汉坤法律评述

2023 年 3 月 23 日

HANKUN
汉 坤 律 师 事 务 所
Han Kun Law Offices

北京 | 上海 | 深圳 | 海口 | 武汉 | 香港

“未履行开源许可证义务的专有软件不受法律保护” — GPL 抗辩的探讨（上）

作者：段志超 | 鲁学振 | 蒋海楠[1]

在专有软件中使用开源软件但未履行开源许可证义务可能会影响专有软件权利人的后续行权。我们注意到近期南京市中级人民法院（“**南京中院**”）在一起涉及开源代码的软件著作权侵权案（（2021）苏 01 民初 3229 号）中，首次支持了被告的“GPL 抗辩”，驳回了原告的侵权主张[2]。根据南京中院在本案中确立的裁判规则，在计算机软件著作权侵权案件中，如果作为原告权利基础的计算机软件使用了 GPL 开源代码而未履行开源义务，权利人就该计算机软件所享有的著作权将不再得到法律保护，第三人可使用专有软件相关代码而无需承担侵权责任。**南京中院在本案中的裁判观点为使用开源软件的企业提出了更高的合规要求，值得业界关注。**

本文将以案涉 GPL 开源软件为切入点，从技术分析和法律适用层面逐步厘清法院对 GPL 抗辩的审理思路，辨析 GPL 抗辩的法律依据，并最终就企业合规给出建议。我们对该案件的分析将分为上下两篇，本篇主要从技术层面出发，分析 GPL v2 及 Classpath Exception 的适用问题；下篇主要从法律适用的角度探讨 GPL 抗辩成立的法理基础。

目次

[1] 实习生梁杰对本文的写作亦有贡献。

[2] 参见知产宝 — 版权案例 | 全国首起民事生效判决—GPL 抗辩获得法院支持！，https://mp.weixin.qq.com/s/GiMWAdzbqg83OUlwdAQPrQ。

一、案件简述

本案中，原告研发了“未来软件—投标文件制作工具”软件，该软件使用了开源软件 SharpZipLib（适用 GPL v2.0 with Classpath Exception 许可证[3]）。原告认为被告所开发的“云蜻蜓软件—投标文件制作工具”使用了其软件代码，且被告对代码亦存在接触可能性，因此以软件著作权侵权为由将被告诉至法院。被告认为，原告软件使用 GPL 组件，整体被 GPLv2 协议约束；即便认定原被告软件构成实质性相似，被告也因和原告存在 GPL 许可关系而不构成侵权。此外，原告未以 GPL 协议开源其代码，属于违反诚实信用原则，不应该支持其侵权诉讼请求。

法院最终认定，原告在**主程序**中使用了 GPL 组件，且其使用方式不满足 Classpath Exception，原告负有将其软件代码以 GPL 协议开源的义务但未履行，不支持就相关代码的侵权诉讼请求。原告开发软件的**预览程序**未使用 GPL 代码，被告对该部分代码的使用侵犯了原告的著作权，需承担惩罚性赔偿的法律责任。

二、何为 Classpath Exception？

涉案开源软件为 SharpZipLib，主要功能是实现文件压缩，采用 GPL v2.0 with Classpath Exception 许可证。根据该许可证要求，如果在使用过程中将 SharpZipLib 与独立模块链接以生成可执行软件，那么最终形成的可执行文件因满足 Classpath 例外，而不受 GPL 协议约束，无需履行许可证义务。

Classpath Exception（也称 Classpath 例外）的原文及译文我们摘录如下:

- *Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.*

  As a special exception, ***the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules****, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module.* ***An independent module is a module which is not derived from or based on this library****. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.*

  将此库与其他模块静态或动态地链接就是基于该库制作组合作品。因此，GNU 通用公共许可证的条款和条件涵盖了整个组合。

  *作为一个特殊例外，****此库的版权所有者允许您将此库与独立模块链接以生成可执行文件，而不管这些独立模块的许可条款如何****，并根据您选择的条款复制和分发生成的可执行文件，前提是对于每个链接的独立模块，您还满足该模块许可证的条款和条件。****独立模块即不从该库派生或基于该库的模块****。如果您修改此库，则可以将此例外扩展到您的库版本，但您没有义务这样做。如果您不希望这样做，请从您的版本中删除此异常声明。*

追根溯源，Classpath Exception 并非由 SharpZipLib 作者首创，其最早出现在为 Java 程序设计而开发的

[3] https://github.com/icsharpcode/SharpZipLib，SharpZipLib 经历过协议变更，由之前的 GPL v2 with classpath exception 已经变更为 MIT 协议。

