



---

# ICT FOR SME'S

## Dossier "Open Source"

ITWS4SMES-11102004-Open Source-v203.doc

**Réalisé par**



**Octobre 2004**



---

# Index

Licences Libres .....	3
Qu'est-ce que l'Open Source .....	22
La définition de l'Open Source.....	39
Justifications de l'OSD .....	43
Certification OSI et processus de certification.....	43
Licence de Documentation Libre GNU .....	49



---

# ***Licences Libres***



---

## **1. Diffusion du présent document**

Ce document issu de recherches sur Internet, ainsi qu'au sein des réseaux INDIALLEY et OPERATION appartient à la NCB. Il est librement diffusable dans les termes de la [Licence de Documentation Libre GNU \(GNU Free Documentation License\)](#).

Nous remercions la société IDEALX pour son apport à la réalisation de ce dossier.

Toutes précisions, remarques ou correctifs seront reçues avec plaisir et considérées avec le plus grand soin.

## **2. Préambule**

Nous allons traiter dans ce document de questions liées à la propriété intellectuelle en matière d'oeuvres numériques (programmes d'ordinateur, textes). Il faut savoir que le côté juridique est souvent artificiellement technique et qu'il est difficile de prédire le résultat d'un éventuel procès, sans oublier que la plupart des petits acteurs n'ont pas la volonté ou les moyens d'engager une procédure en cas de violation de licence. Les lois traitant de la propriété intellectuelle changent d'un pays à l'autre: les États-Unis d'Amérique fonctionnent autour du copyright (plutôt comparable, dans une certaine mesure, au droit de reproduction) alors que le droit d'origine latine appliqué en France repose sur le droit d'auteur, consigné dans le *Code de la Propriété Intellectuelle*, qui se décompose en droit moral (incessible) et droit patrimonial. La plupart des auteurs de licences ont rédigé ces dernières dans l'esprit de la loi de leur pays, c'est-à-dire le droit anglo-saxon.

## **3. Code source versus code binaire**

Il est important de comprendre ces deux notions, car de nombreuses licences en parlent et établissent des distinctions de droits et de devoirs selon qu'on parle de code source ou de code binaire. Elles n'ont que peu d'équivalents dans la vie courante, c'est pourquoi il est difficile de les expliquer à quelqu'un qui n'a jamais programmé et de trouver une comparaison pertinente.

La plupart des programmes d'ordinateurs (pour être précis, les programmes écrits dans un langage compilé, ce qui représente la majeure partie des programmes utilisés et diffusés) ont deux formes: le code source, et la version binaire du programme.

### 3.1 Qu'est-ce que le «code source»?

Le code source est le texte écrit par le ou les programmeurs sur son écran. Il est constellé d'informations redondantes qui en facilitent la lecture et la compréhension, ainsi que de commentaires explicatifs. Depuis de nombreuses années maintenant, les langages de programmation utilisés font usage de mots de la langue anglaise (espéranto *de facto* dans le monde de l'informatique). C'est un texte écrit par l'homme pour l'homme. S'il est «bien» écrit, c'est-à-dire dans le but d'en faciliter la correction et la maintenance, il est très facile de trouver rapidement la portion qui nous intéresse quand on a une correction en tête. Comme dans la plupart des activités humaines, il est plus long et fastidieux de «bien» programmer, mais on y gagne beaucoup de temps à moyen terme: les autres membres de l'équipe de développement, ou l'auteur lui-même, quelques mois plus tard, entreront d'autant mieux dans le code source qu'il a été soigneusement écrit et documenté.

Le code source n'est pas utilisable en l'état par la machine. Si on souhaite exécuter le programme, c'est-à-dire le faire fonctionner sur la machine, il faut passer par une phase de **compilation**. Cette phase est assurée par un **compilateur**, qui produit une version **compilée**, qui est la version binaire, directement exécutable et donc utilisable par l'ordinateur. Le compilateur inspecte l'ensemble du code source, fait les simplifications et les raccourcis qu'il trouve, élimine les commentaires, et traduit l'ensemble en une succession d'instructions élémentaires que l'ordinateur pourra suivre à pleine vitesse, sans aucun recul, sans «comprendre» ce qu'il est en train de faire. On est passé d'un langage de **haut niveau** à un langage de **bas niveau**.

Si on dispose du code source:

On peut s'assurer que le programme ne fait rien de «bizarre» (inspection de nos fichiers personnels et envoi à notre insu par le réseau);

Dans le cadre de programmes de chiffrement manipulant des données confidentielles, on peut plus facilement (même si la cryptographie est une science complexe) vérifier que l'algorithme de chiffrement utilisé n'est pas faible ou qu'il n'a pas été prévu de trou de sécurité volontaire pour que tel ou tel, qui intercepterait la communication, puisse savoir ce qui se dit sans disposer des clefs de déchiffrement;

Si le programme a un défaut («bogue», ou *bug* en anglais) ou si on souhaite lui ajouter une fonctionnalité (*feature* en anglais), on peut très bien modifier le code source et le compiler de nouveau;

Il est envisageable de réutiliser tout ou partie du code pour l'intégrer dans d'autres programmes.

### **3.2 Liberté et coopération**

Bien sûr, si on n'a pas la compétence de faire ces vérifications, souvent longues et difficiles, ni le temps de les acquérir, on peut s'adresser à quelqu'un pour qu'il les fasse à notre place. Toute personne s'intéressant à la programmation est susceptible de nous rendre ce service, ce qui assure avec le mécanisme de l'offre et de la demande, un «juste prix» pour ce travail. Souvent, dans le cas de programmes communs d'intérêt général, utiles à tous, ces vérifications sont continuellement menées par des bénévoles de par le monde, communiquant entre eux par l'Internet, car ils en ont besoin pour eux-mêmes. On peut alors faire confiance à ces vérifications si plusieurs personnes indépendantes aboutissent aux mêmes conclusions, ou si un spécialiste, ou un groupe de spécialistes reconnu et de confiance prend position (ils ont souvent une réputation à laquelle ils tiennent).

L'Internet joue ici un rôle important de factorisation de l'information. Ce qui coûte cher en valeur ajoutée ou en temps humain, c'est de produire l'information, le programme, la vérification. La transmettre à un coût négligeable, et tous peuvent en profiter. Si de nombreuses personnes d'horizons différents et de formations différentes s'intéressent au même programme, elles remarqueront un plus grand nombre de problèmes différents. Pratiquement chacun des problèmes réels ou potentiels semblera évident à l'une de ces personnes.

Du point de vue de l'entreprise qui utilise du logiciel libre, elle a la garantie qu'en cas de problème, de souhait de mise à jour ou d'évolution du logiciel, elle n'est pas tributaire du bon vouloir du fournisseur ou de l'éditeur du logiciel. Si son besoin ou souhait n'est pas déjà couvert par un programme librement disponible sur l'Internet, parce qu'il est trop ciblé, fastidieux, ou ennuyeux, elle pourra le commanditer ou le réaliser en interne sans obstacle technique artificiel imposé par un éditeur qui ne diffuserait que la version binaire de son programme.

Si on ne dispose que du code binaire, on est extrêmement limité. Il est possible au prix d'efforts et de temps sans commune mesure avec ceux qu'il faudrait fournir si on disposait du code source, de faire quelques vérifications ou modifications mineures, mais la souplesse et l'éventail de possibilités sont extrêmement limités par rapport à ce qu'il est possible de faire avec le code source.

### **3.3 Un code source peut être rendu délibérément abscons**

Mais si l'on dispose d'un code source qui a été rendu délibérément difficile à comprendre, alors la difficulté peut être presque aussi grande que dans le cas précédent. Plusieurs degrés d'«illisibilité volontaire» peuvent être envisagés: soit par la suppression de tous les commentaires dont un programmeur consciencieux parsème son code afin de pouvoir «replonger» dans son propre code lorsque cela s'avérera nécessaire et qu'il en aura forcément oublié les tenants et aboutissants; soit par l'usage systématique d'algorithmes abscons, l'appel de fonctions non, mal, voire insidieusement documentées... bref toutes sortes de ruses dont il est évidemment impossible de dresser la liste.



Dans le premier cas, il peut s'agir d'une simple négligence ou inexpérience de la part du programmeur, qui estime sincèrement inutile de documenter son code au moment où il l'écrit (peut être parce qu'il est pressé, même si le calcul s'avère peu payant à moyen et long terme). Dans le second, il y a nécessairement hypocrisie de la part du programmeur, qui feint de «libérer» son code, alors qu'il s'efforce en réalité d'en borner la compréhension, donc l'utilisation, par un tiers.

Bref, la simple disponibilité du code source ne suffit pas, encore faut-il que le ou les programmeurs aient eu la volonté de le rendre réellement accessible à leurs pairs.

### **3.4 Code source et sécurité**

On peut être tenté de croire, dans les questions de sécurité et de chiffrement, que ne diffuser que la version binaire du programme de chiffrement améliorera la sécurité, car cela dissuadera les curieux de plonger dans les détails des calculs de chiffrement. C'est très souvent faux. C'est ce qu'on appelle la "security through obscurity" («sécurité par «obscurité», ou masquage») et de nombreux exemples de par le passé prouvent que qui est vraiment motivé ou en a les moyens, peut recomposer le détail des calculs effectués par un programme de chiffrement dont on n'a qu'une version binaire.

La méthode conseillée est au contraire de donner à tous un accès facile à tous au code source du programme de chiffrement, afin que la revue des pairs s'exerce, car il est très facile de laisser une faille dans de telles matières. Bien sûr, les agences de contre-espionnage gouvernementales gardent jalousement secrètes l'ampleur des moyens de calcul dont elles disposent, ainsi que les éventuelles percées théoriques permettant de «casser» des codes en faisant moins de calculs.

## **4. Comparaisons «valables» dans la vie de tous les jours**

Pour tenter d'apporter un meilleur éclairage sur ces notions, et en prenant le risque d'enfermer la conception du lecteur dans une image d'Épinal réductrice, nous allons donner des analogies, comparaisons, dans la vie de tous les jours, représentant un parallèle acceptable avec les notions de code source et de code binaire. Commençons par démentir une idée reçue.

La partition n'est **pas** le code source de l'oeuvre interprétée par le musicien ou l'orchestre. On n'a pas les informations de timbre de l'instrument, des libertés prises par l'interprète, son jeu, et on peut reconstituer assez facilement une partition à partir du phonogramme du morceau interprété, et absolument pas le contraire! C'est donc une mauvaise analogie.

Une meilleure analogie est celle de la recette de cuisine: le gâteau cuisiné sorti du four s'adresse directement aux papilles gustatives en langage de bas niveau: «c'est sucré, salé, amer, acide». La recette contient elle des informations transmises de cuisinier à cuisinier qui seront détruites lors de la cuisson: «bien remuer pour éviter les grumeaux», ... Si on souhaite obtenir un gâteau un peu plus ou moins sucré, il est beaucoup plus facile de modifier les quantités dans la recette que de saupoudrer uniformément de sucre tout le volume du gâteau. De plus, les clubs de cuisine amateurs ont pour tradition de partager et d'améliorer les recettes des uns et des autres. L'Internet leur permet maintenant de faire ce travail de mise en commun de conseils et d'expériences au niveau mondial.

Tentons maintenant d'expliquer la différence entre les notions de langage de «haut niveau» et de «bas niveau».

Le «haut niveau» se réfère à l'être humain, fait appel par associations d'idées à ses connaissances dans d'autres domaines (par exemple, cela utilise certains mots de la langue anglaise), contient un vocabulaire riche et varié correspondant à de nombreux cas particuliers possibles. Le «bas niveau» se réfère à l'électronique de la machine. Le vocabulaire est plus restreint, et le document est moins facile à comprendre, notamment parce que certaines informations redondantes et complémentaires ne sont plus présentes, et qu'elles facilitaient l'explication "à la main" de ce qui se passait pour que d'autres lecteurs du programme, ou son propre auteur, après quelques mois, comprennent plus vite de quoi il retourne, mais aussi car les descriptions sont beaucoup plus longues et difficiles à suivre pour des raisons de pure forme, de syntaxe et de vocabulaire réduit.

## 5. [Exemples](#)

Dans un langage de haut niveau: «Viens me chercher à la gare à 17h» Il faut faire appel à tout un vécu et une expérience pour interpréter cela correctement, ou se référer à une bibliothèque mettant en relation les notions impliquées avec les séquences de bas niveau correspondantes. Le terme «gare» est choisi parmi une liste très riche comportant plusieurs centaines ou milliers de mots clés reconnus.



---

Dans un langage de bas niveau cela deviendrait:

Sors de la maison à 16h

Engage la clef dans la portière de la voiture

Déverrouille la serrure

Dégage la clef

Ouvre la portière

Prends place sur le siège

Ferme la portière

...

Règle le rétroviseur

...

100 m en avant

Tourne de 30 degrés à droite

300 m en avant

...

Il faut suivre pas à pas et c'est parfois très long avant de comprendre ce qui se passe vraiment. Mais une machine peut exécuter ces ordres simples très vite, sans bien sûr devoir «comprendre» ce dont il retourne.

La transformation d'un programme de langage de haut niveau en langage de (plus) bas niveau s'appelle la compilation.

