formulation pour «libre»

La formulation ici a été faite sans soin. L'intention était que personne n'aurait à payer pour la **permission** d'utiliser le système GNU. Mais les mots ne rendent pas cela clairement, et les gens les interprètent souvent comme indiquant que les copies de GNU devraient toujours être distribuées à bas coût ou gratuitement. Cela n'a jamais été l'intention.

### Message original

Pour être complet, le courriel original est reproduit ici, dans sa forme originale.

```
De CSvax:pur-ee:inuxc!ixn5c!ihnp4!houxm!mhuxi!eagle!mit-vax!mit-eddie!RMS@MIT-OZ
De: RMS%MIT-OZ@mit-eddie
Newsgroups: net.unix-wizards,net.usoft
Sujet: nouvelle implémentation de Unix
Date: Tue, 27-Sep-83 12:35:59 EST
Organisation: MIT AI Lab, Cambridge, MA

Libérez Unix!

À partir de Thanksgiving je vais écrire un
système logiciel complet compatible avec Unix appelé GNU
(pour Gnu's Not Unix - Gnu N'est pas Unix) et le distribuer librement
[1] à quiconque voudra
l'utiliser. Il y a un grand besoin de contributions sous forme de
temps, d'argent, de programmes et d'équipement.

Pour commencer, GNU comprendra un noyau ainsi que tous les utilitaires
requis pour écrire et faire tourner des programmes C :
éditeur, interpréteur de commandes, compilateur C,
éditeur de liens, assembleur et quelques autres encore. Par la
suite, nous ajouterons un formateur de texte, un YACC, un jeu Empire,
un tableur et des centaines d'autres choses. Nous espérons
suppléer par la suite à tout composant utile venant normalement
avec un système Unix ainsi que n'importe quoi d'autre d'utile,
incluant de la documentation en ligne et imprimée.

GNU sera capable de faire tourner des programmes Unix mais ne sera pas
identique à Unix. Nous y apporterons toute amélioration
pratique en nous basant sur notre expérience d'autres
systèmes d'exploitation. En particulier, nous avons l'intention
de mettre en oeuvre des noms de fichiers plus longs, des
numéros de version sur les fichiers, un système de
fichiers résistant aux plantages, la terminaison automatique des
noms de fichiers peut-être, l'affichage indépendant du
terminal et éventuellement un système de fenêtrage
basé sur le Lisp grâce auquel plusieurs programmes Lisp ou
programmes Unix ordinaires pourront se partager l'écran.
Le C et le Lisp seront tous les deux disponibles comme langages de
programmation système. Nous aurons des logiciels réseaux
basés sur le protocole chaosnet du MIT, bien supérieur
à UUCP. Nous pourrions aussi avoir quelque chose de compatible
avec UUCP.

Qui suis-je?

Je suis Richard Stallman, inventeur de la version originale de
l'éditeur très imité Emacs, maintenant au Labo
d'Intelligence Artificielle du MIT. J'ai travaillé
intensément sur des compilateurs, des éditeurs, des
débogueurs, des interpréteurs de commandes, sur
l'Incompatible Timesharing System (Système à Temps
partagé Incompatible) ainsi que sur le système
d'exploitation de la Machine Lisp.  J'ai été un pionnier
lors du support de l'affichage indépendant du terminal dans
l'ITS. De plus, j'ai implémenté un système de
fichiers robuste et deux systèmes de fenêtrage pour
machines Lisp.


Pourquoi je dois écrire GNU

Je considère que la règle d'or requiert que si j'aime un
programme je dois le partager avec d'autres personnes qui
l'aiment. Je ne peux pas en bonne conscience signer un accord de
non-révélation ni un accord de licence pour logiciel.
```

```
Afin de pouvoir continuer à utiliser les ordinateurs sans
violer mes principes, j'ai décidé de rassembler une
quantité suffisante de logiciels libres, de manière
à ce que je puisse m'en tirer sans aucun logiciel qui ne soit
pas libre.


Comment vous pouvez participer

Je demande aux constructeurs d'ordinateurs des dons sous forme de
machines et d'argent. Je demande aux individus une participation sous
forme de programmes et de travail.

Un constructeur d'ordinateurs a déjà offert de nous fournir
une machine. Mais nous pourrions en employer d'autres. Une
conséquence à laquelle vous pouvez vous attendre si vous
donnez des machines est que GNU tournera sur elles à une date
proche. La machine devrait pouvoir fonctionner dans une zone
résidentielle sans requérir des systèmes de courant
ou de refroidissement sophistiqués.

Les programmeurs individuels peuvent apporter leur contribution en
écrivant des clones de certains utilitaires Unix
et en me les donnant. Pour la plupart des projets, un tel travail
distribué à temps partiel serait très difficile
à coordonner; les parties écrites indépendamment
ne pourraient pas fonctionner ensemble. Mais pour la tâche
particulière de remplacer Unix, ce problème est
absent. La plupart des spécifications d'interface sont
résolues par la compatibilité avec Unix. Si chacune des
contributions peut fonctionner avec le reste de Unix, elle a de fortes
chances de fonctionner avec le reste de GNU.

Si je recevais des dons d'argent, je pourrais engager quelques
personnes à temps complet ou à temps partiel. Le salaire
ne sera pas élevé mais je recherche des gens pour qui
aider l'humanité est aussi important que l'argent. Je vois cela
comme un moyen de permettre aux personnes dévouées de
mettre toute leur énergie à travailler sur GNU en leur
épargnant le besoin de gagner leur vie d'une autre
manière.

Pour de plus amples informations, contactez-moi.
Courriel Arpanet:
  RMS@MIT-MC.ARPA

Usenet:
  ...!mit-eddie!RMS@OZ
  ...!mit-vax!RMS@OZ

Adresse postale (États-Unis):
  Richard Stallman
  166 Prospect St
  Cambridge, MA 02139
```

# Le Manifeste GNU

Le manifeste GNU, reproduit ci-dessous, a été écrit par Richard Stallman au commencement du projet GNU pour encourager la participation et le soutien de tous. Au cours des premières années, il y a eu quelques petites mises à jour pour tenir compte des développements, mais il nous semble maintenant plus judicieux de le laisser en l'état.

Nous avons appris depuis qu'il y avait quelques incompréhensions; celles-ci peuvent être corrigées en changeant quelques mots. Des notes de bas de page ajoutées en 1993 aident à clarifier ces points.