OpenJDK 开源项目中[4]。

Java 应用程序的开发需要使用 JDK（Java Development Kit）将编写好的源代码文件转换为可供运行的应用程序。在此过程中，编写好的源文件.java 在 JDK 中编译成.class 字节码文件，JVM 再将.class 字节码文件翻译为机器可读的机器码（二进制语言）[5]。在解释器（Interpreter）进行解释的过程中，链接（Link）操作将会被执行[6]，目标程序（开发者自有部分）与被调用的库函数（存在于 JDK 中）将被**链接**在一起。

OpenJDK 是常用的开源的 JDK，早期版本适用 GPLv2 协议。而根据 GPLv2 协议，链接将导致 GPL 下的传染问题。因此，通过 OpenJDK 进行源代码编译将导致整个软件需按照 GPL 协议开源，这对于 Java 生态而言是极不友好的。针对该问题，OpenJDK 的项目作者在 GPL v2.0 上附加了 Classpath 例外[7]，允许用户通过静态/动态链接方式使用 OpenJDK 而不履行开源义务。

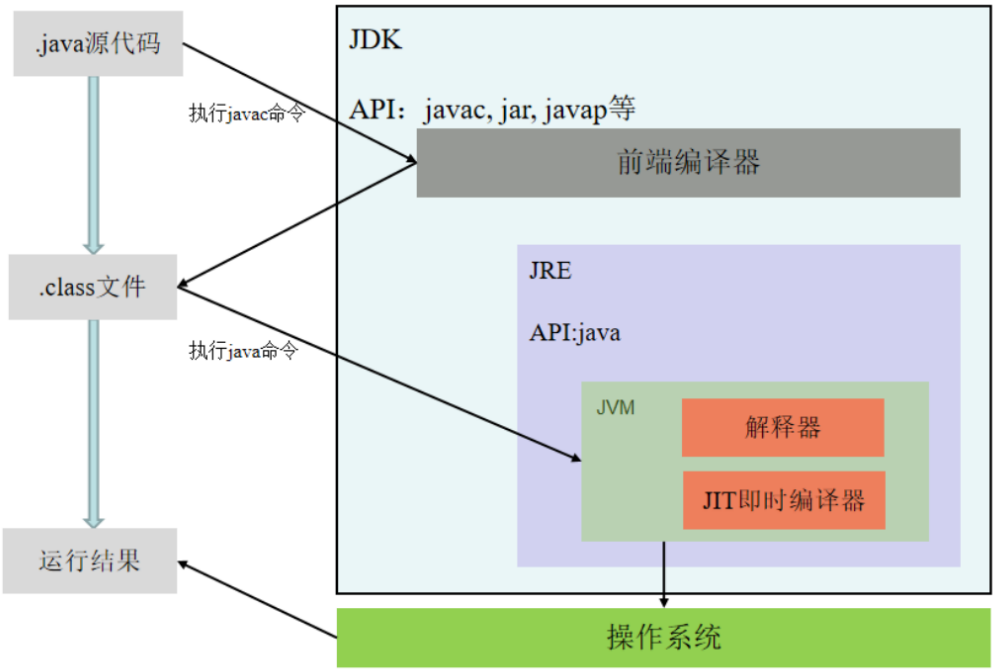

图 1 Java 应用程序与 JDK 的关系

通过对 GPL 协议设置例外，以促进开源软件传播的情形并不少见。例如，开源软件 MySQL 即设定了 Universal FOSS Exception，允许 MySQL 与特定开源软件通过**接口**通信而不需履行开放源代码的义务[8]。

[4] 参见维基百科词条—GPL Linking Exception，https://wiki.alquds.edu/?query=GPL_linking_exception。

[5] 一倾而尽，什么是 JDK，JRE，JVM---深入分析，https://blog.csdn.net/weixin_38339025/article/details/90313695。

[6] chun_soft，Java 代码编译和执行的整个过程，https://chunsoft.blog.csdn.net/article/details/113697528?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~Rate-1-113697528-blog-5904542.pc_relevant_recovery_v2&depth_1-utm_source=distribute.pc_relevant.none-task-。