J'espère que vous comprenez bien maintenant la différence entre «code source» et «code binaire», ou «exécutable»: on peut utiliser un programme en ne disposant que du code binaire, mais on est limité et astreint à ce que son auteur a prévu, et on doit subir les erreurs qui s'y sont glissées sans espoir de pouvoir les corriger, même si on en a, ou connaît quelqu'un qui en a, la possibilité.

## **6. Signification du mot «libre»**

Le logiciel libre en général et Linux en particulier gagnant en couverture et visibilité médiatiques, de plus en plus d'entreprises se réclament du «logiciel libre», de plus en plus d'articles de presse en parlent, ce qui fait que tel quel, ce mot ne signifie plus rien. Personne n'a le monopole de son utilisation et ne peut imposer des conditions strictes à quiconque l'utilise, car ce terme n'est pas une marque déposée. Il est donc important de toujours se faire préciser dans quel sens précis on dit d'un logiciel ou d'un document numérique qu'il est «libre».

### **6.1 Les quatre niveaux de liberté selon Richard M. Stallman**

Richard Matthew Stallman (dit «RMS»), qui joue un rôle important dans le logiciel libre, comme on le détaillera plus tard) distingue quatre niveaux de liberté dans les programmes d'ordinateurs:

le niveau 0, qui consiste à utiliser l'exécutable en fonctionnement normal, pour n'importe quel but. On dispose de cette liberté dans la plupart des cas (sans elle, on ne peut rien faire). Cela impose donc de disposer du code binaire (ou d'être capable de le produire à partir du code source);

Le niveau 1, qui est la liberté de faire des améliorations et des corrections dans le programme. Cela impose donc de disposer du code source;

Le niveau 2, qui est la liberté de diffuser des copies du programme. Cela impose donc que celui qui dispose des droits sur le programme (selon la ou les législations qui s'appliquent) autorise quiconque à en donner ou vendre des copies sans pour cela toucher des royalties lors de chaque transaction. Cela peut surprendre, car cela va à l'encontre de nos idées reçues ou habitudes en la matière, mais cela a des conséquences intéressantes, et l'argent ne disparaît pas du système, il change de lieu, et passe du produit logiciel au «service»;

Le niveau 3, qui est la liberté d'écrire des améliorations et de publier les améliorations apportées par nous au programme de telle sorte que toute la communauté puisse en bénéficier. Cela permet de mettre en place des projets de développement libres réunissant de nombreuses personnes de par le monde, qui peuvent développer une grosse charge de travail pour peu qu'ils arrivent à s'organiser (ce qui est possible, et la constatation de cette réalité a surpris les analystes traditionnels en génie logiciel). Certains projets disposent d'une main d'oeuvre qu'aucune société de développement traditionnelle ne pourrait s'offrir.

---

## 6.2 Problèmes de terminologie et de respect de l'esprit du «libre»

Le contraire du mot «libre» est «propriétaire» (en anglais *proprietary*, et non par *owner*, mais en français on ne remarque pas la nuance). «Libre» n'implique pas directement «gratuit» et ne s'oppose pas directement à «commercial»: on peut fort bien vendre des logiciels libres, au prix qu'on souhaite. La gratuité ou le prix fort faible de solutions équivalentes, sans emballage ni service, est une conséquence des conditions du logiciel libre, un effet de bord. Cette confusion vient du fait qu'en anglais les mots «libre» et «gratuit» se disent tous deux *free*. L'*Open Source Initiative* a choisi d'utiliser le terme *Open Source* (pour parler de logiciels libres selon un sens que l'on verra plus bas) pour éviter cette confusion, mais RMS n'est pas de cet avis, pense qu'il est important d'insister sur le critère de liberté, et que si la langue anglaise est mal fichue, le travail en sera plus difficile mais tant pis.

Certains jouent assez malhonnêtement sur les mots et on voit surgir de façon récurrente des polémiques sur le fait qu'un programme «libre» pour ses utilisateurs ne donne pas à son programmeur ou à la société qui a financé son développement toutes les «libertés» dont ils pourraient rêver. Évidemment! Ces parties ont des intérêts contradictoires. Mais les sociétés à but lucratif ont largement leur place dans le logiciel libre, notamment en vendant du service (lire *The Magic Cauldron* à ce sujet [cité en référence]) ou en inventant de nouveaux plans stratégiques financiers (*business plans*). De même, bien sûr, certaines licences libres cherchant à garantir aux utilisateurs leurs libertés, sont coercitives en ce sens qu'il faut les respecter. Encore une évidence! Il faut bien mettre en place un attirail compatible avec la loi sur la propriété intellectuelle telle qu'elle est. Pourtant, on entend régulièrement des gens qui semblent trouver cela trop contraignant. C'est aussi pour cela qu'il est important de se faire préciser les limites de la «liberté» d'un logiciel déclaré «libre», et la façon dont cette liberté s'exerce.

Enfin il convient d'insister sur le fait que le recours à des outils logiciels «libres» ne garantit pas que les programmes réalisés le seront effectivement. Le compilateur gcc du GNU peut naturellement produire du code propriétaire... Si la licence sous laquelle il est distribué comporte des clauses contraignant les auteurs de programmes incorporant du code libre à «libérer» leur propre code, gcc n'est, en revanche, doté d'aucun pouvoir métaphysique de «contamination»... Plus sérieusement, sa licence n'implique pas que tout programme compilé sous gcc doit être libre au sens de la GPL. Il existe par ailleurs de nombreux programmes propriétaires (compilés ou non avec gcc) qui «tournent» sous Linux, c'est à dire font appel à des bibliothèques de code libre.

## 7. Historique

Voir à ce sujet les chapitres de *Tribune Libre* [cités en référence].

Les principaux systèmes d'exploitation libres qui existent actuellement sont:

Le système GNU autour du noyau Linux (souvent abrégé en «Linux»)

Les systèmes BSD libres: FreeBSD, NetBSD, OpenBSD.

Ce sont tous des systèmes à base Unix (inventé en 1969, qui a rapidement occulté la plupart des systèmes qui se trouvaient en concurrence à l'époque. Lire à ce sujet *Le dernier dinosaure...* [cité en référence]).

Ils diffèrent principalement par les licences qui les couvrent:

Le système GNU est couvert par la GPL (ou la LGPL pour certaines bibliothèques); le noyau Linux également (même si son auteur d'origine et principal, Linus Torvalds, a appliqué un petit amendement à la GPL précisant un cas particulier qui était resté flou). Tous les utilitaires et programmes fournis avec les distributions ne sont pas sous cette licence, certains sont couverts par d'autres licences libres. C'est le cas en particulier de X, Perl, TeX et LaTeX...

Les systèmes BSD sont couverts principalement par une licence de type X ou BSD, même s'ils utilisent pour leur développement le compilateur du projet GNU, gcc.

Ils diffèrent aussi par le nombre d'utilisateurs et de développeurs: Linux est bien plus connu et médiatisé, utilisé et développé par plusieurs milliers de programmeurs de par le monde, les projets BSD sont menés eux par quelques dizaines de programmeurs, et ne cherchent pas à conquérir autant d'utilisateurs que possible.

Le code source de BSD a été rendu disponible au début des années 1980 selon la loi américaine qui impose de publier et de placer dans le «domaine public» tout ce qui a été financé par le contribuable (or les premiers systèmes BSD ont été mis au point dans des universités).

---

## 7.1 Le succès du projet GNU

Le projet GNU, pour lequel a été écrite la GPL (et la LGPL), a été lancé en 1983-84 par Richard Stallman. Lire à ce sujet *Le système d'exploitation du projet GNU...* [Cité en référence]. RMS, attristé par le départ de ses camarades de laboratoire, a cherché à créer une nouvelle communauté de partage du logiciel. Échaudé par des expériences malheureuses, il s'entoure d'avocats et blinde au maximum le côté légal pour ne pas que ses efforts soient vains suite à une faiblesse dans la licence ou autre problème technique. Il respecte donc et insiste pour faire respecter la loi telle qu'elle est, ce qui n'est pas la loi telle qu'il souhaiterait qu'elle soit: il a aussi pour but de faire évoluer les mentalités pour -- à terme -- faire changer la loi, notamment en matière de brevets logiciels (voir à ce sujet les sites d'AFUL et d'APRIL [cités en référence]). C'est pourquoi RMS a parfois des exigences que l'on peut trouver contraignantes: le détenteur du copyright d'un programme ayant seul le droit de poursuivre en justice des personnes en violant la licence, il est nécessaire de transférer son copyright à la FSF pour que celle-ci assure le suivi légal des programmes que l'on fait et place sous GPL (les auteurs français, eux, disposent toujours de leur droit moral d'auteur, qui est incessible). De même, une entreprise utilisant et modifiant en interne un logiciel couvert par la GPL n'a pas à en diffuser le code source à l'extérieur, car elle est considérée comme une entité atomique, et le passage du code source d'un employé à l'autre n'est pas considéré comme une diffusion.

Aussi incroyable que cela paraisse, la FSF, qui vend très cher des articles qu'on a le droit de revendre et de recopier, **gagne** de l'argent. C'est une forme de remerciement de la part de ceux qui lui font des dons (exemple: [Ikarios](#)) et c'est un modèle difficile à étendre car RMS et la FSF disposent d'un prestigieux capital d'image de marque.

En 1991, le projet GNU avait remplacé tous les composants d'Unix par des équivalents libres (au sens GPL le plus souvent), à l'exception du noyau, et s'était engagé pour ce dernier sur une voie escarpée et lente. C'est alors que Linus Torvalds, pour son besoin personnel, démarre le développement de la dernière pièce du puzzle, suscitant un grand engouement: on pouvait enfin disposer d'un système libre! Il appelle son noyau «Linux», et peu à peu les distributeurs pressent des CD-ROM, facilitent l'installation, financent des développements complémentaires. La courbe du boom de Linux suit celle de la diffusion de l'Internet auprès du grand public (l'Internet existant bien des années avant que la presse généraliste n'en parle). Tout le monde s'engouffre dans la mode, en se figurant parfois pouvoir cumuler le beurre et l'argent du beurre: licences se déclarant «libre» mais qui violent telle ou telle clause de la GPL pour tenter de conserver un avantage (en escomptant tromper un acheteur naïf et mal informé qui ne se fierait qu'à l'étiquette du produit). Peu après, la distribution Debian de GNU/Linux se heurte au problème des logiciels de confort qu'elle choisit de fournir en sus des logiciels systèmes de base. Certains programmes intéressants en effet ne sont pas couverts par la GPL ou une licence de type BSD: où tracer la limite? La discussion et la réflexion mènent à la définition de l'*Open Source*, marque déposée, donnant une liste de critères qu'une licence doit honorer pour s'y conformer. La création de cette organisation à but non lucratif est fortement liée à l'annonce faite par la société Netscape de «libérer» (au sens d'une licence spécifique écrite par eux pour l'occasion) le code source de son produit phare, Navigator.



## 8. Licences

Un trait commun à l'ensemble des licences présentées ci-dessous, repose sur l'absence de garantie de la part de l'auteur ou des auteurs concernant d'éventuelles conséquences du fonctionnement ou dysfonctionnement de leurs logiciels. Cette caractéristique n'est d'ailleurs pas propre au logiciel libre, puisque revendiquée pour la quasi totalité des licences propriétaires.

Domaine public Après un délai qui varie selon les lois et les pays (ou avant, si l'auteur le veut), une oeuvre du domaine public est librement modifiable et distribuable (y compris commercialement) sans contrepartie. C'est le cas «libre de droits».

GPL (qui n'est PAS «libre de droits») ou [General Public License](#), ( [Licence Publique Générale](#)) utilisée dans le projet GNU Si on distribue (y compris commercialement) un programme sous GPL ou une version modifiée de celui-ci, on est tenu de fournir les sources (ou les sources de la version modifiée). Si on le modifie, on est tenu de placer le résultat également sous GPL. C'est également le cas des programmes qui sont développés à partir d'une base sous GPL. Ce caractère infectieux et viral de la GPL empêche dans certain cas l'interopérabilité avec des programmes placés sous d'autres licences, mais a des conséquences intéressantes en matière de garanties pour l'utilisateur ou le programmeur bénévole, qui se traduisent par des projets de développement souvent plus dynamiques. Ce côté viral et infectieux (à ne pas prendre dans un sens péjoratif) est appelé le "copyleft", ou «gauche d'auteur»: c'est utiliser le droit d'auteur ("copyright") pour le prendre à revers. Il n'est pas nécessaire qu'une licence soit "copyleftée" pour remplir les 4 critères du logiciel libre selon RMS.

LGPL [Lesser General Public License](#) ( [Licence Publique Générale Limitée](#)) Version particulière de la GPL (appelée *Library* General Public License avant de devenir *Lesser* General Public License), pour les bibliothèques de programmation. Ce sont les mêmes idées, reformulées dans le cas particulier des bibliothèques. Désormais, la FSF et le projet GNU dissuadent d'utiliser cette licence, car ils estiment que stratégiquement ce n'est plus nécessaire.

FDL [Free Documentation License](#) Licence pour la documentation technique, reposant sur les idées de la GPL. Les notions de codes «source» et «binaire» sont remplacées par des formats «transparents» ou «opaques».



---

BSD1, BSD2, X11 Licences fort proches, utilisées dans les systèmes BSD libres ( [FreeBSD](#), [OpenBSD](#), [NetBSD](#)) ou le système de fenêtrage X ( [X Window System](#)), qui assure les fondations de tous les modes graphiques pour les Unix. On a le droit d'utiliser ce code source et de ne redistribuer qu'une version binaire, c'est donc libre au sens de RMS, mais pas viral. Leur différence principale avec la GPL est qu'il est possible de rendre du code propriétaire, en n'en publiant qu'une version binaire. Ce sont des licences libres fort simples, proches du domaine public, non «copyleftée».

