Richard M. **Stallman**
Sam **Williams**
Christophe **Masutti**

Richard Stallman
et la révolution du logiciel libre

Une biographie autorisée

2e édition

Une initiative Framasoft

 ISBN : 978-2-212-13635-7

EYROLLES

1

Une histoire d'imprimante

« Je crains les Grecs. Même lorsqu'ils offrent des présents. »
– Virgile, *L'Énéide*.

La nouvelle imprimante était encore bloquée. Richard M. Stallman, vingt-sept ans, programmeur au laboratoire d'intelligence artificielle (AI Lab) du Massachussetts Institute of Technology (MIT), le constata à ses dépens. Une heure après l'envoi d'un fichier d'une cinquantaine de pages à l'imprimante laser du bureau, il devait interrompre une séance productive de travail pour aller récupérer ses documents. À l'arrivée, il ne trouvait dans le bac que quatre pages, qui ne lui appartenaient pas. Son travail d'impression, ainsi que celui, inachevé, d'un autre utilisateur,

étaient coincés quelque part dans les mailles électriques du réseau informatique du laboratoire.

Être tributaire du bon vouloir d'une machine fait partie des risques du métier de programmeur. Stallman devait donc prendre son mal en patience… À cette différence de taille près qu'il lui fallait rester planté devant la machine comme un valet au chevet de son maître. Ce n'était pas la première fois qu'il se voyait réduit à regarder les pages sortir une à une.

Consacrant le plus clair de son temps à améliorer l'efficacité des appareils et des logiciels qui les contrôlent, il ressentit tout naturellement le besoin d'ouvrir la bête, de regarder dans ses entrailles et de mettre à jour la faille. Malheureusement, ses talents de programmeur ne s'étendaient pas au monde de l'ingénierie mécanique. Alors que les documents sortaient fraîchement imprimés de la machine, il réfléchissait à un moyen de remédier aux problèmes de bourrage papier.

Combien de temps s'était-il écoulé depuis la réception enthousiaste de la nouvelle imprimante par le personnel du AI Lab ? Stallman se le demandait. La machine était un don de la Xerox Corporation : un prototype de pointe, version modifiée du photocopieur rapide Xerox. Au lieu de faire de simples photocopies, elle transformait les données issues du réseau en documents d'aspect professionnel. Conçu par les ingénieurs du célèbre centre de recherches de Xerox à Palo Alto, le prototype annonçait déjà la révolution de l'impression bureautique qui allait marquer l'industrie informatique à la fin des années 1980.

Poussés par le besoin instinctif de jouer avec l'équipement dernier cri, les programmeurs du AI Lab avaient rapidement intégré la nouvelle machine au sein de l'infrastructure sophistiquée du laboratoire. Les résultats avaient immédiatement plu. Comparée à la vieille imprimante, celle-ci était rapide. Les pages défilaient au rythme d'une par seconde : un travail d'impression de vingt minutes n'en durait plus que deux. Et l'ensemble était plus précis : les cercles ressemblaient à des cercles, pas à des ovales ; les lignes droites étaient

véritablement droites, fort éloignées des anciennes sinusoïdes de faible amplitude.

C'était, à tout point de vue, un cadeau impossible à refuser.

C'est à l'usage que les défauts de la machine firent surface, en particulier une certaine propension aux bourrages papier. Les programmeurs, ingénieurs dans l'âme, devaient rapidement en comprendre la raison : comme tout photocopieur, la machine réclamait une surveillance humaine. Stipulant qu'un opérateur humain serait toujours disponible pour réparer ces incidents s'ils se produisaient, les ingénieurs de Xerox avaient investi temps et énergie à résoudre d'autres problèmes. En théorie, la diligence des utilisateurs faisait partie intégrante du système.