Pour les dernières informations sur les logiciels GNU actuellement disponibles, veuillez vous référer à notre site web, en particulier notre liste de logiciels. Pour savoir comment contribuer, consultez http://www.gnu.org/help.

### Qu'est ce que GNU? Gnu N'est pas Unix!

GNU, l'acronyme de GNU's Not Unix (GNU n'est pas Unix), est le nom du système complet de logiciels compatible Unix que j'écris pour pouvoir le donner librement à tous ceux qui en auraient besoin.

Pour l'instant, nous avons un éditeur de texte, Emacs, utilisant le Lisp pour écrire des commandes d'édition, un débogueur, un générateur d'analyseurs syntaxiques compatible avec YACC, un éditeur de liens, et environ trente-cinq autres utilitaires. Un shell (un interprète de commandes) est presque terminé. Un nouveau compilateur C portable et optimisé s'est compilé lui-même et devrait être disponible cette année. Un premier noyau existe, mais nécessite plus de fonctions pour émuler Unix. Quand le noyau et le compilateur seront terminés, il sera possible de distribuer un système GNU propice au développement. Nous utiliserons TeX comme formateur de textes, mais un nroff est en cours de développement. Nous utiliserons aussi le système libre et portable X Window System. Par la suite, nous ajouterons un Common Lisp portable, le jeu Empire, un tableur et des centaines d'autres choses, plus une documentation en ligne. Nous espérons fournir, finalement, tout ce qui peut être utile et qui est normalement inclus dans un système Unix et plus encore.

GNU pourra exécuter des programmes Unix mais ne sera pas identique à Unix. Nous ferons toutes les améliorations que nous jugerons appropriées, en nous fondant sur nos expériences avec d'autres systèmes d'exploitation. En particulier, nous prévoyons d'avoir des fichiers avec des noms longs, des numéros de version de fichier, un système de fichiers à tolérance de panne, éventuellement un système de complétion des noms de fichiers, un dispositif d'affichage indépendant du terminal, et peut-être, finalement, un système de fenêtrage fondé sur Lisp, au travers duquel plusieurs programmes Lisp et autres programmes Unix pourront partager un écran. Le C et le Lisp seront tous les deux disponibles comme langages de programmation système. Nous essayerons de supporter UUCP, MIT Chaosnet, et les protocoles de l'Internet pour la communication.

Initialement, GNU vise les machines de classe 68000/16000 avec de la mémoire virtuelle, car ce sont les machines les plus simples pour le faire fonctionner. Nous laissons l'effort supplémentaire pour l'adapter sur de plus faibles machines à ceux qui voudront l'utiliser sur celles-ci.

Pour éviter d'horribles confusions, veuillez bien prononcer le 'G' de 'GNU' quand vous parlez de ce projet.

### Pourquoi dois-je écrire GNU

J'estime que la Règle d'or est que, si j'aime un programme, je dois le partager avec d'autres qui aiment ce programme. Les éditeurs de logiciels cherchent à diviser et à conquérir les utilisateurs, en interdisant à chacun de partager avec les autres. Je refuse de rompre la solidarité avec les autres utilisateurs de cette manière. Je ne peux pas, en mon âme et conscience, signer un accord de non-divulgation ou une licence de logiciels. Pendant des années, j'ai oeuvré au sein du Laboratoire d'Intelligence Artificielle du MIT pour résister à ces tendances, mais finalement, ils sont allés trop loin : je ne pouvais pas rester dans une institution où de telles choses avaient lieu contre ma volonté.

Pour pouvoir continuer à utiliser les ordinateurs en accord avec ma conscience, j'ai décidé de rassembler un ensemble suffisant de logiciels libres, pour pouvoir me débrouiller sans logiciels non libres. J'ai démissionné du laboratoire d'Intelligence Artificielle pour que le MIT ne puisse invoquer toutes les excuses légales pour m'empêcher de distribuer GNU librement.

### Pourquoi GNU sera compatible avec Unix

Unix n'est pas pour moi le système parfait, mais il n'est pas trop mauvais. Les fonctions essentielles d'Unix semble être les bonnes, et je pense pouvoir compléter ce qui manque à Unix sans les gâcher. Et un système compatible Unix serait commode à adopter par de nombreuses personnes.

### Disponibilité de GNU

GNU n'est pas dans le domaine public. Tout le monde aura le droit de modifier et redistribuer GNU, mais aucun distributeur ne pourra restreindre ces futures redistributions. C'est-à-dire que des modifications propriétaires seront interdites. Je veux être sûr que toutes les versions de GNU restent libres.

### Pourquoi beaucoup de programmeurs veulent contribuer.

J'ai rencontré beaucoup de programmeurs enthousiasmés par GNU et qui souhaitaient contribuer.

De nombreux programmeurs sont insatisfaits de la commercialisation de logiciels systèmes. Il se peut que cela leur permette de gagner plus d'argent, mais cela les amène forcement à se sentir en conflit avec les autres programmeurs en général, plutôt que d'être camarades. L'acte fondamental d'une amitié entre des programmeurs est le partage des programmes; les arrangements commerciaux typiquement utilisés de nos jours interdisent aux programmeurs de considérer les autres comme des amis. L'acheteur de programmes doit choisir entre l'amitié et l'obéissance à la loi. Naturellement, un grand nombre décident que l'amitié est plus importante. Mais ceux qui respectent la loi se sentent souvent mal à l'aise face à ce seul choix. Ils sont désabusés et pensent que programmer n'est qu'une façon de gagner de l'argent.

En utilisant GNU plutôt que des programmes propriétaires, nous pouvons être amicaux envers tout le monde tout en respectant la loi. De plus, GNU est une source d'inspiration et une bannière sous laquelle d'autres peuvent nous rejoindre dans le partage. Ceci peut nous procurer un sentiment d'harmonie, impossible à atteindre avec des logiciels qui ne sont pas libres. Pour environ la moitié des programmeurs avec lesquels j'ai discuté, c'est une satisfaction importante que l'argent ne peut pas remplacer.

### Comment vous pouvez contribuer

(Actuellement, pour la liste des tâches logicielles sur lesquelles travailler, consultez la liste des tâches GNU.
Pour d'autres manières de contribuer, consultez http://www.gnu.org/help/help.fr.html.)

