Cours Logiciels Libres

Licences libres

Théo Zimmermann

Télécom Paris, Institut Polytechnique de Paris

Vendredi 27 janvier 2023

Introduction : pourquoi c'est important

- Les licences de logiciels déterminent ce que vous avez le **droit** ou non de faire avec.
- Votre entreprise encourt des **risques économiques** à ne pas respecter les termes des licences.
- À titre individuel, si vous ne courez pas de tel risques, il y a tout de même de bonnes raisons de prêter attention aux licences :
  - Une raison **morale** : le logiciel libre est une forme de dons sous conditions. Acceptez les conditions ou refusez le don.
  - Une raison **pratique** : vous pouvez mettre en danger vos utilisateurs et certains refuseront donc d'utiliser vos logiciels si vos pratiques de gestion des licences laissent à désirer.

La propriété intellectuelle en bref

La propriété intellectuelle est composée de plusieurs champs, dont les trois suivants :

- Le **droit d'auteur** (**copyright** dans les pays anglo-saxons).
- Les **brevets**.
- Le droit des **marques**.

Ces trois domaines peuvent concerner le **logiciel**.

Note : le terme "propriété intellectuelle" est controversé (car très loin de la notion de propriété sur des objets physiques). Mais il est pratique pour regrouper les différentes notions sous-jacentes.

Le droit d'auteur

- Il protège des **œuvres originales de l'esprit**, indépendamment de leur support.
- Il protège l'**expression** (créative) et non les idées.
- En France, il s'applique automatiquement, sans besoin d'enregistrement préalable.
- Il comporte deux parties :
  - Le **droit moral** (notamment reconnaissance de la paternité de l'œuvre) inaliénable.
  - Les **droits patrimoniaux** (monopole d'exploitation pour durée limitée, par exemple 70 ans après la mort de l'auteur) qui peuvent être cédés.
- Correspond au **copyright** dans les pays anglo-saxons (sans la partie droit moral).

Les brevets

- Ils permettent d'obtenir un **monopole** sur l'exploitation d'inventions industrielles.
- Ce monopole est à **durée limitée** (généralement 20 ans) et nécessite un **dépôt** (de brevet).
- L'obtention d'un brevet nécessite de prouver que l'idée est **nouvelle**, **utile** et **non-évidente**.
- Le principe des brevets est basé sur un compromis censé favoriser l'**innovation** :
  - Le dépôt de brevet nécessite de rendre l'invention **publique**.
  - En échange, on accorde un monopole temporaire à son inventeur.
  - Mais au terme de cette durée, n'importe qui peut se servir de l'invention.

Les marques

- Noms, slogans, logos, etc.
- Elles permettent de protéger le consommateur car elles indiquent la **provenance** d'un produit.
- Elles ne nécessitent pas toujours un enregistrement, mais doivent être **activement utilisées** et sont limitées à un **marché spécifique**.

Application au logiciel

- Copyright / droit d'auteur : depuis les années 70.
  - Plus simple à appliquer (pas besoin d'enregistrement).
  - Protège les logiciels qui ne contiennent pas d'idées nouvelles.
  - S'applique aux codes **sources**, aux codes **binaires**/**objets**, aux éléments **graphiques**, et aussi à la **documentation**.
  - Règles spéciales pour les droits patrimoniaux du logiciel.
- Brevets :
  - Initialement considérés comme ne s'appliquant pas au logiciel.
  - Puis accordés de plus en plus facilement, aux USA et dans l'UE.
  - Peuvent empêcher les **clones** libres de logiciels propriétaires.
  - Générateurs de **risques juridiques**.
- Marques : peuvent s'appliquer aux noms et logos de logiciels, y compris de logiciels libres.

Licences

- Une licence est un **contrat** qui permet au détenteur (du copyright, d'un brevet) d'accorder des droits au bénéficiaire de la licence.
- Tous les logiciels (libres et propriétaires) sont distribués avec des licences (**pas de licence = pas de droits**) :
  - Lorsque un logiciel propriétaire est vendu, les utilisateurs doivent "acheter" des licences.
  - Dans le cas du libre, les licences accordent des droits (qui peuvent être plus ou moins vastes) à tous ceux qui **reçoivent** le logiciel.
- Les licences logicielles reposent **principalement sur le droit d'auteur/copyright**, et secondairement sur les brevets (certaines licences libres modernes incluent des clauses sur les brevets).

