浅谈市面上存在的主流开源许可证的区别

从 1946 年第一台计算机真正开始使用，软件都还没有体系化，工程化，软件的开发还没有像现在这样简便快速，都只能是写死在计算机中。而 70 多年后的如今，简洁高效的软件开发语言层出不穷，实现复杂服务的框架也是开箱即用，软件的工程化体系也不断完善，出现了非常多的开发流程管理模式和工具，软件开发再也不像以前那样低效冗杂。

而可以说软件开发世界有现在这样的动能和活力，绝对离不开开源这一划时代的思想。开源给整个开发世界带来了新鲜的活力，人们再也不需要重复地制造达到同一目的的模块，同时软件开发世界的人都能从优秀的项目中学习和吸收创作者的奇思妙想，然后开发者们一起开发和维护整个项目，使得这个项目不断地注入新鲜的血液，不断地修复漏洞和长期更新，同时也为整个软件开发世界带来了稳定高可用的产品。这样的想法所带来的收益几乎惠及了所有在这个世界生活探索的人。Linux 社区长青的缘由很大一部分就源自这股开源的魔力。

但是，软件开发走到现在，不可能再像以前那样拥有长时间的开发周期，大量人力物力的支持，因此当一个团队开发出优秀的项目时，我们既希望这个团队能开源这个项目供大家学习参考，同时这个团队也需要资金来维持这个项目的不断开发和维护。

这时，相当多的团队出于经济层面和市场竞争层面的考量，就会选择将自己的项目包装成一份产品或是服务对外售卖，同时将整个项目闭源。这种考量无可厚非，因为现如今软件的开发和维护需要大量的人力物力支持，不得不选择通过这种方式来确保自己的收益。

但是，之前就说过了，开源永远是软件开发世界的核心力量和新鲜血液的来源。

有的团队允许任何人使用修改自己的项目，同时能直接将这个修改版直接发布出去，但是必须带上同样的开源许可证。也有团队选择将自己的项目开源，只要使用时带上相应的开源许可证就能免费使用，同时允许修改代码后能直接将项目闭源发布。

不断地市面上出现了各式各样的开源模式，也诞生了相当多不同含义的开源许可证。充分地扩充了开源世界的内涵，同时也为本来不允许商业化的完全开源带来了新的血液。

如今市面上存在的开源许可证大概有上百种，其中的区别还是很大的，但大多数人可能并不知道其中具体到底有什么区别，不知道自己使用的开源项目到底是属于什么模式，也不知道自己准备开源自己开发的项目时应该如何选择开源许可证。最大的问题可能还是误以为自己使用的某一项目的服务是开源的，自己就能随意使用，这种想法在商业用途时，非常容易带来版权问题。

开源协议和许可证基本可以分为两种大的类型，宽松式和 Copyleft 许可证，两者有很明显的区别。宽松式包含没有使用限制，没有担保，披露要求的特点。也就是说，用户可以做任何想做的事，但是不保证代码质量，自担风险且用户必须披露原始作者的信息。Copyleft 虽然允许用户随意复制，但限制会更多。例如分发二进制格式时必须要提供源码；修改后的源码的许可证必须与修改前保持一致；不得在原始许可证以外添加其他限制。

BSD 开源协议以及 BSD 开源许可证就是属于宽松式的一员。这是一个给与使用者极大自由的协议，基本上可以说是为所欲为。在这个协议下的项目，能修改源码，也能将修改后的代码开源或是闭源地发布出去。但为所欲为也不是完全的，必须遵守相应的规则。如果再发布的产品中包含源代码，则在源代码中必须带有原来代码中的 BSD 协议。如果再发布的只是二进制类库/软件，则需要在类库/软件的文档和版权声明中包含原来代码中的 BSD 协议。可以用开源代码的作者/机构名字和原来产品的名字做市场推广。

BSD 代码鼓励代码共享，但需要尊重代码作者的著作权。BSD 由于允许使用者修改和重新发布代码，也允许使用或在 BSD 代码上开发商业软件发布和销售，因此是对商业集成很友好的协议。而很多的公司企业在选用开源产品的时候都首选 BSD 协议，因为可以完全控制这些第三方的代码，在必要的时候可以修改或者二次开发。

Apache License 是著名的非盈利开源组织 Apache 采用的协议。该协议和 BSD 类似，同样鼓励代码共享和尊重原作者的著作权，同样允许代码修改，再发布（作为开源或商业软件）。需要满足的条件也和 BSD 类似，需要给代码的用户一份 Apache License；如果你修改了代码，需要再被修改的文件中说明。在延伸的代码中（修改和有源代码衍生的代码中）需要带有原来代码中的协议，商标，专利声明和其他原来作者规定需要包含的说明。如果再发布的产品中包含一个 Notice 文件，则在 Notice 文件中需要带有 Apache License。你可以在 Notice 中增加自己的许可，但不可以表现为对 Apache License 构成更改。

Apache License 也是对商业应用友好的许可。使用者也可以在需要的时候修改代码来满足需要并作为开源或商业产品发布/销售。

MIT 是和 BSD 一样的宽松式许可协议,作者只想保留版权,而无任何其他了限制.也就是说,你必须在你的发行版里包含原许可协议的声明,无论你是以二进制发布的还是以源代码发布的.