Je demande aux fabricants d'ordinateurs de faire don de machines et d'argent. Je demande aux individus de faire don de programmes et de travail.

Une des conséquences à laquelle vous pouvez vous attendre si vous donnez des machines, c'est que GNU tournera dessus très rapidement. Les machines doivent être complètes, prêtes à l'utilisation, sans besoin de système particulier de climatisation ou d'alimentation.

J'ai trouvé de nombreux programmeurs impatients de contribuer à mi-temps pour GNU. Pour la plupart des projets, un tel travail distribué à temps partiel serait très difficile à coordonner; les diverses parties codées indépendamment ne fonctionneraient pas ensemble. Mais ce problème n'existe pas dans le cas du projet de remplacement d'Unix. Un système Unix complet contient des centaines d'utilitaires, chacun étant documenté séparément. La plupart des spécifications des interfaces sont déterminées par la compatibilité avec Unix. Si chaque collaborateur peut écrire un remplacement compatible pour un seul utilitaire Unix et l'intégrer proprement à la place de l'original sur un système Unix, il s'ensuit que ces utilitaires fonctionneront ensemble sans problème. Même en faisant quelques concessions à la loi de Murphy qui créera quelques problèmes inattendus, l'assemblage de ces composants sera une tâche réalisable. (Le noyau demandera quand même une communication plus soutenue et sera réalisé par un petit groupe.)

Si je reçois des dons financiers, je pourrais embaucher quelques personnes à temps plein ou à mi-temps. Le salaire ne sera peut-être pas très élevé par rapport au marché, mais je cherche des personnes pour lesquelles l'esprit de communauté est aussi important que l'appât du gain. Je considère que c'est une façon de permettre à quelques personnes dévouées de consacrer toutes leurs ressources au projet GNU, en leur évitant d'avoir à gagner leur vie autrement.

### Pourquoi tous les utilisateurs en bénéficieront.

Une fois GNU achevé, tout le monde pourra obtenir de bons logiciels libres comme l'air [2].

Ceci représente beaucoup plus que l'économie d'une licence Unix. Cela veut dire que l'on va éviter la duplication inutile du travail de programmation. Cet effort pourra plutôt se diriger vers l'avancement du domaine informatique.

Les sources du système complet seront disponibles pour tous. Et cela aura pour résultat qu'un utilisateur qui a besoin de changer un composant du système aura toujours la liberté d'effectuer des changements lui-même, ou d'engager une personne ou une société capable d'effectuer ces changements pour lui. Les utilisateurs ne seront plus à la merci d'une seule personne ou d'une seule société qui possède les sources du programme et qui est la seule à pouvoir effectuer des changements.

Les écoles pourront fournir un milieu beaucoup plus éducatif en encourageant chaque étudiant à étudier et à améliorer le code du système. Le laboratoire informatique d'Harvard avait comme politique de n'installer aucun programme sur le système si ses sources n'étaient pas disponibles, et ils soutenaient cette politique en refusant carrément d'installer certains programmes. Cela m'a beaucoup inspiré.

Enfin, les frais engendrés par les questions d'appartenance et de droits des logiciels ne seront plus d'actualité.

Les mesures pour faire payer les licences des programmes et de leurs copies génèrent toujours un coût important pour la société en général, à cause des mécanismes nécessaires pour calculer combien (c'est-à-dire quels programmes) chacun doit payer. Et il faudrait un État policier pour appliquer parfaitement ces mesures. Prenons une station orbitale, où l'air doit être fabriqué à un coût important: facturer chaque litre inspiré peut être justifié, mais porter un masque/compteur toute la journée et toute la nuit est intolérable même si on a de quoi payer la facture. Et les caméras de surveillance placées partout pour vérifier que vous ne retirez jamais le masque/compteur seraient inacceptables. Il vaut mieux financer la fabrication de l'air avec une taxe par personne et se débarrasser des masques.

De copier tout ou des parties d'un logiciel semble aussi naturel à un programmeur que de respirer, tout aussi productif. Cela aussi devrait être libre.

### Quelques objections facilement contrées aux objectifs de GNU

**«Personne ne s'en servira si c'est gratuit, car cela veut dire que l'on ne peut compter sur aucun support.»**
**«Il faut faire payer le logiciel pour financer le service après-vente.»**
Il y a des gens qui préfèrent payer pour GNU et le service plutôt que d'obtenir GNU sans service. Une société qui propose le service uniquement à ceux qui ont obtenu GNU gratuitement, devrait être rentable [3].

Nous devons faire la distinction entre le support en termes de réel travail de programmation et le simple support d'assistance. On ne peut pas compter sur le premier de la part d'un simple revendeur. Si votre problème n'est pas suffisamment répandu, le revendeur vous enverra balader.

Si votre société a besoin d'un support fiable, la seule façon est d'avoir toutes les sources et tous les outils nécessaires. À partir de là, vous pouvez engager n'importe quelle personne qualifiée pour régler les problèmes. Vous n'êtes pas à la merci d'une seule personne. Avec Unix, le prix des sources rend cette solution inabordable pour la plupart des sociétés. Avec GNU ce sera facile. Il serait éventuellement concevable qu'il n'y ait personne de disponible, mais les modalités de distribution ne sont pas responsables de ce problème. GNU ne propose pas de régler tous les problèmes, mais seulement quelques-uns.

Pendant ce temps, les utilisateurs qui n'y connaissent rien en informatique ont besoin d'assistance et de personnes pour les aider à faire ce qu'ils pourraient faire très bien eux-mêmes si seulement ils s'y connaissaient.

De tels services pourraient être proposés par des sociétés qui ne font que des cours d'initiation et des réparations. S'il est vrai que les utilisateurs préfèrent dépenser de l'argent pour un logiciel intégrant un service après-vente, ils seront aussi d'accord pour payer simplement le service, ayant obtenu le logiciel gratuitement. Les sociétés de service se feront concurrence sur la qualité et le prix de leurs prestations. Les utilisateurs ne seront pas limités à une société particulière. En même temps, ceux d'entre nous qui n'ont pas besoin du service pourront utiliser le logiciel sans payer le service.