Conditions standards des licences libres

License	Commercial use	Distribution	Modification	Patent use	Private use	Disclose source	License and copyright notice	Network use is distribution	Same license	State changes	Liability	Trademark use	Warranty
BSD Zero Clause License	●	●	●		●						●		●
Academic Free License v3.0	●	●	●	●	●		●			●	●	●	●
GNU Affero General Public License v3.0	●	●	●	●	●	●	●	●	●	●	●		●
Apache License 2.0	●	●	●	●	●		●			●	●	●	●
Artistic License 2.0	●	●	●	●	●		●			●	●	●	●
BSD 2-Clause "Simplified" License	●	●	●		●		●				●		●
BSD 3-Clause Clear License	●	●	●	●	●		●				●		●
BSD 3-Clause "New" or "Revised" License	●	●	●		●		●				●		●
BSD 4-Clause "Original" or "Old" License	●	●	●		●		●				●		●
Boost Software License 1.0	●	●	●		●		●				●		●

Figure 1: Capture d'écran de https://choosealicense.com/appendix/

Conditions standards des licences libres

Toujours autorisés :

- **Usage** privé, usage commercial.
- **Modification**.
- **Distribution** (à l'identique, ou d'une version modifiée).

Brevets :

- Licence explicite.
- Absence de mention de la question des brevets.
- Absence explicite de licence sur les brevets.

Conditions possibles :

- **Paternité** / préservation de la licence / du copyright.
- Partage des sources, modifications sous la **même licence** (copyleft).
- Traçage des modifications.

Licences libres les plus connues

Permissives :

- MIT.
- BSD3, BSD2.
- Apache 2.0 (protection vis-à-vis des brevets, obligation de traçage des changements).

Copyleft faible :

- MPL 2.0 (partage des modifications dans les mêmes fichiers).
- GNU LGPL 3.0, LGPL 2.1 (utilisation de bibliothèque).

Copyleft fort :

- GNU GPL 2.0, GPL 3.0 (ajoute protection vis-à-vis des brevets, de la “tivoization”).
- GNU AGPL 3.0 (modification sur serveur $\Longrightarrow$ distribution).

Compatibilité entre licences

Les licences de deux logiciels libres sont compatibles si on peut créer un logiciel qui **dérive des deux à la fois**.

Exemple d'**incompatibilité** (entre licences copyleft) :

- La licence GNU GPL 2.0 (de Linux par exemple) impose que tout travail dérivé soit distribué sous licence GNU GPL.
- La licence CDDL (de Solaris) impose que tout travail dérivé soit distribué sous licence CDDL.

Compatibilité entre licences

Autre exemple d'incompatibilité (plus subtil) :

- La licence GNU GPL impose que tout travail dérivé soit distribué sous cette licence et empêche l'ajout de nouvelle restrictions.
- La licence BSD originale (4 clauses) contient une clause ajoutant des restrictions sur la publicité :

  All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by [project].

Compatibilité entre licences

La compatibilité avec les licences GNU est très utile, car les logiciels sous ces licences sont très répandus.

En pratique, de nombreuses licences ont été **modifiées suite à des problèmes d'incompatibilité** avec les licences GNU (en particulier la GPL) :

- BSD,
- Apache,
- MPL,
- CeCILL,
- etc.

Les licences copyleft ont souvent besoin d'une clause spécifique autorisant l'**inclusion dans un logiciel GPL**.

Licences multiples

- On trouve parfois des logiciels qui sont distribués sous plusieurs licences. Dans ce cas, **l'utilisateur peut choisir** quelle licence s'applique.
- Certaines licences (notamment la GPL) peuvent être augmentées de **clauses additionnelles** (qui peuvent seulement accorder des droits supplémentaires, et peuvent être retirées) :
  - Classpath exception (Java)
  - LGPL 3.0
- L'auteur peut choisir d'autoriser l'utilisation de **nouvelles versions** d'une licence.
  - C'est une **bonne pratique** mais qui nécessite de faire confiance à l'organisme qui produit ces licences).
  - GPL-2.0-only (Linux) vs GPL-2.0-or-later.
  - Certaines licences le rendent automatique (exemple MPL 2.0).

Domaine public

Définition : une œuvre est dans le domaine public lorsqu'elle **n'est plus protégée** par des droits patrimoniaux.