[Artistic License](#) ( [Licence Artistique](#)) Licence de distribution de Perl, moins contraignante que la GPL du point de vue des entreprises, et permettant l'interopérabilité que la GPL interdit, dans certains cas.

OSD [Open Source Definition](#) ( [Définition de l'Open Source](#)) Ensemble de conditions que doit remplir une licence de logiciel pour avoir la marque "Open Source". GPL, BSD, Artistic les remplissent. C'est le critère utilisé par la distribution Debian pour juger de ce qui est «libre» ou pas. Ce n'est pas en soi une licence.

Autres licences proches:

[NPL](#) (Mozilla);

[QPL](#) (bibliothèques utilisées dans KDE);

[OPL](#) (OpenContent License, pour les contenus);

[Bugroff](#) (à lire par curiosité);

SCSL Sun Community Source License (SCSL) pas libre);

Apple (pas libre).

## 9. Annexe: quelle confiance peut-on accorder à un code source?

Même quand on dispose du code source d'un programme, on peut avoir de mauvaises surprises si on ne prend pas certaines précautions. Commençons par planter le décor, en posant les jalons qu'on ne pourra jamais dépasser.

### 9.1 L'affaire Ken Thompson

[Ken Thompson](#) est l'un des meilleurs spécialistes au monde d'Unix et du langage C (très utilisés aujourd'hui), puisqu'il fut l'un de leurs inventeurs et premiers utilisateurs. Lorsqu'il a reçu le [Turing award](#), prix d'informatique dont le prestige est comparable à celui du Nobel, il a prononcé un discours intitulé [Reflections on Trusting Trust](#) («De la défiance en la confiance»).

Dans ce discours, il explique comment il a imaginé, lui qui concevait un système d'exploitation utilisé dans de grands comptes, se réserver une [back-door](#) («porte d'entrée cachée, dans un système, réservée à ceux qui en connaissent le secret») dans un programme utilisé par tous les systèmes Unix pour s'y connecter (à l'époque): login.

Bien sûr, il était hors de question d'indiquer cela en lettres de feu dans le code source de ce programme, il a donc caché ce petit truc dans le *compilateur*: le compilateur reconnaissait qu'il compilait le programme login, et ajoutait la *back-door* de son propre chef!

Pour effacer davantage ses traces, il s'est arrangé pour que le compilateur reconnaisse même qu'il était en train de se recompiler lui-même, afin d'ajouter dans le code binaire de l'exécutable du compilateur engendré les instructions prenant l'initiative de placer la *back-door* dans le programme login, ainsi que les instructions prenant l'initiative de placer la *back-door* de niveau supérieur dans le compilateur lui-même!

Il ne lui restait plus qu'à distribuer des codes sources innocents pour le programme login et le compilateur, sachant que s'il ne distribuait que des versions malignes du binaire du compilateur, tous les binaires de compilateurs ou programmes login jamais engendrés souffriraient de ces mêmes *back-doors*!

C'est probablement la bidouille la plus diabolique de tous les temps, et la conclusion de Ken est sans appel: **on ne peut pas avoir confiance en un code qu'on n'a pas complètement écrit soi-même**. Ce qui est vrai du compilateur l'est aussi pour de nombreux autres programmes de la chaîne de fabrication: éditeur de liens, chargeur, microcode au niveau du matériel. Plus le niveau de programmation considéré est bas, plus il est difficile de repérer le piège, et cela est presque impossible dans les derniers niveaux.

## 9.2 Les bizarreries de ssh

Avec le développement de l'Internet et des attaques où l'indélicat écoute tranquillement les mots de passe et les données transiter en clair sur le réseau, la nécessité d'une méthode de connexion plus fiable est apparue. La société [SSH Communications Security Ltd.](#) a proposé le logiciel [SSH](#), qui utilise des méthodes cryptographiques pour chiffrer le contenu des connexions, et notamment une authentification *Zero knowledge* (qui n'apporte aucune information à un indélicat qui espionnerait l'ensemble de la procédure).

La loi française limitant à l'époque (et encore de nos jours, mais moins durement) la taille des clés utilisables en cryptographie, il a fallu corriger ssh pour qu'il reçoive l'autorisation de mise sur le marché en France par les autorités compétentes (sous le nom *ssf*). Il était donc nécessaire que la personne qui se charge de cette opération comprenne bien l'ensemble du code source du programme, afin de ne rien laisser au hasard, car, en cryptographie, une erreur est vite arrivée.

[Bernard Perrot](#) a remarqué des détails troublants, qu'il a fait paraître dans un bulletin du [CNRS](#) intitulé [Sécurité informatique](#) (numéro [23](#), pages [5](#) et [6](#); merci à [DaLinuxFrenchPage](#) pour cette [info](#)) info. Il a depuis donné des détails dans une [conférence](#) qu'il a donnée au sujet de [SSF](#), diapositives [61](#) et suivantes, dont de nombreuses sont intitulées «Bogues de SSH». Ces bogues sont parfois très graves, et incompréhensibles dans un programme vieux de trois ans d'âge, comme l'était ssh au moment de son examen par le chercheur français. Ce dernier n'exclut pas que ce soit des *back-doors* insérées sciemment à la demande du gouvernement; et depuis que l'existence du programme [Echelon](#) a été rendue publique, on ne saurait trop se méfier en la matière.

La cryptographie est une science très difficile, et la moindre faille peut profiter à qui a les moyens de l'exploiter. Ce n'est pas le [Handbook of Applied Cryptography](#) («manuel de cryptographie appliquée») qui vous enseignera le contraire; pas non plus les diverses FAQ de cryptographie qu'on peut trouver, [en anglais](#) ou [en français](#). On ne s'improvise pas cryptographe; ce n'est pas pour rien que les gouvernements font des ponts d'or aux meilleurs mathématiciens et spécialistes depuis qu'ils ont compris l'intérêt de cette science dans l'espionnage et le contre-espionnage (c'est-à-dire la première guerre mondiale, mais surtout la deuxième).



On est ici dans un cas de figure où même la lecture de la totalité du code source du programme, avec l'impression de le comprendre, ne vous aurait rien apporté. Ainsi **personne** n'avait en trois ans signalé ces failles que le Français, permettant au gouvernement français de signaler que la loi française permet de produire des logiciels plus sûrs, même s'ils utilisent des clefs moins longues, juge inadmissibles! On ne connaît pas, de manière générale, d'exemple de logiciel de cryptographie vérifié et certifié par un cryptographe professionnel, et les plus folles rumeurs courent également sur [PGP](#) (*Pretty Good Privacy*, ou «confidentialité sacrément bonne»; programme permettant de chiffrer et d'authentifier les documents et en particulier les courriers électroniques). Rien de plus facile bien sûr que d'insinuer sans donner de détails, mais en la matière, on n'est jamais assez méfiant.

On peut quand même signaler à ceux que la question inquiète, que désormais on trouve des équivalents fonctionnels (vraiment libres, de plus) à ces deux programmes (dont certaines conditions de distribution n'en faisaient pas des programmes libres, malgré la mise à disposition du code source): [GPG](#) (*GNU Privacy Guard*, ou «gardien de la vie privée du projet GNU») peut remplacer `pgp`, et [OpenSSH](#) peut remplacer `ssh`. De plus, ces deux programmes ont été mis au point par des fondations ou groupements de volontaires que la protection de la vie privée intéresse au plus haut point.

### 9.3 Le cheval de Troie d'IRC2

L' [Internet Relay Chat](#) («bavardage par relais sur l'Internet») est un réseau de discussion organisées en canaux très utilisé et célèbre qui a vu le jour en Finlande. Longtemps, le client le plus utilisé était `irc2`, dont les sources étaient proposés par de nombreux sites FTP, et notamment la machine `ftp.funet.fi`.

Ce n'est qu'après plusieurs fois qu'on a remarqué que les sources avaient été corrompues par un petit malin et que les personnes qui les avaient téléchargées, compilées et installées entre-temps avaient laissé le cheval de Troie s'introduire dans leur système.

Il n'est pas question ici de remise en question des mécanismes fondamentaux de la machine, de subtilités diverses, ou de théories mathématiques compliquées, mais d'absence de contrôle de l'intégrité des fichiers en lesquels on a placé sa confiance: les lire suffisait à remarquer la supercherie.

De nos jours, les différentes distributions de GNU/Linux et des BSD libres fonctionnent avec des signatures pour chaque paquetage logiciel, signature, que seuls peuvent produire les responsables officiels de chaque paquetage, en lesquels il faut bien avoir confiance!

#### **9.4 La fonction de «vérification» de GnuPG ne vérifiait rien**

Le 13 octobre 2000, on recevait [confirmation](#), sur une liste de diffusion spécialisée dans ces alertes de sécurité, que la fonction de vérification de GnuPG, pour toutes ses versions jusqu'à la version 1.0.3, ne vérifiait en cas de signatures multiples, que la première signature, et marquait chacun des documents signés juste ou bon en fonction du résultat de cet unique test.

Il est significatif que la personne qui a confirmé et corrigé ce bogue signale:

This problem has been in GnuPG since the beginning but Jim's seems to be the first one who noticed that. We need better auditing folks! This bug is just one more prove that "given enough eyeballs all bugs are shallow" can not be held true when it comes to the security bugs; well, the bugs are probably found faster - but most times only be coincidence.

*(Ce problème était présent dans GnuPG depuis ses toutes premières versions mais Jim est apparemment la première personne à l'avoir remarqué. Il nous faut de meilleurs relecteurs de code! Ce bogue prouve une fois de plus, s'il en était besoin, que l'aphorisme «Étant donnés suffisamment d'observateurs, tous les bogues sautent aux yeux.»*

*Référence à la «loi de Linus», mentionnée dans la [La cathédrale et le bazar](#) ne s'applique pas dans le cas particulier des bogues de sécurité; il est certes exact que les bogues sont trouvés plus rapidement, mais le plus souvent, uniquement par hasard.)*

Une fois de plus, la libre disposition d'un code source est une condition nécessaire mais pas suffisante si personne ne prend la peine de l'examiner, ou si ceux qui le font ne sont pas compétents, ou si ceux qui trouvent des problèmes ne le signalent pas publiquement.

#### **9.5 Conclusion: ne plus utiliser d'ordinateur?**

Avant de fondre dans le pessimisme, rappelons l'un des premiers préceptes que tous les étudiants en cryptographie apprennent: de même qu'il est inutile de placer une porte en titane renforcé dans une cloison de plâtre (c'est le maillon faible de la chaîne qui compte), de même il est inutile de protéger ses données ou ses transmissions d'une manière engendrant des coûts pour celui qui veut casser le secret sans commune mesure avec la valeur du secret. Pour évaluer ces coûts, il faut bien sûr se fier aux dernières découvertes théoriques et à la puissance des dernières machines sorties, sachant que les agences gouvernementales entretiennent un flou impénétrable quant à leurs moyens et possibilités réels, et qu'elles se font fort d'avoir au moins «5 ans d'avance sur l'état de l'art».



---

C'est vraiment un monde imparfait, et nous n'avons chacun que 24 heures dans une journée. Puisque se méfier de tout et de tous implique en particulier de ne pas utiliser d'ordinateur et de ne presque plus rien faire, il faut bien faire quelques concessions (à la paranoïa, à la vie privée, ou au degré de sécurité recherché!). Nous nous sommes échinés à démontrer dans la présente section que disposer des sources (ce qui n'est pas toujours suffisant pour que le logiciel incriminé soit qualifiable de «libre») n'était pas l'herbe du Pantagrué lion. «Il faudrait» plutôt ou aussi, et par ordre décroissant de préférence:

les avoir écrites soi-même (ainsi que le compilateur, et toute la chaîne de compilation, et avoir fondu la puce, extrait le minerai pour la fabriquer, travailler profondément enfoui sous une montagne dans une cage de Faraday en produisant soi-même son électricité... Rappel: il faut savoir arrêter l'escalade au bon moment!)

les avoir relues entièrement soi-même (et comprises; attention on peut avoir l'impression de comprendre et ne pas remarquer telle ou telle subtilité vicieuse)

que ces sources aient été relues par beaucoup d'autres personnes, spécialistes et «indépendantes», régulièrement (ce qui implique des diffusions régulières et fréquentes, même de versions imparfaites ou incomplètes du logiciel, lors de son développement)

Est-il encore utile de faire remarquer que, comparés aux astuces à mettre en place lorsque les codes sources sont disponibles, les pièges dont on peut truffier des programmes qui ne sont livrés que sous forme binaire sont du nanan à mettre en place, et que dans tous les cas, si on veut être cohérent avec soi-même, on ne doit accorder aux logiciels uniquement diffusés sous forme binaire une confiance proportionnelle à celle qu'on accorde aux autres (et, si on est un brin paranoïaque et cohérent, absolument **nulle**).