**«On ne peut pas être connu sans publicité et il faut payer le logiciel pour financer la publicité.»**
**«Ca ne sert à rien de faire de la publicité si on peut obtenir le logiciel gratuitement.»**

Il existe plusieurs formes de publicité gratuite ou bon marché qui peuvent être utilisées pour informer de nombreux utilisateurs au sujet de GNU. Cependant, il est peut-être vrai que l'on peut atteindre plus d'utilisateurs avec de la publicité. Si cela est vrai, une société qui fait de la publicité sur le service payant de copie et de distribution de GNU doit être suffisamment rentable pour assurer sa propre publicité et bien davantage. Ainsi, seuls les utilisateurs qui bénéficient de la publicité la payent.

En revanche, si de nombreuses personnes obtiennent GNU par leurs relations, de telles sociétés ne seraient pas rentables, et cela démontrerait que la publicité n'était pas vraiment nécessaire pour répandre GNU. Pourquoi est-ce que les partisans du libre échange ne veulent pas laisser cette décision au marché libre [4]?

**«Ma société a besoin d'un système propriétaire pour être compétitive.»**

GNU va retirer les systèmes d'exploitation du domaine de la concurrence. Vous ne pourrez pas être avantagé dans ce domaine, mais votre concurrent non plus. Vous pourrez rivaliser dans d'autres domaines. Si votre domaine est la vente de systèmes d'exploitation, vous n'aimerez pas GNU, et c'est tant pis pour vous. Si votre domaine est différent, GNU peut vous éviter d'être poussé dans le domaine onéreux de la vente de systèmes d'exploitation.

J'aimerais bien voir le développement de GNU financé par des dons de fabricants et utilisateurs, réduisant ainsi les coûts pour chacun [5].

**«Les programmeurs ne méritent-ils pas d'être récompensés pour leur créativité?»**

Si quelque chose mérite une récompense, c'est bien la contribution sociale. La créativité peut être une contribution sociale, mais seulement tant que la société est libre de profiter des résultats. Si les programmeurs méritent d'être récompensés pour la création de logiciels innovants, de même, ils méritent d'être punis s'ils limitent l'utilisation de leurs programmes.

**«Un programmeur ne devrait-il pas avoir le droit de demander une récompense pour sa créativité?»**

Il n'y a rien de mal à vouloir être payé pour son travail, ou à chercher à augmenter ses revenus, tant que l'on n'utilise pas de méthodes destructives. Mais les méthodes pratiquées dans le domaine du logiciel sont fondées sur la destruction. Extraire de l'argent aux utilisateurs d'un programme en limitant son utilisation est destructeur, car ces restrictions réduisent l'utilité du programme. Ce qui à son tour réduit la richesse apportée par ce programme à l'humanité. Quand le choix de limiter est délibéré, les conséquences néfastes qui en découlent sont de la destruction délibérée.

La raison pour laquelle un bon citoyen ne doit pas utiliser de telles méthodes destructrices pour augmenter sa richesse personnelle est que si tout le monde faisait de même, il y aurait un appauvrissement général dû à la destruction mutuelle. C'est ce que l'on appelle la morale kantienne, ou la Règle d'or. Puisque je n'apprécie pas les conséquences qui adviennent si tout le monde fait de la rétention d'informations, je ne dois pas trouver acceptable qu'un individu le fasse. Plus précisément, le désir d'être récompensé pour sa création ne justifie pas que l'on prive le monde en général de toute ou partie de cette créativité.

**«Les programmeurs ne vont-ils pas mourir de faim?»**

Je peux répondre que personne n'est forcé d'être un programmeur. La plupart d'entre nous n'arriverait pas à se faire payer pour faire des grimaces dans la rue. Mais nous ne sommes pas pour autant condamnés à passer notre vie dans la rue à faire des grimaces et à mourir de faim. On fait autre chose.

Mais c'est une mauvaise réponse, car elle accepte l'a priori de la question : c'est-à-dire que sans possession du logiciel, les programmeurs ne pourraient pas recevoir le moindre sou. C'est, soi-disant, tout ou rien.

La vraie raison pour laquelle les programmeurs ne vont pas mourir de faim est qu'il sera quand même possible pour eux d'être payés pour programmer; seulement, peut-être pas autant qu'actuellement.

La restriction des copies n'est pas la seule base des affaires du domaine des logiciels. C'est la base la plus commune, car c'est la plus rentable. Si ces restrictions étaient interdites ou rejetées par le client, les éditeurs passeraient à d'autres formes d'organisation, qui sont actuellement moins utilisées. Il y a de nombreuses façons d'organiser une entreprise.

Il est probable qu'avec ce nouveau système, la programmation serait moins rentable qu'elle ne l'est actuellement. Mais ce n'est pas un argument valable contre le changement. Il n'est pas considéré comme injuste que les caissières gagnent ce qu'elles gagnent. Si les programmeurs gagnaient la même chose, ce ne serait pas non plus une injustice. (En pratique, ils gagneraient quand même beaucoup plus.)

**«Les gens n'ont-ils pas le droit de gérer l'utilisation de leur créativité? »**

«Contrôler l'utilisation que l'on fait de ses idées» revient à contrôler la vie des autres; et c'est souvent utilisé pour leur rendre la vie plus difficile.

Ceux qui ont étudié le problème de la propriété intellectuelle [6] à fond (les avocats, les juristes, etc.) soutiennent qu'il n'existe aucun droit intrinsèque à la propriété intellectuelle. Les différents droits de soi-disant propriété intellectuelle reconnus par le gouvernement ont été créés par des législations précises dans des buts bien précis.

Par exemple, le système de brevets a été établi pour encourager les inventeurs à divulguer les détails de leurs inventions. Sa raison d'être était d'aider la société plutôt que d'avantager les inventeurs. À l'époque, la durée de vie de 17 ans pour un brevet était court par rapport à la cadence des évolutions technologiques. Puisque les brevets ne concernent que les fabricants, pour lesquels le coût et l'effort d'établir une licence sont minimes comparés à la mise en production, les brevets ne font souvent pas trop de tort. Ils ne gênent pas la plupart des individus qui utilisent des produits brevetés.

Le concept de droit d'auteur n'existait pas dans l'Antiquité, les auteurs copiaient souvent, et beaucoup, l'oeuvre des autres. Cette pratique était utile, et c'est de cette seule façon que les travaux de certains auteurs ont survécus ne serait-ce qu'en partie. Le système du droit d'auteur a été créé expressément pour encourager les auteurs. Dans le domaine pour lequel ce système a été inventé, les livres, qui pouvaient seulement être copiés en imprimerie ne causait pas beaucoup de tort, et ne gênait pas la plupart des personnes qui lisaient ces livres.