"Mettre une œuvre dans le domaine public" : ce n'est **pas juridiquement possible** dans tous les pays. Par conséquent, des licences existent qui accordent des **droits équivalents** au cas où ce ne serait pas permis (Unlicense, CC-0).

Choisir une licence libre

Règle 1 : on n'oublie pas de mettre une licence sur son code !

Sans licence :

- Vos **utilisateurs** n'ont aucun droit.
- Vous n'avez aucun droit sur les **contributions externes** !

Choisir une licence libre

Règle 2 : on n'invente pas sa propre licence !

Conséquence de la prolifération de licences :

- Risques juridiques (licences moins comprises, moins testées).
- **Incompatibilité** entre licences (difficile de combiner des logiciels libres entre eux).

Il vaut mieux choisir une licence **populaire** :

- Soit une licence populaire en général (MIT, BSD2, BSD3, Apache 2.0, MPL 2.0, licences GNU).
- Soit une licence populaire dans l'**écosystème** auquel on contribue (mais en général elle fait partie de la liste précédente).

Choisir une licence libre

Règle 3 : on choisit une licence qu'on comprend !

- Certaines licences ont des conditions **complexes**. Par exemple la licence GNU LGPL donne des droits supplémentaires, mais à des conditions techniques qui ne sont pas autant adaptées à tous les langages de programmation.
- Il est **difficile de changer de licence** plus tard, une fois qu'on a reçu de nombreuses contributions externes.

Choisir une licence libre

- Domaine public ?
- Permissive ?
- Protection contre les brevets ?
- Copyleft faible ?
- Copyleft fort ?
- Le logiciel est-il destiné à tourner sur un serveur ?

Et pour la documentation ?

- Le plus commun est que la documentation soit sous la **même licence que le code**.
- Mais des licences spécifiques aux contenus autres que logiciels existent (par exemple, les licences **Creative Commons**).