[7] 参见 https://openjdk.org/legal/gplv2+ce.html。

[8] https://oss.oracle.com/licenses/universal-foss-exception/。

三、涉案开源软件是如何与原告的其他代码通信的？

回归本案，本案中法院已查明原告软件中就开源软件 SharpZipLib 调用方式是基于函数的调用，其调用方式见下图所示：

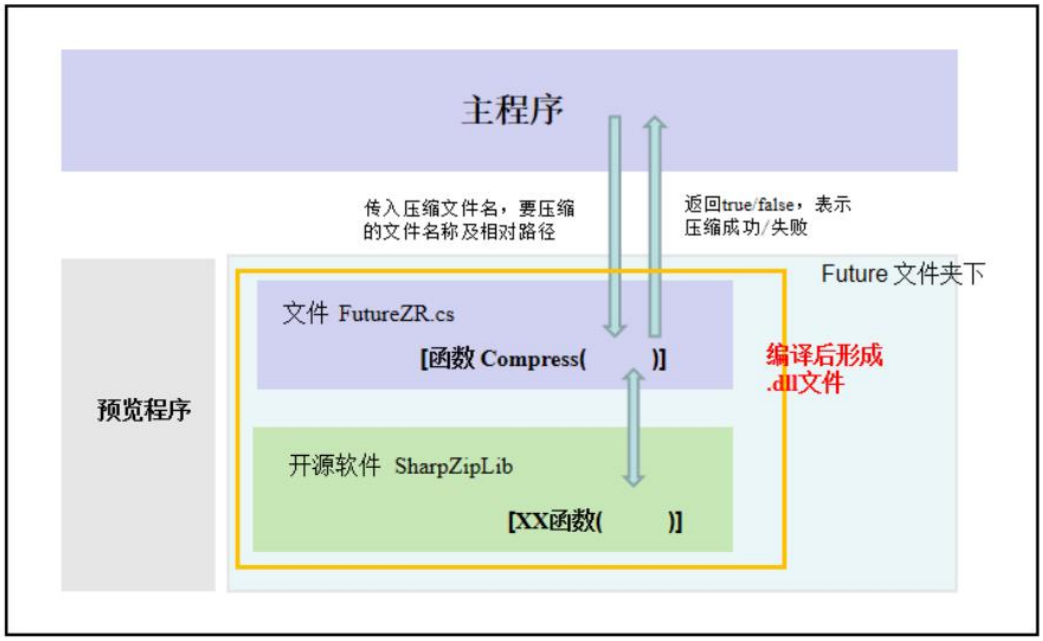

图 2 涉案软件与开源软件 SharpZipLib 的关系

具体而言：

1. 原告的软件分为主程序和预览程序两部分，两部分相互独立；
2. 预览程序不调用开源软件 SharpZipLib；
3. 主程序将开源软件 SharpZipLib 作为动态链接库的一部分而调用，该过程的具体实现方式如下：

- 原告在自行编写的 FutureZR.cs 文件中创建函数 CompressZip()；
- 该.cs 文件和 SharpZipLib 代码被一同编译为.dll（动态链接库）文件；
- 主程序运行时调用 dll 文件。

在实际使用中，为实现压缩功能，主程序调用.cs 文件中的 CompressZip()函数 1 次，CompressZip()继而调用 SharpZipLib 中的函数以实现压缩，并根据执行结果向主程序返回 true/false 表示压缩成功还是失败。

四、原告软件是否被 GPL 组件传染？

由于原告软件主程序调用了开源软件 SharpZipLib，原被告双方就主程序是否受到 GPL v2.0 约束未进行过多争辩。法院综合（i）主程序与涉案 GPL 开源代码存在函数调用关系，且（ii）涉案 GPL 开源代码实现的压缩功能系投标文件上传前不可或缺的功能两点，认为主程序与开源代码结合为衍生作品，受 GPL 协议约束。

就判断 GPL 协议的“传染”范围，本判决提出，需对软件中原告独立创作的“自有代码”部分和从开源社区引入的“开源代码”进行区分，明确二者的交互关系以及开源代码的功能及其在软件中所起的作用，被传染的程序应当是与原开源软件形成密切通信使得二者高度牵连融合成一体的程序，而非只要有数据交换就会构成传染。（本文概括为“**密切通信**”规则）

法院的上述“密切通信”规则与自由软件基金会（FSF）[9]提出的判断是否“*通过共享或交换复杂数据结构而建立了密切的通信*”[10]的判断规则是一致的。FSF 还认为“If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program”[11]（如果两个模块运行时是在共享地址空间连接在一起的，那几乎肯定意味着它们被组合成一个程序）。也即，FSF 认为运行时共享地址空间的两个模块大概率存在“复杂数据结构”的交换，属于“密切通信”，会引发 GPL 传染。回归到本案中，主程序采用动态链接库形式调用开源软件，主程序处于**共享的地址空间**内，一般意义上认为已被传染。