La transformation d'un photocopieur en imprimante avait subtilement mais profondément changé le rapport entre l'utilisateur et la machine. Autrefois dépendant d'un seul opérateur humain, le bon fonctionnement de l'appareil relevait désormais de tout un ensemble d'opérateurs en réseau. Ce n'était plus un seul utilisateur posté à proximité qui envoyait sa commande d'impression, mais un utilisateur situé à l'autre extrémité du réseau. À lui de se déplacer pour constater le peu de pages finalement imprimées.

Stallman n'était bien sûr pas le seul locataire du AI Lab à connaître le problème, mais c'est lui qui trouva un remède. Des années plus tôt, il avait résolu un problème similaire en modifiant le logiciel qui pilotait l'ancienne imprimante depuis un petit PDP-11, ainsi que le système ITS (*Incompatible Timesharing System*) qui tournait sur le PDP-10[1] – l'ordinateur central du laboratoire. Stallman ne pouvait rien aux bourrages mécaniques, mais il put programmer un morceau

1. Les ordinateurs de la gamme PDP (Programmable Data Processor) furent commercialisés dès le début des années 1960 par Digital Equipment Corporation (DEC), le PDP-8 marquant en 1965 une étape importante dans la miniaturisation et dans le rapport qualité/prix. On peut noter que ce fut sur un PDP-1 que le jeu SpaceWar fut créé en 1962. – Voir au chapitre 4 l'explication de l'écriture de l'ITS en réponse au système CTSS (Compatible Time Sharing System).

de code sur le PDP-11, vérifiant périodiquement l'imprimante et envoyant les rapports de bourrage à l'ordinateur central. Sur ce dernier, il ajouta aussi un programme pour qu'en cas de blocage, toute personne en attente d'un tirage soit informée. Le message de notification était simple et tenait en quelques mots : « L'imprimante est bloquée, merci de vérifier. » Les utilisateurs pressés d'obtenir leur impression étaient susceptibles d'accourir rapidement.

La solution de Stallman contournait certes le problème, mais avec élégance. Elle ne résolvait pas la question mécanique du bourrage, mais offrait un retour indispensable à l'utilisateur dans le dialogue avec la machine. Ces quelques lignes de code épargnaient chaque semaine aux employés du AI Lab plusieurs allers-et-retours à l'imprimante. En termes de programmation, cette solution tirait parti de l'interconnexion des opérateurs. Stallman se rappelle la logique du procédé : « La personne qui recevait ce message ignorait si elle était seule ou pas à l'avoir reçu. Elle n'avait d'autre choix que de se rendre jusqu'à l'imprimante. En quelques minutes seulement, deux ou trois personnes arrivaient ; l'une d'entre elles au moins savait résoudre le problème. »

De tels stratagèmes constituaient l'une des marques de fabrique du laboratoire et de sa population résidente de programmeurs. En fait, les meilleurs d'entre eux dédaignaient ce dernier terme, lui préférant celui, plus argotique, de *hacker* (bidouilleur). Un titre couvrant une foule d'activités – tout, de la plaisanterie créative à l'optimisation de programmes et de systèmes existants. Une appellation au parfum légèrement suranné qui évoquait aussi le bon vieux système D à l'américaine[2].

Des sociétés comme Xerox avaient pour politique d'offrir machines et logiciels sur les lieux fréquentés par des hackers. Si ces derniers utilisaient (et souvent amélioraient) le logiciel, ils pouvaient par la

2. Pour un hacker (cf. annexe A), écrire un logiciel fonctionnel n'est qu'un début. Il faut ensuite afficher son ingéniosité et impressionner ses confrères en relevant un autre défi : faire briller le programme par sa rapidité, sa puissance et son élégance.

suite travailler pour ces compagnies. Des années 1960 au milieu des années 1970, celles-ci redistribuaient même parfois directement à leurs clients des programmes élaborés par des hackers.

Face aux bourrages fréquents de l'imprimante, Stallman pensait recourir au même vieux remède – ou *hack*. Or, en cherchant le logiciel pilote de l'imprimante Xerox, il fit une découverte troublante : rien de tel n'était présent, du moins aucun code intelligible ni pour lui-même ni pour d'autres programmeurs.