## 10. [Remerciements](#)

## 11. [Références](#)

<http://www.linux-france.org/article/these/> Traductions de licences: GPL, LGPL, X, Artistic, OpenContent... Textes relevant de la «philosophie» du logiciel libre, thèses sur le sujet, questions de droit, mémoires de mercatique... Lire notamment les articles de fond: *La cathédrale et le bazar*; *Le chaudron magique*; *Le dernier dinosaure*... Versions originales: <http://www.tuxedo.org/~esr/writings/> <http://muq.org/~cynbe/rants/lastdino.htm>



---

<http://www.fsf.org/>, notamment <http://www.fsf.org/philosophy/license-list.html> (Analyse des différentes licences), et <http://www.april.org/> La FSF (et le projet GNU). (Licences GPL, LGPL, FDL.) et son représentant en France, l'APRIL.

<http://www.editions-oreilly.fr/catalogue/tribune-libre.html> V.O.:  
<http://www.oreilly.com/catalog/opensources/book/toc.html> Recueil d'essais des principaux acteurs du logiciel libre, avec notamment un historique des projets BSD, du projet GNU, et un commentaire de l'OSD. Lire en particulier les chapitres: « Deux décennies ... », « Le système d'exploitation du projet GNU ... », « La définition de l'Open Source »

<http://www.crao.net/gpl/> Étude juridique de la GPL du point de vue du droit français. Elle conclut que la GPL, bien qu'écrite pour le droit anglo-saxon et que sa seule version officielle soit en langue anglaise, est compatible avec le Code de la Propriété Intellectuelle si on l'interprète comme un contrat international.

<http://www.iful.org/> Une autre association de promotion du logiciel libre en France, luttant en particulier contre les brevets logiciels.

<http://www.scam.fr/fr/auteur/documentation/cpi.htm> Code de la Propriété Intellectuelle. La plupart des textes de loi sont disponibles sur <http://www.legifrance.gouv.fr/> et sur de nombreux autres sites (lire la FAQ du groupe Usenet fr.misc.droit)

Linux magazine France ( <http://www.linuxmag-france.org/>) présente un article de deux pages d'analyse des licences du logiciel libre dans son numéro d'avril 2000, qui devrait être publié sur leur Web 6 mois après cette date.

Les définitions en Français de milliers de termes relevant du logiciel libre, de l'Internet ou de l'informatique en général ( <http://www.linux-france.org/prj/jargonf/>). L'équivalent avec les termes et les définitions anglais: <http://www.netmeg.net/jargon/>.



---

# *Qu'est-ce que l'Open Source*



## 1. Diffusion du présent document

Ce document appartient à [IDEALX](#). Il est librement diffusable dans les termes de la [Licence de Documentation Libre GNU](#) ( [GNU Free Documentation License](#)).

## 2. Une méthode de génie logiciel dérivée de la recherche scientifique

3.

Bien qu'il lui préexiste, le logiciel *Open Source* doit son extraordinaire développement à l'Internet, son milieu écologique de prédilection. C'est pour ainsi dire naturellement que les habitudes de travail coopératif des scientifiques --- parfois eux-mêmes programmeurs --- ont été assimilées et plébiscitées par les créateurs des logiciels libres.

Mais il faut aussi remarquer que ces logiciels libres proviennent essentiellement du monde Unix, où la diversité matérielle des plates-formes et des environnements impose souvent la diffusion des sources des logiciels afin de pouvoir les compiler « sur site » et prendre ainsi en compte les particularités matérielles et environnementales. D'où une culture « spontanée » de la disponibilité du code source, qu'il ne faut pas confondre avec la culture de l'*Open Source* ou du logiciel libre, mais qui y prédispose plus naturellement que dans le monde de la micro-informatique de ces deux dernières décennies, où *distribution de programmes* ne semblait pas pouvoir signifier autre chose que *distribution de binaires*.

L'Internet constitue un terrain privilégié d'expérimentation et de décantation des normes. Lesquelles ne deviennent effectives et ne sont finalement adoptées qu'après sélection et confrontation pratique, eù égard surtout à leur pertinence technique --- par opposition à des critères qui ne relèveraient, par exemple, que de l'opportunisme ou de l'influence de grandes entreprises soucieuses avant tout de défendre des intérêts particuliers.

En soi, l'évident succès de l'Internet (*i.e.* le fait que les protocoles de la galaxie TCP/IP se sont imposés face aux « solutions » propriétaires) démontre la supériorité de la normalisation, de l'interopérabilité et du code source ouvert. Mais il faut insister aussi sur les effets de démultiplication des échanges, qui permettent de diffuser et corriger des logiciels en quelques heures.

## **2.1 Qu'est-ce que le « code source publié » ?**

Au sens purement technique, il faut entendre l'expression « code source publié » (*open source code*) comme la mise à disposition sous forme numérique et facilement duplicable du code de programmation tel qu'il a été saisi, le plus souvent par des programmeurs humains.

À cette étape, il demeure compréhensible --- toutes choses égales par ailleurs --- et modifiable par des informaticiens maîtrisant le langage dans lequel le logiciel a été écrit. Après « compilation », en revanche, c'est-à-dire après sa transformation en code exécutable par l'ordinateur, toute intervention de correction, maintenance ou évolution devient beaucoup plus difficile, voire impossible.

Toutefois pour permettre une intervention efficace sur un programme informatique à partir de son code source, encore faut-il que le programmeur ait produit un code réellement *lisible* et bien documenté. Ce n'est pas parce qu'un algorithme peut être identifié et « compris » que la tâche qu'il accomplit en pratique s'en trouve moins obscure. Faute de commentaires, il peut être nécessaire de reconstituer mentalement --- en suivant la piste du code source comme on démêlerait un écheveau --- ce que fait réellement une section de code. Au delà d'un certain nombre de lignes de codes cela relève de la gageure.

On voit par là que si les programmeurs originaux n'ont pas eu l'intention de rendre le fruit de leur travail accessible à des tiers, la simple mise à disposition du code source (au même titre que la décompilation d'un programme binaire) peut s'avérer inutile ou, à tout le moins, ne simplifier qu'une partie dérisoire du travail d'analyse. Pour user d'une autre métaphore, ce n'est pas parce que l'on connaît la définition de chacun des mots d'une phrase que l'on en comprend nécessairement le sens. Même s'il s'agit d'une langue non naturelle.

## **2.2 Coopération et reproductibilité**

Ainsi, de la même manière qu'un scientifique s'efforcera, en établissant le protocole d'une expérimentation, de faire en sorte qu'un collègue puisse en reproduire l'agencement et, si possible, en tirer des conclusions identiques, un informaticien travaillant en *Open Source* veillera à documenter et expliciter son code afin que ses pairs soient en mesure de le critiquer, et le cas échéant, d'y apporter les corrections ou compléments qui s'avèreraient nécessaires.

Les théoriciens de l'*Open Source* identifient, chez les scientifiques et chez les programmeurs, un même souci d'établir les faits ou, à tout le moins, le comportement des instruments de mesure ou le comportement d'un programme informatique avec autant de « robustesse » dans les deux cas. Ainsi, l'activité de « débogage » et d'examen critique d'un code source par des personnes compétentes peut s'apparenter à la validation d'un protocole expérimental par sa reproduction dans un autre laboratoire.

Corollairement, cette « disponibilité » du code source favorise l'intégration et la complémentarité des ressources humaines. Au même titre que des briques logicielles peuvent être utilement conçues en vue d'une intégration dans de multiples projets, les programmeurs *Open Source* mettent tout simplement le potentiel de création spécifique de chacun à la disposition de l'ensemble de la communauté. Une entreprise fonctionnant suivant le mode de l'*Open Source* bénéficiera du développement d'un patrimoine commun de la même façon qu'elle dispose de la publication des travaux de la recherche scientifique.

### **2.3 Secrets industriels et valeur ajoutée**

Bien sûr, en ce qu'elle valorise la coopération et la rationalisation des ressources matérielles et humaines, cette approche « modulaire » ne présente rien de révolutionnaire en soi. De par ses similitudes avec la recherche scientifique, elle prévaut dans les services de programmation des éditeurs de logiciel ou les sociétés d'ingénierie informatique, au même titre que dans les laboratoires. Mais l'effort de transparence et de coopération s'arrête le plus souvent aux portes de l'entreprise. Parfois même à celles des différents services, qui veillent jalousement à conserver la maîtrise de leurs créations.

Contrairement au monde scientifique où, en principe, une découverte va rapidement faire le tour du monde et contribuer à l'émergence de travaux complémentaires --- voire radicalement novateurs --- le culte du secret industriel a au contraire prévalu dans le monde de l'informatique propriétaire. Tant en termes de « secrets de fabrication » qu'en termes de « spécifications » s'apparentant aux secrets industriels. Or la confidentialité du code alliée à la non diffusion des caractéristiques essentielles d'un logiciel, contrarie sérieusement toute implémentation dans un ensemble intégré plus large, en dehors de tout problème de licence.

Au lieu de considérer la nature de la valeur ajoutée par leur intervention dans leur domaine de compétence comme un « service » rendu au client, les éditeurs de logiciels propriétaires ont au contraire axé leur politique sur la facturation de « produits », dont il importe de maintenir un prix élevé par l'organisation artificielle de la rareté. Quand bien même la nature immatérielle du « produit » se prête à une duplication à des coûts dérisoires --- avec pour conséquence la multiplication des normes « de fait » et l'impossibilité de faire communiquer des applications pourtant complémentaires sur le plan fonctionnel.

---

### **3. Une philosophie du développement adaptée aux NTIC**

Parce qu'il recouvre des pratiques « communautaires » issues de la recherche scientifique et du monde Unix, le concept d'*Open Source* relève de positions philosophiques (ou, si l'on préfère, sociales et éthiques) et déborde, là encore, le cadre de la simple mise à disposition publique du code source. Il implique liberté et responsabilité.

Sans constituer une licence au sens juridique ni se substituer à une licence de distribution particulière, la définition de l'*Open Source* (OSD) vise avant tout à garantir les droits des utilisateurs et des auteurs des programmes. Droit d'utilisation, de modification et de duplication ; mais aussi droit de voir son travail reconnu et --- le cas échéant --- rémunéré, selon des modalités bien adaptées aux nouvelles technologies de l'information et de la communication (NTIC).

S'agissant des droits concédés aux utilisateurs et du thème connexe de la liberté, il n'est guère possible d'éviter la figure de Richard M. Stallman, qui fut, dès 1984, un des premiers à promouvoir et organiser ce mode de développement, en insistant au moins autant sur les conséquences sociales, nécessairement favorables à ses yeux, que sur les implications économiques induites par le phénomène du « logiciel libre ». « RMS » est ainsi le créateur du *Copyleft*, parfois appelé « gauche d'auteur » en France, qui garantit explicitement le droit de copie.

#### **3.1 « Logiciel libre » vs « Open Source »**

Tandis que Richard Stallman, créateur de la FSF (*Free Software Foundation*) et du projet GNU (la « première » communauté de partage du logiciel) insiste avant tout sur la dimension sociale (voire messianique) du mouvement du logiciel libre, les créateurs du concept *Open Source* insistent plus volontiers sur la dimension économique du phénomène. Lequel a manifestement trouvé un terrain favorable et une écoute à tout le moins attentive dans l'univers entrepreneurial, au croisement de l'esprit d'innovation et de la réticence à l'égard de la constitution de monopoles. Il rejoint également une tendance vers un mouvement de normalisation suivi par les composantes les plus dynamiques de l'industrie logicielle, et qui répond à une demande croissante d'interopérabilité de la part des utilisateurs.

Avec oecuménisme, et dans le souci d'écartier toute polémique et risque de schisme dans le mouvement rassemblant des composantes très diverses du logiciel libre, Bruce Perens, qui a été à l'origine de la rédaction du manifeste *Open Source* en collaboration avec Eric Raymond, estime que ce dernier et Richard Stallman « ne font qu'utiliser des méthodes différentes pour défendre le même concept ». On pourrait également remarquer que les uns et les autres insistent chacun sur un aspect différent (implications sociales/ambitions économiques) d'un même phénomène interférant sur des domaines que l'on n'a pas coutume d'associer. Cette richesse conceptuelle se retrouve par ailleurs dans l'association d'individualités et de compétences qui contractent, au sein du mouvement *Open Source*, des alliances jusque là inédites.

---

Il n'en demeure pas moins que pour Richard Stallman, le but du projet GNU est bien « de faire des avancées sociales, c'est-à-dire d'accroître la liberté des utilisateurs ». C'est sur cette base qu'il juge un logiciel libre en toutes circonstances préférable à un logiciel propriétaire, fût-il techniquement plus satisfaisant. Plus mesurés sur ce point, les promoteurs de l'Open Source se sont montrés favorables à des rapprochements de circonstance avec des compagnies à priori moins réputées pour leur engagement en faveur du « libre » que pour leur puissance logistique et financière, voire leur propension avérée à l'hégémonie. Concessions « tactiques », pour autant que les visées stratégiques demeurent inflexibles quant aux quelques principes clés exposés dans la « définition de l'*Open Source* ».

### **3.2 Open Source : un concept normatif et régulateur**

Normatif et régulateur, le concept d'*Open Source* s'efforce de tracer une ligne de démarcation explicite entre ce qui relève ou non de l'esprit du logiciel libre. Comme on vient de le voir, il recouvre cependant une réalité qui préexiste à sa formulation « canonique », plus récente que le phénomène qu'elle entend décrire et encadrer --- d'autant qu'il ne s'agit, en fin de compte, que de lui décerner un brevet d'honorabilité à l'égard du marché.