本文认为，涉案软件是否被 GPL 传染是有探讨空间的。本案的特殊之处在于，主程序函数调用交换的数据并不复杂。FSF 也提出“如果主程序动态链接了插件，但是它们之间的通信限于调用插件的‘主’函数及其参数并等待其返回，那么这就是个可以算单一组合程序也可以算两个独立程序的**临界情况**”[12]。换言之，在虽处于共享空间但**数据结构交换较为简单**的情况下，主程序是否会被 GPL 插件传染，FSF 是保有疑议的。本文注意到，本判决在在界定传染范围时将该开源软件所起到的功能同样纳入分析“涉案 GPL 开源代码实现的压缩功能系投标文件上传前**不可或缺的功能**，因此，主程序系涉案 GPL 开源代码的衍生作品”。本文认为这种分析路径值得商榷，一者，这种基于“功能论”的观点在 GPL 文本以及 FSF 官方问答中没有依据；二来，既然特定的开源软件代码已经被用于所开发的软件中，那么在通常的情形下，它就会承担一定的功能。

五、原告软件是否适用 Classpath Exception？

在认定主程序已被 GPL v2 传染后，南京中院就 Classpath Exception 是否适用进行了分析，并认定本案并不适用该例外。

双方争论焦点主要集中与本案是否符合 Classpath Exception 中“**独立模块**”、“**链接调用**”的例外规定，而这两点恰好也是适用 Classpath 例外的构成要件。

为方便阅读，本文总结相关事实、原被告主张和法院认定如下：

Classpath Exception（原文）	本库文件的版权所有者允许你将本库文件与**独立模块链接**以生成可执行文件，而不管这些独立模块的开源许可条款如何……一个独立的模块是一个**不从本库衍生或**

[9] FSF 是一个致力于推广自由软件的美国民间非营利性组织。它于 1985 年 10 月由理查德·斯托曼创建。其主要工作是执行 GNU 计划，推动施行 GPL 协议，开发更多的自由软件。GPL 许可证最初由 FSF 的理查德·斯托曼为 GNU 项目所撰写，FSF 分别于 1991 年和 2007 年发布了 GPL v2 和 GPL v3。参见维基百科词条“FSF”、“GPL”。

[10] “什么时候一个程序和它的插件会被认为是一个单一的结合在一起的程序？这依赖于主程序如何调用插件。如果主程序使用 fork 和 exec 调用插件，那么它们之间通过共享或交换复杂数据结构而建立了密切的通信，这样它们就是一个单一的结合在一起的程序。如果主程序使用的是简单的 fork 和 exec 调用插件，并没有建立密切的通信，那么插件就是一个单独的程序。”参见 https://www.gnu.org/licenses/gpl-faq.html。

[11] 参见 https://www.gnu.org/licenses/gpl-faq.en.html#MereAggregation。

[12] “If the main program dynamically links plug-ins, but the communication between them is limited to invoking the ‘main’ function of the plug-in with some options and waiting for it to return, that is a borderline case.”参见 https://www.gnu.org/licenses/gpl-faq.en.html#GPLPlugins。

	基于本库的模块。
原告主张	FutureZR.cs 为独立模块与主程序链接，满足例外情况。
被告主张	SharpZipLib 本身提供了用于链接调用的“ICSharpCode.SharpZipLib.dll”，但是原告却自行编写 FutureZR.cs 以调用 SharpZipLib 中的函数，并一起重新编译为 FZR.dll 以进行调用。“FZR.dll”以及相应的源代码属于对开源代码的演绎。原告调用 FZR.dll 不属于链接调用 SharpZipLib，不满足例外情况。
法院认定	原告自行编写的 FutureZR.cs 文件与 SharpZipLib 开源代码一同编译成 FZR.dll，并由主程序对其进行调用的方式，不符合“例外声明”中“一个独立的模块是一个不从本库衍生或基于本库的模块”关于独立模块的约定，**例外声明对其不适用。**

