Chapitre 1

Logiciel Libre, Open Source, formats et protocôles ouverts : une introduction.

Qu'est-ce qu'un logiciel libre ? Quel est son histoire ? Que différence y-at-il avec l'*Open Source* ? Qu'est-ce qu'un format ou un protocôle *ouvert* ? Comment cela se compare à une *norme* ? Qui écrit des logiciels libres ? Quel est le modèle économique du logiciel libre ? Pourquoi est-il préféré au logiciel propriétaire dans certaines administrations ? Comment on dévéloppe du logiciel libre ? Comment fonctionne la communauté de dévéloppeur d'un logiciel libre ? Comment peut-on légalement choisir le statut de son propre logiciel ? Qu'es-te qu'une *licence*, un *brevet*, une *marque* ?

Voilà quelques-unes des questions auxquelles nous allons essayer d'apporter des éléments de réponse dans ces notes de cours.

1.1 Définitions

On retrouve utilisés quelques fois de façon confuse les termes *freeware*, *shareware*, *free software*, donc il vaut la peine de préciser leur nature :

Freeware
: ce terme indique le logiciel *gratuit*, mais la gratuité est une propriété qui peut changer avec le temps sans préavis, pour des simples raisons commerciales

Shareware
: ce terme indique les logiciels payants, mais fournis avec une période d'essai gratuite

Free Software, Open Source, Logiciel Libre
: Là, il s'agit de quelque chose de radicalement différent, à l'origine du succés du Web et de l'Internet

1.1.1 Free Software (Logiciel Libre)

Le terme *free software*, introduit par Richard Stallman au début des années 1980, a malheureusement donné lieu à confusion en raison de la surcharge du mot *free* en anglais, au point que l'on a pris l'habitude en anglais de repeter

> Free software as in free speech, not as in free beer.

Heureusement, dans les langues latines, cette confusion ne paraît pas :

Logiciel *Gratuit* (anglais : freeware) :
logiciel non payant (aujourd'hui)

Logiciel *Libre* (anglais : free software) :
logiciel avec 4 droits fondamentaux
- Liberté d'utiliser le logiciel
- Liberté d'étudier les sources du logiciel et de l'adapter à ses besoins
- Liberté de distribuer des copies
- Liberté de distribuer les sources (éventuellement modifiées)

Il y a des obligations aussi, qui varient selon la licence : GPL/BSD/Mozilla/X, et qui font l'objet d'une étude spécifique dans le chapitre sur les aspects légaux.

Malgré la confusion introduite par l'omonymie en anglais, liberté et gratuité sont des attributs absolument orthogonaux du logiciel, comme l'on peut voir en énumerant des exemples de logiciels qui ont les quatre possibles combinaisons de ces attributs :

non libre, gratuit :
Internet Explorer, MacTCP, Acrobat Reader, freeware, etc.

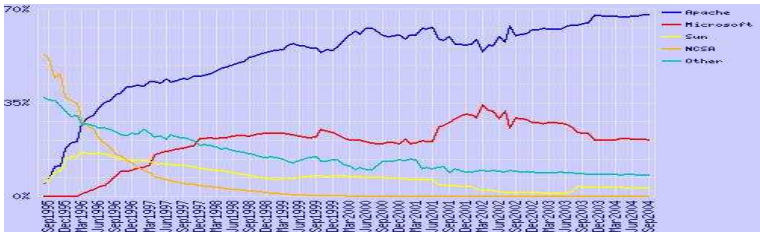
non libre, non gratuit :
le plus connu ...

libre, gratuit :
Mozilla, Linux, FreeBSD, OpenBSD, sendmail, perl, etc.

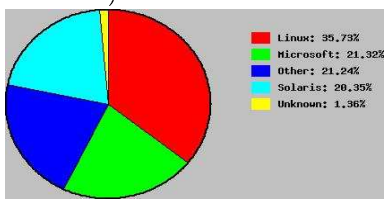
libre, non gratuit :
distributions commerciales de Linux, etc.

1.1.2 Quelques exemples et statistiques sur l'usage des logiciels libres

L'infrastructure même de l'Internet fonctionne grace à des logiciels libres : `sendmail`, `bind`, `TCP/IP`, `apache`, etc.
Ces logiciels ont un très haut dégré de penétration dans le marché des serveurs :



Apache (free software) domine le marché des serveurs web



Linux (free software) leader sur le marché des OS pour serveur web
(Source : NetCraft, Octobre 2004)

1.1.3 Le logiciel libre respecte le droit des auteurs

n'est pas Napster
L'auteur choisit *librement* d'écrire du logiciel libre

n'est pas du "domaine public", ni "libre de droits"
L'auteur *protège* la *liberté* de son logiciel par une licence

ne relève pas d'une "logique d'abandon"
L'auteur choisit une logique de valorisation innovante pour son logiciel

protège la propriété intellectuelle
La disponibilité de logiciels équivalents libres ...
reduit la copie illégale !

1.1.4 Logiciel libre vs. propriétaire

Logiciel libre accès au code, liberté de modifier et distribuer :
- avantages pédagogiques indéniables : accès à une meilleure formation (à l'informatique)
- multiplie[1] le nombre des programmeurs qui vérifient le code, divise les pirates : l'accès au code source attire les programmeurs compétents
- redonne le contrôle aux utilisateurs
- permet d'échapper à la fuite en avant technologique

Logiciel propriétaire : pas d'accés au code, pas de modifications, pas de distribution
- ne permet pas d'adapter le logiciel, ni de le comprendre
- aucun contrôle de l'évolution technologique
- multiplie les pirates, divise les vérificateurs

[1] Ecrire du logiciel à plusieurs est *difficile*, mais verifier le logiciel à plusieurs est *faisable !*