La rédaction de la définition de l'*Open Source* a été réalisée à posteriori, afin de prendre en compte les licences existantes avec pragmatisme. De ce fait, son champ d'acceptation effectif s'avère peut-être moins restrictif que ses auteurs n'auraient pu le souhaiter. La définition officielle n'en propose pas moins un cadre conceptuel à la fois suffisamment rigoureux pour éviter les dérives tout en conservant une relative souplesse, afin de demeurer compatible avec des licences préexistantes et mutuellement exclusives. Comme par exemple la licence GNU, à l'initiative de la FSF, plus intransigeante en ce qu'elle interdit l'appropriation (voire la confiscation des améliorations) de code par des programmeurs tiers, alors que, par exemple, la licence BSD envisage cette hypothèse avec « réalisme » et même un sang froid revendiqué.

Toutefois, dans l'esprit de ses créateurs (Bruce Perens, Eric Raymond, Larry Augustin et Sam Ockman) le cadre théorique de l'Open Source doit trouver son prolongement dans une campagne de promotion active auprès des différentes parties prenantes --- tout particulièrement auprès de l'industrie du logiciel. Autour du manifeste de référence, se développe un corpus de commentaires, illustrations et analyses prospectives. Dans le même esprit, Eric Raymond n'hésite pas à s'engager personnellement dans les débats qui se déroulent à l'intérieur des sociétés qui envisagent de passer tout ou partie de leur code en *Open Source*, allant même jusqu'à proposer ses services en vue de faciliter les négociations. Ce fut notamment le cas chez Netscape où il a collaboré à la rédaction de la licence qui « libérait » le code du navigateur Mozilla.

---

### **3.3 Partage des connaissances et « coopération »**

Au delà des aspects formels relevant d'une définition purement technique ou « juridique » (bien que, encore une fois, ce sont les licences particulières qui régissent ce dernier aspect --- la définition de l'*Open Source* (OSD) n'ayant pas *en soi* valeur contractuelle) l'*Open Source* peut à bon droit prétendre au titre de philosophie, en ce qu'il engage ses adeptes dans une vision du monde. Cette doctrine privilégie la coopération, le partage des connaissances et donc l'enrichissement mutuel, par opposition à une conception privative de la connaissance et de la propriété intellectuelle qui assimile le savoir à une marchandise qu'il convient de thésauriser pour en accroître la valeur.

Comme les idées, les algorithmes et les programmes informatiques ne viennent pas spontanément au monde. Plus divers sont les échanges et multiforme la coopération, plus riches sont les créations. Certes, des entreprises continuent de parier sur la valeur du secret, veillant à ne pas révéler leurs propres innovations tandis qu'elles s'efforcent à rebours de tirer profit du travail intellectuel des autres (intelligence économique). Ce schéma, dérivé du monde matériel où le partage d'un bien implique une part de dépossession, s'avère de plus en plus coûteux dans l'univers de l'informatique, où toute tentative de verrouillage finit par se payer au prix fort. Si, comme on le dit, la connaissance est pouvoir, ce n'est certainement pas dans le sens où, dans le dessein de régner, les plus habiles pratiqueraient la rétention d'information. Au contraire, ne pas partager l'information (fluide et éphémère par définition), c'est prendre le risque de l'isolement et de se faire dépasser tandis qu'on consacre l'essentiel de son énergie à la protection de trouvailles rapidement obsolètes.

La coopération, a fortiori dans un domaine où règne une forte compétition --- comme les nouvelles technologies de l'information et de la communication --- s'impose donc comme un enjeu stratégique vital. Pour décrire cette ambivalence, le terme « coopération » s'est imposé dans le vocabulaire économique et entrepreneurial, après avoir été utilisé dans la sphère politico-économique internationale, comme au forum de Davos.

Même les pratiques opportunistes --- voire prédatrices --- ne sont pas à craindre. Si une entreprise s'avisait d'incorporer du code libre dans ses propres logiciels sans rien faire bénéficier en retour la communauté *Open Source* de ses innovations, alors elle se priverait de l'apport le plus significatif du développement à code source ouvert. Elle ne témoignerait que de son incompréhension profonde du phénomène, en réifiant la valeur marginale de quelques lignes de code, alors que leur richesse réelle repose sur le travail *coopératif* des programmeurs et sa constante remise en cause.

---

#### 4. Pourquoi faut-il choisir l'Open Source ?

*Comment se peut-il qu'un bien distribué à plusieurs possesseurs, rende plus riche de soi, que s'il n'appartenait qu'à quelques-uns ?* Dante, *La Divine Comédie*, Purg. XV 61-63, trad. J. Risset.

C'est tout bonnement « en marchant » que le mode de développement *Open Source* fait la preuve de son efficacité. Le succès --- voire d'ores et déjà l'hégémonie dans certains domaines --- des logiciels qui ont été conçus selon ce mode de développement constitue sans doute, plus encore que des arguments rhétoriques, le meilleur avocat auprès des utilisateurs, des programmeurs ou des entreprises.

On a souvent relevé que si les logiciels libres devaient disparaître d'un coup, l'Internet n'y survivrait pas. De fait, et malgré les tentatives récurrentes de mainmise et de « propriétérisation » des normes de la part de puissants éditeurs de logiciels « fermés », l'essentiel des programmes qui constituent le coeur du réseau des réseaux répondent aux critères définis par la définition de l'*Open Source* (OSD) et doivent tout à la collaboration de la communauté mondiale des développeurs « libres ».

Ainsi, fin mars 2000, le serveur HTTP *Apache* avait franchi le cap des 60% de « parts de marché » (<http://www.netcraft.com/survey/>) laissant loin derrière les applications propriétaires concurrentes. *Bind*, pour sa part, constitue l'implémentation de référence en matière de serveurs de noms (DNS), tandis que le langage *Perl*, particulièrement bien adapté aux technologies du World Wide Web, s'est répandu sur la quasi totalité des plates-formes matérielles et des systèmes d'exploitation --- même là où la culture des logiciels libres ne s'impose pas aussi spontanément que dans le monde Unix. Pourtant, si le simple constat de ce succès représente la meilleure preuve de sa suprématie technique, il convient de ne pas se dérober à l'argumentaire. En d'autres termes, il ne suffit pas de montrer que *ça marche*, encore faut-il assurer que cela peut être *profitable*.

##### **4.1 Limiter les coûts de développement**

Comme le relève Bernard Lang, responsable scientifique du projet ATOLL à l'INRIA,

« l'énergie naguère dépensée pour multiplier les biens est maintenant consacrée à trouver les moyens d'empêcher leur multiplication, ce qui entrave la mise en oeuvre efficace des outils informationnels et met en péril la pérennité des contenus » ([Des logiciels libres à la disposition de tous, Bernard Lang, Le Monde Diplomatique, janvier 1998](#))

Ceci alors même que, par la magie de l'électronique, les coûts de duplication de l'information tendent vers zéro. D'où l'instauration d'un état d'abondance et de disponibilité virtuelle de fait dont n'osaient rêver les générations précédentes. La situation s'est renversée : c'est maintenant pour empêcher la copie ou faire en sorte qu'elle ne soit pas gratuite que des entreprises investissent, dans la mesure où elles ne conçoivent pas d'autre issue pour rentabiliser leurs coûts de développement. Mais le serpent se mord la queue.

---

On sait qu'en matière de brevets, par exemple, les coûts relatifs à la veille à posteriori représentent l'essentiel des dépenses, bien plus que les droits de dépôt initiaux. Car il ne suffit pas de déposer un brevet pour assurer la « protection » de son savoir-faire. Encore faut-il mettre en place un réseau de surveillance parfois extrêmement étendu afin de pouvoir, le cas échéant, « piéger » le concurrent indélicat. Hormis pour les firmes de dimension internationale, ce mandat sera la plupart du temps confié à des sociétés spécialisées. Or les fonds consacrés à ces prestations ne constituent pas à proprement parler un investissement productif, bien qu'ils représentent la part principale d'un budget « brevet ». Certes, cette stratégie de la citadelle assiégée a pu donner des résultats (permettre d'accumuler des bénéfices assis sur la défense et l'exploitation de technologies propriétaires) dans les domaines qui ne progressent pas rapidement.

Dans le secteur informatique, où les évolutions des techniques et des savoir-faire sont au contraire très rapides, cette stratégie a peu de chances de s'avérer payante à moyen et long terme. Quelques sociétés, néanmoins, se sont efforcées de l'adapter en imaginant des protections matérielles contre la copie logicielle. Elles accroissaient ainsi les coûts de développement et de conditionnement de leurs produits sans que leurs clients --- tout au contraire, puisqu'on les contraint ainsi à financer leurs propres chaînes --- bénéficient de ces investissements. Enfin elles permettaient à leurs concurrents de développer des solutions équivalentes se concentrant sur l'essentiel à des coûts inférieurs, et surtout moins contraignantes pour les utilisateurs.

#### **4.2 Réduire le cycle de développement**

Au delà du pur et simple abandon des investissements consacrés à la « protection » des produits, la méthode de développement *Open Source* autorise une diminution sensible des *coûts* par la réduction du *cycle* de développement. Comme le note un observateur,

« Les compagnies innovatrices découvrent qu'en marchant avec *Linux*, elles peuvent influencer une base logicielle dont l'équivalent commercial dépasserait déjà les dix milliards de dollars américains doublant régulièrement --- au lieu de devoir réinventer ces roues et d'être par conséquent en retard sur le marché --- et peuvent se faire un allié de la plus grosse équipe de développement de la planète. » [Le dernier dinosaure et les mares de goudron du destin, Ross Nesbitt, trad. Roland Trique](#). Et ce sans devoir en passer par un schéma désormais classique d'acquisition de petites sociétés en vue de tirer parti de leurs innovations technologiques.

Spontanément en phase avec l'approche modulaire, la démarche *Open Source* autorise l'exploitation de riches gisements de ressources humaines et logicielles et, corollairement, suscite des économies considérables.

Elle peut ainsi rendre accessible le développement de solutions ambitieuses à des sociétés qui n'en n'auraient probablement pas les moyens dans le cadre d'un développement propriétaire fermé. Parallèlement, le constat conserve la même pertinence pour les clients des sociétés de service. Gagnantes sur les délais --- donc les coûts --- ces dernières gagnent aussi en liberté et en indépendance vis-à-vis de leurs fournisseurs. Tandis que les solutions propriétaires aspirent la clientèle dans un cycle de mises à jour et de renouvellement (matériel et logiciel) dont il est extrêmement difficile de se dégager, le choix de l'*Open Source* garantit la possibilité de pouvoir changer à tout moment de prestataire sans pour autant bouleverser le système d'information. Notamment grâce à l'adoption de formats de données et d'échange normalisés et pérennes.

Or, insiste Eric S. Raymond, « plus de 75 % des coûts du cycle de vie typique d'un logiciel se trouvent dans la maintenance ». Dans un contexte propriétaire, cette phase fondamentale peut virer au cauchemar. Alors que les problèmes s'accroissent avec la complexité des solutions mises en oeuvre, il paraît finalement déraisonnable de confier son système d'information à des sociétés qui devront elles-mêmes s'en remettre à des tiers pour maintenir et corriger leur code. Une disposition vraiment fiable pour s'assurer de sa qualité consiste au contraire à en permettre l'examen par des personnes compétentes, ce que l'on a coutume d'appeler dans le monde de l'*Open Source* la « revue des pairs », selon une expression empruntée à la recherche scientifique. À rebours, l'impossibilité de maintenir le code pour en corriger les inévitables dysfonctionnements et adapter les applications à l'évolution des besoins, entraîne dans une spirale de greffes successives, dont l'issue s'avère chaque fois plus problématique, ne faisant que mettre en relief les inconséquences de l'approche initiale.

#### **4.3 Accorder la primauté au service et non au produit**

Bien que l'univers de la production des biens matériels ait fétichisé les « produits » en leur conférant une improbable valeur intrinsèque censée justifier le prix de commercialisation, l'extrême variabilité des coûts de production (si l'on tient compte, en particulier, du cycle de vie et d'amortissement d'une unité de manufacturation) invite déjà à relativiser. Mais cette déconnexion des coûts de production et du prix public est encore plus manifeste en ce qui concerne l'industrie du logiciel, au point d'entretenir un malaise, tant auprès des consommateurs que des producteurs eux-mêmes. Une ambiguïté qui tombe cependant dès lors qu'on prend en compte la notion de service, qui correspond finalement à ce pourquoi le client est prêt à payer, pour autant qu'il juge l'offre adéquate à sa demande.

Face à cette « crise de la valeur » à laquelle se trouve confrontée l'industrie du logiciel et toute activité commerciale à l'ère des médias électroniques, Rishab Aiyer Gosh, rédacteur en chef de la revue en ligne *First Monday* souligne :

« La clé est de donner de la valeur à la diversité, de telle sorte que plusieurs copies d'un seul produit ajoutent peu de valeur ajoutée --- l'utilité marginale est quasi nulle --- alors que des copies simples de produits multiples sont d'une immense valeur aux yeux d'un utilisateur. Si un nombre suffisant de gens contribue à la production de biens libres, la marmite les clones pour tous, de telle sorte que chacun obtient bien plus de valeur que ce que lui même a contribué. » [Les marchés « marmite » : un modèle économique pour le commerce de biens et de services gratuits sur l'Internet](#) », trad. Sébastien Blondeel.

