www.jusletter-it.eu

Daniel Ronzani

Linking Exceptions under OSS Licenses

Category of articles: TechLawNews by Ronzani Schlauri Attorneys
Field of law: IP-Law

Citation: Daniel Ronzani, Linking Exceptions under OSS Licenses, in: Jusletter IT 30 March 2023

[1] In order not to reinvent the wheel each time software code is written, developers often link their (base) code to other software components. This process is called linking. A code link editor combines one or more object files into a single executable file, library file[1], or another «object» file.[2] There are two basic types of linking: static and dynamic.

[2] *Static linking* is the result of the code link editor making a (dedicated) *copy* of all used library functions to the executable (binary) file. The code is extracted from the linked component at the time the executable is *built*, i.e. at «*compile time*». Examples of libraries which are statically linked are «.a» files in Linux and «.lib» files in Windows. A practical use case could be integrating a tool library directly into an application.[3]

[3] *Dynamic linking* is the result of the dynamic code editor loading and linking the shared libraries needed by a binary file only when it is *executed*, i.e. at «*run time*».[4] This can be achieved by linking only the *name* of the linked component (e.g. library) to the binary file. Examples of libraries which are dynamically linked are «.so» files in Linux and «.dll» files in Windows.[5] A practical use case could be linking to a maths library if the base code needs to resolve a mathematical equation.[6]

[4] The benefit of dynamic linking deferring the linking process until a program starts running is, among others, that the code does not need to be copied and it is thus also easier to update should the linked component (e.g. library) change; whereas the benefit of static linking can be stability (everything need is «in one file» and it does not change). A disadvantage of dynamic linking, however, might be higher costs due to repeated linking at each runtime; whereas a disadvantage of static linking is that each updated library needs to be repackaged and redeployed.[7]

[5] Creating a derivative work based on software licensed under open source (OSS) will, depending on the OSS license applied, trigger a copyleft effect (also) for the newly created code. Does this mean that linking *OSS components* (e.g., libraries) to non-OSS (or OSS with weak copyleft effect) qualifies as a derivative work? If so, it would mean that the downstream user of the compiled, binary software linking to the OSS component would be entitled to receive the *source* code of the (compiled) binary code under the OSS license of the linked OSS component(s).[8]

[6] A solution to this problem is the *linking exception* foreseen by certain OSS licenses, by which parts of the software only linking (whether statically or dynamically) to the open source component (e.g. library) are *not* considered a derivative work and thus do *not* need to be disclosed. Three selected examples are:

a. Generally, under the Mozilla Public License 2.0 (MPL-2.0)[9] derivative works must be made available to anyone to whom the source code is distributed. However, new files containing no MPL-licensed code are *not* considered *derivative works*, and therefore do *not* need

1 IBM, Libraries (30.1.2023), tinyurl.com/mrrz5wv3.

2 Wikipedia, Linker (computing), tinyurl.com/3rup4wey.

3 Abhijit Saha, GeeksforGeeks, Static and Dynamic Libraries (Set 1), 14.10.2019, tinyurl.com/2hr78snd.

4 Wikipedia, Dynamic linker, tinyurl.com/3e3wmjf4.

5 Abhijit Saha, GeeksforGeeks, Static and Dynamic Libraries | Set 1, 14.10.2019, tinyurl.com/2hr78snd.

6 Lithmee, PEDIAA, What is the Difference Between Linker Loader and Compiler, 1.11.2018, tinyurl.com/bdewdtfk.

7 IBM, Libraries (30.1.2023), tinyurl.com/mrrz5wv3.

8 See in this context also the article on «Derivative Works in OSS» by Daniel Ronzani, in Jusletter IT 30 March 2023.

9 Open Source Initiative, MPL-2.0, tinyurl.com/yv9a44aa.

to be distributed under the terms of the MPL-2.0 license, even if using, compiling, or distributing the non-MPL files together with MPL-licensed files.[10] The software code licensed under MPL-2.0 can be kept in separate files (aka «file-based copyleft license»), allowing the software code linking to the MPL-component (e.g., library) being released under different license terms despite being an aggregate work (derivative work).[11] A similar concept applies under the Common Development and Distribution License 1.0 (CDDL-1.0)[12].

b. The GNU Lesser General Public License version 3 (LGPL-3.0)[13] is also a weak copyleft OSS license[14]. Under this license it is permissible to convey a derivative work *without* being bound by section 3 of GNU GPL-3.0 (i.e., the section protecting the users' legal rights from DRM[15]), granting, generally, the right to incorporate *material from header files* of the linked into the *object code* of the application. The object code may then be conveyed under terms of choice, i.e. other than the OSS terms of the linked library.
Whereas, static linking is possible under LGPL-3.0, dynamic linking is considered best practice; the reason being that dynamic linking requires neither access to the library's source code nor other information or materials for the downstream user to be able to rebuild the program.[16]

c. Finally, the GNU Classpath[17], a free software implementation of the standard class library for the Java programming language[18], is a *special exception* under the GNU General Public License version 3 (GPL-3.0)[19]. The copyright holders of a GPL-3.0 library may grant permission (i) to link such library with independent modules to produce an executable, *regardless of the license terms* of these independent modules, and (ii) to copy and distribute the resulting executable *under terms of choice*, provided one also meets, for each linked independent module, the terms and conditions of the license of that module.[20]

[7] The foregoing examples show that using OSS components in proprietary software code does not necessarily contaminate the latter. The process of creating and conveying derivative works with the possibility of dual licensing requires interplay between the objective of the code (technical components) and selection of suitable OSS licenses, and their linking exceptions, respectively.

10 moz://a, MPL 2.0 FAQ, Q11: How «viral» is the MPL? If I use MPL-licensed code in my proprietary application, will I have to give all the source code away? tinyurl.com/3wyhxkfs.

11 FOSSA, Open Source Software Licenses 101: The LGPL License, 20.8.2021, tinyurl.com/2p8799tz.

12 Open Source Initiative, CDDL-1.0, tinyurl.com/bddmadtk.

13 Open Source Initiative, LGPL-3.0, tinyurl.com/msp35t8n.

14 FOSSA, Open Source Software Licenses 101: The LGPL License, 20.8.2021, tinyurl.com/2p8799tz.

15 Cf. art. 11 of the WIPO Copyright Treaty 1996, tinyurl.com/y7whwwrn.

16 FOSSA, Open Source Software Licenses 101: The LGPL License, 20.8.2021, tinyurl.com/2p8799tz.

17 GNU Classpath, 2.4.2018, tinyurl.com/4aduytft (own emphasis).

18 Wikipedia, GNU Classpath, tinyurl.com/4bktxtzy.

19 Open Source Initiative, GPL-3.0, tinyurl.com/ycksdafs.

20 GNU Classpath, 2.4.2018, tinyurl.com/4aduytft (own emphasis).