
Que signifie logiciel libre ?

Aux origines de l'informatique : les précurseurs du Libre

Au début étaient les machines, puis vint l'idée d'automatiser leur fonctionnement. C'est en 1834 que l'Anglais Charles Babbage (1791-1871) inventait la « machine analytique », une extension de l'invention du Français Joseph Jacquard, qui, pour fabriquer des brocards de soie, avait conçu un métier à tisser guidé par des cartes perforées. Mais c'est une femme, la mathématicienne britannique Augusta Ada Byron (1815-1862), comtesse de Lovelace et fille de Lord Byron, qui popularisa les travaux de Babbage, en traduisant l'article fondateur – rédigé en français par... un Italien – décrivant sa « machine », mais surtout en les enrichissant. Elle inventa diverses techniques de ce qu'on n'appelait pas encore la programmation, en particulier le principe de la « boucle », qui permet de répéter une suite d'instructions jusqu'à la vérification d'une condition donnée.

À la veille de la Seconde Guerre mondiale, de 1936 à 1938, l'Allemand Konrad Zuse travailla à l'un des premiers calculateurs réellement programmables, qui fonctionnait avec des relais électromécaniques. Pour cela, il dut renoncer à utiliser le classique système décimal et recourir au système binaire, ne comportant que deux chiffres, 0 ou 1, correspondant à l'état de chaque relais, fermé ou ouvert, ce qui répondait également à la bascule faux ou vrai de la logique symbolique.

Après la guerre, les chercheurs états-uniens du premier ordinateur de grande taille, le Mark I, inauguré en 1948,

constatant que l'écriture des programmes de calcul à lui faire exécuter impliquait la répétition fréquente de mêmes tâches, écrivirent ce qu'on appela des « routines » : des petites fonctions, ou sous-programmes, régulièrement utilisées dans les programmes. Ils écrivaient la routine nécessaire, et lorsqu'un programmeur en avait besoin, ils la lui donnaient, afin de lui faire gagner un temps précieux pour tous et qu'il puisse se concentrer sur des recherches nouvelles. La notion actuelle de « logiciel libre » – distribution libre de la chose programmée – était l'évidence pour ces chercheurs.

C'est également une mathématicienne, l'Américaine Grace Murray Hopper (1906-1992), qui fut à l'origine de ce qui se révéla une nécessité et fit avancer rapidement la programmation : le « compilateur ». Le premier texte scientifique à ce sujet fut publié en 1952 : réunissant les routines en une bibliothèque de sous-programmes sur lesquels se fondaient les programmeurs pour ne pas avoir à tout réécrire, elle permit aux chercheurs de ne pas réinventer la roue de la programmation. Ce logiciel-compileur était libre – le code de la programmation était ouvert à tous –, ce qui était considéré comme naturel.

Notons que Grace Hopper fut également à l'origine de la découverte d'un autre classique de la programmation : le *bug*. C'est lors d'un travail à Harvard en tant qu'officier de marine que, dans les années 1940, elle dépanna la grosse machine coincée, composée de milliers de relais électromécaniques, dans laquelle un papillon (insecte, *bug* en anglais) s'était introduit... Il y avait un *bug* et les programmeurs durent « déboguer ». Ils le font toujours plus d'un demi-siècle plus tard...

C'est encore elle qui inventa à la fin des années 1950 le premier langage de programmation humain, le Cobol (pour Common Business Oriented Language), alors qu'auparavant on ne pouvait programmer qu'en « langage machine ». Au lieu de parler avec des chiffres – des 0 et des 1 –, on écrit désormais les programmes avec des mots. En 1983, elle expliquera que c'était pour que plus de gens puissent accéder à la programmation qu'elle avait créé un langage destiné aux

machines, mais que l'on puisse écrire en anglais ¹. Et ce pour une raison fort triviale : « Je pense que les êtres humains sont allergiques au changement. Ils n'aiment pas changer. Ils ont appris quelque chose et ils sont absolument satisfaits de ce qu'ils font ; vous arrivez et vous leur dites : vous allez faire ceci de cette manière-là. Ils le rejettent. Ils sont naturellement allergiques au changement. »