Creative Commons

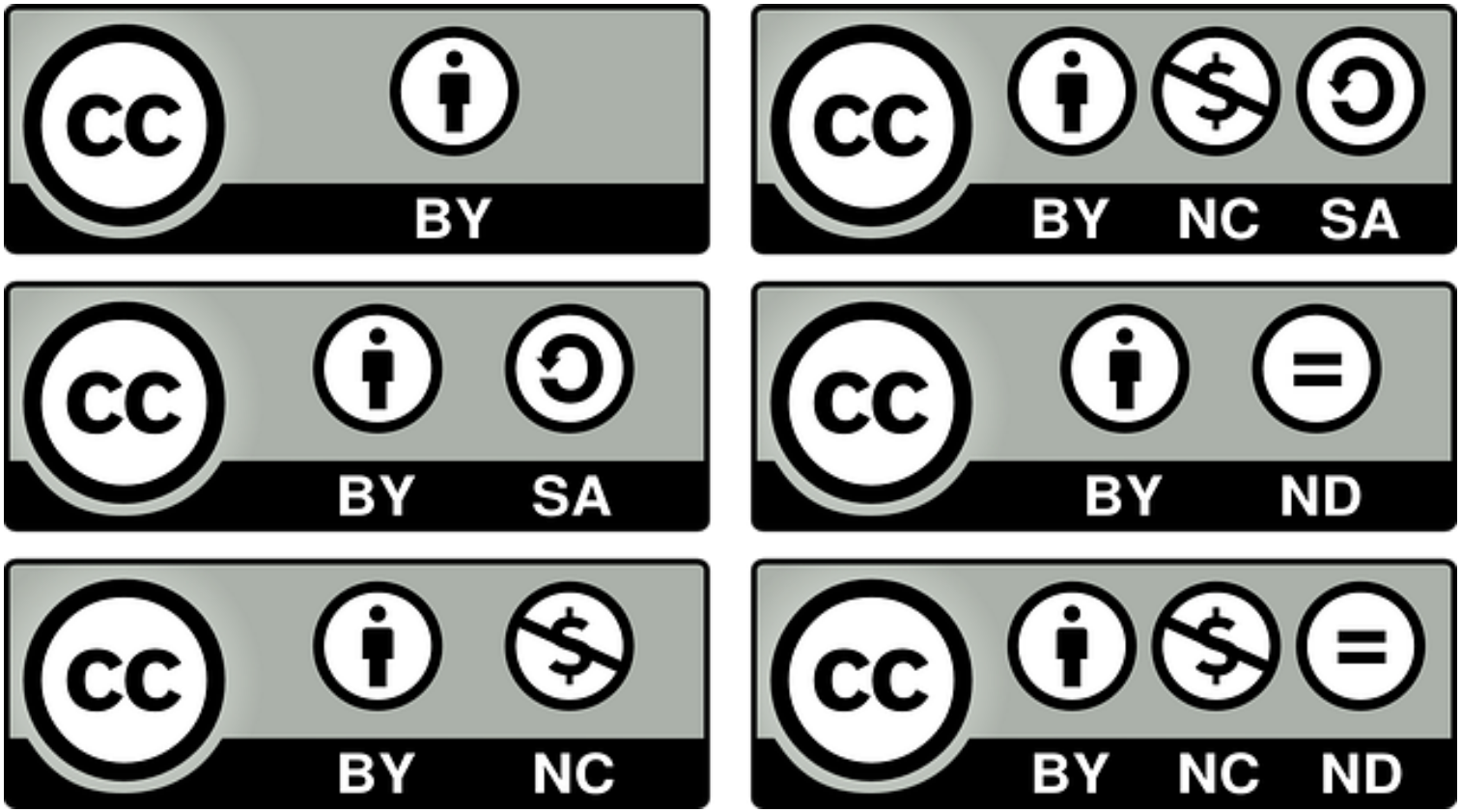

Figure 2: Les différentes licences Creative Commons

Attention ! Seules CC-BY et CC-BY-SA sont des licences libres.

Appliquer une licence libre

Les recommandations varient d'une licence à l'autre :

- En général, on inclut le **texte de la licence** dans un fichier `LICENSE` à la racine du projet.
- Certaines licences recommandent en plus l'ajout d'un **commentaire en haut de chaque fichier** de code source.
- Les fichiers de **métadonnées pour gestionnaires de paquets** (tel que `package.json` pour npm) ont souvent un champ pour indiquer la licence. On utilise alors en général la typologie **SPDX** pour ce champ.

Typologie SPDX https://spdx.org/licenses/

Licences les plus communes :

- MIT
- BSD-2-Clause, BSD-3-Clause
- Apache-2.0
- MPL-2.0
- LGPL-2.1-only, LGPL-2.1-or-later, LGPL-3.0-only, **LGPL-3.0-or-later**
- GPL-2.0-only, GPL-2.0-or-later, GPL-3.0-only, **GPL-3.0-or-later**
- AGPL-3.0-only, **AGPL-3.0-or-later**

Expressions :

- MIT **OR** Apache-2.0 (licences multiples)
- GPL-3.0-only **WITH** Classpath-exception-2.0 (clauses additionnelles)

Application de licences aux contributions

Certains projets demandent de “signer” des documents :

- DCO (Developer Certificate of Origin),
- CLA (Contributor License Agreement),
- CTA (Copyright Transfer Agreement).

Quelles peuvent être les raisons de ces demandes ?

DCO (Developer Certificate of Origin)

- Créé pour le projet Linux, également utilisé pour d'autres projets.
- Le contributeur s'engage sur l'origine de la contribution et sur son droit à la soumettre à la licence du projet.
- Très léger : nécessite une ligne par commit (Signed-off-by: Your Name <your@email.com>), option `-s` de git.

CLA (Contributor License Agreement)

Même utilité qu'un DCO, plus des droits supplémentaires (par exemple pour l'entreprise qui maintient le projet) :

- Par exemple, le droit d'inclure la contribution dans un code propriétaire.
- Ou bien le droit de changer la licence.

CTA (Copyright Transfer Agreement)

- Cession des droits partrimoniaux.
- Utilisé par la Free Software Foundation pour pouvoir lancer des procès au nom de tous les contributeurs.

Application de licences aux contributions

Extrait des Terms of Services de GitHub :

> *Whenever you add Content to a repository containing notice of a license,* ***you license that Content under the same terms, and you agree that you have the right to license that Content under those terms****. If you have a separate agreement to license that Content under different terms, such as a contributor license agreement, that agreement will supersede.*
> *Isn't this just how it works already? Yep. This is widely accepted as the norm in the open-source community; it's commonly referred to by the shorthand "inbound=outbound".* ***We're just making it explicit.***