Jusqu'à présent, la plupart des sociétés avaient la courtoisie de publier le code source de leurs logiciels sous la forme de fichiers texte lisibles, qui tenaient lieu de documentation détaillant chaque commande. Or cette fois, Xerox n'avait fourni les fichiers du logiciel que sous une forme binaire (compilée). Si les programmeurs tentaient de l'ouvrir, ils ne pouvaient voir qu'une incompréhensible suite sans fin de 0 et de 1.

Ils auraient certes pu faire appel à des programmes appelés « désassembleurs », capables de convertir ces suites de 0 et de 1 en instructions machine de bas niveau. Cependant, se représenter ce que ces instructions sont censées *faire* est un travail long et fastidieux, plus connu sous le terme de *rétro-ingénierie*. Un travail qui, pour le code de l'imprimante Xerox, aurait pris au moins le temps perdu en bourrages papier sur cinq années. Stallman, qui n'était pas encore assez désespéré pour s'y coller, mit le problème de côté.

La politique de rétention de Xerox contrastait vivement avec le mode coopératif en cours au sein de la communauté des hackers. Ainsi, pour développer le pilote de l'ancienne imprimante ainsi que celui permettant l'affichage des terminaux depuis deux PDP-11 distincts, le AI Lab avait eu besoin d'un assembleur croisé, compilant les codes du PDP-11 sur le PDP-10 principal.

Les hackers du laboratoire auraient pu écrire un tel programme. Stallman en avait cependant déniché un similaire auprès du département de sciences informatiques de son université, Harvard. Ce programme avait bien été conçu pour tourner sur un PDP-10,

mais avec un autre système d'exploitation. Stallman ne sut jamais qui en était l'auteur – aucun nom n'étant mentionné. Il en rapporta malgré tout une copie au laboratoire, et l'adapta au système ITS en place. Le AI Lab acquit ainsi sans peine une composante de son infrastructure logicielle. Mieux, Stallman y ajouta des fonctions puissantes. « Nous l'avons quand même utilisé plusieurs années », indique-t-il.

Pour un programmeur des années 1970, ces emprunts de code étaient aussi anodins que la visite d'un voisin venu emprunter un peu de sucre ou un appareil ménager. À ceci près qu'en effectuant une copie du logiciel pour le AI Lab, Stallman ne privait personne de la possibilité d'utiliser le programme. Au contraire, tous étaient même invités à utiliser à leur tour les fonctions nouvellement intégrées. Ainsi Stallman se rappelle qu'un programmeur de chez Bolt, Beranek & Newman – une société d'ingénierie privée – réutilisa le programme sur un système nommé Twenex, y ajoutant des fonctions que Stallman put à son tour intégrer dans l'archive du code source du AI Lab. Ils décidèrent même de maintenir une version commune dont le code s'exécutait tant sur Twenex que sur le système ITS.

« Un programme évoluait comme une ville », dit Stallman, évoquant l'infrastructure logicielle du AI Lab. « Certains quartiers étaient remplacés, reconstruits ; de nouveaux éléments étaient ajoutés. Mais il était toujours possible d'en regarder un bout et de dire : 'Bon, d'après le style, cette partie a été écrite dans les années 1960 et cette autre au milieu des années 1970'. »

Ce système simple de capitalisation intellectuelle, mis en place par les hackers au AI Lab, donna naissance à bien des programmes robustes. Même si ceux qui baignaient dans cette culture ne se disaient pas « hackers », la plupart convenaient du fait qu'un programme – ou l'un de ses correctifs – assez bon pour résoudre un problème pouvait servir à d'autres. Y avait-il une quelconque raison de ne pas le partager, ne serait-ce que par le plus élémentaire altruisme ?