- **争议点：本案是否存在 Classpath Exception 中的“独立模块”，是否以链接方式调用？**

  本案中，原告主张 FutureZR.cs 作为“独立模块”与主程序链接满足例外情况，南京中院未予认可。具体地，南京中院认为“通过将自行编写的 FutureZR.cs 文件与 SharpZipLib 开源代码一同编译成 FZR.dll，并由主程序对其进行调用的方式，不符合‘例外声明’中‘一个独立的模块是一个不从本库衍生或基于本库的模块’关于**独立模块**的约定，‘例外声明’对其不适用”。

本文认为，原告若将主程序主张为 Classpath Exception 中的“独立模块”，本案结果或有不同。具体原因如下：

1. **主程序满足 Classpath Exception 中的“独立模块”。**

   原告的主程序非基于或衍生（Derive From）自开源组件，满足独立模块的概念。

2. **主程序与开源软件的调用满足“链接”调用。**

   前文分析可知，主程序通过动态链接的形式调用 SharpZipLib 的 modified version（即 FutureZR.cs 文件与 SharpZipLib 开源代码一同编译成的 FZR.dll 文件），满足链接调用。该环节可能存在的争议点是主程序并非直接与 SharpZipLib 链接调用，然而本文认为这并不影响 Classpath Exception 的成立。一方面，Classpath Exception 仅限定是以“链接形式结合独立模块”，并没有限定链接用文件的来源或产生方式；另一方面，即使原告自行编写了.dll 文件，.dll 以及相应的源代码确如被告所述属于对开源代码的演绎，但这仅是使得这部分代码被认定为衍生作品受 GPL 开源义务约束，与主程序调用方式是否满足 Classpath 例外无关。

六、案件之外：企业开源合规的启示

南京中院此项裁判实际上对企业的开源合规提出了较高的要求，在不满足开源义务的情形下甚至可以除却软件权利人请求侵权赔偿的胜诉权。因此，企业的开源合规便显得更为必要。

（一）使用 GPL 组件的合规方式仍未被颠覆

本判决对 GPL 协议的解读仍然是遵循 GPL 协议文本，且与 FSF 官方的理解基本保持一致。

本案中，原告将 GPL 组件与自由代码在同一地址空间运行，存在整个程序被认定为“衍生自”GPL 组件的风险。**作为风险缓释措施**，原告可采取管道（Pipes）、Sockets、命令行方式（Command-line）等方式以确保自有部分与开源组件在隔离的地址空间中进行通信，将自有代码和开源组件部分加以区隔。

增加自有代码独立性，避免被认定为“通过共享或交换复杂数据结构而建立了密切的通信[13]”以致被认定为衍生自（Derived From）开源组件。就 Classpath Exception 而言，考虑到尚不存在官方的解释，其具体合规调用方式存在一定的不确定性，可采用业界较为成熟的 GPL 的合规方式对相关组件进行处理。

（二）建立完整的开源合规体系尤为关键

本案中，原告因未能履行开源协议，其基于自有代码的软件著作权实施受到阻碍。**企业通过在内部搭建开源软件合规体系，在软件开发阶段引入开源合规理念，则可通过小成本撬动大收益。通过企业内部成体系的开源合规搭建，可实现对开源软件的引入、更新和退出进行全生命周期的管理。**

我们注意到，本案中原告软件所使用的 SharpZipLib 经历了多次协议的变更，从最初的 GPL v2 with Classpath Exception 变更到 LGPL，再到 2016 年变为非常宽松的 MIT 协议。本案被告被认定的侵权行为系从 2017 年 1 月 1 日至 2021 年 8 月 30 日，在此期间，SharpZipLib 已有适用 MIT 的版本[14]。如果原告对其软件中的 SharpZipLib 开源组件存在监控和管理，则可直接换用 MIT 版本，届时将不存在自有代码被传染的问题，以致侵权行权受限，难以主张足额损害赔偿的问题。

[13] https://www.gnu.org/licenses/gpl-faq.en.html#MereAggregation。

[14] 参见 SharpZibLib 1.0.0 版（2018 年发布）的 MIT 协议，https://github.com/icsharpcode/SharpZipLib/blob/v1.0.0/LICENSE.txt。

特别声明

汉坤律师事务所编写《汉坤法律评述》的目的仅为帮助客户及时了解中国或其他相关司法管辖区法律及实务的最新动态和发展，仅供参考，不应被视为任何意义上的法律意见或法律依据。

如您对本期《汉坤法律评述》内容有任何问题或建议，请与汉坤律师事务所以下人员联系：

段志超

电话： +86 10 8516 4123
Email： kevin.duan@hankunlaw.com