Comment fonctionne un ordinateur, et avec quels logiciels

Le seul langage que comprend la machine – le *langage machine* – est composé de « mots » composés à partir de deux lettres seulement, 1 ou 0. Même si les relais de l'époque de Konrad Zuse ont techniquement évolué depuis, la machine ne comprend toujours que des 1 et des 0. Les *langages de programmation* poursuivent donc tous le même but, depuis l'invention du Cobol : permettre aux humains d'écrire des instructions dans une « langue » compréhensible par les humains.

Les *logiciels* (aussi appelés programmes, ou applications) sont des suites de commandes écrites par des humains et compréhensibles par des machines, après diverses transformations, transparentes tant pour les programmeurs qui les écrivent que pour les utilisateurs des logiciels. C'est l'ensemble des instructions écrites en un quelconque langage de programmation que l'on appelle le *code source*. Ce code source est ensuite traduit (par un compilateur) en langage machine, exécutable par l'ordinateur, et composé uniquement de 0 et de 1 : le résultat est ce que l'on appelle le *code binaire*.

Le *système d'exploitation* est un ensemble de logiciels qui font correspondre entre eux les divers composants élémentaires gérant l'ordinateur : clavier, souris, mémoire vive,

1. <ei.cs.vt.edu/~history/Hopper.Danis.html>.

disques durs, lecteur de disquettes, de CD-Rom, de DVD, imprimantes, scanner, programmes, etc. Il coordonne le fonctionnement interne de la machine, il agit, en quelque sorte, comme un chef d'orchestre.

On peut classer les logiciels, du point de vue de leur mode de distribution et de leur type d'utilisation, en plusieurs catégories :

– le *logiciel propriétaire* n'est utilisable qu'avec restriction – souvent sur une seule machine par exemple –, on ne peut ni le copier ni le diffuser, on ne peut accéder au code source, donc on ne peut pas savoir exactement les commandes qu'il préconise à l'ordinateur ; il n'est donc pas modifiable par l'utilisateur ;

– le *freeware* (gratuciel) est un logiciel propriétaire gratuit ; on ne le paie pas, mais on ne peut accéder au code source, ni le modifier ;

– le *shareware* (partagiciel) est intermédiaire entre le *freeware* et le propriétaire : c'est un logiciel propriétaire, disponible gratuitement, soit à l'essai durant un certain temps (après quoi l'utilisateur doit le payer), soit sans conditions mais avec des fonctions limitées ;

– le *logiciel libre*. La « liberté » d'un tel logiciel est multiple. S'il existe diverses « licences » de logiciels libres (c'est-à-dire diverses formulations juridiques des droits et devoirs de l'utilisateur), certains critères sont communs à tous les logiciels définis comme « libres » : on peut les utiliser et les copier sans limite, en toute légalité – donc, par exemple, les utiliser sur autant de machines que l'on veut, et les donner à ceux avec qui l'on travaille sur des fichiers communs ; on peut lire et étudier leurs codes sources et les modifier (les finesses de modifications possibles dépendent du type de liberté octroyé par la licence à laquelle ils sont soumis).

Les bases juridiques du Libre

L'idée de base du logiciel libre dépasse largement les intérêts techniques et rejoint ce que défendait Grace Hopper : c'est un

but en soi que de permettre à tous d'utiliser des outils qui peuvent faciliter la vie. Mais ce qui fonctionnait parfaitement du temps de Grace Hopper n'est plus aussi simple depuis que la technicisation de la société et l'emprise des grands groupes informatiques ont atteint un niveau impressionnant : pour permettre que le travail de chacun reste le bien commun de tous, et que cette autogestion à tendance anarchiste puisse fonctionner de manière durable, il fallut l'idée de génie de structurer ce concept sur des fondements juridiques (l'« utilisation du droit au service de la philosophie ² »).