Cet esprit de coopération fut peu à peu sapé par le secret commercial et l'appât du gain, ce qui conduisit parfois à des situations bancales, entre partage et cloisonnement. Ainsi les chercheurs de l'université de Californie à Berkeley avaient-ils élaboré un puissant système d'exploitation nommé BSD (*Berkeley Software Distribution*) à partir du système Unix qu'ils avaient obtenu d'AT&T. L'université distribuait BSD pour le simple coût de la copie sur bande, mais uniquement aux écoles justifiant de 50 000 dollars de licence chez AT&T. Les hackers de Berkeley s'accommodaient de cette contrainte tant qu'AT&T les laissait faire, sans percevoir alors de conflit entre les deux pratiques.

§

Stallman fut donc ennuyé que Xerox ne fournisse pas les fichiers du code source, sans pour autant y trouver matière à se mettre en colère. Il ne prit même pas la peine de contacter Xerox. « Ils nous avaient déjà donné l'imprimante laser, dit-il. Je ne pouvais pas prétendre qu'ils nous devaient quoi que ce soit. De plus, j'avais bien conscience que le fait de ne pas livrer le code source était une décision calculée, et qu'il était inutile d'essayer de leur faire changer d'avis. »

Un jour, une nouvelle se mit à circuler : un chercheur du département informatique de l'université de Carnegie Mellon possédait une copie du code source de l'imprimante laser.

Hélas, l'évocation de l'université de Carnegie Mellon n'augurait rien de bon. En 1979, le doctorant Brian Reid y avait provoqué de vifs remous en refusant de partager avec ses confrères son programme de formatage de texte nommé Scribe. Ce logiciel était le premier à interpréter des commandes de balisage vraiment sémantiques (« mettre ce mot en évidence » ou « ce paragraphe est une citation »), au lieu de commandes de pure mise en forme de bas niveau (« mettre ce mot en italique » ou « réduire les marges de ce paragraphe »). Plutôt que d'en faire profiter la communauté, Reid vendit Scribe à

une société informatique de la région de Pittsburgh appelée Unilogic et déclara qu'il avait cherché à la fin de ses études à « confier » le programme à une équipe de développeurs soucieux de le conserver hors du domaine public – quoique l'on puisse se demander en quoi une telle perspective était si indésirable.

En guise de bonus commercial, Reid convint également d'intégrer un ensemble de fonctions programmées dans le temps : des « bombes à retardement », dans le langage des programmeurs, désactivant au bout de quatre-vingt-dix jours les versions du programme copiées gratuitement. Pour empêcher la désactivation du logiciel, les utilisateurs devaient payer une somme à la société informatique, laquelle fournissait alors un patch pour désamorcer « l'anti-fonction » à retardement.

Pour Stallman, c'était là une trahison pure et simple de l'éthique du programmeur. Au lieu d'honorer l'idéal de partage entre pairs, Reid initiait une pratique contraire : celle, pour les entreprises, de forcer les programmeurs à payer l'accès à l'information. Stallman, qui n'utilisait guère Scribe, n'en fit pas grand cas sur le moment. Unilogic en avait donné un exemplaire gratuit au AI Lab sans en retirer la « bombe à retardement », ni en faire mention. Le programme avait donc fonctionné un moment, puis s'était bloqué du jour au lendemain. Un autre hacker, Howard Cannon, passa des heures à le déboguer avant de trouver la « bombe » et de la supprimer en corrigeant le logiciel. Révolté, il ne manqua pas de se plaindre auprès de ses collègues de la manière dont Unilogic lui avait fait perdre son temps avec une erreur intentionnelle.

Quelques mois après, c'est avec l'épisode de Scribe en tête que Stallman, à l'occasion d'une visite professionnelle au campus de Carnegie Mellon, rendit visite à la personne censée détenir le code source du pilote de l'imprimante. Par chance, l'homme était à son bureau. Comme toute conversation entre ingénieurs, celle-ci fut cordiale mais directe. Après s'être brièvement présenté comme venant du MIT, Stallman demanda une copie du code source du pilote de l'imprimante afin de le modifier. À sa grande déception,

le chercheur refusa. « Il m'a expliqué qu'il s'était engagé à ne pas en donner de copie », précise Stallman.