En concentrant leur offre sur la notion de service, les sociétés d'ingénierie informatique ne font pas autre chose. En optant pour l'*Open Source* elles permettent de juger de la validité d'un programme ou d'une solution intégrée à l'aune de son utilité réelle (soumise à la critique, ouverte aux améliorations et développements postérieurs) plutôt que de spéculer sur sa valeur marchande par une tarification arbitraire, ayant pour toute justification la rareté du service supposé rendu --- et non les coûts de production réels. La multiplication, à l'inverse de représenter un facteur de dévalorisation constitue au contraire une aubaine et ne fait qu'accroître la valeur ajoutée, en contribuant à améliorer la pertinence et la robustesse du service proposé. Aussi,

« il ne s'agit plus de déterminer si le capital-risque investira dans l'*Open Source*, mais plutôt de déterminer pourquoi le flot a commencé à s'écouler dans cette direction » [L'avenir du capital risque et de l'investissement dans Linux, in Tribune Libre : Ténors de l'informatique libre](#) Par chance, cela revient à tirer profit commun du partage des connaissances et des ressources, plutôt que fonder le profit sur le secret et la privatisation des idées.

## 5. [La définition de l'Open Source \(OSD\)](#)

Traduction française des textes canoniques de la définition et de la procédure de certification *Open Source*.

[La définition de l'Open Source](#)

[Justifications de l'Open Source](#)

[La procédure de certification OSI](#)

[Les licences certifiées OSI](#)



---

## 6. [Licences \(adaptations françaises\)](#)

Les traductions (non officielles) des principales licences couvertes par l'*Open Source* sont disponibles :

[Licence Publique Générale GNU](#), adaptée par René Cougnenc et Nat Makarévitch

[Licence de documentation libre GNU](#)

[Licence artistique](#), l'une des deux licences sous lesquelles Perl est distribué, adaptée par S. Blondeel

[Adaptations françaises de diverses licences](#) (BSD, X, DEC)

## 7. [Références](#)

[La définition de l'Open Source](#) (B. Perens et la communauté Debian, adapté par S. Blondeel)

[Open Source: Software Gets Honest](#) (en anglais), qui « propose de nombreux points de vues complémentaires sur le phénomène open source ».

[Free Software Foundation](#), créée par Richard M. Stallman. En particulier (en Anglais) [Various Licenses and Comments about Them](#), où figurent les pointeurs vers de nombreuses licences, et les commentaires qu'elles inspirent à la FSF.

[De nombreux textes](#) et auteurs clés de l'*Open Source* et du logiciel libre ont été traduits par Sébastien Blondeel

[Tribune Libre : ténors de l'informatique libre](#), juin 1999, ISBN 2-84177-084-2. Adaptation française de [Open Sources, voices of the open source revolution](#), O'Reilly, janvier 1999, ISBN 1-56592-582-3

[Logiciels libres, Liberté, Égalité, business](#), par Jean-Paul Smets-Solanes et Benoît Faucon, publié aux éditions Edispher, mars 1999 ISBN 2-911-968-7

[Libres enfants du savoir numérique](#), une anthologie du "libre" préparée par Olivier Blondeau et Florent Latrive, publiée aux éditions de l'Éclat, mars 2000, ISBN 2-84162-043-3

Un essai consacré aux nouvelles technologies de l'information et de la communication (NTIC) évoquant la notion de « coopération » : [Ce que va changer la révolution informationnelle](#) par Joël de Rosnay

[Une étude](#) plus particulièrement consacrée aux aspects juridiques soulevés par la GPL, rédigée par Madame Mélanie Clément-Fontaine (doctorante en droit de la propriété intellectuelle)



---

## 8. Les neufs critères de l'OSD

Libre redistribution ;

Disponibilité du code source ;

Autoriser les travaux dérivés ;

Préserver l'intégrité du code source de l'auteur ;

Pas de discrimination envers les personnes ou les groupes ;

Pas de discrimination envers les domaines d'application ;

La licence doit se suffire à elle-même ;

La licence ne doit pas être spécifique à un produit ;

La licence ne doit pas contaminer d'autres logiciels.

## 9. Des sociétés qui contribuent ou promeuvent l'Open Source

[RedHat Software](#)

[MadrakeSoft](#)

[Corel](#), en particulier pour le projet [wine](#)

[Cygnus \(redirection vers <http://www.redhat.com/>\)](#), qui contribue au développement du fameux compilateur [gcc](#)

[IBM](#), notamment pour les projets [Postfix](#) et [Apache](#)



---

## 10. [Les créateurs du concept Open Source](#)

### **Bruce Perens**

Rédacteur de la définition, dérivée d'un travail préalablement effectué pour la distribution Debian. Il se présente comme un « scientifique » spécialisé dans l'informatique. [page personnelle](#)

### **Eric S. Raymond**

. Principal « théoricien » de l'*Open Source*, convaincu de la nécessité de conduire une campagne d'opinion sur ce thème. [page personnelle](#)

Les principaux écrits d'ESR ont été adaptés en français par Sébastien Blondeel :

[La cathédrale et le Bazar](#) --- Expose l'*Open Source* du point de vue de l'ingénierie logicielle ;

[Le chaudron magique](#) --- L'*Open Source* du point de vue du développement économique. Où l'on notera des anticipations de l'actuelle fortune que rencontre le concept de « nouvelle économie » ;

[À la conquête de la noosphère](#) --- Revient plus en détail sur les motivations et ressorts des programmeurs libres, à travers une étude éthologique (comportements et moeurs).

### **Larry Augustin**

président co-fondateur de *VA Research* [VA Linux Systems](#) ;

### **Sam Ockman**

président fondateur de PenguinComputing [PenguinComputing](#).

Autres personnalités, figures historiques du mouvement du logiciel libre :

### **Richard M. Stallman**

Fondateur de la Free Software Foundation. [page personnelle](#)

### **Linux Torvalds**

Créateur du noyau Linux. [page personnelle](#)

### **Alan Cox**

Responsable du développement du noyau Linux, dont il tient le [journal en ligne](#).



---

## **11. Quelques logiciels fameux dont les licences de distribution relèvent de l'Open Source**

[Sendmail](#)

[Postfix](#)

[Apache](#)

[Bind/named](#)

[noyau Linux](#)

[Emacs](#)

[XEmacs](#)

[Perl](#)

[Gimp](#)

[Samba](#)

## **12. Associations**

Voici, classées par ordre alphabétique, les références des principales organisations françaises visant à la promotion des logiciels libres.

[AFUL](#) Association Francophone des Utilisateurs de Linux et des Logiciels Libres

[APRIL](#) Association Pour la Promotion et la Recherche en Informatique Libre

[Freenix](#) qui se présente tout simplement comme « un groupe de personnes utilisant toutes un Unix libre »

[Proselux](#) qui se propose de compiler département par département (voire pays par pays), les coordonnées des utilisateurs du système GNU/Linux et disposés à apporter une aide directe et gracieuse aux personnes désireuses d'installer ce système.



---

### **13. Open Source vs freeware, shareware...**

Afin de circonvier toute confusion sur les notions elles-mêmes, entretenue parfois involontairement par un usage trop peu rigoureux des termes, voici quelques définitions ayant trait aux modes de diffusion de logiciels (voire aux logiciels eux-mêmes) qui *ne relèvent pas* de l'*Open Source* ou du logiciel libre. En pratique, ces notions ne sont pas toujours aussi tranchées qu'il n'y paraît, et certains programmes participent de plusieurs catégories simultanément. Un des buts de la définition officielle de l'*Open Source* (OSD) consistant justement à proposer un cadre suffisamment régulateur et normatif pour écarter toute ambiguïté.

#### **Logiciels propriétaires**

Le code appartient à l'auteur (ou l'entreprise qui l'emploie) et ne concède, généralement, aucun droit particulier sur le devenir de ses programmes, dont les sources sont rarement mises à disposition, et dont il prétend la plupart du temps prévenir toute tentative de décompilation.

#### **shareware**

(ou partagiciel, encore appelé ironiquement *guiltyware* pour stigmatiser la culpabilité des utilisateurs opportunistes). Ces programmes ne sont pas gratuits. L'auteur octroie aux utilisateurs le droit de tester son travail durant une durée de temps limitée, à l'issue de laquelle il faudra verser une certaine somme d'argent si l'on souhaite continuer à utiliser le programme.



---

### **domaine public**

l'auteur qui place un logiciel dans le domaine public renonce à tous ses droits, y compris de propriété intellectuelle. Pas de copyright. Les sources d'un logiciel du domaine public peuvent ne pas être disponibles (et ne le sont généralement pas).

### **freeware**

(ou graticiel) Parfois confondu avec les logiciels du domaine public. En pratique, le terme est réservé aux programmes dont l'utilisation est libre, mais dont le code source n'est pas disponible. Le préfixe *free* désigne ici plutôt la gratuité que la libre disposition, car certaines clauses de licence restreignent parfois ces programmes à une utilisation non commerciale.

### **semi-libres**

Notion improbable, qui pourrait rassembler, par exemple, les logiciels ne correspondant pas *stricto sensu* à la définition de l'*Open Source*. Beaucoup de variantes sont imaginables, à condition qu'on octroie ici un sens très large à « libre » qui, en principe, n'accepte pas la demi-mesure.

### **Voire simple disponibilité du code source**

Dans certains cas, une société de services informatiques peut confier le code source de ses réalisations à un client sans pour autant les « libérer ». Même en dehors d'une diffusion restreinte, la publication du code source ne saurait faire en soi d'un logiciel un logiciel *Open Source*. Ainsi, *Pretty Good Privacy* de Phillip Zimmerman, dont le code source est disponible, notamment pour des raisons de sécurité (puisque'il en va de la crédibilité d'un programme de chiffrement) mais qui n'est pas libre pour autant. Il est à noter, en revanche, que cette pratique contribue à accréditer la supériorité de l'*Open Source*.

# *La définition de l'Open Source*

### 1. Libre redistribution

La licence ne devra pas limiter le droit de vendre ou de donner le logiciel en tant que composant d'un ensemble de programmes. Elle ne doit pas exiger que cette vente soit soumise à l'acquittement de droits d'auteur ou d'une redevance.

### 2. Code source

Le programme doit inclure le code source, et la distribution sous forme de code source comme sous forme compilée doit être autorisée. Si un mode de distribution n'inclut pas le code source, il doit exister un moyen clairement indiqué de se procurer ce dernier pour un coût raisonnable destiné à compenser les frais de reproduction, ou, de préférence, de le télécharger sans frais supplémentaire depuis l'Internet. Par « code source » nous désignons ici la forme la plus adéquate pour un programmeur qui souhaiterait modifier le programme. Il n'est pas autorisé à proposer un code source rendu volontairement difficile à comprendre. Il n'est pas autorisé à proposer des formes intermédiaires, comme ce qu'engendre un préprocesseur ou un convertisseur.

### 3. Travaux dérivés

La licence doit autoriser les modifications et travaux dérivés ainsi que leur distribution sous les mêmes conditions que celles qui protègent le programme originel.

### 4. Intégrité du code source de l'auteur

Toute licence restreignant le droit de redistribution du code source modifié doit autoriser la distribution de la version originelle accompagnée de fichiers de modification (*patches*). La licence doit explicitement permettre la distribution de logiciels construits à partir du code source modifié. La licence peut exiger que les travaux dérivés portent un nom différent ou un numéro de version distinct de ceux du logiciel originel.



---

5. **Pas de discrimination envers les personnes ou les groupes**

La licence ne doit pas établir de discrimination à l'encontre de personnes ou de groupes de personnes.

6. **Pas de discrimination envers les domaines d'application**

La licence ne doit pas limiter le champ d'application du programme. Par exemple, elle ne doit pas interdire l'utilisation du programme à des fins commerciales ou dans le cadre de la recherche génétique.

7. **Distribution de la licence**

Les droits attachés au programme doivent s'appliquer à tous ceux qui l'obtiennent, et ne doivent pas être soumis à une licence complémentaire.

8. **La licence ne doit pas être spécifique à un produit**

Les droits accordés à l'utilisateur du programme ne doivent pas dépendre du fait que le programme appartient à un ensemble donné. Si le programme est extrait d'une distribution et utilisé ou distribué selon les conditions de sa licence, toutes les parties auxquelles il est ainsi redistribué doivent bénéficier des droits accordés lorsque que le programme se trouve au sein d'un ensemble.

9. **La licence ne doit pas contaminer d'autres logiciels**

La licence ne doit pas imposer de restrictions à d'autres logiciels distribués avec le programme. Par exemple, la licence ne doit pas exiger que tous les programmes distribués sur le même support soient des logiciels *Open Source*.



---

## 10. Conformité

(Cette section de fait pas partie de la définition de l'Open Source.)

Nous estimons que la définition de l'Open Source englobe ce qu'une large majorité de la communauté logicielle entendait, et continue d'entendre par l'expression « Open Source ». Cependant, l'usage de cette expression s'est tellement répandu qu'elle a perdu en précision. La marque de certification OSI (*OSI Certified*) est le moyen pour l'OSI de certifier qu'une licence sous laquelle un logiciel est distribué est bien conforme à l'OSD (définition de l'Open Source) ; l'expression générique « Open Source » ne peut pas conférer cette assurance, mais nous continuons à encourager l'usage de l'expression « Open Source » pour signifier sa conformité avec l'OSD. Pour des informations concernant la marque de certification (*OSI Certified*), et pour une liste des licences que l'OSI a approuvées comme conforme à l'OSD, consulter ce [lien](#).