Ainsi avec la création, en 1989, de la GNU General Public License ³ (GPL) par Richard Stallman et la Free Software Foundation ⁴ (FSF) pour soutenir le premier projet d'écriture d'un logiciel libre, le projet GNU ⁵, des bases de travail saines étaient posées. Cette licence, extrêmement pensée et, en fin de compte, complexe (mais pas compliquée), permet un résultat qui correspond parfaitement aux besoins recherchés : elle ne permet à personne de s'approprier à son seul profit le travail des autres, tout en permettant à tous d'utiliser et de modifier le travail des autres. Car l'altruisme des partisans du Libre ne vaut que si tout le monde peut s'approprier le travail de tous, mais également que si personne ne peut s'approprier, à son unique profit, le travail de tous.

Ce n'est que grâce à cette licence, qui accorde à tous le droit de récupérer, transformer et adapter les codes sources d'un Libre, si – et uniquement si – le résultat de ces transformations est également distribué sous cette même licence GPL, que le Libre peut croître et espérer faire vaciller les monopoles en ce domaine. Chacun conserve la liberté d'adapter et de transformer un Libre, chacun peut même

2. <www.droit-technologie.org/pda/dossier.asp?dossier_id=118>.

3. Initié en 1984, ce projet visait à créer un système d'exploitation libre, à la différence d'Unix, système alors dominant (<www.gnu.org/copyleft/gpl.html>).

4. <www.fsf.org/>.

5. GNU (prononcer Gnou) est un acronyme récursif, « GNU's Not Unix », qui signifie « GNU n'est pas Unix » (<www.gnu.org/gnu/thegnuproject.html>).

intégrer des composants propriétaires et des transformations du code source sans le divulguer. À condition que ce soit à son usage propre et qu'il ne redistribue pas le résultat. Chacun conserve la liberté de diffuser, éventuellement de manière payante, le logiciel libre, avec ou sans changement, avec ou sans document, avec ou sans soutien technique, à condition que les règles de la licence GPL soient respectées.

Ce travail juridique indispensable pour échapper aux tentations a été effectué très précisément, en prenant en compte les composantes éthiques et politiques du mouvement du Libre. Il est remarquable qu'aient pu être ainsi traduits en termes juridiques les besoins, nécessaires et suffisants, pour atteindre des buts éthiques : préserver la plus grande liberté possible dans la création, la modification et l'utilisation de logiciels, tout en laissant la plus grande liberté de faire du commerce avec eux. Le beurre et l'argent du beurre, en quelque sorte...

Mais le double sens de *free* en anglais (à la fois libre et gratuit), conjugué à l'intransigeance de certains défenseurs du Libre, a vite effrayé les médias de l'*establishment*. Ajoutons à cela l'image anarchiste de son créateur, Richard Stallman (voir *infra*, chapitre 2), et il ne fallut pas longtemps pour que le Libre soit associé à l'« hostilité aux droits d'auteur » et même... au « communisme », injure suprême aux États-Unis. Au printemps 1998, pour rassurer les milieux d'affaires – sur le thème : on peut faire du Libre et gagner sa vie – tout en faisant sortir le Libre d'une certaine image de marginalité, des militants du logiciel libre réunis au sein de l'Open Source Initiative (autour d'Éric Raymond, Tim O'Reilly et Larry Augustin) ont alors promu une nouvelle appellation, celle d'*open source* (pour « code source ouvert », ce qui est *une* des libertés d'un logiciel libre, mais pas la seule)⁶. La formule sera ensuite largement popularisée, notamment par la presse, mais aussi

6. L'appellation *open source* comme synonyme de logiciel libre appartient à l'Open Source Initiative (<www.opensource.org>).

mise à toutes les sauces, y compris par des éditeurs de logiciels propriétaires... Un peu comme, dans un tout autre domaine, le terme « développement durable », revendiqué par les pires pollueurs pour se refaire une virginité.