La mémoire humaine est étrange. Vingt ans après les faits, l'enregistrement mental de Stallman comporte quelques blancs. Non seulement ne se souvient-il ni des raisons motivant ce voyage ni de la période de l'année, mais encore a-t-il oublié le nom de son interlocuteur. Selon Reid, la personne la plus susceptible d'avoir reçu Stallman est Robert F. Sproull, ancien chercheur au Xerox PARC et actuel directeur de Sun Laboratories, une division de recherche et développement du conglomérat informatique Sun Microsystems. Dans les années 1970, ce chercheur avait été au Xerox PARC le principal développeur du logiciel en question. Au début des années 1980, il avait intégré le département de recherche de Carnegie Mellon pour y poursuivre ses travaux sur les imprimantes laser, entre autres projets.

Toutefois, interrogé directement par courriel, Sproull ne s'en rappelle rien. « Je ne peux commenter les faits », écrit-il en retour. « Je n'ai aucun souvenir de cet incident. »

« Le code source demandé par Stallman était le nec plus ultra. Un code pointu rédigé par Sproull, une année environ avant d'aller à Carnegie Mellon », se rappelle Brian Reid. Peut-être y eut-il un malentendu, puisque Stallman ne désirait que la version ancienne du code source, utilisée au MIT depuis un certain temps – et non celle développée récemment ? La conversation fut si brève que la question de la version ne fut pas abordée.

En public, Stallman fait souvent référence à cet incident. Il souhaite faire comprendre que le refus du chercheur de partager son code source était directement lié à la signature de l'accord de non-divulgation. Xerox consentait à lui donner un accès au code source, contre l'assurance de sa discrétion.

§

Aujourd'hui fort répandue dans l'industrie logicielle, cette clause de confidentialité (*non-disclosure agreement* – NDA) n'en était alors qu'à ses balbutiements. Elle résultait d'une réflexion de la part de Xerox : la valeur commerciale de l'imprimante résidait aussi dans les informations nécessaires à son fonctionnement. « Xerox, à l'époque, essayait de faire de l'imprimante laser un produit commercial, explique Reid. Ils auraient été fous de faire cadeau du code source. »

Pour Stallman cependant, cette clause de non-divulgation avait une tout autre signification. Elle signifiait le refus, de la part d'un chercheur de Carnegie Mellon, de prendre part à une société où tout programme était considéré comme une ressource collective. Tel le paysan qui voit le ruisseau irriguant ses champs depuis des siècles se tarir brutalement, Stallman était remonté jusqu'à la source. Il n'y avait trouvé qu'un barrage hydroélectrique flambant neuf, orné d'un beau logo Xerox.

Il ne prit pas conscience de suite de tout ce que cela impliquait, et qu'ainsi tout un système pervers allait se mettre en place. Dans un premier temps, il ne vit au refus qu'un caractère personnel. « J'étais tellement en colère que je ne pouvais pas l'exprimer. J'ai fait demi-tour, et suis sorti sans un mot, se souvient Stallman. J'ai même peut-être claqué la porte, qui sait ? Je ne me rappelle qu'une chose : je voulais sortir de là. En venant, pas un instant je n'avais imaginé que ce chercheur pourrait me refuser son aide, et je ne m'y étais pas préparé. Sa réponse m'a laissé sans voix, déçu et furieux. »

Vingt ans après les faits, la colère est toujours là, et Stallman présente cet événement comme l'un des plus décisifs parmi d'autres – dans sa réflexion sur les questions d'éthique autour du logiciel libre. Les mois suivants, la série d'événements impliquant la communauté des hackers du AI Lab aurait pourtant dû, en comparaison, renvoyer au rang de simple détail ces trente secondes de tension dans un bureau obscur de Carnegie Mellon. Pourtant c'est cette rencontre que Stallman invoque pour expliquer comment lui, hacker isolé, instinctivement méfiant vis-à-vis des autorités centralisées, est devenu

un activiste à la tête d'une croisade informatique se réclamant des traditionnels principes de liberté, d'égalité et de fraternité.