## 11. Historique des révisions:

1.0 -- identique à la DFSG (Debian Free Software Guidelines), excepté l'ajout des MPL et QPL à la clause 10.

1.1 -- ajout de la LGPL à la clause 10.

1.2 -- ajout du domaine public à la clause 10.

1.3 -- changement de titre pour la clause 10 et séparation de la liste des licences, ajouts importants sur les procédures.

1.4 -- Maintenant explicite quant à la nécessité du code source pour les logiciels du domaine public.

1.5 -- autorise « les coûts de reproduction raisonnables » pour se conformer aux termes de la GPL.

1.6 -- Édition de la section 10 ; cet élément a été déplacé.

1.7 -- La section 10 a été remplacée par la nouvelle section « conformité ».

1.8 -- Section 1: remplacement de « ne doit pas » par « ne devra pas ».

Bruce Perens a rédigé un premier brouillon de ce document comme « Directives du logiciel libre Debian » (*The Debian Free Software Guidelines*), puis l'a affiné en tenant compte des commentaires des développeurs Debian, durant un long mois de débats par messagerie électronique en juin 1997. Il a retiré du document les références propres à Debian pour créer la définition de l'Open Source (*Open Source Definition*).



---

# *Justifications de l'OSD*

### 1. Libre redistribution

En contraignant la licence à imposer la libre redistribution, on supprime la tentation de sacrifier d'importants gains à long terme afin de gagner rapidement de l'argent en vendant le logiciel. Sans cette contrainte, les contributeurs subiraient de nombreuses pressions pour l'abandonner.

### 2. Code source

On impose l'accès à un code source qui ne soit pas délibérément rendu difficile à comprendre car on ne peut pas faire évoluer un programme sans le modifier. Puisque notre but est de faciliter l'évolution des programmes, on impose que ces modifications soient facilitées.

### 3. Travaux dérivés

La simple possibilité de lire le code source ne suffit pas pour permettre un examen par les pairs et une rapide évolution par sélection. Pour ce faire, il convient d'accorder l'autorisation d'expérimenter des modifications et de les distribuer.

### 4. Intégrité du code source de l'auteur

C'est une bonne chose que d'encourager de nombreuses améliorations, mais les utilisateurs ont le droit de savoir qui est à l'origine du logiciel qu'ils utilisent. Les auteurs et ceux qui font évoluer un logiciel ont aussi le droit, réciproque, de savoir ce qu'on leur demande de faire évoluer, et de protéger leur réputation.

C'est pourquoi une licence Open Source *doit* garantir que le code source soit facilement accessible, mais *peut* exiger que le code soit distribué dans sa forme originelle accompagnée des fichiers de modification (*patches*). De cette manière, les modifications « non officielles » sont rendues disponibles mais aisément distinguables des sources de base.



---

## 5. ***Pas de discrimination envers les personnes ou les groupes***

Pour tirer le meilleur profit du processus, il faut autoriser, à égalité, la plus grande diversité de personnes et de groupes à contribuer aux logiciels Open Source. C'est pourquoi nous interdisons à toute licence Open Source d'exclure qui que ce soit.

Quelques états, dont les Etats-Unis d'Amérique, restreignent l'exportation de certains types de logiciels. Une licence conforme à l'OSD peut avertir les utilisateurs de ces restrictions et leur rappeler qu'ils doivent obéir à la loi, mais n'imposera pas de telles restrictions par elle-même.

## 6. ***Pas de discrimination envers les domaines d'application***

Le but premier de cette clause est d'interdire les licences piégées qui interdisent d'exploiter commercialement un logiciel Open Source. Nous voulons que les utilisateurs commerciaux rejoignent notre communauté, et ne s'en sentent pas exclus.

## 7. ***Distribution de la licence***

Le but de cette clause est d'interdire la fermeture du logiciel par des moyens indirects tels que l'imposition d'un accord de non divulgation.

## 8. ***La licence ne doit pas être spécifique à un produit***

Cette clause permet d'éviter un autre piège classique des licences.

## 9. ***La licence ne doit pas contaminer d'autres logiciels***

Quiconque souhaite utiliser ou redistribuer des logiciels Open Source doit avoir le droit d'appliquer à ses propres logiciels les conditions de son choix. Oui, la licence GPL est bien conforme à cette clause. Les bibliothèques sous licence GPL « contaminent » seulement les logiciels avec lesquels elles seront liées statiquement lors de la compilation, pas les logiciels avec lesquels elles sont distribuées.



# ***Certification OSI et processus de certification***



---

## 1. Pour faire approuver une licence

Envoyez la licence par E-mail à [licence-approval@opensource.org](mailto:licence-approval@opensource.org). Indiquez dans votre message si vous souhaitez que la licence soit postée sur la liste de discussion *licence-discuss* avec vos références ou bien de façon anonyme. (Nous sommes prêts à prendre en compte des licences que leur auteur ne souhaiterait pas poster sur la liste de discussion, mais comme l'examen par la communauté est une part importante du processus d'approbation, nous devons faire circuler ces licences de manière privée entre différents relecteurs : de ce fait, les licences non postées dans la liste de discussion *licence-discuss* prendront plus de temps à être approuvées, et exigeront certainement plus d'interaction de votre part.)

Si nous estimons que la licence n'est pas conforme à la définition de l'Open Source (*Open Source Définition*), nous travaillerons avec vous à la résolution des problèmes.

Parallèlement, nous supervisons l'activité de la liste de discussion sur les licences (*licence-discuss*), et travaillerons avec vous à résoudre tout problème qui ne serait pas soulevé publiquement.

Au titre même de ce processus de validation, il se peut que nous devions faire appel à un avis légal extérieur au sujet des implications de la licence.

Lorsque la licence conforme à la définition de l'Open Source (*Open Source Définition*) a été acceptée par la liste de discussion ou d'autres relecteurs sans qu'il demeure de point controversé que nous jugerions crucial, nous vous avertirons de son approbation, la placerons sur notre site Web, et l'ajouterons à la liste ci dessous.

## 2. Utilisation du Label

Vous pouvez utiliser le label « Certifié OSI » (*OSI Certified*) avec n'importe quelle distribution logicielle qui comprend -- et se trouve conforme aux exigences de -- toute licence figurant dans la liste ci-dessous, ou avec des logiciels dont le code source est explicitement identifié comme relevant du domaine public.

Pour identifier vos logiciels comme étant « Certifié OSI » (*OSI Certified*), vous devez y adjoindre l'une des deux mentions, non modifiées, telles qu'indiquées ci-dessous. La mention complète est :

```
This software is OSI Certified Open Source Software.  
OSI Certified is a certification mark of the Open Source Initiative.
```

```
Ce logiciel est un logiciel Open Source Certifié OSI.  
La Certification OSI est un label de certification délivré par l'Open Source  
Initiative.
```



---

La forme abrégée est :

OSI Certified Open Source Software

Logiciel Open Source Certifié OSI

Chaque forme de distribution de vos logiciels comporte ses propres exigences :

si le logiciel est distribué sous forme électronique (sous une forme immatérielle), vous devez faire apparaître sa mention complète dans un fichier LISEZMOI (*README*), ou tout autre fichier similaire destiné à être lu en premier par un utilisateur humain ;

Si le logiciel est distribué sous une forme matérielle, vous devez dans la mesure du possible, entreprendre les actions suivantes :

Si le logiciel est distribué avec un support imprimé, vous devez faire figurer la mention complète sous forme imprimée ;

Si le logiciel est distribué via un support informatique amovible, du type disquette, CD-ROM, cartouche, etc., sur lequel il est matériellement possible de placer au moins une courte mention, de telle sorte qu'un oeil humain puisse la lire sans assistance, et sans perturber le fonctionnement du média, vous devez y faire figurer la version longue ou courte de la mention ;

Si l'objet matériel contenant le logiciel est distribué dans un paquetage ne permettant pas de faire figurer la mention de telle sorte qu'elle puisse être lue, vous devez placer la mention complète sur un support accompagnant la distribution.



---

***Licence de  
Documentation Libre  
GNU***



---

## 1. DOMAINE D'APPLICATION ET DÉFINITIONS

La présente Licence s'applique à tout manuel ou travail contenant une mention placée par le détenteur du copyright indiquant que le document peut être distribué selon les termes de la présente Licence. Le terme « Document », ci-dessous, se réfère à tout manuel ou travail remplissant cette condition. Tout membre du public est bénéficiaire de la licence et se trouve ici désigné par « vous ».

Une « Version Modifiée » du Document signifie : tout travail contenant le Document, en intégralité ou en partie, aussi bien une copie *verbatim* ou avec des modifications qu'une traduction dans une autre langue.

Une « Section Secondaire » est une annexe ou un avant-propos du Document qui concerne exclusivement le rapport de l'éditeur ou des auteurs du Document avec le sujet général du Document (ou des domaines voisins) et ne contient rien qui puisse tomber directement sous le coup du sujet général (par exemple, si le Document est en quelque partie un manuel de mathématiques, une Section Secondaire n'enseignera pas les mathématiques). Le rapport peut être une connexion historique avec le sujet ou des domaines voisins, ou une précision légale, commerciale, philosophique, éthique ou politique les concernant.

Les « Sections Invariables » sont certaines Sections Secondaires désignées par leurs titres comme Sections Invariables dans la mention qui indique que le Document est couvert par la présente Licence.

Les « Textes de Couverture » sont certains courts passages du texte listés comme « Textes de Première de Couverture » ou « Textes de Quatrième de Couverture » dans la mention qui indique que le Document est couvert par la présente Licence.

Une copie « Transparente » du Document signifie : une copie lisible par une machine, réalisée dans un format dont les spécifications sont disponibles au grand public, et dont le contenu peut être directement visualisé et édité avec des éditeurs de texte génériques ou (pour les images composées de pixels) avec des programmes de composition d'images génériques ou (pour les figures techniques) un éditeur de dessin vectoriel largement disponible, et qui soit approprié aux logiciels qui mettent le texte en forme et le calibrent (formateurs de texte) ou au transcodage automatique vers un assortiment de formats appropriés aux formateurs de texte. Une copie réalisée dans un format de fichier habituellement Transparent mais dont le balisage a été conçu pour contrecarrer ou décourager des modifications ultérieures par le lecteur n'est pas Transparente. Une copie qui n'est pas « Transparente » est appelée « Opaque ».

Les formats appropriés aux copies Transparentes sont par exemple l'ASCII brut sans balises, le format Texinfo, le format LaTeX, SGML ou XML utilisant une DTD publiquement disponible, et l'HTML simple et conforme à la norme, conçu en vue d'une modification manuelle. Les formats Opaques incluent PostScript, PDF, les formats propriétaires qui ne peuvent être lus et édités que par des traitements de texte propriétaires, SGML et XML dont les DTD et/ou les outils de rendu ne sont pas généralement disponibles, et l'HTML généré automatiquement par certains traitements de texte à seule fin d'affichage.

La « Page de Titre » désigne, pour un livre imprimé, la page de titre proprement dite, plus les pages suivantes qui sont nécessaires pour faire figurer, lisiblement, les éléments dont la présente Licence requiert qu'ils apparaissent dans la Page de Titre. Pour les travaux dont le format ne comporte pas de page de titre en tant que telle, « Page de Titre » désigne le texte jouxtant l'apparition la plus marquante du titre de ce travail, qui précède le début du corps du texte.

## **2. COPIES VERBATIM**

Vous pouvez copier et distribuer le Document sur tout support, aussi bien commercialement que non, pour autant que la présente Licence, les mentions de copyright, et les mentions de licence indiquant que la présente Licence s'applique au Document soient reproduites sur toutes les copies, et que vous n'ajoutiez aucune autre condition à celles de la présente Licence. Vous ne pouvez pas user de moyens techniques à des fins d'obstruction ou de contrôle de la lecture ou de la duplication des copies que vous réalisez ou distribuez. Vous pouvez cependant accepter des compensations en échange de la cession de copies. Si vous distribuez un assez grand nombre de copies, vous devez aussi suivre les conditions de la section [Copies en quantité](#).

Vous pouvez aussi prêter des copies, selon les mêmes conditions que celles mentionnées ci-dessus, et vous pouvez exposer publiquement des copies.

## **3. COPIES EN QUANTITÉ**

Si vous publiez des copies imprimées du Document à plus de 100 exemplaires, et que la mention de la licence du Document exige des Textes de Couverture, vous devez inclure les copies dans des couvertures où figurent, clairement et lisiblement, tous ces Textes de Couverture :

les Textes de Première de Couverture sur la première de couverture, et les Textes de Quatrième de Couverture sur la quatrième de couverture. Les deux faces de la couverture doivent également clairement et lisiblement vous identifier comme étant l'éditeur de ces copies. La première de couverture doit présenter le titre complet, titre dont tous les mots doivent être également mis en valeur et visibles. Vous pouvez ajouter des éléments supplémentaires sur les couvertures. Toute copie avec des changements limités aux couvertures, pour autant qu'ils préservent le titre du Document et satisfont ces conditions, peut être considérée comme une copie *verbatim* à tous les autres égards.

Si les textes destinés à l'une ou l'autre page de couverture sont trop volumineux pour y figurer lisiblement, vous devez en mettre les premiers (autant qu'il est raisonnablement possible) sur la couverture proprement dite, et poursuivre sur les pages adjacentes.