Tous les droits de propriété intellectuelle ne sont que des licences accordées par la société parce que nous pensions, à tort ou à raison, que la société en général bénéficierait de ces accords. Mais dans chaque situation précise, nous devons nous demander : bénéficierons-nous vraiment d'accorder cette licence? Quels actes autorisons-nous avec cette licence?

Le cas des logiciels aujourd'hui est très différent de celui des livres il y a un siècle. Le fait que la manière la plus répandue de copier un programme est entre voisins, le fait qu'un programme contient à la fois du code source et du code binaire bien distinct, et le fait qu'un programme est utilisé plutôt que lu comme divertissement, se réunissent pour créer une situation dans laquelle celui qui applique le droit d'auteur fait du tort à la société, matériellement et spirituellement; cette personne ne devrait pas appliquer le droit d'auteur, que la loi l'y autorise ou non.

**«La compétition permet de mieux faire les choses.»**

Le paradigme de la compétition est une course : en récompensant le vainqueur, nous encourageons tout le monde à courir plus vite. Quand le capitalisme fonctionne réellement de cette façon, tout marche bien; mais ses partisans ont tort s'ils pensent que cela fonctionne toujours de cette façon. Si les coureurs oublient le pourquoi de la récompense, et deviennent obsédés par la victoire, quelles que soient les méthodes employées, ils risquent de trouver d'autres stratégies telles qu'agresser les autres concurrents. Si tous les coureurs s'engageaient dans un combat, ils finiraient tous en retard.

Les logiciels propriétaires et secrets sont l'équivalent moral des coureurs qui se battent. Malheureusement, le seul arbitre que l'on ait ne semble pas s'opposer aux combats ; il se contente de les réguler («Pour dix mètres parcourus, vous avez le droit de tirer un coup de feu»). Il devrait en fait séparer les combattants, et punir les coureurs qui tentent de se battre.

**«Les gens s'arrêteront-ils de programmer sans l'appât du gain?»**

En fait, beaucoup de gens programmeront même sans aucun bénéfice financier. La programmation exerce une fascination irrésistible pour quelques-uns, généralement ceux qui programment le mieux. Il n'y a aucune pénurie de musiciens professionnels qui continuent à jouer, même sans l'espoir de pouvoir en faire leur gagne-pain.

Mais en fait cette question, bien qu'elle soit souvent posée, ne convient pas à la situation. Les salaires des programmeurs ne disparaîtront pas mais diminueront peut-être. La question devient donc, trouvera-t-on des programmeurs qui travailleront pour une moindre rémunération? D'après mon expérience, la réponse est oui.

Pendant plus de dix ans, plusieurs des meilleurs programmeurs du monde ont travaillé au laboratoire d'Intelligence Artificielle du MIT pour un salaire bien moins important que ce qu'ils auraient touchés ailleurs. Ils étaient récompensés de plusieurs autres manières: la notoriété, le respect des autres, par exemple. Et la créativité est une récompense en soi.

Et puis la plupart sont partis pour faire le même travail pour beaucoup plus d'argent.

Les faits démontrent que les gens programmeront pour d'autres raisons que l'accumulation de richesses; mais si on leur propose beaucoup plus d'argent, ils s'y attendront finalement et l'exigeront. Les organismes qui payent moins bien ont du mal face à ceux qui payent bien, mais ils devraient pouvoir s'en sortir si les gros payeurs sont bannis.

**«Nos besoins en programmeurs sont tellement importants que s'ils interdisent le partage, nous ne pouvons que leur obéir.»**

La situation n'est jamais aussi désespérée au point d'être amené à obéir à une telle interdiction.

**«Les programmeurs doivent bien gagner leur pain.»**

À court terme, cela est vrai. Cependant, il y a de nombreuses possibilités offertes à un programmeur pour vivre décemment sans pour autant vendre le droit d'utiliser un programme. Cette façon est la plus répandue actuellement, car c'est celle qui engendre le plus de profit pour les programmeurs et les hommes d'affaires, et non parce que c'est la seule manière de gagner son pain. Vous pouvez facilement trouver d'autres manières si vous le voulez. Voici quelques exemples.

Un fabricant arrivant avec un nouvel ordinateur payera pour le portage des systèmes d'exploitation sur le nouveau matériel.

L'offre de services d'enseignement, de conseil et de maintenance peut permettre la création d'emplois.

Les personnes avec des idées nouvelles peuvent distribuer des logiciels librement [7], en demandant des dons aux utilisateurs satisfaits ou en offrant un service de conseil. J'ai déjà rencontré des personnes travaillant ainsi.

Les utilisateurs ayant des besoins en commun, peuvent créer des groupes d'utilisateurs et payer des cotisations. Un tel groupe pourrait faire appel à une société de développement pour écrire les programmes spécifiques pour ses membres.

Toutes sortes de développement pourraient être financés par une taxe sur les logiciels :

supposons que chaque personne qui achète un ordinateur doive payer x pour cent du prix en tant que taxe sur les logiciels. Le gouvernement reverserait cette somme à un organisme tel que la NSF pour subvenir au développement de logiciels.

Mais si l'acheteur fait lui-même un don au développement de logiciels, il pourra être crédité pour cette taxe. Il pourrait donner au projet de son choix, car il espérera profiter des résultats à l'achèvement du projet. Il pourra donc être exempté de la taxe si le montant de sa donation recouvre celle-ci.

Le taux de la taxe pourrait être déterminé par un vote de ceux qui la payent, pondéré par le montant de l'imposition.

Les conséquences :

- La communauté des utilisateurs soutient le développement des logiciels.
- Cette communauté décide du niveau du soutien nécessaire.
- Pour les utilisateurs qui se soucient de quels projets profitent de leur participation, ils pourront les choisir eux-mêmes.

À terme, rendre les programmes libres est un pas vers le monde d'après pénurie, quand personne ne devra travailler très dur juste pour survivre. Les gens seront libres de se consacrer à des activités ludiques telles que la programmation, après avoir, bien entendu, passé les dix heures par semaine nécessaires pour des oeuvres telles que la législation, la thérapie de famille, la réparation de robots et l'exploration d'astéroïdes. Il n'y aura donc plus besoin de gagner sa vie en programmant.