« C'était la première fois que j'étais confronté à une clause de confidentialité ; j'ai tout de suite compris que ces clauses font des victimes, déclare-t-il. En l'occurrence, j'en étais la victime ; [mon labo et moi] nous en étions les victimes. »

Stallman poursuit : « S'il avait refusé sa collaboration pour des raisons personnelles, j'en serais resté là. Je l'aurais sans doute considéré comme un imbécile, mais sans plus. Ce qui rendait l'enjeu important était le caractère systématique et impersonnel de son refus, le fait qu'il s'était engagé d'avance à ne coopérer ni avec moi ni avec aucune autre personne. C'est cela qui a rendu l'enjeu important. »

Si la rencontre de Carnegie Mellon n'était pas le premier événement à provoquer ses foudres, elle fit réaliser à Stallman la menace qui pesait sur une culture qu'il tenait pour sacro-sainte. « J'avais déjà le sentiment que les logiciels devaient être partagés, mais sans l'avoir jamais clairement formulé. Mes idées sur la question n'étaient alors pas assez claires ni organisées pour pouvoir les exprimer au reste du monde de manière concise. C'est cet incident qui m'a fait prendre conscience de l'importance de l'enjeu. »

Appartenant à l'élite des programmeurs dans l'une des meilleures institutions au monde, Stallman se souciait peu des compromis et marchandages de ses collègues, tant qu'ils n'interféraient pas avec son travail. Il s'en était tenu à considérer avec dédain ces machines et ces programmes que d'autres toléraient en maugréant…

Ce, jusqu'à l'arrivée de l'imprimante laser de Xerox, qui provoqua un changement imperceptible mais profond : si la machine fonctionnait très bien, malgré des bourrages épisodiques, la possibilité d'en modifier le programme à son goût ou selon des besoins communs, avait disparu.

Pour l'industrie logicielle toute entière, la rétention d'information autour des pilotes d'impression annonçait un changement radical de stratégie commerciale. Le logiciel était devenu un actif d'une

valeur telle qu'il n'était plus question d'en publier les secrets de fabrication (le code source). Surtout si cette publication offrait aux concurrents potentiels la possibilité de le reproduire à meilleur marché.

Pour Stallman, l'imprimante était un cheval de Troie. Après dix ans de mise en échec, la « propriétarisation » des logiciels qui allait instaurer l'impossibilité pour l'utilisateur de modifier ou de partager ses logiciels – les futurs hackers parleront de « logiciels propriétaires » ou, aujourd'hui, de « logiciels privateurs », – avait établi une tête de pont à l'intérieur du AI Lab par la méthode la plus sournoise qui soit : déguisée en cadeau.

Le fait que Xerox faisait preuve de générosité envers certains programmeurs, en échange de leur discrétion était également exaspérant, – Stallman reconnaît toutefois qu'il aurait peut-être accepté l'offre dans sa jeunesse – fût-ce à ce prix. Cela dit, l'entretien de Carnegie Mellon avait éveillé en lui une colère qui allait le forcer à abandonner son indolence morale et à considérer désormais avec suspicion tous les marchandages du même acabit. Cela l'avait aussi amené à pousser le raisonnement jusqu'au bout : et s'il arrivait qu'un ami hacker surgisse dans son bureau pour lui demander le code source et que, du jour au lendemain, le travail de Stallman consiste à le lui refuser ?