Si vous publiez ou distribuez des copies Opaques du Document à plus de 100 exemplaires, vous devez soit inclure une copie Transparente dans un format lisible par une machine, adapté au traitement automatisé, en accompagnement de chaque copie Opaque, soit indiquer aux côtés de ou dans chaque copie Opaque une adresse de réseau électronique publiquement accessible, qui permette d'obtenir une copie Transparente du Document, sans éléments ajoutés, à laquelle le grand public puisse accéder pour téléchargement anonyme et sans frais en utilisant des protocoles de réseau publics et standard. Si vous retenez la dernière option, vous devez procéder prudemment et prendre les mesures nécessaires, lorsque vous commencez la distribution de copies Opaques en quantité, afin de vous assurer que cette copie Transparente demeurera accessible au public pendant au moins une année après le moment de la distribution (directement ou par l'intermédiaire de vos agents ou revendeurs) de la dernière copie Opaque de cette édition.

Il est souhaité, mais non exigé, que vous contactiez les auteurs du Document bien avant la redistribution de tout grand nombre de copies, afin de leur laisser la possibilité de vous fournir une version mise à jour du Document.

#### **4. MODIFICATIONS**

Vous pouvez copier et distribuer une Version Modifiée du Document selon les conditions des sections [Copies verbatim](#) et [Copies en quantité](#) qui précèdent, pourvu que vous diffusiez la Version Modifiée sous couvert précisément de la présente Licence, avec la Version Modifiée remplissant alors le rôle du Document, et ainsi autoriser la distribution et la modification de la Version Modifiée à quiconque en possède une copie. En complément, vous devez accomplir ce qui suit sur la Version Modifiée :

- A. Utilisez dans la Page de Titre (et sur les couvertures, le cas échéant) un titre distinct de celui du Document et de ceux des précédentes versions (qui doivent, s'il en existe, être citées dans la section « Historique » du Document). Vous pouvez utiliser le même titre qu'une version précédant la vôtre si l'éditeur original vous en donne la permission.
- B. Indiquez sur la Page de Titre, comme auteurs, une ou plusieurs personnes ou entités responsables de l'écriture des modifications de la Version Modifiée, ainsi qu'au moins cinq des principaux auteurs du Document (ou tous les auteurs principaux, s'ils sont moins de cinq).
- C. Apposez sur la Page de Titre de nom de l'éditeur de la Version Modifiée, en tant qu'éditeur.
- D. Préservez toutes les mentions de copyright du Document.
- E. Ajoutez une mention de copyright appropriée à vos modifications, aux côtés des autres mentions de copyright.
- F. Incluez, immédiatement après les mentions de copyright, une mention de licence qui accorde la permission publique d'utiliser la Version Modifiée selon les termes de la présente Licence, sous la forme présentée dans la section [Addendum](#) ci-dessous.
- G. Préservez dans cette mention de licence les listes complètes des Sections Invariables et des Textes de Couverture exigés, données dans la mention de licence du Document.
- H. Incluez une copie non altérée de la présente Licence.
- I. Préservez la section intitulée « Historique », et son titre, et ajoutez-y un article indiquant au moins le titre, l'année, les nouveaux auteurs, et l'éditeur de la Version Modifiée telle qu'elle apparaît sur la Page de Titre. Si le Document ne contient pas de section intitulée « Historique », créez-en une et indiquez-y le titre, l'année, les auteurs et l'éditeur du Document tels qu'indiqués sur la Page de Titre, puis ajoutez un article décrivant la Version Modifiée, comme exposé dans la phrase précédente.
- J. Préservez, le cas échéant, l'adresse de réseau électronique donnée dans le Document pour accéder publiquement à une copie Transparente du Document, et préservez de même les adresses de réseau électronique données dans le Document pour les versions précédentes, sur lesquelles le Document se fonde. Cela peut être placé dans la section « Historique ». Vous pouvez omettre l'adresse de réseau électronique pour un travail qui a été publié au moins quatre ans avant le Document lui-même, ou si l'éditeur original de la version à laquelle il se réfère en donne l'autorisation.
- K. Dans toute section intitulée « Remerciements » ou « Dédicaces », préservez le titre de section et préservez dans cette section le ton et la substance de chacun des remerciements et/ou dédicaces donnés par les contributeurs.



---

L. Préservez toutes les Sections Invariables du Document, non altérées dans leurs textes et dans leurs titres. Les numéros de sections ou leurs équivalents ne sont pas considérés comme faisant partie des titres de sections.

M. Supprimez toute section intitulée « Approbations ». Une telle section ne doit pas être incluse dans la Version Modifiée.

N. Ne changez pas le titre d'une section existante en « Approbations » ou en un titre qui entre en conflit avec celui d'une Section Invariable quelconque.

Si la Version Modifiée inclut de nouvelles sections d'avant-propos ou des annexes qui remplissent les conditions imposées aux Sections Secondaires et ne contiennent aucun élément tiré du Document, vous pouvez, à votre convenance, désigner tout au partie de ces sections comme « Invariables ». Pour ce faire, ajoutez leurs titres à la liste des Sections Invariables dans la mention de licence de la Version Modifiée. Ces titres doivent être distincts de tout autre titre de section.

Vous pouvez ajouter une section intitulée « Approbations », pourvu qu'elle ne contienne rien d'autre que l'approbation de votre Version Modifiée par diverses parties -- par exemple, indication d'une revue par les pairs ou bien que le texte a été approuvée par une organisation en tant que définition de référence d'un standard.

Vous pouvez ajouter un passage de cinq mots ou moins en tant que Texte de la Première de Couverture, et un passage de 25 mots ou moins en tant que Texte de Quatrième de Couverture, à la fin de la liste des Textes de Couverture de la Version Modifiée. Toute entité peut ajouter (ou réaliser, à travers des arrangements) au plus un passage en tant que Texte de la Première de Couverture et au plus un passage en tant que Texte de la Quatrième de Couverture. Si le Document inclut déjà un texte de Couverture pour la même couverture, précédemment ajouté par vous ou, selon arrangement, réalisé par l'entité pour le compte de laquelle vous agissez, vous ne pouvez en ajouter un autre ; mais vous pouvez remplacer l'ancien, avec la permission explicite de l'éditeur qui l'a précédemment ajouté.

Le ou les auteur(s) et le ou les éditeur(s) du Document ne confèrent pas par la présente Licence le droit d'utiliser leur nom à des fins publicitaires ou pour certifier ou suggérer l'approbation de n'importe quelle Version Modifiée.

## 5. MÉLANGE DE DOCUMENTS

Vous pouvez mêler le Document à d'autres documents publiés sous la présente Licence, selon les termes définis dans la section [Modifications](#) ci-dessus, traitant des versions modifiées, pour autant que vous incluez dans ce travail toutes les Sections Invariables de tous les documents originaux, non modifiées, et en les indiquant toutes comme Sections Invariables de ce travail dans sa mention de licence.

Le travail issu du mélange peut ne contenir qu'une copie de cette Licence, et de multiples Sections Invariables identiques peuvent n'être présentes qu'en un exemplaire qui les représentera toutes. S'il existe plusieurs Sections Invariables portant le même nom mais des contenus différents, faites en sorte que le titre de chacune de ces sections soit unique, en indiquant à la fin de chacune d'entre elles, entre parenthèses, le nom de l'auteur original ou de l'éditeur de cette section s'il est connu, ou un numéro unique dans les collisions restantes. Pratiquez les mêmes ajustements pour les titres de sections, dans la liste des Sections Invariables de la mention de licence de ce travail mélangé.

Dans le mélange, vous devez regrouper toutes les sections intitulées « Historique » dans les divers documents originaux, afin de constituer une unique section intitulée « Historique » ; combinez de même toutes les sections intitulée « Remerciements », et toutes les sections intitulées « Dédicaces ». Vous devez supprimer toutes les sections intitulées « Approbations ».

## 6. RECUEILS DE DOCUMENTS

Vous pouvez réaliser un recueil regroupant le Document et d'autres documents publiés sous la présente Licence, et remplacer les diverses copies de la présente Licence figurant dans les différents documents par une copie unique incluse dans le recueil, pour autant que vous suiviez les règles de la présente Licence relatives à la copie *verbatim* pour chacun de ces documents, dans tous les autres aspects.

Vous pouvez n'extraire qu'un seul document d'un tel recueil, et le distribuer individuellement sous la présente Licence, pour autant que vous insériez une copie de la présente Licence dans le document extrait, et que vous suiviez la présente Licence dans tous ses autres aspects concernant la reproduction *verbatim* de ce document.



---

## **7. AGRÉGATION AVEC DES TRAVAUX INDÉPENDANTS**

Une compilation du Document ou de ses dérivés avec d'autres documents ou travaux séparés et indépendants, ou bien sur une unité de stockage ou un support de distribution, ne compte pas comme une Version Modifiée de ce Document, pour autant qu'aucun copyright de compilation ne soit revendiqué pour la compilation. Une telle compilation est appelée une « agrégation », et la présente Licence ne s'applique pas aux autres travaux contenus et ainsi compilés avec le Document, sous prétexte du fait qu'ils sont ainsi compilés, s'ils ne sont pas eux-mêmes des travaux dérivés du Document.

Si les exigences de la section [Copies en quantité](#) en matière de Textes de Couverture s'appliquent aux copies du Document, et si le Document représente moins du quart de la totalité de l'agrégat, alors les Textes de Couverture du Document peuvent n'être placés que sur les couvertures qui entourent le document, au sein de l'agrégation. Dans le cas contraire, ils doivent apparaître sur les couvertures entourant tout l'agrégat.

## **8. TRADUCTION**

La traduction est considérée comme un type de modification, de sorte que vous devez distribuer les traductions de ce Document selon les termes de la section [Modifications](#). La substitution des Sections Invariables par des traductions requiert une autorisation spéciale de la part des détenteurs du copyright, mais vous pouvez ajouter des traductions de tout ou partie des Sections Invariables en sus des versions originales de ces Sections Invariables. Vous pouvez inclure une traduction de la présente Licence pourvu que vous incluiez la version originale, en anglais, de la présente Licence. En cas de désaccord entre la traduction et la version originale, en anglais, de la présente Licence, la version originale prévaudra.

## **9. RÉVOCACTION**

Vous ne pouvez copier, modifier, sous-licencier ou distribuer le Document autrement que selon les conditions expressément prévues par la présente Licence. Toute tentative de copier, modifier, sous-licencier ou distribuer autrement le Document est nulle et non avenue, et supprimera automatiquement vos droits relatifs à la présente Licence. De même, les parties qui auront reçu de votre part des copies ou des droits sous couvert de la présente Licence ne verront pas leurs licences révoquées tant que ces parties demeureront en pleine conformité avec la présente Licence.



---

## **10. RÉVISIONS FUTURES DE LA PRÉSENTE LICENCE**

La *Free Software Foundation* (« fondation du logiciel libre ») peut publier de nouvelles versions révisées de la présente *GNU Free Documentation License* de temps à autre. Ces nouvelles versions seront similaires, dans l'esprit, à la présente version, mais peuvent différer dans le détail pour prendre en compte de nouveaux problèmes ou de nouvelles inquiétudes. Consultez <http://www.gnu.org/copyleft/>.

Chaque version de la Licence est publiée avec un numéro de version distinctif. Si le Document précise qu'une version particulière de la présente Licence, « ou toute version postérieure » s'applique, vous avez la possibilité de suivre les termes et les conditions aussi bien de la version spécifiée que de toute version publiée ultérieurement (pas en tant que brouillon) par la *Free Software Foundation*. Si le Document ne spécifie pas un numéro de version de la présente Licence, vous pouvez choisir d'y appliquer toute version publiée (pas en tant que brouillon) par la *Free Software Foundation*.

## **11. ADDENDUM : Comment utiliser la présente licence dans vos documents**

Pour utiliser la présente Licence dans un document que vous avez rédigé, insérez une copie de la présente Licence dans le document et placez le copyright et les mentions de licence suivants juste après la page de titre :

```
Copyright (c) ANNÉE VOTRE NOM.  
Permission est accordée de copier, distribuer et/ou modifier ce  
document selon les termes de la Licence de Documentation Libre GNU  
(GNU Free Documentation License), version 1.1 ou toute version  
ultérieure publiée par la Free Software Foundation ; avec les  
Sections Invariables qui sont LISTE DES TITRES ; avec les  
Textes de Première de Couverture qui sont LISTE, et avec les  
Textes de Quatrième de Couverture qui sont LISTE. Une copie de  
la présente Licence est incluse dans la section intitulée  
« Licence de Documentation Libre GNU ».
```

Si vous n'avez pas de Sections Invariables, écrivez, « sans Sections Invariables » au lieu d'en indiquer la liste. Si vous n'avez pas de Textes de Première de Couverture, écrivez « sans Texte de Première de Couverture » au lieu de « les Textes de Quatrième de Couverture qui sont LISTE » ; et de la même manière pour les Textes de Quatrième de Couverture.

Si votre document contient des exemples non triviaux de code de programmation, nous recommandons de diffuser ces exemples en parallèle sous la licence libre de votre choix, comme la [Licence Publique Générale GNU](#) ( [GNU General Public License](#)), afin de permettre leur utilisation dans des logiciels libres.



---

## **12. Les sources de l'Open sources**

[www.sourceforge.net](http://www.sourceforge.net)

[www.phpscript-fr.net](http://www.phpscript-fr.net)