Nous avons déjà beaucoup réduit la quantité de travail que la société entière doit fournir pour sa productivité, mais seulement une petite part se traduit en temps de loisirs pour les travailleurs, car beaucoup d'activités non productives sont nécessaires pour accompagner l'activité productive. Les raisons principales sont la bureaucratie et la lutte isométrique contre la concurrence. Le logiciel libre va réduire grandement ces fuites du domaine du développement logiciel. Nous devons faire cela, pour que les gains de productivité se traduisent en moins d'heures de travail pour nous.

### Notes

[1] Le choix des mots ici était irréfléchi. L'idée était que personne n'aurait à payer l'autorisation d'utiliser le système GNU. Mais cela n'était pas clair, et les gens ont souvent compris que les copies de GNU devaient toujours être distribuées peu chères ou gratuitement. Cela n'a jamais été l'intention; plus tard, le manifeste mentionne pour les sociétés la possibilité de fournir un service de distribution rentable. Par la suite, j'ai appris à bien faire la distinction entre «free» dans le sens de libre, et «free» dans le sens de gratuit. Le logiciel libre est un programme que les utilisateurs ont la liberté de distribuer et de modifier. Certains utilisateurs peuvent obtenir des copies gratuitement, tandis que d'autres les paieront et si cela peut rapporter de quoi financer l'amélioration de programmes, tant mieux. L'important est que toute personne détenant une copie a le droit de l'utiliser en collaboration avec d'autres.

[2] Voilà un autre endroit où je n'ai pas fait la distinction entre les deux définitions de «libre». La phrase telle quelle n'est pas fausse, vous pouvez obtenir des copies de logiciels GNU gratuitement, par vos amis ou par l'Internet. Mais elle suggère effectivement la mauvaise idée. [N.d.t. - En français, la distinction entre «libre» et «gratuit» est évidente.]

[3] De telles sociétés existent actuellement.

[4]Depuis dix ans, la Free Software Foundation a levé la plupart de ses fonds à travers un service de distribution, bien qu'elle soit une association plutôt qu'une société. Vous pouvez commander des logiciels, des livres, ... à la FSF.

[5] Un groupe de sociétés a réuni des fonds vers 1991 pour financer la maintenance du compilateur C de GNU.

[6] Dans les années 80, je n'avais pas encore réalisé à quel point il était déroutant de parler du «problème» de la «propriété intellectuelle». Ce terme est évidemment partial; plus subtil dans le fait qu'il mélange diverses lois disparates qui traitent de problàmes tràs différents. De nos jours, je presse les gens à rejeter totalement le terme «propriété intellectuelle», de peur qu'il ne conduise d'autres personnes à supposer que ces lois forment une solution cohérente. La façon d'être clair est de parler de brevets, de droits d'auteur, et de marques déposées séparément. Voir des explications plus détaillées sur la manière dont ce terme sème la confusion et le parti-pris.

[7] Par la suite, nous avons appris la distinction entre «logiciel libre» et «graticiel» («freeware»). Le terme «graticiel» signifie logiciel que vous êtes libre de redistribuer, mais généralement, vous n'êtes pas libre de l'étudier ou de changer le code source; donc la plupart d'entre eux ne sont pas des logiciels libres. Voir la page Termes prêtant à confusion, que vous devriez éviter pour plus d'explications.

## Histoire du système GNU

Le système d'exploitation GNU a développé un système complet de logiciels libres qui a une compatibilité ascendante avec Unix. «GNU» signifie «GNU's Not Unix». Richard Stallman a fait l'annonce initiale du projet GNU en septembre 1983. Une version plus longue appelée Manifeste GNU (31ko) a été publiée en septembre 1985. Il a été traduit dans différentes langues.

Le nom «GNU» fut choisi parce qu'il satisfait à quelques conditions : premièrement c'était un acronyme récursif pour «GNU's Not Unix», en second lieu, c'était un mot réel, et troisièment il était drôle à prononcer (ou à chanter).

Le mot «libre» ci-dessus fait référence à la liberté, et non au prix. Vous pouvez avoir payé des copies d'un logiciel GNU, ou les avoir obtenues sans frais. Mais quoi qu'il en soit, vous possédez trois libertés spécifiques. Premièrement, la liberté de copier le programme et de le donner à vos amis ou collègues. Deuxièmement, la liberté de modifier le programme comme vous l'entendez, grâce à un accès complet au code source. Troisièmement, la liberté de distribuer une version améliorée et ainsi aider la communauté. (Si vous redistribuez des logiciels GNU, vous pouvez percevoir un droit pour le transfert de la copie, ou vous pouvez les offrir).

Le projet pour développer le système d'exploitation GNU est appelé le «Projet GNU». Le projet GNU a été conçu en 1983 comme une manière de rétablir l'esprit coopératif qui prévalait dans la communauté informatique aux premiers jours, pour nouveau à rendre la coopération possible en supprimant les barrières à la coopération imposées par les possesseurs de logiciels propriétaires.

En 1971, quand Richard Stallman démarra sa carrière au MIT, il travaillait dans un groupe qui utilisait exclusivement des logiciels libres. Même les sociétés informatiques distribuaient des logiciels libres. Les programmeurs étaient libres de coopérer entre eux, et ils le faisaient souvent.

Au début des années 80, presque tous les logiciels étaient des logiciels propriétaires (18ko), ce qui signifie que les propriétaires de logiciels interdisaient et empêchaient la coopération entre utilisateurs. Ceci rendit nécessaire le projet GNU.

Chaque utilisateur d'ordinateur a besoin d'un système d'exploitation; s'il n'y a pas de système d'exploitation libre, alors vous ne pouvez même pas commencer à utiliser un ordinateur sans avoir recours au logiciel propriétaire. Ainsi la première question à l'ordre du jour du logiciel libre a été édemment d'être un système d'exploitation libre.

Nous avons décidé de rendre le système d'exploitation compatible avec Unix parce que le concept d'ensemble avait déjà fait ses preuves et est portable, et parce que la compatibilité rend plus facile le passage de Unix à GNU, pour les utilisateurs d'Unix.

Un système d'exploitation de type Unix est bien plus qu'un noyau; il inclut également des compilateurs, des éditeurs, des éditeurs de texte, des logiciels de courrier, et bien d'autres choses. Ainsi, l'écriture d'un système d'exploitation complet est un travail important. Cela a pris de nombreuses années. La Free Software Foundation a été foundée en octobre 1985, dans le but de lever des fonds pour aider au développement de GNU.