« Je fus à mon tour invité à trahir mes collègues de la même façon, et je me suis alors souvenu de la colère que j'avais ressentie lorsque c'était nous qui étions trahis, le labo et moi, dit Stallman. Alors j'ai répondu : 'Je vous remercie beaucoup pour cette superbe collection de logiciels, mais je ne peux l'accepter aux conditions demandées, je vais donc m'en passer'. »

Vocabulaire Logiciel « privateur » – plutôt que « propriétaire »[1]

L'emploi de l'expression « logiciel privateur » (en anglais *proprietary software*) nécessite quelques explications. Pour traduire l'expression anglaise, la plupart des francophones utilisaient le mot « propriétaire » en procédant à une dérivation impropre du nom vers l'adjectif. Or, depuis quelque temps, dans ses discours en français, Richard Stallman préfère qualifier ces logiciels de « privateurs » pour en souligner les effets nocifs. Notons que ce terme n'est pas un néologisme. D'après le *Trésor de la Langue Française* : « **Privateur**, **-trice**, adj., rare. Qui prive. »

Plusieurs raisons justifient le choix de ce qualificatif. D'abord, l'expression « logiciel propriétaire » laisse croire à tort qu'il faudrait renoncer à ses droits d'auteurs (assimilés aux droits de propriété) pour créer un logiciel libre. Or, comme il en sera notamment question au chapitre 9, la GNU General Public Licence créée par Richard Stallman emprunte à la logique du droit d'auteur en stipulant les conditions d'exercice de ce droit sous la forme de quatre libertés accordées à l'utilisateur du logiciel : utiliser le programme, en étudier le code source, le modifier et distribuer des copies de la version originale ou modifiée. Enfin, l'adjectif « privateur » exprime bien la privation de liberté – celle induite par le fait de ne pas rendre libre un programme. Quant à l'adjectif « privatif », il accentuerait encore la confusion évoquée plus haut.

1. Voir également la définition donnée par l'Association pour la promotion et la recherche en informatique libre (APRIL) : http://www.april.org/articles/intro/privateur.html.

§

Stallman n'oublierait pas la leçon au cours des tumultueuses années 1980, une décennie durant laquelle de nombreux collègues du MIT quittèrent le AI Lab et signèrent eux-mêmes des clauses

de confidentialité. Certains se dirent sans doute que c'était un mal nécessaire leur permettant de travailler sur les meilleurs projets. Pour Stallman au contraire, la simple signature d'une telle clause annulait toute légitimité d'un point de vue moral. À quoi bon un projet techniquement passionnant s'il ne peut bénéficier à la communauté ?

Comme Stallman devait très vite l'apprendre, refuser de telles offres représentait davantage qu'un sacrifice matériel. Cela allait le conduire à s'isoler des autres hackers qui, partageant la même aversion pour la rétention d'information, tendaient à l'exprimer avec plus de flexibilité morale.

Pour Stallman, refuser de partager un code source avec un confrère revenait non seulement à trahir la mission scientifique sous-tendant le développement logiciel depuis la fin de la Seconde Guerre mondiale, mais aussi à violer la règle d'or fondant toute morale : « ne faites pas aux autres ce que vous ne voudriez pas qu'on vous fît ».

Ainsi l'épisode de l'imprimante laser et l'entrevue qui en découla eurent-ils leur importance. Sans cela, nous dit Stallman, sa vie aurait peut-être suivi un chemin plus ordinaire, alliant le confort matériel d'un programmeur de logiciels commerciaux et la frustration suprême d'une vie passée à écrire des codes invisibles. Une vie dénuée de ce sentiment d'évidence, de cette urgence à résoudre un problème auquel personne d'autre ne s'attaquait. Plus important encore, il n'y aurait pas eu cette grande et juste colère qui, comme nous le verrons bientôt, devait animer son action aussi sûrement qu'une idéologie politique ou une croyance morale.

« J'ai décidé de ne jamais me rendre complice de ce système », dit Stallman, faisant référence à la fois au mécanisme de la clause de confidentialité (qui à ses yeux revient à « brader sa liberté pour des raisons de commodité »), et à l'esprit qui encourageait ce marchandage moralement douteux. « J'ai décidé de ne jamais faire d'autres victimes comme moi. »