Pour le reste, il n'existe pas de différence cruciale entre Libre et *open source* : ce n'est que le contenu de la licence revendiquée par un logiciel qui détermine sa catégorie. Il n'y a pas que la licence GPL qui peut être qualifiée de « libre ». On peut concevoir de nombreuses licences de logiciel *open source*, recensées par l'Open Source Initiative⁷. Les critères auxquels ces licences doivent répondre, pour être qualifiées d'*open source*, sont les suivants.

– *Aucune restriction à la redistribution gratuite de tout ou partie du logiciel ne peut être imposée.* En interdisant le paiement obligatoire (tout en autorisant un « développeur » à vendre sa version spécifique), on permet que le logiciel continue à s'améliorer, grâce aux apports de ceux qui l'adaptent à leurs besoins propres et décident d'en faire profiter les autres gratuitement. Ainsi, les gains immédiats qui peuvent être engrangés par des ventes de versions intermédiaires n'empêchent pas la progression du logiciel à long terme, au bénéfice d'un nombre croissant d'utilisateurs.

– *Le code source du programme doit être disponible, de manière lisible, sans nécessité d'une traduction intermédiaire, sans dépenser un sou.* En d'autres termes, le code source doit être clair et non délibérément abscons, pour faciliter la tâche des programmeurs travaillant à l'amélioration du logiciel libre.

– *Le logiciel doit pouvoir accepter les modifications et travaux dérivés, ainsi que la redistribution sous la même licence.* C'est la « clause virale » : la licence GPL se transmet automatiquement à chaque distribution de modification (gratuite ou payante). Pour éviter l'appropriation de code, et permettre que le bénéfice du logiciel et de ses améliorations profite à tous,

7. <www.opensource.org/licenses/>.

celui qui travaille sur le code ne peut le redistribuer que sous la licence sous laquelle le code initial se trouvait ou une autre licence d'au moins autant de liberté. C'est aussi cette clause qui permet l'évolution rapide du logiciel, chaque personne y travaillant étant sûre que son travail profitera à tous, de la même manière qu'elle-même pourra profiter du travail de tous.

– *La licence peut exiger que le code source modifié ne soit distribué que s'il porte un nom ou numéro de version différent de l'original.* L'auteur initial se protège ainsi d'une responsabilité qui pourrait lui être attribuée pour des transformations qu'il n'a pas effectuées ou choisies.

– *Aucune discrimination, visant des personnes ou des groupes de personnes, ne peut être autorisée.* Parfois, en effet, des conditions sont imposées par l'État à l'exportation d'un logiciel : c'est notamment le cas aux États-Unis, où des matériels et des logiciels peuvent être répertoriés comme « sensibles » et placés sous embargo pour l'exportation, en particulier vers certains pays (considérés comme terroristes ou susceptibles de développer des armes de haute technologie) ⁸.

– *Aucune restriction ne peut être imposée à l'utilisation d'un logiciel libre dans un domaine spécifique.* Cette clause est destinée à éviter que certains ne déclarent qu'un logiciel libre ne peut être utilisé, par exemple, pour une activité commerciale.

– *Les droits attachés au programme doivent être appliqués à tous ceux à qui il est redistribué.* De cette manière, il ne peut y avoir restriction avec l'évolution du logiciel ; des droits existant à un moment donné ne pourront être restreints après transformation du logiciel.

8. Dans les années 1990, cette censure a par exemple donné lieu à un bras de fer homérique entre le gouvernement américain et Phil Zimmermann, le développeur d'un logiciel libre de cryptage particulièrement efficace, PGP (acronyme de Pretty Good Privacy, euphémisme pour désigner une « sécurité [des données personnelles] à peu près correcte »), affrontement dont ce dernier sortira finalement vainqueur. Voir *infra*, chapitre 5 ; et aussi : Jean GUISNEL, *Guerre dans le cyberspace*, La Découverte, Paris, 1997.