Le but initial d'un système d'exploitation libre a été atteint. Au début des années 90, nous avions trouvé ou écrit les composants majeurs, excepté un : le noyau. Puis le noyau libre Linux, un noyau de type Unix, fut développé par Linus Torvald en 1991 et fut un logiciel libre en 1992. La combinaison du noyau Linux avec le système à peu près complet GNU a pour résultat un système d'exploitation complet : un système d'exploitation GNU/Linux. On estime que des centaines de milliers de personnes utilisent actuellement des systèmes d'exploitation GNU/Linux, typiquement des distributions comme Slackware, Debian, Red Hat, etc.

Cependant, le projet GNU n'est pas limité aux systèmes d'exploitation. Nous aspirons à fournir tout l'éventail du logiciel, tout ce qui peut intéresser de nombreux utilisateurs. Ceci inclut des logiciels applicatifs. Consulter le Répertoire du logiciel libre pour un catalogue d'applications logicielles libres.

Nous voulons également fournir des logiciels pour les utilisateurs qui ne sont pas des experts en informatique. Nous avons donc développé une interface graphique (appelée GNOME) pour aider les débutants à utiliser le système GNU.

Nous voulons également fournir des jeux et autres logiciels récréatifs. Pleins de jeux libres sont déjà disponibles.

Jusqu'où le logiciel libre peut-il aller ? Il n'y a pas de limites, excepté quand les lois régissant la propriété intellectuelle (brevets, par exemple) interdisent complètement le logiciel libre. L'ultime but est de fournir des logiciels libres pour effectuer toutes les tâches informatiques - et de cette manière rendre obsolète le logiciel propriétaire.

# Le Projet GNU

par **Richard Stallman**

publié à l'origine dans le livre «Open Sources»

## La première communauté qui partageait le logiciel

En 1971, quand j'ai commencé à travailler au laboratoire d'intelligence artificielle (IA) du (N.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 (système à temps partagé incompatible) que les hackers de l'équipe avaient écrit et mis au point en langage assembleur pour le Digital -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 radicale 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», ce qu'ils *disent* est secondaire. 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 :

The term "free software" is sometimes misunderstood--it has nothing to do with price. It is about freedom. Here, therefore, is the definition of free software: a program is free software, for you, a particular user, if:

- 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 TeX; 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 Kit (N.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 platesformes.

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 dollars amécains. 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 tout 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 œuvre 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.

La mise en oeuvre 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.(2)

(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.

(2) Nous utilisons maintenant la GNU Free Documentation License pour la documentation.

### La Free Software Foundation, ou Fondation pour le Logiciel Libre

Fondation pour le logiciel libre (en anglais FSF, sigle de Free Software Foundation) est une organisation américaine à but non lucratif, fondée par Richard Stallman en 1985 pour aider au financement du projet GNU et de la communauté du Logiciel libre. Les fonds sont réunis en vendant notamment des tee-shirts, CD-ROMs et des distributions GNU.

La FSF Europe, la FSF India, et la FSF Latin America sont trois organisations sœurs distinctes de la FSF.

Le 25 novembre 2002, la FSF lance la «FSF Associate Membership program for individuals». En mars 2005, plus de 3400 membres sont adhérents. Le 5 mars 2003 est lancé le Corporate Patron program for commercial entities, un programme concernant les entreprises. En avril 2004, 45 entreprises participent à ce programme.

http://fr.wikipedia.org/wiki/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'œil 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 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 platesformes.

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 ont tous 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 (1) (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.

(1) Cette licence s'appelle maintenant la GNU Lesser General Public License, pour éviter de laisser penser que toutes les bibliothèques doivent l'utiliser. .

### 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 mis en oeuvre 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.

La rétro-ingénierie 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» (1), 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.

(1) Il aurait été plus clair d'écrire « du modèle bazaar»', étant donné qu'il s'agissait de la nouvelle alternative, controversée au début.

### 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.

# Licences

### Introduction

Les logiciels publié devraient être des logiciels libres. Pour rendre un logiciel libre, vous devez le diffuser sous les termes d'une licence de logiciel libre. Nous utilisons normalement la GNU General Public License (GNU GPL), mais occasionnellement nous utilisons d'autres licences de logiciel libre. Pour les logiciels du projet GNU, nous utilisons uniquement des licences compatibles avec la GNU GPL.

La documentation pour les logiciels libres devraient être de la documentation libre, de façon à que les gens puissent la redistribuer et l'améliorer avec le logiciel qu'elle décrit. Pour rendre une documentation libre, vous devez la diffuser sous les termes d'une licence de documentation libre. Nous utilisons normalement la GNU Free Documentation License (GNU FDL), mais occasionnellement nous utilisons d'autres licences de documentation libre.

### La Licence publique générale (GPL) de GNU

La Licence publique générale de GNU (en anglais General Public License) est souvent appelée de l'acronyme «GNU GPL» ou «GPL de GNU». C'est la licence de la plupart des programmes GNU et de plus de la moitié de l'ensemble des logiciels libres actuellement distribués. La dernière version est la version 3.

### La Licence publique générale amoindrie (LGPL) de GNU

La Licence publique générale amoindrie de GNU est adoptée par certaines bibliothèques de composants (mais en aucun cas toutes). La dernière version est la version 3.

### La licence GNU Affero General Public

La licence publique générale GNU Affero est basée sur la GNU GPL, mais a une clause additionnelle pour autoriser les utilisateurs qui inter-agissent avec le logiciel sous licence via un réseau de recevoir les sources de ce programme. Nous recommandons que les personnes envisagent d'utiliser la GNU AGPL pour tout logiciel exécuté sur un réseau. La dernière version est la version 3.

### La Licence de documentation libre (FDL) de GNU

La Licence de documentation libre de GNU est une forme de copyleft destinée aux manuels, aux recueils de textes et autres documents. Son objectif est de garantir à tous la possibilité effective de copier et de redistribuer librement le document avec ou sans modifications, et que ce soit ou non dans un but commercial. La dernière version est la version 1.2.

### Traductions non officielles

Juridiquement parlant, la version originale - en anglais - de la GPL est celle qui décrit exactement les conditions de distribution des programmes GNU. Cependant, pour aider à une meilleure compréhension des licences, nous autorisons la publication de traductions dans d'autres langues, à condition que ces traductions respectent notre protocole pour les traductions non officielles.