GPL 协议作为 Copyleft 协议中的一员和 BSD, Apache License 等鼓励代码重用的许可很不一样。我们很熟悉的 Linux 就是采用了 GPL。GPL 的出发点是代码的开源/免费使用和引用/修改/衍生代码的开源/免费使用，但不允许修改后和衍生的代码做为闭源的商业软件发布和销售。这也就是为什么我们能用免费的各种 linux，包括商业公司的 linux 和 linux 上各种各样的由个人，组织，以及商 业软件公司开发的免费软件了。

GPL 协议的主要内容是只要在一个软件中使用(“使用”指类库引用，修改后的代码或者衍生代码)GPL 协议的产品，则该软件产品必须也采用 GPL 协议，既必须也是开源和免费。这就是所谓的“传染性”。GPL 协议的产品作为一个单独的产品使用没有任何问题，还可以享受免费的优势。

由于 GPL 严格要求使用了 GPL 类库的软件产品必须使用 GPL 协议，对于使用 GPL 协议的开源代码，商业软件或者对代码有保密要求的部门就不适合集成/采用作为类库和二次开发的基础。其它细节如再发布的时候需要伴随 GPL 协议等和 BSD/Apache 等类似。

LGPL 是 GPL 的一个为主要为类库使用设计的开源协议。和 GPL 要求任何使用/修改/衍生之 GPL 类库的的软件必须采用 GPL 协议不同。LGPL 允许商业软件通过类库引用(link)方式使用 LGPL 类库而不需要开源商业软件的代码。这使得采用 LGPL 协议的开源代码可以被商业软件作为类库引用并发布和销售。

但是如果修改 LGPL 协议的代码或者衍生，则所有修改的代码，涉及修改部分的额外代码和衍生的代码都必须采用 LGPL 协议。因此 LGPL 协议的开源 代码很适合作为第三方类库被商业软件引用，但不适合希望以 LGPL 协议代码为基础，通过修改和衍生的方式做二次开发的商业软件采用。GPL/LGPL 都保障原作者的知识产权，避免有人利用开源代码复制并开发类似的产品。

MPL 是 The Mozilla Public License 的简写，是 1998 年初 Netscape 的 Mozilla 小组为其开源软件项目设计的软件许可证。MPL 许可证出现的最重要原因就是，Netscape 公司认为 GPL 许可证没有很好地平衡开发者对源代码的需求和他们利用源代码获得的利益。同著名的 GPL 许可证和 BSD 许可证相比，MPL 在许多权利与义务的约定方面与它们相同（因为都是符合 OSIA 认定的开源软件许可证）。但是，相比而言 MPL 还有以下几个显著的不同之处。

MPL 虽然要求对于经 MPL 许可证发布的源代码的修改也要以 MPL 许可证的方式再许可出来，以保证其他人可以在 MPL 的条款下共享源代码。但是，在 MPL 许可证中对“发布”的定义是“以源代码方式发布的文件”，这就意味着 MPL 允许一个企业在自己已有的源代码库上加一个接口，除了接口程序的源代码以 MPL 许可证的形式对外许可外，源代码库中的源代码就可以不用 MPL 许可证的方式强制对外许可。这些，就为借鉴别人的源代码用做自己商业软件开发的行为留了一个豁口。

MPL 许可证第三条第 7 款中允许被许可人将经过 MPL 许可证获得的源代码同自己其他类型的代码混合得到自己的软件程序。

MPL 许可证不像 GPL 许可证那样明确表示反对软件专利，但是却明确要求源代码的提供者不能提供已经受专利保护的源代码（除非他本人是专利权人，并书面向公众免费许可这些源代码），也不能在将这些源代码以开放源代码许可证形式许可后再去申请与这些源代码有关的专利。

在 MPL（1.1 版本）许可证中，对源代码的定义是:“源代码指的是对作品进行修改最优先择取的形式，它包括:所有模块的所有源程序，加上有关的接口的定义，加上控制可执行作品的安装和编译的‘原本’（原文为‘Script’），或者不是与初始源代码显著不同的源代码就是被源代码贡献者选择的从公共领域可以得到的程序代码。”

MPL 许可证第 3 条有专门的一款是关于对源代码修改进行描述的规定，就是要求所有再发布者都得有一个专门的文件就对源代码程序修改的时间和修改的方式有描述。

未来的开发世界必然是开源的世界，即使现在开源世界式微，但有着相当多的个人和团队选择为开源贡献自己的一份力量，也不断地有大的商业公司加入也开源的行列。而未来的知识产权也会更加完善，开源协议和许可证也会更加完善，这些协议就会成为那时新的协议和许可证的基石，不断为开源世界和开发世界注入新的能量。