### Copie et distribution conforme (verbatim)

La notice de copyright standard pour les pages web de GNU est : *Verbatim copying and distribution of this entire article are permitted worldwide without royalty in any medium, provided this notice is preserved.* (en français : *La reproduction exacte et la distribution intégrale de cet article sont autorisées dans le monde entier sans redevance et sur tous supports pourvu que la présente notice soit préservée.*). Noter le commentaire suivant d'Eben Moglen :

«Notre intention dans la phrase «verbatim copying in any medium» n'est pas d'obliger la rétention des en-têtes et des pieds de page, ou d'autres mises en forme graphique. La rétention des liens hypertextes à la fois sur les supports de type hypertextes ou non (tels que les notes ou les URL imprimées dans des medias non-HTML) est requise.»

### Qu'entend-on par «Copyleft», ou «gauche d'auteur»?

Le *Copyleft* est un cadre permettant de faire d'un programme un logiciel libre et d'exiger que les versions modifiées ou étendues deviennent elles aussi des logiciels libres.

Le moyen le plus simple pour faire d'un programme un logiciel libre est de le placer dans le domaine public (18 Ko), sans aucune licence de droit de copie. Une telle publication permet effectivement aux personnes bien intentionnées de s'échanger le programme ainsi que d'éventuelles améliorations. Malheureusement elle permet aussi à des personnes à l'esprit peu coopératif de transformer le programme en logiciel propriétaire (18 Ko). Ces personnes peuvent écrire des modifications, beaucoup de modifications, ou bien un petit peu, puis le distribuer en tant que produit propriétaire. Les gens qui reçoivent le logiciel sous sa forme modifiée ne disposent plus des libertés que l'auteur leur avait données à l'origine : un intermédiaire s'est interposé et les a supprimées.

En ce qui concerne le projet GNU, nous voulons garantir à *tous* les utilisateurs la possibilité de redistribuer et de modifier les logiciels. Si des intermédiaires pouvaient supprimer ces libertés, nous aurions peut-être plus d'utilisateurs mais ces utilisateurs n'auraient pas les libertés que nous voulons leur donner. Alors, au lieu de placer les logiciels GNU dans le

domaine public, nous les mettons sous «gauche d'auteur». La gauche d'auteur (copyleft) énonce que quiconque redistribue le logiciel, avec ou sans modifications, doit transmettre aussi la liberté de copier et de modifier ce logiciel. Le copyleft est une garantie des libertés de tous les utilisateurs.

Le copyleft constitue aussi un moyen d'encourager d'autres programmeurs à contribuer aux logiciels libres. D'importants programmes libres comme le compilateur GNU C++ n'existeraient pas sans cela.

Le copyleft aide les programmeurs qui souhaitent contribuer à améliorer les logiciels libres à obtenir la permission de le faire. En effet, ces programmeurs travaillent souvent pour les entreprises ou pour les universités et celles-ci feraient presque n'importe quoi pour obtenir de l'argent. Si un programmeur ou une programmeuse désire contribuer à l'œuvre commune, il se peut que son employeur veuille au contraire tirer parti de ce travail par la vente d'une licence de logiciel propriétaire.

Quand nous expliquons à l'employeur qu'il est illégal de redistribuer la version améliorée autrement que sous la forme d'un logiciel libre, ce qui se passe habituellement c'est que l'employeur préfère tout de même le publier comme logiciel libre plutôt que de tout jeter.

Pour placer un programme sous gauche d'auteur, nous commençons par énoncer le droit de copie, puis nous y ajoutons les conditions de distribution. Celles-ci constituent un instrument juridique donnant à toute personne le droit d'utiliser, de modifier et de redistribuer le code du programme *ou de tout programme dérivé*, sous réserve que les conditions distribution demeurent inchangées. Ainsi, le code et les libertés attenantes deviennent juridiquement insécables.

Les développeurs de logiciels propriétaires utilisent le droit de copie pour priver les utilisateurs de leurs libertés. De notre côté, nous l'utilisons pour garantir ces libertés. Voilà pourquoi nous avons inversé le mot «copyright» («droit d'auteur») en «copyleft» («gauche d'auteur»).

Le copyleft est un concept d'une portée générale et peut être décliné de diverses manières. En ce qui concerne le projet GNU, les conditions spécifiques de distribution que nous utilisons sont énoncées par la Licence publique générale (GPL), la Licence publique générale amoindrie (LGPL) et la Licence de documentation libre (FDL) de GNU.

Le texte de la licence appropriée est présent dans de nombreux manuels et dans chacune des distributions de code source du projet GNU.

La GPL de GNU a été conçue pour que vous puissiez facilement l'appliquer à votre programme si vous en êtes l'auteur. Il n'est pas nécessaire de modifier le texte de la licence, il vous suffit d'insérer des notices faisant référence à la licence GPL. Attention, pour placer votre programme sous GPL vous devez impérativement utiliser le texte intégral de la licence. Elle forme un tout, la copie partielle n'est pas autorisée. (De même pour la LGPL ou la FDL).

Le fait d'utiliser les mêmes conditions de distribution pour de nombreux programmes différents facilite la recopie de code entre divers programmes. En effet, si les termes de distribution sont identiques, on n'a plus besoin de se poser de questions sur la compatibilité des conditions de distribution. La GPL amoindrie comprend une clause permettant de modifier les termes de distribution en faveur de la GPL normale, ce qui permet aussi de recopier du code dans un programme sous GPL.

### Licences pour d'autres types d'œuvres

Nous pensons que le logiciel et la documentation publiés devraient être du logiciel libre et de la documentation libre. Nous recommandons que les œuvres de référence et les œuvres éducatives soient libres également, en utilisant les licences de documentation libre telle que la GNU Free Documentation License (GNU FDL).

Pour des essais d'opinion et des articles scientifiques, nous recommandons la simple licence «verbatim copying only» («copie intégrale uniquement») qui est utilisée pour cette page Web.

Nous ne sommes pas convaincus que les œuvres artistiques ou de divertissement doivent être libres, mais si vous voulez en rendre une libre, nous recommandons la Free Art License (Licence Art libre).



Dernière mise-à-jour:Date: 2007/12/11 18:53:43

Traduction: Odile Bénassy.
Révision: trad-gnu@april.org